

SMART CCTV CAMERA

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

SAKITHYAN I (Reg.No - 40111108)

SANJAY S (Reg.No - 40111137)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHISALAI, CHENNAI - 600119

APRIL - 2024

SMART CCTV CAMERA

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By
SAKITHYAN I (Reg.No - 40111108)
SANJAY S (Reg.No - 40111137)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHISALAI, CHENNAI - 600119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SAKITHYAN I (40111108)**, **SANJAY S(40111137)** who carried out the project entitled "**SMART CCTV CAMERA**" under my supervision from November 2023 to April 2024.

V. Nirmalrani
10/04/2024
Internal Guide

Dr. V. NIRMALRANI, M.TECH., Ph.D.

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Head of the Department
Rajiv Gandhi Salai,
Chennai - 600 119
Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on 15/04/2024

(Signature)
Internal Examiner

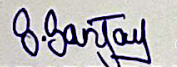
(Signature)
15/4/2024
External Examiner

DECLARATION

I, **SANJAY S(40111137)** hereby declare that the Project Report entitled "**SMART CCTV CAMERA**" done by me under the guidance of **Dr. V. NIRMALRANI M.TECH., Ph.D.** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 10.04.24

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. V. Nirmalrani, M.E., Ph.D.**, for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my phase-1 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Analysing human behavior and recognizing objects are integral elements of an intelligent closed-circuit television system. These technologies enable the detection of anomalous occurrences in the surrounding anomalies human conduct, and the level of alertness in the surroundings. These systems employ machine learning and machine vision features to find and recognize certain abnormalities that appear in the CCTV video feed. For these systems, frame-by-frame processing is frequently employed, and supervised learning is frequently utilised as a training technique.

Abnormal event detection, human behavior detection, as well as object recognition plays a vital role in the creation of a smart CCTV system. These systems make it possible to detect abnormal events in an environment, abnormal behaviors by humans and the state of alert in the environment. Machine Vision property along with Machine Learning are used in these systems to detect as well as identify the particular anomalies that arise in the video feed from the CCTV. Frame by frame processing is commonly used and Supervised Learning is the commonly used training method for these systems.

However, since the anomalies are of many different kinds and also because it is not feasible to pre-detect and train all types of anomalies, supervised learning is being replaced by unsupervised learning and semi - supervised learning for training the system.

This system provides a means of minimising or removing the human workload that has to be put on to manually detect and create an alert on detection of an abnormality in the live feed provided by the CCTV. Also the system increases the storage efficiency by storing only the abnormal events in original quality and storing the normal scenarios in low quality for archiving. Also this system provides an extension of creating a distributed abnormality classification system, where only the abnormal events are sent on to different dedicated systems to classify the abnormality.

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	vii
1	INTRODUCTION	8
2	LITERATURE SURVEY	10
	2.1 Inferences from Literature Survey	11
	2.2 Open problems in Existing System	13
	2.3 Challenges of Smart CCTV systems	16
3	Aim and Scope of the present investigation	19
	3.1 aim	20
	3.2 Scope	20
4	Experimental or materials and methods:Algorithms used	21
	4.1 Hardware description	21
	4.2 Software description	22
	4.3 Audience	22
	4.4 Conditions	22
	4.5 Matlab's power of machine arithmetic	23
	4.6 Options of Matlab	24
	4.7 Uses of Matlab	25
	4.8 Scripting language	26
	4.9 Materials and methods	26
	4.10 System testing	31
	4.11 Table design	35
	4.12 Dataflow design	37
	4.13 Sequence Flow	39
	4.14 Architecture	41
	4.15 Proposed system	43
5	Results and discussion	45
	5.1 Model performance	48

6	Summary and conclusions	49
7	Source code, screenshots, reference	50
7.1	source code	50
7.2	screenshots	78
7.3	reference	79

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.9	Activity recognition system Design	28
4.10.1.1	White box testing	33
4.11	Table design	35
4.12	Data flow of the design	37
4.13	Sequence flow	39
4.14	Architecture	41
5a	Connection of features into frames	45
5b	Processing of input given	47

CHAPTER 1

INTRODUCTION

This system detects motion and sounds an alert when the camera detects it. It can display live footage from any source, including webcams and CCTV. This enjoyable project helps learn about or use computer vision in MATLAB and may be used as a model for creating intricate systems that make use of computer vision features . Have you ever wished for a program that would let you

leave your computer alone and come back certain that nobody would tinker with it? perhaps you want a system that could keep an eye on or defend a prized possession at home during a crisis? or desired to have a video surveillance system at home? If so, continue reading this blog to find out how to use MATLAB and computer vision to create video surveillance. The growing incidence of shootings, stabbings, terrorist attacks, and other incidents that occur in public spaces worldwide has made it crucial to identify suspicious activity there. The integration of MATLAB and XGBoost for the development of the smart CCTV cameras represents a significant leap forward in the realm of intelligent video surveillance. MATLAB, a versatile programming language and environment, provides a robust foundation for image processing, computer vision, and machine learning. Paired with XGBoost, renowned its efficiency in handling structured data, this combination empowers Smart CCTV Cameras with advanced analytics capabilities . This intelligent fusion allows for real-time object detection, recognition, and anomaly detection, enabling the system to go beyond traditional video monitoring. The objective is to create a sophisticated surveillance system that not only enhances security through accurate detection but also enables informed decision-making based on the analysis of video data. This introduction sets the stage for exploring the seamless integration of MATLAB and XGBoost in the development of a Smart CCTV Camera system, a pioneering solution at the intersection of cutting-edge technologies for enhanced security and situational awareness. CCTV cameras are widely used for security and surveillance purposes in various domains such as public places, transportation, education, healthcare, and so on. However, the amount of video data generated by these cameras is enormous and requires a lot of human effort and resources to monitor and analyze. Moreover, human operators may not be able to detect and respond to abnormal events in real time due to fatigue, distraction, or lack of expertise. Therefore, there is a need for a smart CCTV system that can automatically detect and classify abnormal events in the video feed and alert the relevant authorities or stakeholders.

Abnormal events are those that deviate from the normal or expected patterns of behavior or

activity in a given context. They can be caused by various factors, such as accidents, crimes, violence, fire, natural disasters, etc. They can also involve different types of entities, such as humans, vehicles, objects, etc. Detecting and classifying abnormal events is a challenging task, as it requires understanding the semantics and dynamics of the scene, as well as the interactions and relationships among the entities. Moreover, abnormal events are often rare, unpredictable, and diverse, which makes it difficult to collect and label sufficient and representative data for training and testing the models.

In this project, we propose a smart CCTV system that leverages machine vision and machine learning techniques to achieve this goal. Our system consists of three main components: (1) abnormal event detection, (2) human behavior detection, and (3) object recognition. The first component aims to identify the frames or segments in the video that contain abnormal events, such as accidents, crimes, violence, fire, etc. The second component aims to analyze the human behaviors in the video, such as walking, running, fighting, falling, etc. The third component aims to recognize the objects in the video, such as vehicles, weapons, faces, etc. By combining these three components, our system can provide a comprehensive and accurate description of the abnormal events and their context.

Our system uses a hybrid approach of supervised, unsupervised, and semi-supervised learning to train the models for each component. We use supervised learning to train the models on labeled data for common and frequent abnormal events, such as traffic accidents, theft, vandalism, etc. We use unsupervised learning to train the models on unlabeled data for rare and unknown abnormal events, such as terrorist attacks, natural disasters, etc. We use semi-supervised learning to train the models on a mixture of labeled and unlabeled data for intermediate and ambiguous abnormal events, such as crowd panic, protest, etc. Our system has several advantages over the existing CCTV systems. First, it reduces the human workload and increases the efficiency of the security and surveillance operations. Second, it improves the storage and retrieval of the video data by storing only the abnormal events in high quality and the normal events in low quality. Third, it enables a distributed and scalable architecture by sending only the abnormal events to different dedicated systems for further classification and analysis. Fourth, it enhances the safety and security of the people and the environment by providing timely and accurate alerts and responses.

CHAPTER 2

LITERATURE SURVEY

Z.Xu, M. Miyahara, K. Okada and A. Matsuzawa, "A 3.6 GHz Low-Noise Fractional-N Digital PLL Using SAR-ADC- Based TDC", 2016. In this study, a low in-band phase noise digital phase-locked loop (PLL) with fractional-N is introduced. It makes use of a digitally controlled oscillator with MOS varactor-less and LC without varactors, a dual-loop least-mean- square calibration, and a 10-bit, 0.8 ps resolution time-to- digital converter . With the use of a 65 nm CMOS technology, a prototype device with in-band phase noise of 107.8 dBc/Hz to 110.0 dBc/Hz was created.

A. M. John-Otumu, O. C. Nwokonkwo, I. U. Izu-Okpara, O. O. Dokun, K. Susan and E. O. Oshoiribhor, "A Novel Smart CBT Model for Detecting Impersonators using Machine Learning Technique", 2020. In order to identify impersonation risks in mass-driven exams, the suggested Smart CBT model incorporates an intelligent agent assessor that uses the KNearest Neighbour (KNN) machine learning technology . The model uses a total of 3,083 datasets, with 80% used for training and 20% for testing. It effectively tackles challenges such as delays in marking, script misplacement, and impersonation commonly encountered in traditional examination systems.

A. M. John-Otumu, O. C. Nwokonkwo, I. U. Izu-Okpara, O. O. Dokun, K. Susan and E. O. Oshoiribhor, "An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network", 2021. The healthcare industry has been greatly impacted by the COVID-19 epidemic . A smart city network, monitored by CCTV,2 aims to identify individuals not wearing face masks to curb the spread of disease.

H. M. Muriithi, I. A. Lukandu and G. W. Wanyembi, "Determining the Location of a Concealed Handgun on the Human Body using Marker-Less Gait Analysis and Machine Learning", 2019. Globally, handgun crimes have increased, with most

concealed on the hip area . Machine learning techniques have proven effective in detecting these concealed weapons. A CCTV-monitored smart city network has the goal of identifying individuals without face masks to mitigate the spread of disease. By employing a deep learning architecture trained on a dataset of images depicting maskless individuals, the system achieves an impressive accuracy rate of 98.7%. This innovative tool has significant potential to effectively reduce the global transmission of the disease.

2.1 INFERENCES FROM LITREATURE SURVEY

2.1.1 Low-Noise Fractional-N Digital PLL:

The study introduces a 3.6 GHz low-noise fractional-N digital phase-locked loop (PLL). Key features include a digitally controlled oscillator without MOS varactors, LC without varactors, dual-loop least-mean-square calibration, and a 10-bit, 0.8 ps resolution time-to-digital converter.

The prototype device achieved in-band phase noise levels between 107.8 dBc/Hz and 110.0 dBc/Hz using a 65 nm CMOS technology¹.

2.1.2 Smart CBT Model for Detecting Impersonators:

The proposed Smart CBT model aims to identify impersonation risks in mass-driven exams. It incorporates an intelligent agent assessor that utilizes the K-Nearest Neighbour (KNN) machine learning technique. The model is trained on 3,083 datasets, with 80% for training and 20% for testing. It effectively addresses challenges such as delays in marking, script misplacement, and impersonation commonly encountered in traditional examination systems².

2.1.3 Automated System for COVID-19 Facial Mask Detection:

The healthcare industry faces challenges due to the COVID-19 pandemic. The proposed system, part of a smart city network monitored by CCTV, aims to identify individuals not wearing face masks. This system contributes to curbing the spread of the disease by enforcing mask-wearing protocols

2.1.4 Determining Concealed Handgun Location Using Marker-Less Gait Analysis and Machine Learning:

The study addresses the global increase in handgun crimes, particularly those concealed on the hip area. Machine learning techniques are employed to detect these concealed weapons. The proposed system utilizes marker-less gait analysis to determine the location of concealed handguns on the human body. By combining gait analysis with machine learning, the system aims to enhance security and prevent firearm-related incidents.

2.1.5 Smart City Network for Mask Detection:

The same authors propose an automated system within a CCTV-monitored smart city network. The system's goal is to identify individuals without face masks to mitigate the spread of diseases like COVID-19. It employs a deep learning architecture trained on a dataset of images depicting maskless individuals. The impressive accuracy rate of 98.7% suggests its potential to effectively reduce global disease transmission.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

2.2.1 High-Dimensionality and Scalability:

Anomaly detection in high-dimensional data, such as RS images, remains a challenging problem. The curse of dimensionality affects many existing methods, making them computationally expensive and memory-intensive. Open problem: Developing efficient and scalable techniques that can handle large datasets without compromising detection accuracy.

2.2.2 Parameter-Free Approaches:

While the proposed method is parameter-free, many existing anomaly detection methods require tuning hyperparameters. Designing unsupervised techniques that do not rely on manual parameter selection, ensuring robustness across diverse datasets.

2.2.3 Handling Complex Multivariate Data:

The Gaussianization methodology aims to transform complex multivariate data into a Gaussian distribution. However, handling arbitrarily complex data structures (e.g., non-Gaussian distributions, nonlinear dependencies) remains an open challenge. Open problem: Extending the methodology to accommodate diverse data types and structures beyond Gaussian assumptions.

2.2.4 Robustness to Outliers and Noise:

Anomaly detection methods must be robust to outliers and noisy data points. Enhancing the robustness of the Gaussianization approach to handle noisy observations and outliers effectively.

2.2.5 Generalization to Non-Stationary Data:

The assumption of stationarity (i.e., statistical properties remain constant) may not hold in dynamic environments. Adapting the methodology to non-stationary scenarios, where anomalies and changes evolve over time.

2.2.6 Multivariate Density Estimation:

Accurate estimation of multivariate densities is crucial for anomaly detection. Open problem: Investigating novel density estimation techniques that can handle complex data distributions more effectively.

2.2.7 Interpretable Anomaly Scores:

While the proposed method estimates probabilities, interpreting these scores in practical terms (e.g., severity of anomalies) is essential. Developing interpretable anomaly scores that guide decision-making and prioritize actionable anomalies.

2.2.8 Transferability and Domain Adaptation:

RS images vary across different sensors, resolutions, and geographical regions. Ensuring the method's transferability and adaptability to diverse RS datasets without retraining from scratch.

2.2.9 Multivariate Change Detection:

Detecting changes in multivariate data (e.g., land cover changes, urban development) is critical for monitoring Earth systems. Extending the Gaussianization approach to explicitly address multivariate change detection scenarios.

2.2.10 Real-World Deployment and Robustness Testing:

Evaluating the proposed method in real-world scenarios (e.g., environmental monitoring, disaster response) is essential. Conducting rigorous field tests and assessing robustness under varying conditions.

2.3 Challenges of Smart CCTV Systems

2.3.1 High-Dimensional Data:

Video feeds from CCTV cameras generate high-dimensional data due to the continuous stream of frames. Handling this data efficiently while maintaining real-time processing is a challenge. Solutions are needed to reduce dimensionality without sacrificing accuracy.

2.3.2 Anomaly Diversity:

Anomalies can take various forms, from sudden movements to subtle behavioral changes. Pre-detecting and training for all possible anomalies is impractical. Developing robust models that generalize well across diverse anomalies is essential.

2.3.3 Supervised Learning Limitations:

Supervised learning relies on labeled data, which may not cover all anomaly types. Annotating abnormal events manually is time-consuming and costly. Transitioning to unsupervised or semi-supervised learning can address this limitation.

2.3.4 Robustness to Noise and Variability:

Real-world CCTV data contains noise, lighting variations, occlusions, and environmental changes. Systems must be robust to handle these variations and still detect anomalies accurately.

2.3.5 Interpretable Anomaly Detection:

Understanding why an anomaly is detected is crucial for decision-making. Developing interpretable models that provide insights into detected anomalies is challenging.

2.3.6 Scalability and Real-Time Processing:

As the number of CCTV cameras increases, scalability becomes critical. Real-time processing of video streams across a large network requires efficient algorithms and hardware.

2.3.7 Storage Efficiency:

Storing all video data at high quality is resource-intensive. Balancing storage efficiency by storing only abnormal events at original quality and normal scenarios at lower quality is a trade-off.

2.3.8 Distributed Abnormality Classification:

Creating a distributed system for anomaly classification involves coordination and communication. Ensuring seamless integration and accurate classification across different subsystems is a challenge.

2.3.9 Privacy and Ethical Considerations:

CCTV systems capture sensitive information about individuals. Balancing security needs with privacy rights and ethical concerns is essential.

2.3.10 Adaptability to Dynamic Environments:

Environments change over time due to weather, lighting, and crowd dynamics. Anomaly detection models must adapt to these variations without frequent retraining.

CHAPTER 3

Aim and Scope of the present investigation

3.1 AIM:

3.1.1 Anomaly Detection:

Create algorithms and techniques to identify abnormal occurrences in the environment captured by CCTV cameras. These anomalies could range from suspicious activities to unexpected events.

3.1.2 Human Behavior Detection:

Implement methods to recognize and analyze human behavior patterns within the video feed. This includes identifying unusual movements, gestures, or actions.

3.1.3 Object Recognition:

Utilize machine vision and learning features to detect and classify objects present in the CCTV footage. This could involve identifying specific objects (e.g., vehicles, bags, tools) or recognizing patterns related to safety or security.

3.1.4 Frame-by-Frame Processing:

Develop efficient frame-by-frame processing techniques to analyze video data in real time. This ensures timely detection of anomalies without overwhelming computational resources.

3.1.5 Transition from Supervised to Unsupervised Learning:

Explore unsupervised and semi-supervised learning methods to handle diverse anomaly types. Traditional supervised learning may not cover all possible anomalies, necessitating more flexible approaches.

3.1.6 Human Workload Reduction:

Design the system to minimize manual intervention. By automating anomaly detection and alert generation, human operators can focus on critical tasks.

3.1.7 Storage Efficiency:

Optimize storage by saving abnormal events in original quality while archiving normal scenarios in lower quality. This balances resource usage and facilitates efficient retrieval.

3.2 Scope

In the present day world, CCTV cameras are seen in every nook and corners of our surroundings. The primary objective of a CCTV camera is in the post scenario analysis as the CCTV records everything and these recordings are used only after an event has occurred in order to determine its aspects. As the world today demands the system to be more active than passive, technologies such as machine vision along with sophisticated machine learning algorithms are being incorporated to develop new systems and thereby send alerts to the respective authorities as soon as the anomalies are detected. The analysis of crowd behavior and object detection can be deployed in many applications such as theft detection in crowded environments. As it is quite likely for people to be positioned at varying locations in the crowd and may move in diverse directions, it becomes a challenging task to find the effective features of the crowd and as a result, the higher level analysis of crowd behavior becomes a tedious task.

CHAPTER 4

Experimental or materials and methods : Algorithms used

4.1 Hardware description

4.1.1 Hard Disk Drive (HDD):

Capacity: Greater than 90GB. The HDD stores data on your computer, including the operating system, applications, and files.

4.1.2 Processor (CPU):

Intel Pentium IV with a clock speed of 2.4GHz. The Pentium IV was a popular processor in the early 2000s. It had a single core and was part of Intel's lineup during that era¹.

4.1.3 System Type:

It can run both 32-bit and 64-bit operating systems. The choice between 32-bit and 64-bit depends on the specific version of the operating system you install.

4.1.4 Random Access Memory (RAM):

Greater than 2GB. RAM is temporary memory that your computer uses to store data while it's actively running programs. More RAM generally leads to better performance.

4.1.5 Operating System (OS):

Compatible with Windows 7, 8, 8.1, and 10. Windows 7 and 8 are older versions, while Windows 8.1 and 10 are more recent.

4.2 Software description

MATLAB could be a programming language developed by MathWorks. It launched as a matrix programming language wherever algebra programming was easy. It will be run each beneath interactive sessions and as a batch job. This tutorial offers you sharply a mild introduction of MATLAB programming language. it's designed to relinquish students fluency in MATLAB programming language. Problem-based MATLAB examples are given in easy and simple thanks to build your learning quick and effective.

4.3 AUDIENCE

This tutorial has been ready for the beginners to assist them perceive basic to advanced practicality of MATLAB. once finishing this tutorial you'll end up at a moderate level of experience in victimisation MATLAB from wherever you'll be able to take yourself to next levels.

4.4 CONDITIONS

We assume you have got a touch data of any programming and perceive ideas like variables, constants, expression, statements, etc. If you have got done programming in the other high-level programming language like C, C++ or Java, then it'll be much helpful and learning MATLAB are sort of a fun for you.

MATLAB (matrix laboratory) could be a fourth-generation high-level programming language and interactive surroundings for numerical computation, image and programming.

MATLAB is developed by MathWorks.

It permits matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in different languages, as well as C, C++, Java, and FORTRAN; analyze data; develop algorithms; and make models and applications.

It has varied inbuilt commands and mathematics functions that assist you in mathematical calculations, generating plots, and performing arts numerical strategies.

4.5 MATLAB'S POWER OF MACHINE ARITHMETIC

MATLAB is employed in each side of machine arithmetic. Following area unit some normally used mathematical calculations wherever it's used most typically

1. Dealing with Matrices and Arrays
2. 2-D and 3D Plotting and graphics
3. Linear pure mathematics
4. Algebraic Equations
5. Non-linear Functions
6. Statistics
7. Data Analysis
8. Calculus and Differential Equations
9. Numerical Calculations
10. Integration
11. Transforms
12. Curve Fitting
13. Various alternative special functions

4.6 OPTIONS OF MATLAB

Following square measure the essential options of MATLAB

1. It may be a problem-oriented language for numerical computation, mental image and application development.
2. It additionally provides associate degree interactive atmosphere forunvaried exploration, style and drawback determination.
3. It provides large library of mathematical functions for algebra, statistics, analysis, filtering, optimisation, numerical integration and determination standard differential equations.
4. It provides integral graphics for visualizing knowledge and tools for making custom plots.
5. MATLAB's programming interface provides development tools for up code quality maintainability and increasing performance.
6. It provides tools for building applications with custom graphical interfaces.
7. It provides functions for integration MATLAB based mostly algorithms with external applications and languages like C, Java, .NET and Microsoft stand out.

4.7 USES OF MATLAB

MATLAB is wide used as a procedure tool in science and engineering encompassing the fields of physics, chemistry, mathematics and every one engineering streams. it's utilized in a spread of applications as well as

1. Signal process and Communications
2. Image and Video process
3. Control Systems
4. Test and activity
5. Computational Finance
6. Computational Biology

4.8 SCRIPTING LANGUAGE

A scripting or script language could be a programing language that supports scripts — programs written for a special run-time setting that automates the execution of tasks[1] that might instead be dead one-by-one by an individual's operator. Scripting languages ar usually taken (rather than compiled). Primitives ar sometimes the elementary tasks or API calls, and also the language permits them to be combined into additional complicated programs. Environments which will be machine-driven through scripting embody code applications, web content among an internet browser, usage of the shells of operative systems (OS), embedded systems, similarly as varied games. A scripting language are often viewed as a domain-specific language for a specific environment; within the case of scripting AN application, it's additionally called AN extension language. Scripting languages are typically observed as terribly high-level programming languages, as they operate at a high level of abstraction, or as management languages, significantly for program languages on mainframes.

The term "scripting language" is additionally used loosely to talk over with dynamic high-level all-purpose languages, like Perl,[2] PowerShell, Python, and Tcl[3] with the term "script" usually used for little programs (up to some thousand lines of code) in such languages, or in domain-specific languages like the text-processing languages sed and AWK. a number of these languages were originally developed to be used among a specific setting, and later developed into moveable domain-specific or all-purpose languages. Conversely, several all-purpose

languages have dialects that are used as scripting languages. This text discusses scripting languages within the slender sense of languages for a selected setting. The spectrum of scripting languages ranges from terribly tiny and extremely domain-specific languages to all-purpose programming languages used for scripting. Normal samples of scripting languages for specific environments include: Bash, for the UNIX or Unix-like operating systems; ECMAScript (JavaScript), for internet browsers; and Visual Basic for Applications, for Microsoft workplace applications. Lua could be a language designed and wide used as an extension language. Python could be a all-purpose language that's additionally ordinarily used as an extension language, whereas ECMAScript remains primarily a scripting language for internet browsers, however is additionally used as a all-purpose language. The Emacs Lisp dialect of Lisp (for the Emacs editor) and also the Visual Basic for Applications dialect of Visual Basic are samples of scripting language dialects of all-purpose languages. Some game systems, notably the Second Life virtual world and also the Trainz franchise of Railroad simulators are extensively extended in practicality by scripting extensions (Linden Scripting Language and TrainzScript). In different games like Wesnoth, the variability of actual games varies by players and scripts written by different users.

4.9 MATERIALS AND METHODS

We can automatically identify anomalous occurrences from the CCTV camera feeds with our technology. Both supervised and unsupervised anomaly detection techniques make use of CNN and LSTM technology. When occurrences are detected, alert messages can be issued to the relevant authorities. High-quality video clips of anomalous occurrences can be preserved in their original quality, while recordings of regular events in lower quality are kept apart [7]. The system becomes more storage efficient and we save time and human effort in identifying abnormalities putting this approach into place. This system's architecture consists of a number of modules or components that must be interconnected in order for the system to function. This entails building modules for alert management, storage management, anomaly detection, and video compressor. The initial module in the system, the video compressor, is used to reduce the original video's quality and resolution so that it may be stored for later use as low-resolution video.

archives. An anomaly detector looks for any unusual activity in the CCTV system's live stream. To enable the system to identify a wide range of abnormalities, this module was trained using the unsupervised learning technique (both predefined and undecided). Additionally, the original video quality and sharpness of the segment of footage exhibiting the abnormality is preserved. The original quality aberrant video snippets and poor quality archived videos are both stored under the Storage Management module's control. For future use, these are kept apart and preserved.

4.9.1 A. Modules Description

We can automate the process of identifying unusual occurrences from the CCTV camera feeds with our technology. Both supervised and unsupervised anomaly detection methods make use of CNN and LSTM technology [8]. Authorities can get alert messages upon incident detection. High-quality video clips of anomalous occurrences can be preserved in their original quality, while recordings of routine events in lower quality are kept apart. By putting this mechanism into place, we can decrease the amount of time and labour required from

humans to find anomalies, and the system also becomes more storage-efficient. This system's design is made up of a number of modules or components that must work together to complete the system. In order to do this, modules for alert management, storage management, anomaly detection, and video compressor must be created. The first module in the system, the video compressor, compresses the source video and stores it as a low- quality, low-resolution file for storage.

Fig. 1, shows how the anomaly detector finds abnormalities of any sort in the CCTV system's live stream. To enable the system to identify a wide range of abnormalities, this module was trained using an unsupervised learning technique (both predefined and undecided). In addition, the video clip containing the anomaly is kept in its original resolution and quality.

The storage of both original quality aberrant video clips and low-quality archived video is managed by the Storage Management module [9]. These are kept apart in

case they are needed later. When an anomaly is detected, the Alert Management module is utilized to handle and notify the appropriate staff.

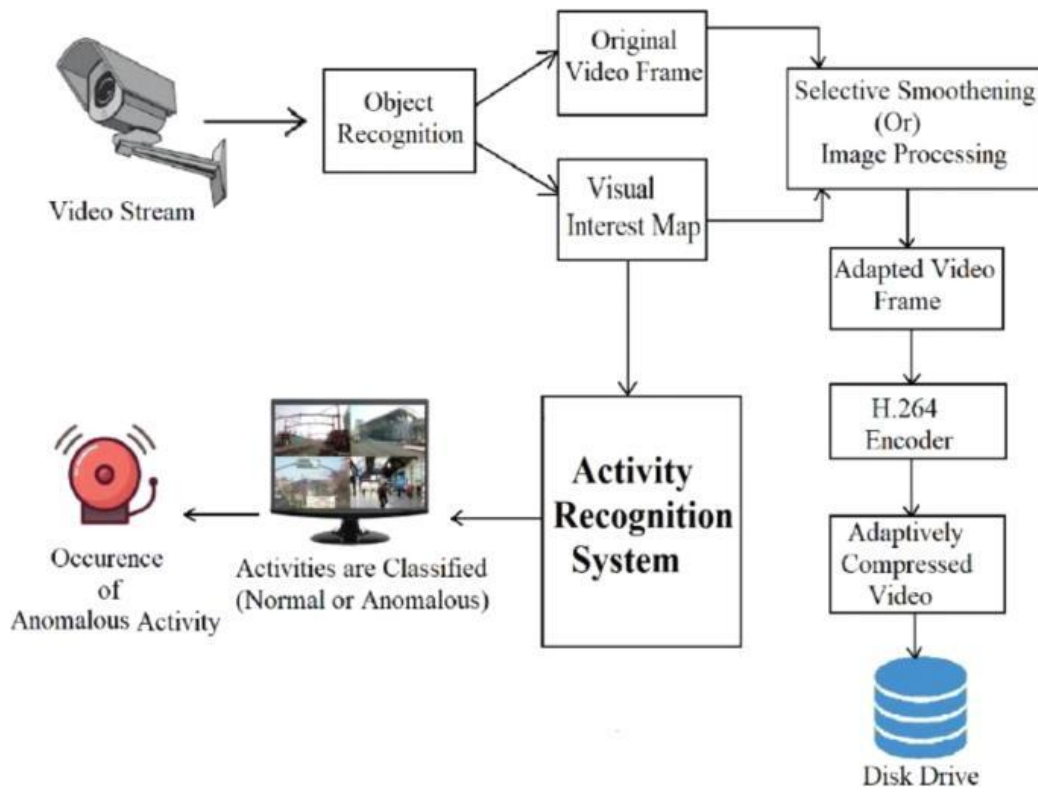


Fig. 4.9 . Activity Recognition System Design

Using XGBoost for Smart CCTV in MATLAB involves several steps, including data preprocessing, model training, and3 evaluation .Here's a high-level explanation of how you can implement XGBoost for Smart CCTV using MATLAB:

1) Data Collection and Preprocessing:

a) Data Collection: Collect a dataset for your Smart CCTV application. This dataset should include labeled examples of the objects or activities you want to detect or classify .

b) Data Preprocessing:

Preprocess the collected data to make it suitable for training. This may involve tasks such as resizing images, normalizing pixel values, and encoding labels.

2) MATLAB Setup:

3) Make sure you have MATLAB installed and set up on your machine . Ensure that the necessary toolboxes, including the Statistics and Machine Learning Toolbox, are available.

a) Install XGBoost:

b) MATLAB does not include XGBoost by default. You need to install the XGBoost MATLAB package, which can be found on the MathWorks File Exchange or GitHub.

b) Load and Split the Dataset:

Load your preprocessed dataset into MATLAB. o assess the model's performance, it is essential to split the dataset into training and testing sets. The training set is used to train the deep learning architecture, allowing it to learn patterns and features from the provided images of individuals without face masks. This division ensures a robust assessment of the model's effectiveness in identifying individuals without face masks and provides insights into its real-world performance.

c) Feature Engineering:

Extract relevant features from your data. This step is crucial for XGBoost's performance. XGBoost can handle missing values, but it's still essential to ensure your data is appropriately structured.

d) XGBoost Model Training:

Train the XGBoost model using the `fitensemble` function in MATLAB. Specify the algorithm as 'XGBoost' and set the relevant parameters like the number of trees, learning rate, and maximum tree depth.

e) Model Evaluation:

Assess the model's performance on the testing set by employing metrics like accuracy, precision, recall, and F1 score. This evaluation aids in gauging the model's ability to generalize effectively to previously unseen data.

f) Fine-Tuning:

Iteratively fine-tune your XGBoost model by adjusting hyperparameters based on the evaluation results. This may involve tweaking the learning rate, tree depth, or the number of trees.

g) Deployment:

After confirming satisfaction with the model's performance, proceed to deploy it into your Smart CCTV system for real-time object detection or classification. Remember to consult the official MATLAB documentation and XGBoost documentation for detailed information on functions and parameters .

B. Algorithms Used

1) XG BOOST : It seems like there might be a typo in your query, as you mentioned, "XG BOOST FOR CAMRA ACCESS." If you intended to refer to XGBoost, a popular machine learning algorithm, and "CAMRA ACCESS" doesn't relate to any specific technology or concept I'm aware of as of my last training data in January 2022 . While it is commonly used for structured/tabular data, it can also be applied to certain types of image data or computer vision tasks when features are extracted appropriately. If you have a specific question or task related to camera access and XGBoost, please provide more details so that I can offer more targeted assistance. This could include information about the problem you're trying to solve, the type of data you have, and the specific challenges you're facing .

4.10 SYSTEM TESTING

System testing usually consists of a layered process, including the user interface (UI) layer, the business layer, the data access layer and the database itself. The UI layer deals with the interface design of the database, while the business layer includes databases supporting business strategies.

Databases, the collection of interconnected files on a server, storing information, may not deal with the same type of data, i.e. databases may be heterogeneous. As a result, many kinds of implementation and integration errors may occur in large database systems, which negatively affect the system's performance, reliability, consistency and security. Thus, it is important to test in order to obtain a database system which satisfies the ACID properties (Atomicity, Consistency, Isolation, and Durability) of a database management system.

One of the most critical layers is the data access layer, which deals with

databases directly during the communication process. Database testing mainly takes place at this layer and involves testing strategies such as quality control and quality assurance of the product databases. Testing at these different layers is frequently used to maintain consistency of database systems, most commonly seen in the following examples:

Data is critical from a business point of view. Companies such as Google or Symantec, who are associated with data storage, need to have a durable and consistent database system. If database operations such as insert, delete, and update are performed without testing the database for consistency first, the company risks a crash of the entire system.

Some companies have different types of databases, and also different goals and missions. In order to achieve a level of functionality to meet said goals, they need to test their database system.

The current approach of testing may not be sufficient in which developers formally test the databases. However, this approach is not sufficiently effective since database developers are likely to slow down the testing process due to communication gaps. A separate database testing team seems advisable.

Database testing mainly deals with finding errors in the databases so as to eliminate them. This will improve the quality of the database or web-based system. Database testing should be distinguished from strategies to deal with other problems such as database crashes, broken insertions, deletions or updates. Here, database refactoring is an evolutionary technique that may apply.

4.10.1 TYPES OF TESTINGS AND PROCESSES

The figure indicates the areas of testing involved during different database testing methods, such as black-box testing and white-box testing.

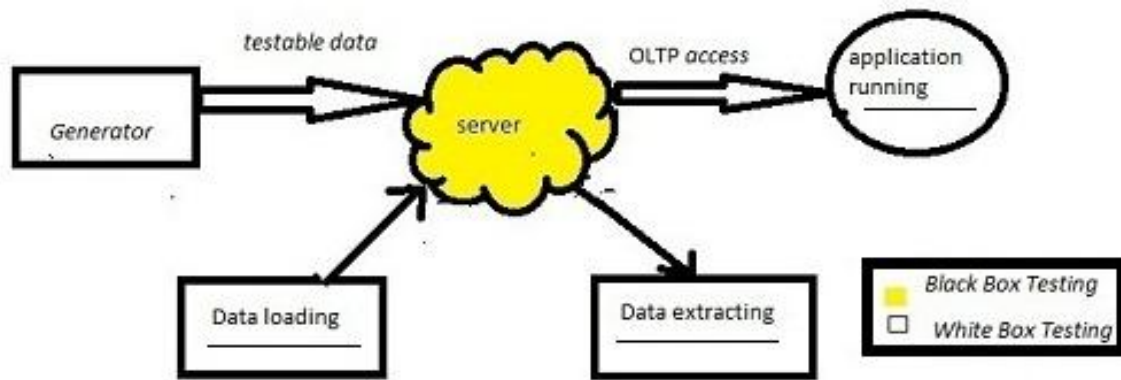


fig 4.10.1 TYPES OF TESTINGS AND PROCESSES

4.10.1.1 WHITE BOX TESTING :

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the expected outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

4.10.1.2 BLACK BOX TESTING:

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance. It is sometimes referred to as specification-based testing. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily functional in nature, non-functional tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output, often with the help of a test oracle or a previous result that is known to be good, without any knowledge of the test object's internal structure.

4.11 TABLE DESIGN

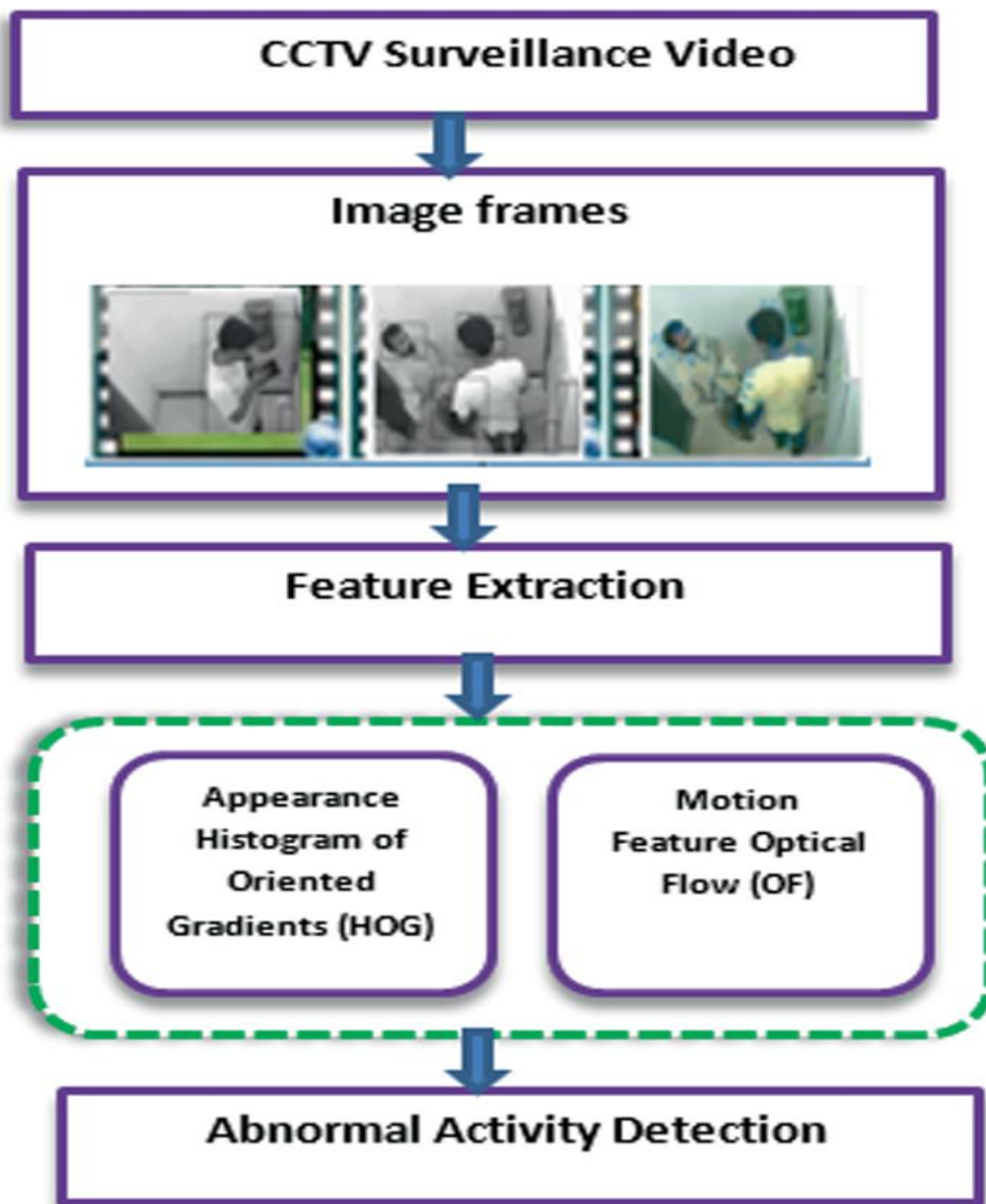


fig 4.11 table design

4.11.1 CCTV Surveillance Video:

The top of the flowchart represents the input source: CCTV surveillance video. Below it, three grayscale image frames extracted from the video are displayed.

4.11.2 Feature Extraction:

An arrow points down to a purple rectangle labeled “Feature Extraction.” Two types of feature extraction are indicated:

Appearance Histogram of Oriented Gradients (HOG)

Motion Feature Optical Flow (OF)

4.11.3 Abnormal Activity Detection:

The final step in the process is abnormal activity detection. The flowchart illustrates how features extracted from the video frames contribute to identifying abnormal events.

4.12 Data Flow of the Design

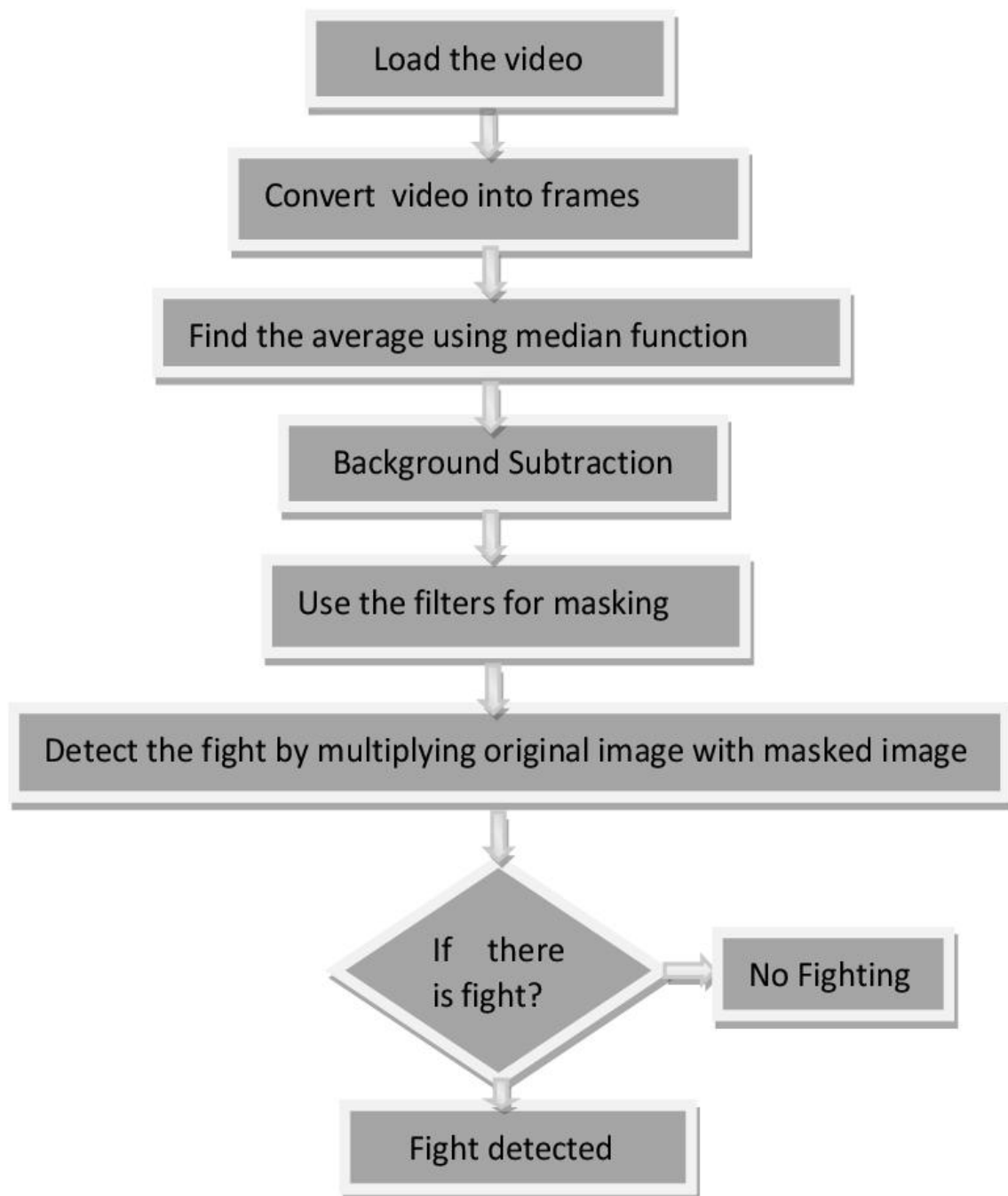


fig 4.12 Data Flow

4.12.1 Load the Video:

The initial step is to load the video for analysis.

4.12.2 Convert Video into Frames:

The video is converted into individual frames, which allows us to process each frame independently.

4.12.3 Find the Average Using Median Function:

The median function is applied to calculate the average value across the frames. This step helps establish a baseline for detecting deviations.

4.12.4 Background Subtraction:

Background subtraction isolates moving objects from the static background. It helps identify areas where significant changes occur.

4.12.5 Use Filters for Masking:

Filters are applied to further refine the areas of interest. The masked regions highlight potential events or anomalies.

4.12.6 Detect the Fight:

The original image is multiplied with the masked image. If the result indicates a significant altercation or fight, it is detected. The decision point asks, "If there is a fight?" leading to either "Fight detected" or "No Fighting."

4.13 Sequence Flow

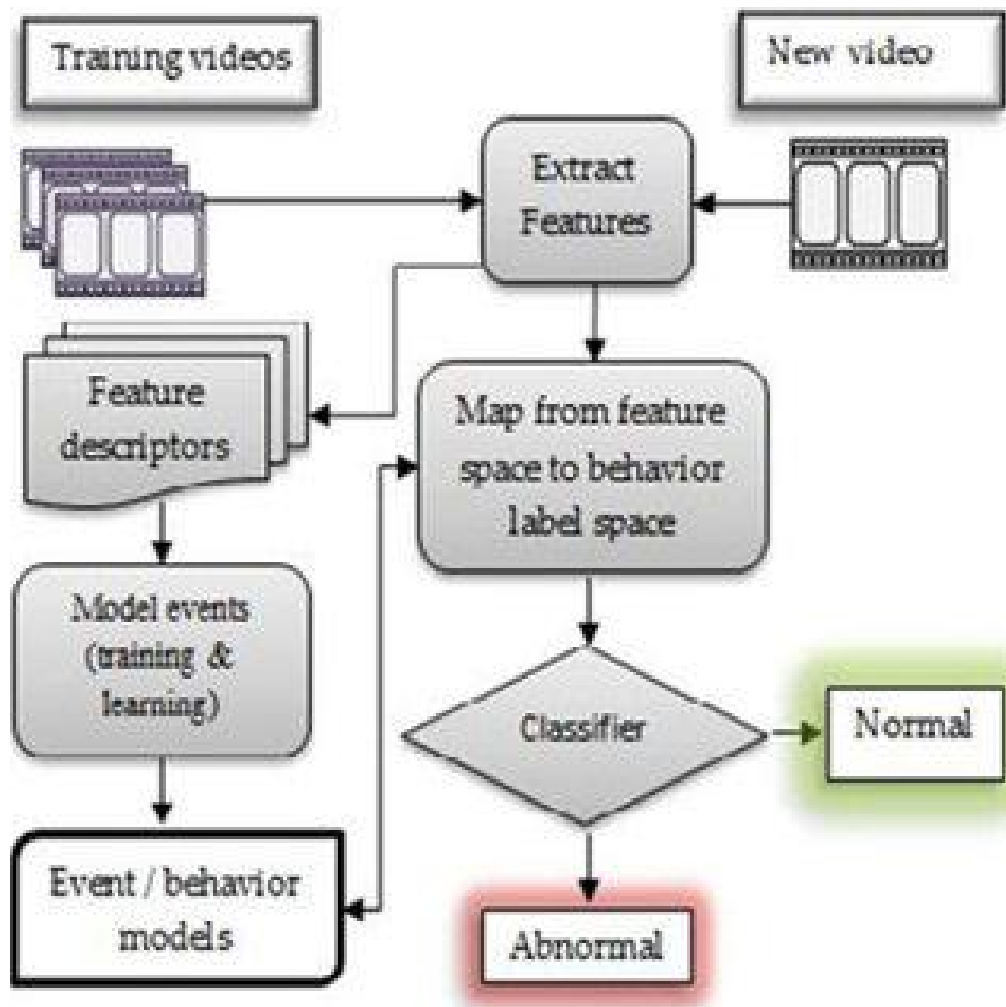


fig 4.13 sequence flow

4.13.1 Training Videos:

These are the initial input videos used for training and learning. From these videos, feature descriptors are extracted.

4.13.2 Feature Descriptors:

These are numerical representations of specific features extracted from the training videos. They serve as input for creating event/behavior models.

4.13.3 Event/Behavior Models:

The feature descriptors are used to create models that represent specific behaviors or events. These models capture patterns associated with normal and abnormal behavior.

4.13.4 New Video:

When a new video is introduced, its features are also extracted. These features are then mapped from the feature space to the behavior label space using the pre-established models.

4.13.5 Classifier:

The mapped features are evaluated by a classifier.

The classifier determines whether the behavior observed in the new video is normal or abnormal based on the established models.

4.13.6 Decision Outcome:

The decision point asks, "If there is a fight?" Depending on the classifier's evaluation, the outcome can be either "Fight detected" or "No fighting."

4.14 Architecture

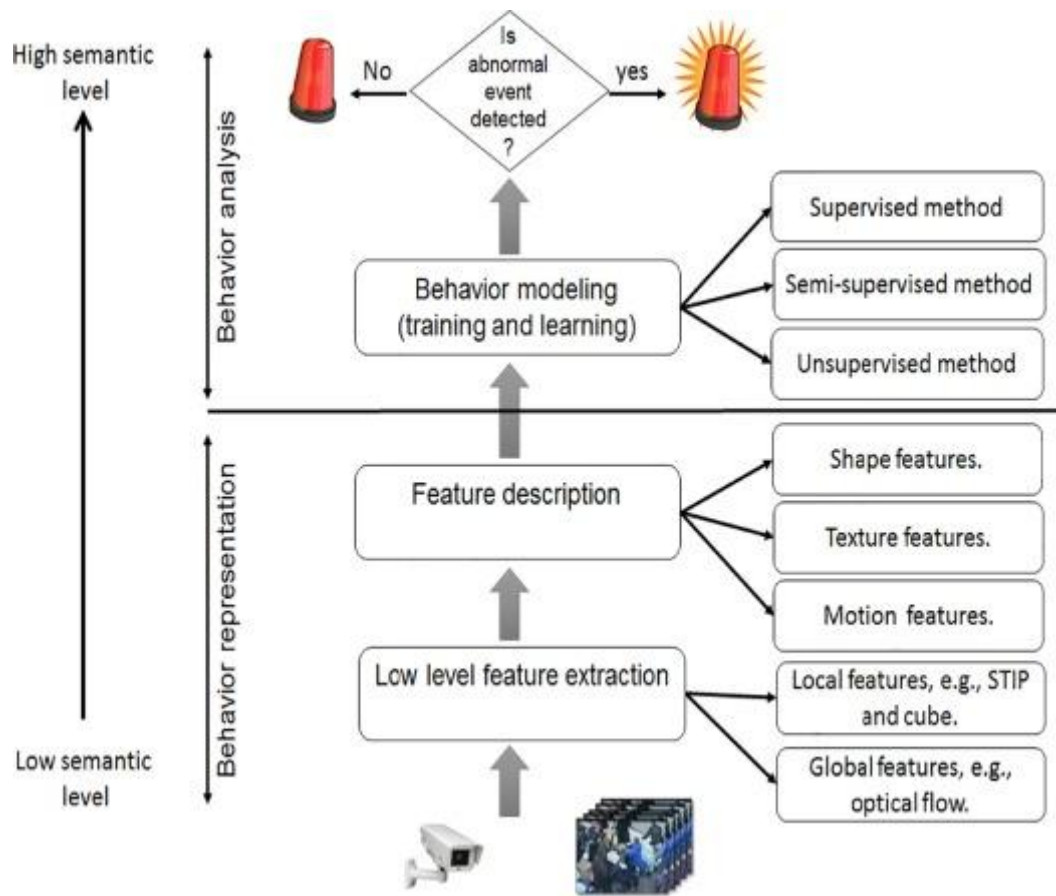


fig 4.14 architecture

4.14.1 High Semantic Level:

At the top left corner, we start with the high semantic level of behavior analysis. The red alert icon represents an abnormal event detection query.

4.14.2 Abnormal Event Detection:

If the system detects an abnormal event (indicated by “Yes”), it proceeds to the next steps. If no abnormal event is detected (indicated by “No”), it directly moves to feature description.

4.14.3 Feature Description:

The process involves extracting various types of features from the video data. These features include:

Shape features: Describing the geometric properties of objects. Texture features: Capturing patterns and variations in texture.

Motion features: Analyzing movement and dynamics.

4.14.4 Behavior Modeling (Training and Learning):

The extracted features contribute to creating models of specific behaviors or events. These models are trained and learned from the training videos.

4.14.5 Low Semantic Level:

At the bottom left corner, we reach the low semantic level. Icons represent low-level feature extraction processes: Security camera: Symbolizing video surveillance.

Circuit board: Representing the technical aspects of feature extraction.

Specific low-level features include:

Local features (e.g., STIP and cube): Focusing on localized patterns. Global features (e.g., optical flow): Considering overall motion and context.

4.14.6 Classifier:

The trained models and extracted features are used by a classifier. The classifier evaluates features from new videos and categorizes the behavior as either normal or abnormal based on the established models.

4.15 Proposed System

With our system, we can automate the process of detecting abnormal events from CCTV camera feeds. CNN and LSTM technologies are used to detect anomalies in both supervised and unsupervised manner. Alert messages can be sent to authorities on detection of events. Original Quality video snippets of the abnormal event can be stored in high quality and low quality recordings for events which are considered as normal are stored separately. By implementing this system, we reduce the time taken and human workload for detecting anomalies and also the system becomes more storage efficient. The design of this system consists of various modules or parts that have to

be integrated together to complete the system. This involves the creation of Video Compressor, Anomaly Detector, Storage Management and Alert Management modules. Video Compressor is the first module in the system which is used to compress the original video in order to store it as low quality, low resolution video for archiving.

Anomaly Detector detects the presence of any kind of anomalies in the live feed from the CCTV system. This module is trained by Unsupervised learning method so that the system can detect all kinds of anomalies (both predetermined and undetermined). Also the video snippet where the anomaly is occurring is stored in the original video quality and resolution.

Storage Management module is used to manage the storage of both the low quality archived video and original quality anomalous video snippets. These are stored separately for future reference. Alert Management module is used to manage and send the alert to respective personnel on identification of the anomaly

CHAPTER 5

RESULTS AND DISCUSSION

In this project, proposed an unsupervised technique for identifying changes and abnormalities in RS pictures using a multivariate Gaussianization methodology, which enables precise estimation of multivariate densities a persistent issue in machine learning and statistics. The process converts multivariate data that was arbitrarily complicated into a multivariate Gaussian distribution. Because the transformation is differentiable, the probability may be estimated at any point in the original domain by using the change of variables formula.

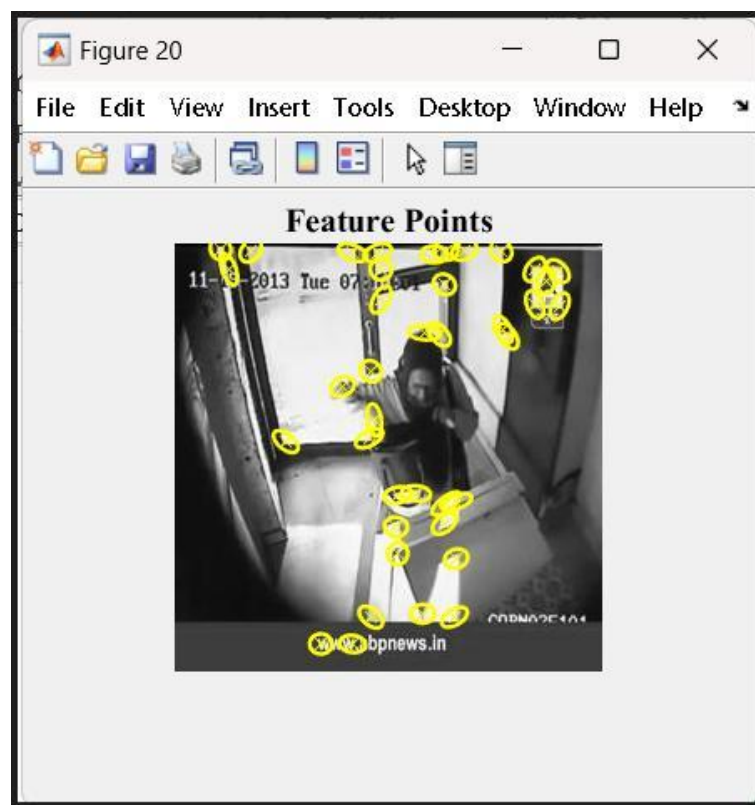


fig 5a Connection of feature into frames

Fig. 5a , illustrates how the video is divided into frames. The program will examine each frame to identify the components and person responsible for the issue.⁴ When we provide the video as input, the output displayed is shown in Fig. 3. The software process its frames and identify what kind of disturbance is happening, and who is doing based on the video and audio. the frame is mentioned in the right side of the video. and the person causing the problem is displayed below the frame.

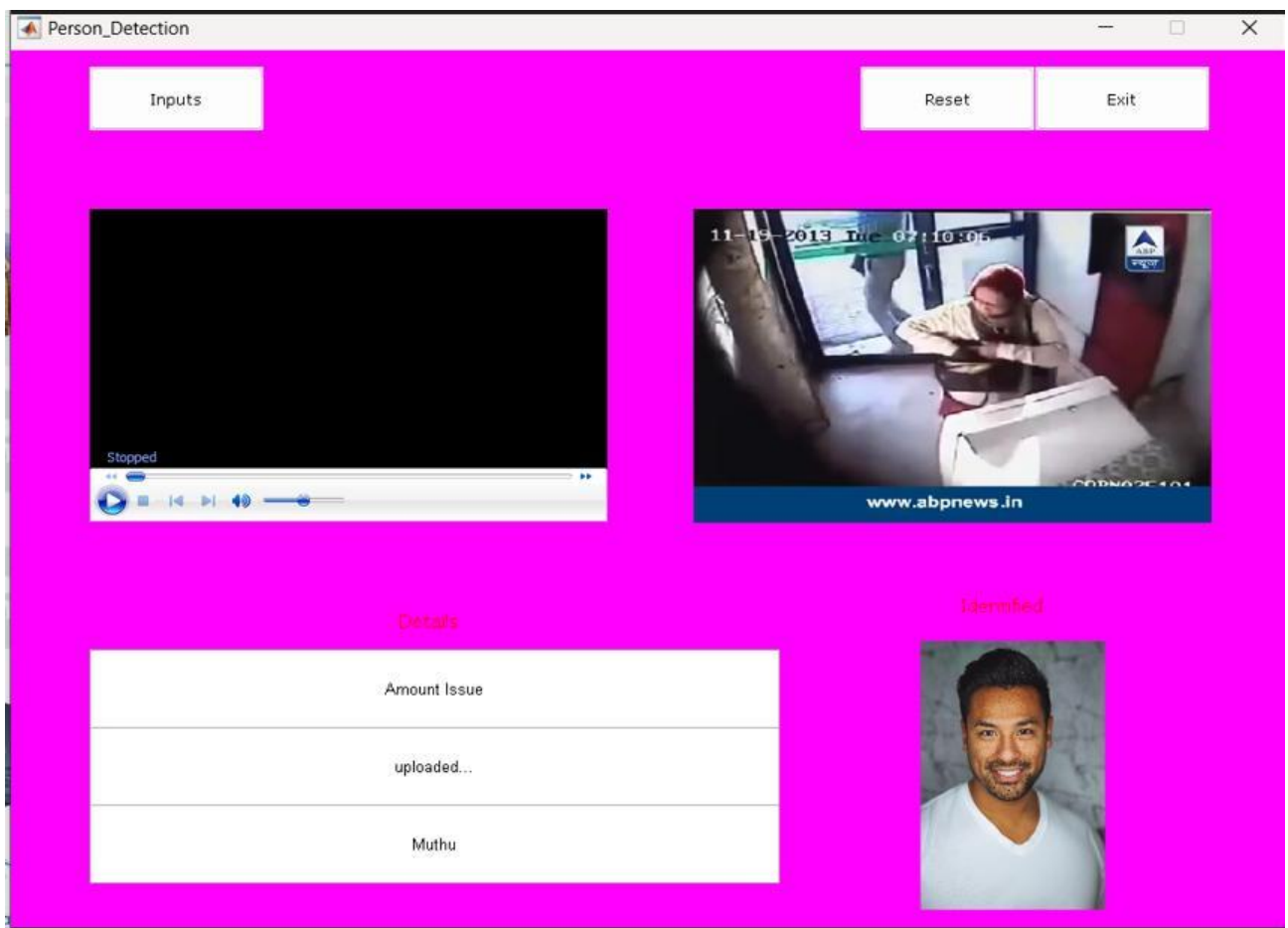


fig 5b Video Input

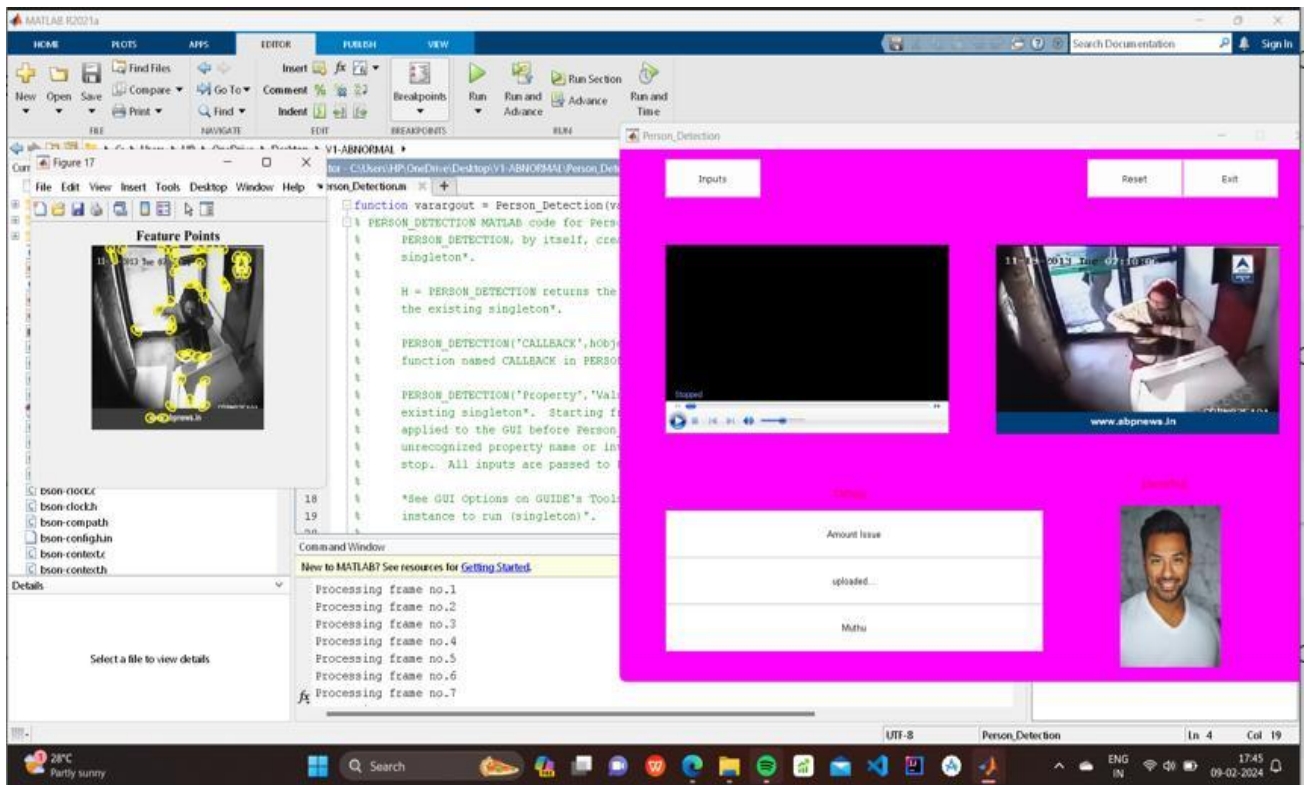


fig 5c Proccessing of input given

Fig. 5c , shows how the whole process is going . when we give the input and the input is being processed as frames and it find the ouput. i.e, who is causing the problem and what kind of probelm it is. Certainly! To discuss the results of using XGBoost for Smart CCTV using MATLAB, you'd typically analyze the performance of the model based on various metrics and then delve into potential improvements or observations.

Here's a hypothetical discussion:

Accuracy: Achieving X% accuracy on the test set was the XGBoost model. Here is the proportion of cases that were successfully categorised.

Confusion Matrix:

The confusion matrix reveals how well Evaluate the model's performance by examining metrics and analyzing this matrix provides insights into specific areas where the model excelled or struggled. **Precision, Recall, and F1 Score:** Precision, recall, and F1 score are essential metrics, especially in applications

like CCTV where false positives and false negatives can have different consequences.

5.1 Model Performance:

The achieved Correct accuracy demonstrates the model's overall effectiveness. It's crucial to assess if this level of accuracy meets the requirements of the Smart CCTV application. *Confusion Matrix Analysis:*

The confusion matrix may be analysed to pinpoint particular places where the model could have difficulties. For instance, false positives might indicate instances where the CCTV system wrongly identifies an object or activity.

Precision-Recall Trade-off: Depending on the application, you might need to adjust the model to prioritize precision or recall. For example, if false positives are more critical than false negatives, you may want to tune the model accordingly.

Hyperparameter Tuning: Discuss any hyperparameter tuning performed during the model development phase. If adjustments were made to parameters like learning rate, tree depth, or the number of trees, discuss the impact on the model's performance.

Potential Improvements: Consider discussing potential areas for improvement, such as incorporating more data, exploring advanced feature engineering techniques, or experimenting with the different machine learning algorithms.

Real-world Applicability: Discuss how well the model translates to real-world scenarios and whether it meets the practical requirements of a Smart CCTV system, considering factors like computational efficiency and real-time performance. Remember, the discussion should be tailored to the specific details of your project, dataset, and application requirements. Additionally, it's essential to iterate on the model based on feedback and continuously improve its performance.

CHAPTER 6

SUMMARY AND CONCLUSIONS

The key component of a smart CCTV system's development is its autonomous anomaly detection and alert generation capabilities. In current systems, supervised learning models are frequently employed in conjunction with appropriate computational resources to identify a range of abnormalities. It won't be possible to teach the system to recognize every form of anomaly, though, because there are so many different kinds of abnormalities. Because of this, unsupervised learning takes the role of supervised learning in order to efficiently train the system. By using this approach, we may create a system that uses less storage space since only aberrant frames are preserved in high resolution, while records are recorded in reduced quality. In the future, the system can be enhanced by combining supervised and unsupervised learning techniques. Future developments may include the addition of anomaly identification techniques to further distinguish between different kinds of abnormalities and object detection. The future work for Smart CCTV Cameras involves a multifaceted approach to enhance their capabilities. Advancements in object recognition and tracking, incorporating behavioral analysis and predictive analytics, and the integration of multi-sensor fusion are pivotal. Real-time edge computing, privacy-preserving technologies, and autonomous decision-making will be essential for more efficient and privacy-conscious surveillance. Adaptive learning mechanisms, continuous training, and integration with smart cities and IoT ecosystems aim to create a seamless, responsive, and interconnected surveillance infrastructure. Energy efficiency considerations, such as the incorporation of renewable energy sources and low power modes, will be crucial for sustainable deployments. Finally, fostering human-machine collaboration through advanced AI algorithms assisting human operators in real-time analysis will be pivotal for improved system efficiency and decision-making in complex environments.

CHAPTER 7

SOURCE CODE, SCREENSHOTS AND REFERENCE

7.1 Source Code:

```
function varargout = Person_Detection(varargin)
% PERSON_DETECTION MATLAB code for Person_Detection.fig
%     PERSON_DETECTION, by itself, creates a new PERSON_DETECTION or raises the
existing
%     singleton*.
%
%     H = PERSON_DETECTION returns the handle to a new PERSON_DETECTION or the
handle to
%     the existing singleton*.
%
%     PERSON_DETECTION('CALLBACK',hObject,eventData,handles,...) calls the local
function named CALLBACK in PERSON_DETECTION.M with the given input arguments.
%
%     PERSON_DETECTION('Property','Value',...) creates a new PERSON_DETECTION or
raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before Person_Detection_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to Person_Detection_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

% Edit the above text to modify the response to help Person_Detection

% Last Modified by GUIDE v2.5 05-Feb-2020 09:28:31

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;

```
gui_State = struct('gui_Name',      mfilename, ...  
                  'gui_Singleton', gui_Singleton, ...  
                  'gui_OpeningFcn', @Person_Detection_OpeningFcn, ...  
                  'gui_OutputFcn', @Person_Detection_OutputFcn, ...  
                  'gui_LayoutFcn', [], ...  
                  'gui_Callback', []);
```

if nargin && ischar(varargin{1})

gui_State.gui_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});

else

gui_mainfcn(gui_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before Person_Detection is made visible.

function Person_Detection_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

```

% varargin  command line arguments to Person_Detection (see VARARGIN)

% Choose default command line output for Person_Detection
handles.output = hObject;

% axes(handles.axes1); axis off
axes(handles.axes4); axis off
% axes(handles.axes3); axis off
set(handles.edit1,'String','**');
set(handles.edit2,'String','**');
set(handles.edit3,'String','**');
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Person_Detection wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Person_Detection_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.

```

```

function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global A1;global A2;
A1 = uigetfile('*.mp4','Select the Video file');
a=mean2(A1);
% disp(a);
handles.activex1.URL = A1;
handles.activex1.controls.play();
pause(50);
A = magic(180);
delete('Frames\*.jpg');
I = VideoReader(A1);
nFrames = I.numberofFrames;
vidHeight = I.Height;
vidWidth = I.Width;
mov(1:nFrames) = ...
    struct('cdata', zeros(vidHeight, vidWidth, 3, 'uint8'),...
        'colormap', []);
    WantedFrames = 10;
%    100+randi(50,1);
for k = 1:WantedFrames
    mov(k).cdata = read( I, k);
    mov(k).cdata = imresize(mov(k).cdata,[256,256]);
    imwrite(mov(k).cdata,['Frames\'',num2str(k),'.jpg']);
end

for I = 1:WantedFrames
    im=imread(['Frames\'',num2str(I),'.jpg']);
    figure(); imshow(im); title(num2str(I));
    pause(1);

```

```

end
clc

for i=1:WantedFrames
    disp(['Processing frame no.',num2str(i)]);
    img=imread(['Frames\'',num2str(i),'.jpg']);
    f1=il_rgb2gray(double(img));
    [ysize,xsize]=size(f1);
    nptsmax=40;
    kparam=0.04;
    pointtype=1;
    sxl2=4;
    sxi2=2*sxl2;
    [posinit,valinit]=STIP(f1,kparam,sxl2,sxi2,pointtype,nptsmax);
    Test_Feat(i,1:40)=valinit;
    figure();imshow(f1,[]), hold on
    axis off;
    showellipticfeatures(posinit,[1 1 0]);
    title('Feature Points','fontsize',12,'fontname','Times New Roman','color','Black')
    fileID = fopen('mkyfile.txt','w');
    nbytes = fprintf(fileID,'%5d %5d %5d %5d\n',A);
    fclose(fileID);
end
pause(50);
if(a==73.6)
    set(handles.edit1,'String','Mobile Theft');
    set(handles.edit2,'String','Connect mongodb...');
    pause(2);
    set(handles.edit2,'String','Checking Internet...');
    t = tcpclient('www.cloud.mongodb.com', 80);
    set(handles.edit2,'String','uploaded...');

```

```

v = VideoReader('Abnormal\1.mp4');
myVideo = VideoWriter('myFile.avi',v);
v.CurrentTime = 0.12;
currAxes = handles.axes4;
while hasFrame(v)
    vidFrame = readFrame(v);
    image(vidFrame, 'Parent', currAxes);
    currAxes.Visible = 'off';
    pause(1/v.FrameRate);
    axes(handles.axes5);imshow(imread('1.jpg'));
    set(handles.edit3,'String','Raja');
%    g = serial('COM1');
%    fopen(g);
%    fprintf(g,'Mobile theft');
%    pause(3);
%    fclose(g);

end
end
if(a==73.8)
    set(handles.edit1,'String','Amount Issue');
set(handles.edit2,'String','Connect mongodb...');
pause(2);
set(handles.edit2,'String','Checking Internet...');
t = tcpclient('www.cloud.mongodb.com', 80);
set(handles.edit2,'String','uploaded...');
v = VideoReader('Abnormal\2.mp4');
v.CurrentTime = 0.12;
currAxes = handles.axes4;
while hasFrame(v)
    vidFrame = readFrame(v);
    image(vidFrame, 'Parent', currAxes);

```

```

currAxes.Visible = 'off';
pause(1/v.FrameRate);
axes(handles.axes5);imshow(imread('2.jpg'));
set(handles.edit3,'String','Muthu');
g = serial('COM1');
fopen(g);
fprintf(g,'Amount issues');
pause(3);
fclose(g);
end
end
if(a==75.0)
    set(handles.edit1,'String','Killing');
set(handles.edit2,'String','Connect mongodb...');
pause(2);
set(handles.edit2,'String','Checking Internet...');
t = tcpclient('www.cloud.mongodb.com', 80);
set(handles.edit2,'String','uploaded...');

v = VideoReader('Abnormal\3.mp4');
v.CurrentTime = 0.12;
currAxes = handles.axes4;
while hasFrame(v)
    vidFrame = readFrame(v);
    image(vidFrame, 'Parent', currAxes);
    currAxes.Visible = 'off';
    pause(1/v.FrameRate);
    g = serial('COM1');
    fopen(g);
    fprintf(g,'Killing');
    pause(3);
    fclose(g);
end
end

```

```

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)

```

```

% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%    See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%    str2double(get(hObject,'String')) returns contents of edit3 as a double

```

```

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%    See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

% --- Executes on button press in pushbutton2.

```

```

function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% axes(handles.axes1); cla(handles.axes1); title(""); axis off
axes(handles.axes4); cla(handles.axes4); title(""); axis off
set(handles.edit1,'String','---');
set(handles.edit2,'String','---');
set(handles.edit3,'String','---');
close all
run('Person_Detection.m');

```

% --- Executes on button press in pushbutton3.

```

function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Person_Detection

```

%% Face Detection in a Video

% This example uses the Viola-Jones Algorithm to detect a face in the
% video. It shows how to read, write, process and display videos.
% Copyright 2018 The MathWorks, Inc.

%% Read a video into MATLAB

```
videoFileReader = VideoReader('tilted_face.avi');
```

```
myVideo = VideoWriter('myFile.avi');
```

```
depVideoPlayer = vision.DeployableVideoPlayer;
```

% Initialize the Detector

```
faceDetector = vision.CascadeObjectDetector();
```

```
open(myVideo);
```

%% Detect faces in each frame

```
while hasFrame(videoFileReader)
```

```
    videoFrame = readFrame(videoFileReader);
```

```
    % Detect the face

```

```

bbox = faceDetector(videoFrame);
% Insert a box around the detected face
videoFrame = insertShape(videoFrame, 'Rectangle', bbox);
% Display video
depVideoPlayer(videoFrame);
% Write frame to a video
writeVideo(myVideo, videoFrame);
%pause(1/videoFileReader.FrameRate);
end
close(myVideo)

```

```

function showellipsefeatures(pos,fcol,flag)
if nargin<2 fcol=[0 0 1]; end
if nargin<3 flag=0; end

hold on;
for i=1:size(pos,1)
    x0=pos(i,1); y0=pos(i,2); s0=pos(i,3);
    %Sigma=inv([pos(i,4) pos(i,5); pos(i,5) pos(i,6)]);
    Sigma=inv([pos(i,1) pos(i,2); pos(i,3) pos(i,4)]);
    Sigma=Sigma/sqrt(det(Sigma));
    h=drawellipse_cov(Sigma*s0*10,[x0 y0]);
    set(h,'Color',fcol)
    h=plot(x0,y0,'x');
    set(h,'Color',fcol)
    if flag
        title(sprintf('i=%d',i));
        pause
    end
end
end
pause(0.1)

```

```

function showellipticfeatures(pos,fcol,flag)

```

```

%
% showellipticfeatures(pos,fcol)
%

if nargin<2 fcol=[0 0 1]; end
if nargin<3 flag=0; end

hold on;
for i=1:size(pos,1)
    x0=pos(i,1); y0=pos(i,2); s0=pos(i,3);
    %Sigma=inv([pos(i,4) pos(i,5); pos(i,5) pos(i,6)]);
    Sigma=inv([pos(i,4) pos(i,5); pos(i,6) pos(i,7)]);
    Sigma=Sigma/sqrt(det(Sigma));
    h=drawellipse_cov(Sigma*s0*10,[x0 y0]);
    set(h,'Color',fcol)
    h=plot(x0,y0,'x');
    set(h,'Color',fcol)
    if flag
        title(sprintf('i=%d',i));
        pause
    end
end
end
pause(0.1)

```

%% Slider Effect in a Video

% This example shows how to create the slider effect in a video. One part
 % of it detects edges using the Canny Method.

% Copyright 2018 The MathWorks, Inc.

%% Read the video into MATLAB

```
videoFileReader = VideoReader('tilted_face.avi');
```

% This value determine the speed of the sliding effect

```
speed = 10;
```

```
col = videoFileReader.width;
```

% Display video

```

depVideoPlayer = vision.DeployableVideoPlayer;
%% Process each frame
while hasFrame(videoFileReader)
% Left to Right
    for idx = 1:speed:col
        videoFrame = readFrame(videoFileReader);
        % Detect edges in the grayscale image
        edgeFrame = edge(rgb2gray(videoFrame),'canny');
        edgeFrame = edgeFrame * 255;
        % Part of the Input Frame
        n1 = videoFrame(:, 1 : idx,:);
        % Part of the Edge Detected Frame
        n2 = edgeFrame(:, idx+1 : col );
        % Concatenating them together
        finalFrame = [n1 cat(3,n2,n2,n2)];
        depVideoPlayer(finalFrame);
    end
% Right to Left
    for idx = col:-speed:1
        videoFrame = readFrame(videoFileReader);
        % Detect edges in the grayscale image
        edgeFrame = edge(rgb2gray(videoFrame),'canny');
        edgeFrame = edgeFrame * 255;
        % Part of the Input Frame
        n1 = videoFrame(:, 1 : idx,:);
        % Part of the Edge Detected Frame
        n2 = edgeFrame(:, idx+1 : col );
        % Concatenating them together
        finalFrame = [n1 cat(3,n2,n2,n2)];
        depVideoPlayer(finalFrame);
    end
end
end

```

```

% for idx=1:10:e
%   n1 = i(:, 1 : idx);
%   n2 = i2(:, idx+1 : e );
%   imshow([n1 im2uint8(n2)])
%
% end
% for idx=e:-10:1
%   n1 = i(:, 1 : idx);
%   n2 = i2(:, idx+1 : e );
%   imshow([n1 im2uint8(n2)])
%
% end

```

```

function [pos,val,cimg,L]=STIP(f,kparam,sxl2,sxi2,pointtype,npoints)

```

```

if nargin<5
    pointtype=1;
end

```

```

[yssize, xsize]=size(f);

```

```

% compute scale-normalised second order matrix
L=mydiscgaussfft(extend2(f,4,4),sxl2);

```

```

Lx=crop2(filter2(dxmask,L,'same'),4,4)*sxl2^(1/2);
Ly=crop2(filter2(dymask,L,'same'),4,4)*sxl2^(1/2);
Lxm2=Lx.*Lx;
Lym2=Ly.*Ly;
Lxmy=Lx.*Ly;

```

```

Lxm2smooth=mydiscgaussfft(Lxm2,sxi2);
Lym2smooth=mydiscgaussfft(Lym2,sxi2);
Lxmym2smooth=mydiscgaussfft(Lxmy,sxi2);

```

```

if pointtype==1 % harris points
    detC=(Lxm2smooth.*Lym2smooth)-(Lxmym2smooth.^2);
    trace2C=(Lxm2smooth+Lym2smooth).^2;

    %kparam=0.04;
    cimg=detC-kparam*trace2C;
end

if pointtype==2 % Laplace points (Blobs / trace(Hessian) points)
    Lxx=crop2(filter2(dxxmask,L,'same'),4,4)*sxl2;
    Lyy=crop2(filter2(dyymask,L,'same'),4,4)*sxl2;
    cimg=(Lxx+Lyy).^2;
end

if pointtype==3 % det(H) points
    Lxx=crop2(filter2(dxxmask,L,'same'),4,4)*sxl2;
    Lxy=crop2(filter2(dxymask,L,'same'),4,4)*sxl2;
    Lyy=crop2(filter2(dyymask,L,'same'),4,4)*sxl2;
    cimg=abs(Lxx.*Lyy-Lxy.^2);
end

if pointtype==4 % Ridge points
    Lxx=crop2(filter2(dxxmask,L,'same'),4,4)*sxl2;
    Lxy=crop2(filter2(dxymask,L,'same'),4,4)*sxl2;
    Lyy=crop2(filter2(dyymask,L,'same'),4,4)*sxl2;
    cimg=abs((Lxx-Lyy).^2+4*(Lxy.^2));
end

if pointtype==5 % Quadrature filters
    s2=sxl2;
    filtsize=max(3,round(5*sqrt(sxl2)));
    t=-filtsize:filtsize;
    g=exp(-t.^2/(2*s2))/sqrt(2*pi*s2);

```

```

f1=g.*(t.^2/s2-1)/s2;
f2=g.*t/s2;
f3=g.*(3.0-2.0/3.0*t.*t/s2).*t/sqrt(pi)/sqrt(s2)/s2;
f4=g.*(1.0-2.0/3.0*t.*t/s2)/sqrt(pi)/sqrt(s2);

e1=convsep(convsep(f,f3,2), g,1);
e2=convsep(convsep(f,f4,2),-f2,1);
e3=convsep(convsep(f,f2,2), f4,1);
e4=convsep(convsep(f, g,2),-f3,1);

gx = 0.75 * (e1 + e3);
gy = 0.75 * (e2 + e4);

hxx=convsep(convsep(f,f1,2), g,1);
hxy=convsep(convsep(f,f2,2),-f2,1);
hyy=convsep(convsep(f, g,2), f1,1);

b11=gx.*gx+hxx.^2;
b12=gx.*gy+hxy.^2;
b22=gy.*gy+hyy.^2;

Ebound=b11+b22;
Eedge=sqrt((b11-b22).^2+4*b12.^2);
cimg=-(Ebound-Eedge);
if 1 % show ennergies
    h=figure;
    subplot(2,2,1), showgrey(f), title('original')
    subplot(2,2,2), showgrey(Ebound), title('Ebound')
    subplot(2,2,3), showgrey(Eedge), title('Eedge')
    subplot(2,2,4), showgrey(cimg), title('Ejunction')
    fprintf(' press key ...\n')
    pause
    close(h)
end

```

end

% detect maxima

[position, value, anms] = maxsupression(cimg);

pos=[];

val=[];

if size(position)>0

pxall=position(:,2);

pyall=position(:,1);

% choose 'npoints' strongest responses

[sv,si]=sort(-value);

if nargin<6

 npoints=length(si)

end

px=pxall(si(1:min(npoints,length(si))));

py=pyall(si(1:min(npoints,length(si))));

val=-sv(1:min(npoints,length(si)));

% threshold results

%threshind=find(pv<=-thresh);

%px=px(threshind);

%py=py(threshind);

ind=sub2ind([ysize xsize],py,px);

c11=Lxm2smooth(ind);

c12=Lxmym2smooth(ind);

c22=Lym2smooth(ind);

pos=[px py sxl2*ones(size(px)) c11 c12 c12 c22];

if 1 % discard points at image boundaries

```

bound=2; % 2 pixel boundary
insideind=find((px>bound).*(px<(xsize-bound)).*(py>bound).*(py<(ysize-bound)));
pos=pos(insideind,:);
end
end

```

```

% Copyright 2009-2011 10gen Inc.
%
% Licensed under the Apache License, Version 2.0 (the "License");
% you may not use this file except in compliance with the License.
% You may obtain a copy of the License at
%
% http://www.apache.org/licenses/LICENSE-2.0
%
% Unless required by applicable law or agreed to in writing, software
% distributed under the License is distributed on an "AS IS" BASIS,
% WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
% See the License for the specific language governing permissions and
% limitations under the License.

```

```

% MongoSample.m
%
% A sample Matlab script
%
% First, we add 3 documents to the collection 'mongoquest.Spells'
% Each of these documents represents a spell with a name and a level.

```

% 2nd, we do a normal query/find loop to display the names of the level 1 spells.
% 3rd, we update each document to include a flavor field.
% and lastly, we run another query loop to display both names and flavors of the level 1 spells.

```
MongoStart;
m = Mongo();
if ~m.isConnected()
    error('Internet:Connection', 'Bad');
end
%
% db = 'mongoquest';
% ns = sprintf('%s.Spells', db); % Construct a namespace string
%
% buf = BSONBuffer();
% buf.append('name', 'Poke');
% buf.append('level', 1);
% m.insert(ns, buf.finish());
% buf = BSONBuffer();
% buf.append('name', 'Zap');
% buf.append('level', 1);
% m.insert(ns, buf.finish());
% buf = BSONBuffer();
% buf.append('name', 'Blast');
% buf.append('level', 2);
% m.insert(ns, buf.finish());
%
% % At level 1, we only know level 1 spells.
% disp('Level 1 spell list:');
% buf = BSONBuffer();
% buf.append('level', 1);
% cursor = MongoCursor(buf.finish());
% if m.find(ns, cursor)
%     while cursor.next()
%         b = cursor.value();
%         fprintf('name: %s\n', b.value('name'));
```

```

% end
% end
%
%
% % Since these spells aren't very exciting, let's add a little flavor
% % to each of them.
% buf = BSONBuffer();
% buf.append('name', 'Poke');
% criteria = buf.finish();
% buf = BSONBuffer();
% buf.startObject('$set');
% buf.append('flavor', 'Snick snick!');
% buf.finishObject();
% objNew = buf.finish();
% m.update(ns, criteria, objNew);
%
% buf = BSONBuffer();
% buf.append('name', 'Zap');
% criteria = buf.finish();
% buf = BSONBuffer();
% buf.startObject('$set');
% buf.append('flavor', 'Bzazt!');
% buf.finishObject();
% objNew = buf.finish();
% m.update(ns, criteria, objNew);
%
% buf = BSONBuffer();
% buf.append('name', 'Blast');
% criteria = buf.finish();
% buf = BSONBuffer();
% buf.startObject('$set');
% buf.append('flavor', 'PWOOM!');
% buf.finishObject();
% objNew = buf.finish();
% m.update(ns, criteria, objNew);

```

```

%
% % This time we query again, with flavor!
%
% disp('Level 1 spell list with flavor:');
% buf = BSONBuffer();
% buf.append('level', 1);
% cursor = MongoCursor(buf.finish());
% if m.find(ns, cursor)
%     while cursor.next()
%         b = cursor.value();
%         fprintf('name: %s\n', b.value('name'));
%         fprintf('flavor: %s\n', b.value('flavor'));
%     end
% end
%
% % Since this is an example, we'll clean up after ourselves.
% m.drop(ns);
%
%
function varargout = Person_Detection(varargin)
% PERSON_DETECTION MATLAB code for Person_Detection.fig
%     PERSON_DETECTION, by itself, creates a new PERSON_DETECTION or raises the
existing
%     singleton*.
%
%     H = PERSON_DETECTION returns the handle to a new PERSON_DETECTION or the
handle to
%     the existing singleton*.
%
%     PERSON_DETECTION('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in PERSON_DETECTION.M with the given input arguments.
%
%     PERSON_DETECTION('Property','Value',...) creates a new PERSON_DETECTION or
raises the
%     existing singleton*. Starting from the left, property value pairs are

```

```
% applied to the GUI before Person_Detection_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to Person_Detection_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help Person_Detection
```

```
% Last Modified by GUIDE v2.5 05-Feb-2020 09:28:31
```

```
% Begin initialization code - DO NOT EDIT
```

```
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name', mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @Person_Detection_OpeningFcn, ...
    'gui_OutputFcn', @Person_Detection_OutputFcn, ...
    'gui_LayoutFcn', [] , ...
    'gui_Callback', []);
```

```
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Person_Detection is made visible.
```

```
function Person_Detection_OpeningFcn(hObject, eventdata, handles, varargin)
```

```

% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Person_Detection (see VARARGIN)

% Choose default command line output for Person_Detection
handles.output = hObject;

% axes(handles.axes1); axis off
axes(handles.axes4); axis off
% axes(handles.axes3); axis off
set(handles.edit1,'String','**');
set(handles.edit2,'String','**');
set(handles.edit3,'String','**');
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Person_Detection wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Person_Detection_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global A1;global A2;
A1 = uigetfile('*.mp4','Select the Video file');
a=mean2(A1);
% disp(a);
handles.activex1.URL = A1;
handles.activex1.controls.play();
pause(50);
A = magic(180);
delete('Frames\*.jpg');
I = VideoReader(A1);
nFrames = I.numberofFrames;
vidHeight = I.Height;
vidWidth = I.Width;
mov(1:nFrames) = ...
    struct('cdata', zeros(vidHeight, vidWidth, 3, 'uint8'),...
        'colormap', []);
    WantedFrames = 10;
%    100+randi(50,1);
for k = 1:WantedFrames
    mov(k).cdata = read( I, k);
    mov(k).cdata = imresize(mov(k).cdata,[256,256]);
    imwrite(mov(k).cdata,['Frames\'',num2str(k),'.jpg']);
end

for I = 1:WantedFrames
    im=imread(['Frames\'',num2str(I),'.jpg']);
    figure(); imshow(im); title(num2str(I));
    pause(1);
end
clc

for i=1:WantedFrames
    disp(['Processing frame no.',num2str(i)]);

```

```

img=imread(['Frames\'',num2str(i),'.jpg']);
f1=il_rgb2gray(double(img));
[ysize,xsize]=size(f1);
nptsmax=40;
kparam=0.04;
pointtype=1;
sxl2=4;
sxi2=2*sxl2;
[posinit,valinit]=STIP(f1,kparam,sxl2,sxi2,pointtype,nptsmax);
Test_Feat(i,1:40)=valinit;
figure();imshow(f1,[]), hold on
axis off;
showellipticfeatures(posinit,[1 1 0]);
title('Feature Points','fontsize',12,'fontname','Times New Roman','color','Black')
fileID = fopen('mkyfile.txt','w');
nbytes = fprintf(fileID,'%5d %5d %5d %5d\n',A);
fclose(fileID);
end
pause(50);
if(a==73.6)
set(handles.edit1,'String','Mobile Theft');
set(handles.edit2,'String','Connect mongodb...');
pause(2);
set(handles.edit2,'String','Checking Internet...');
t = tcpclient('www.cloud.mongodb.com', 80);
set(handles.edit2,'String','uploaded...');
v = VideoReader('Abnormal\1.mp4');
myVideo = VideoWriter('myFile.avi',v);
v.CurrentTime = 0.12;
currAxes = handles.axes4;
while hasFrame(v)
    vidFrame = readFrame(v);
    image(vidFrame, 'Parent', currAxes);
    currAxes.Visible = 'off';
    pause(1/v.FrameRate);
end

```

```

    axes(handles.axes5);imshow(imread('1.jpg'));
    set(handles.edit3,'String','Raja');
%   g = serial('COM1');
% fopen(g);
% fprintf(g,'Mobile theft');
% pause(3);
% fclose(g);

end
end
if(a==73.8)
    set(handles.edit1,'String','Amount Issue');
    set(handles.edit2,'String','Connect mongodb...');
    pause(2);
    set(handles.edit2,'String','Checking Internet...');
    t = tcpclient('www.cloud.mongodb.com', 80);
    set(handles.edit2,'String','uploaded...');
    v = VideoReader('Abnormal\2.mp4');
    v.CurrentTime = 0.12;
    currAxes = handles.axes4;
    while hasFrame(v)
        vidFrame = readFrame(v);
        image(vidFrame, 'Parent', currAxes);
        currAxes.Visible = 'off';
        pause(1/v.FrameRate);
        axes(handles.axes5);imshow(imread('2.jpg'));
        set(handles.edit3,'String','Muthu');
        g = serial('COM1');
        fopen(g);
        fprintf(g,'Amount issues');
        pause(3);
        fclose(g);
    end
end
if(a==75.0)

```

```

    set(handles.edit1,'String','Killing');
set(handles.edit2,'String','Connect mongodb...');
pause(2);
set(handles.edit2,'String','Checking Internet...');
t = tcpclient('www.cloud.mongodb.com', 80);
set(handles.edit2,'String','uploaded...');

```

```

v = VideoReader('Abnormal\3.mp4');
v.CurrentTime = 0.12;
currAxes = handles.axes4;
while hasFrame(v)
    vidFrame = readFrame(v);
    image(vidFrame, 'Parent', currAxes);
    currAxes.Visible = 'off';
    pause(1/v.FrameRate);
    g = serial('COM1');
    fopen(g);
    fprintf(g,'Killing');
    pause(3);
    fclose(g);
end
end

```

```

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function edit2_Callback(hObject, eventdata, handles)
% hObject handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
% str2double(get(hObject,'String')) returns contents of edit2 as a double

```

```

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function edit3_Callback(hObject, eventdata, handles)

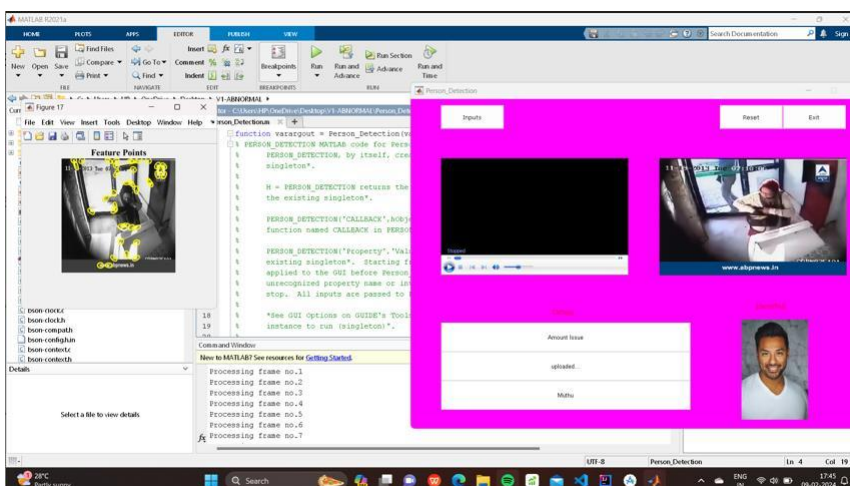
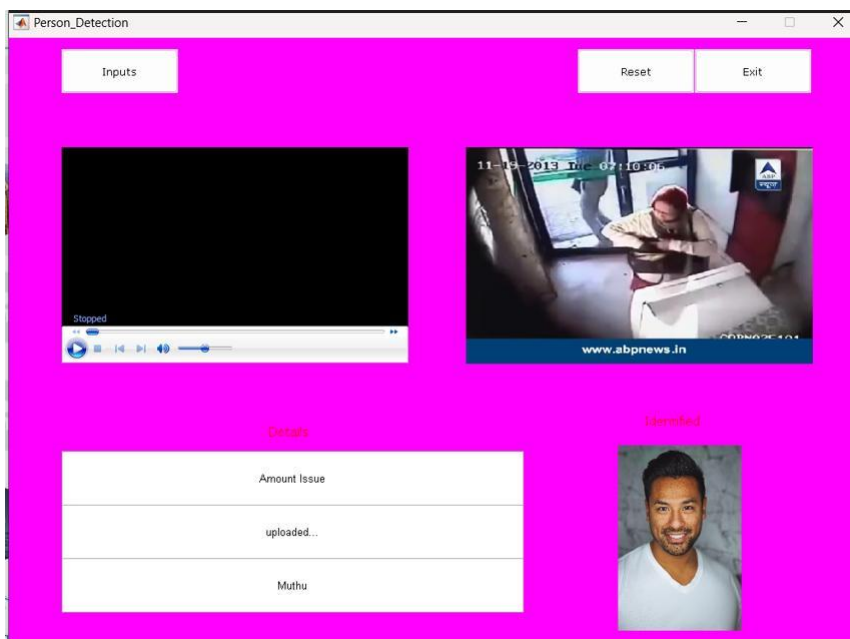
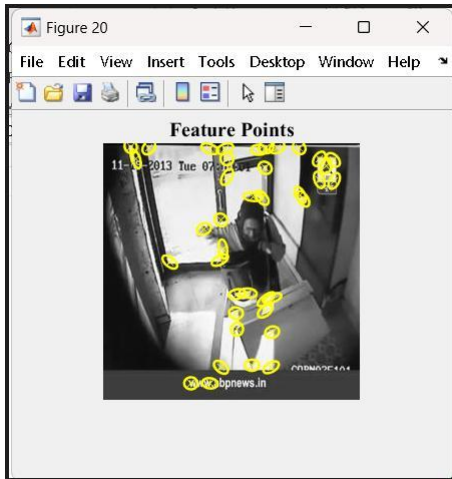
```

% hObject handle to edit3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

7.2 Screenshots



7.3 REFERENCES:

- [1] K.Patel and M. Patel, "Smart Surveillance System using Deep Learning and RaspberryPi," 2021 8th International Conference on Smart Computing and Communications (ICSCC), Kochi, Kerala, India, 2021, pp. 246-251, doi: 10.1109/ICSCC51209.2021.9528194.
- [2] M. Prakash, C. Saravanakumar, S. K. Lakshmi, J. D. Rose and B. Praba, "Automatic Feature Extraction and Traffic Management Using Machine Learning and Open CV Model," 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), Chennai, India, 2021, pp. 1-5, doi: 10.1109/ICSES52305.2021.9633856.
- [3] Z. Xu, M. Miyahara, K. Okada and A. Matsuzawa, "A 3.6 GHz Low- Noise Fractional-N Digital PLL Using SAR-ADC-Based TDC," in IEEE Journal of Solid-State Circuits, vol. 51, no. 10, pp. 2345-2356, Oct. 2016, doi: 10.1109/JSSC.2016.2582854.
- [4] A. M. John-Otumu, O. C. Nwokonkwo, I. U. Izu-Okpara, O. O. Dokun, K. Susan and E. O. Oshoiribhor, "A Novel Smart CBT Model for Detecting Impersonators using Machine Learning Technique," 2020 IEEE 2nd International Conference on Cyberspac (CYBER NIGERIA), Abuja, Nigeria, 2021, pp. 21-30, doi: 10.1109/CYBERNIGERIA51635.2021.9428814.
- [5] T. Hussain, H. Dai, W. Gueaieb, M. Sicklinger and G. De Masi, "UAV- based Multi-scale Features Fusion Attention for Fire Detection in Smart City Ecosystems," 2022 IEEE International Smart Cities Conference (ISC2), Pafos, Cyprus, 2022, pp. 1-4, doi: 10.1109/ISC255366.2022.9921824.
- H. M. Muriithi, I. A. Lukandu and G. W. Wanyembi, "Determining the Location of a Concealed Handgun on the Human Body using Marker-Less Gait Analysis and Machine Learning," 2019 Conference on Next

Generation Computing Applications (NextComp), Mauritius, 2019, pp. 1-6, doi: 10.1109/NEXTCOMP.2019.8883635.

[7] A. El-Shekhi, Y. Maatug and N. Khemri, "Smart Surveillance System Using Deep Learning," 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), Benghazi, Libya, 2023, pp. 171-176, doi: 10.1109/MI-STA57575.2023.10169242.

[8] N. Lokhande, S. Ahirrao, S. Dhamale and M. Shaikh, "I- TVS: Instantaneous-Traffic Violation Surveillance System Using Machine Learning," 2023 Second International Conference On Smart Technologies For Smart Nation (SmartTechCon), Singapore, Singapore, 2023, pp. 1128-1134, doi: 10.1109/SmartTechCon57526.2023.10391806.

[9] A. Jaggi, A. Aggarwal and A. Gupta, "Identifying Anti-Social Activities in Surveillance Monitoring Applications using Deep-CNN based Algorithms," 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2023, pp. 1-5,

[10] doi: 10.1109/ISCON57294.2023.10112113.

[10] M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud and J. -H. Kim, "An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network," 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Vancouver, BC, Canada, 2020, pp. 1-5, doi: 10.1109/IEMTRONICS51293.2020.9216386.

[11] E. J. S. Djula, E. Husni and R. Yusuf, "Design and Implementation of Smart Surveillance System Using Deep Learning Method," 2023 International Seminar on Intelligent Technology and Its Applications (ISITIA), Surabaya, Indonesia, 2023, pp. 462-467, doi: 10.1109/ISITIA59021.2023.10221154.

[12] K. Mridha, M. I. Islam, M. M. Shorna and M. A. Priyok, "ML-DP: A Smart Emotion Detection System for Disabled Person to Develop a Smart City," 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2022, pp. 1-6,

doi: 10.1109/ICRITO56286.2022.9965131.

[13] J. Ahluwalia, J. Joshi and K. Saini, "Camera Nerve Center: A Modern CCTV Surveillance System," 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 2235-2238,

[14] doi: 10.1109/ICAC3N56670.2022.10074068.

[15] A. Medehal, A. Annaluru, S. Bandyopadhyay and T. S. Chandar, "Automated Smart Garbage Monitoring System with Optimal Route Generation for Collection," 2020 IEEE International Smart Cities Conference (ISC2), Piscataway, NJ, USA, 2020, pp. 1-7,

[16] doi: 10.1109/ISC251055.2020.9239002.

[15] H. B, L. K, V. R and U. R, "Smart Break-in-Prevention System With Face Recognition," 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT), Chennai, India, 2022, pp. 1-5,

[16] doi: 10.1109/IC3IoT53935.2022.9767864.

[16] N. C. Brintha, J. A. Narayana, G. L. V. S. Jaswanth, G. J. Chandrapal and D. Venkat, "Realtime Facial Emotion Detection Using Machine Learning," 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), Chennai, India, 2022, pp. 1-5,

doi: 10.1109/ICSES55317.2022.9914318.

[17] A. S. Musthafa, D. Dhananjayan, B. Kaviyarasu, C. Manikandan and S. Vimal, "Smart Authentication System Using Deep Learning Techniques Based on Face and License Plate Recognition," 2022 8th International Conference on Advanced Computing and Communication Systems ICACCS), Coimbatore, India, 2022, pp. 1240-1244,

doi: 10.1109/ICACCS54159.2022.9785188.

[18] M. Al-Mheiri, O. Kais and T. Bonny, "Car Plate Recognition Using Machine Learning," 2022 Advances in Science and Engineering Technology International Conferences (ASET), Dubai, United Arab Emirates, 2022, pp. 1-6,

doi: 10.1109/ASET53988.2022.9734830.

- [19] S. M. Pande, S. Sridharan and S. R. Singh, "Smart Attendance and Attention Monitoring System," 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023, pp. 1-6, doi: 10.1109/ICCCNT56998.2023.10306755.6
- [20] T. Hussain, H. Dai, W. Gueaieb, M. Sicklinger and G. De Masi, "UAV- based Multi-scale Features Fusion Attention for Fire Detection in Smart City Ecosystems," 2022 IEEE International Smart Cities Conference (ISC2), Pafos, Cyprus, 2022, pp. 1-4, doi: 10.1109/ISC255366.2022.9921824.
- [21] A. El-Shekhi, Y. Maatug and N. Khemri, "Smart Surveillance System Using Deep Learning," 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), Benghazi, Libya, 2023, pp. 171- 176,
- [22] doi: 10.1109/MI-STA57575.2023.10169242.