Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Computer Science & Engineering

Mid Term Report

On

Using existing CCTV network for crowd management, crime prevention, and work monitoring using AI/ML

Project-I

BACHELOR OF TECHNOLOGY

(Robotics and Artificial Intelligence)

SUBMITTED BY:

Rahul kumar

2231242

April 2025

Under the Guidance of

Dr. Puneet

Department of Computer Science & Engineering

Chandigarh Engineering College

Jhanjeri, Mohali – 140307

Table of content

Chapter- 1

Introduction

In an increasingly urbanized and security-conscious world, the deployment of closed-circuit television (CCTV) systems has become widespread for surveillance and public safety. However, traditional CCTV networks primarily rely on manual monitoring, which is often inefficient, resource-intensive, and prone to human error. With the rapid advancements in Artificial Intelligence (AI) and Machine Learning (ML), there exists a significant opportunity to enhance the effectiveness of existing CCTV infrastructure by integrating intelligent automation and real-time analytics.

This project explores how AI/ML technologies can be utilized to transform passive video surveillance systems into proactive tools for **crowd management**, **crime prevention**, and **workplace monitoring**. By leveraging computer vision, pattern recognition, and predictive modeling, these systems can detect unusual activities, identify potential threats, optimize crowd flow, and ensure compliance with safety protocols — all without the need for costly hardware overhauls.

The goal of this study is to assess the feasibility, design, and implementation of AI-powered video analysis on existing CCTV networks, highlighting the practical benefits, technical challenges, and ethical considerations involved in this smart surveillance transformation.

Chapter- 2

System Requirements

To implement AI/ML-based enhancements on existing CCTV infrastructure, both hardware and software components are required. The system must be capable of real-time video processing, data storage, and intelligent decision-making. Below is a breakdown of the functional and non-functional system requirements:

1. Hardware Requirements

- **CCTV Cameras**
  - Existing analog or IP-based CCTV cameras with at least 720p resolution
  - Night vision and motion detection capabilities (optional but beneficial)
- **Processing Unit / Server**
  - CPU: Intel i7 or higher (or equivalent AMD Ryzen)
  - GPU: NVIDIA GPU (e.g., RTX 3060 or higher) with CUDA support for real-time AI processing
  - RAM: Minimum 16 GB
  - Storage: SSD with at least 1 TB for high-speed video read/write and model storage
  - Network Interface: Gigabit Ethernet for high-speed video data transfer
- **Edge Devices (Optional)**
  - AI-enabled edge processors (e.g., NVIDIA Jetson, Google Coral) for decentralized processing

2. Software Requirements

- **Operating System**
  - Ubuntu Linux (preferred for AI/ML development) or Windows 10/11 (if needed for compatibility)
- **Development Tools and Libraries**
  - Python 3.x

  - OpenCV for real-time image and video processing
  - TensorFlow / PyTorch for AI/ML model development
  - YOLOv5 or SSD for object detection
  - Scikit-learn for analytics and pattern recognition
  - Flask / FastAPI (for API-based model deployment, if needed)
- **Database**
  - MongoDB or PostgreSQL for storing event logs, metadata, and user activity
  - Cloud storage integration (optional) for scalable video archiving
- **Video Management Software (VMS)**
  - Integration with existing VMS or custom middleware for stream handling and alert generation

3. Functional Requirements

- Real-time video feed acquisition from existing CCTV
- Object and activity detection (e.g., people counting, loitering, violence detection)
- Facial recognition and tracking (with proper consent and ethics)
- Alert system for abnormal behavior or crowd surges
- Reporting dashboard for analytics and compliance monitoring

4. Non-Functional Requirements

- **Scalability**: Should support multiple camera feeds simultaneously
- **Accuracy**: High detection accuracy with minimal false positives/negatives
- **Security**: Secure data transmission and access control for video and analytics
- **Latency**: Low-latency processing for real-time alerting
- **Reliability**: High availability and system uptime for 24/7 monitoring

Software Requirement Analysis

Software Requirement Analysis is essential to ensure the development and deployment of an efficient and reliable AI-based surveillance system using existing CCTV infrastructure. This section outlines the specific software functionalities, constraints, and expected outcomes necessary for successful implementation.

1. Functional Requirements

- **Video Stream Integration**: The system must connect with existing CCTV cameras to access live video feeds for analysis.
- **AI-Powered Detection**: The software should detect objects, identify people, and recognize specific behaviors such as crowd buildup, loitering, or suspicious movement using machine learning models like YOLOv5 or SSD.
- **Event Logging and Alerts**: Detected events should be logged with timestamps and camera IDs, and real-time alerts should be generated for anomalies (e.g., crowding, unusual activity).
- **Workplace Monitoring**: The system should be able to monitor workers' presence, detect PPE usage (helmets, vests), and assess work duration or compliance.
- **User Dashboard**: An interface should be available for users to view live analytics, review past events, and customize system settings.

2. Non-Functional Requirements

- **Performance**: The system should analyze video feeds in real-time or near-real-time with minimal delay.
- **Scalability**: It must support multiple camera feeds and expand without major architectural changes.
- **Security**: Data and user access must be protected through encryption and role-based access control.
- **Usability**: The dashboard should be intuitive and accessible for non-technical users.
- **Reliability**: The software should be able to run continuously and recover from failures automatically.

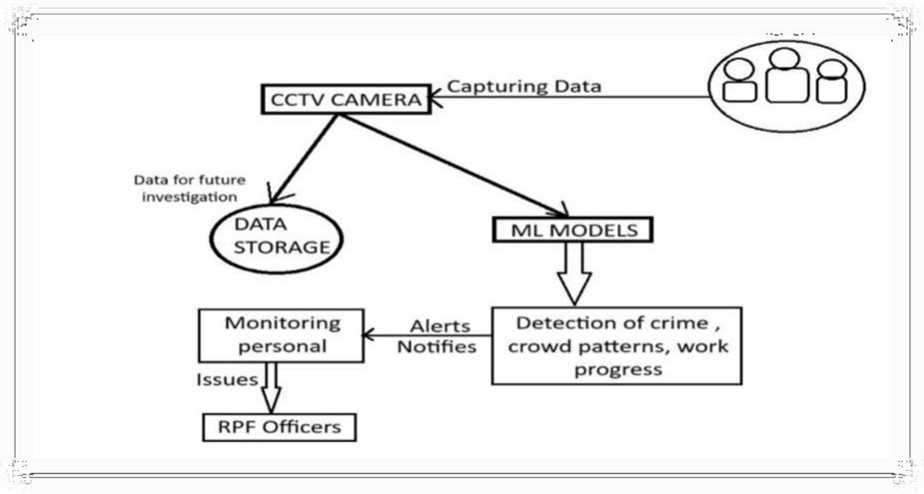
3. Tools and Technologies

- **Programming Language**: Python (for AI/ML logic), JavaScript (for dashboard UI)
- **Libraries & Frameworks**: OpenCV, TensorFlow or PyTorch, Flask or FastAPI
- **Database**: PostgreSQL or MongoDB for storing logs and analytics
- **Deployment Tools**: Docker (for containerization), cloud or local servers
- **Additional Tools**: Git (version control), Jupyter Notebook (for ML model testing)

Chapter- 4

Software Design

The software design phase outlines how the system will be structured and how its components will interact. For this AI-enhanced CCTV system, the design includes data flow representation, database structure, and entity-relationship modeling.



a. Data Flow Diagrams (DFDs)

Level 0 DFD – Context Diagram

This diagram shows the system as a single process interacting with external entities:

- **Entities**:
  - CCTV Camera Feed (External Source)
  - Admin/User (External Actor)
- **Process**:
  - AI-Powered Surveillance System
- **Data Stores**:
  - Event Log Database

- **Data Flows**:
  - Live Video Input → AI Model
  - Detection Alerts → Admin/User
  - Event Metadata → Event Log DB

Level 1 DFD – Detailed Process View

- **Processes**:
  1. Video Stream Input
  2. AI/ML Detection Module (Object Detection, Behavior Analysis)
  3. Alert Generation System
  4. Event Logging
  5. Dashboard Display
- **Data Stores**:
  - D1: Event Log Database
  - D2: User Credentials (for access control)
- **External Entities**:
  - CCTV Cameras
  - Admin/User

b. Database Design

- The system requires a structured database to store events, users, and camera metadata.
- **Main Tables:**

Table Name	Description
`Users`	Stores user credentials and access roles
`Cameras`	Info about each connected CCTV camera
`Events`	Logs all detected events
`Detections`	Details of objects detected per event

Table Name	Description
`Alerts`	Stores active or historical alerts

c. E-R Diagram (Summary)

- **Entities**:
  - User, Camera, Event, Detection, Alert
- **Relationships**:
  - One User can log many Events
  - One Camera can generate many Events
  - One Event can have many Detections and one Alert

Implement

The implementation of the system involves integrating AI/ML models with the existing CCTV infrastructure to analyze video feeds in real-time and generate meaningful insights. Below are the key steps involved:

1. Integration with Existing CCTV Cameras

- Connect to IP or analog CCTV feeds using RTSP (Real-Time Streaming Protocol) or video file input.
- Use software like OpenCV to capture and process video frames.

2. Real-Time Video Processing

- Extract frames from the live video feed.
- Resize and format frames to be compatible with ML models.
- Apply object detection and activity recognition using models like **YOLOv5**, **SSD**, or **OpenPose**.

3. AI/ML Model Application

- **Crowd Management**: Count people in a frame and detect overcrowded zones.
- **Crime Prevention**: Identify unusual or suspicious activities such as loitering, aggression, or trespassing.
- **Work Monitoring**: Detect worker presence, PPE compliance (e.g., helmets), and task duration.

4. Alert and Notification System

- If the AI model detects an abnormal event (e.g., crowd surge, safety violation), generate alerts in real-time.
- Notify security personnel via email, SMS, or a dashboard.

5. Event Logging and Dashboard Display

- Store detected events with time, location, and camera ID in a database.

- Visualize alerts, analytics, and event history through a web-based dashboard using tools like Flask, HTML, and JavaScript.

6. Testing and Optimization

- Test the system with different camera angles, lighting conditions, and crowd scenarios.
- Fine-tune ML model thresholds for accuracy and reduce false positives.