CHAPTER 1
INTRODUCTION

1.1 Introduction

In recent years, ensuring security and safety in both private and public spaces has become a critical concern. Traditional security systems, such as CCTV surveillance, alarms, and physical barriers, while effective to some extent, often suffer from limitations like false positives, human error, and inefficiency in real-time response. As security threats become more sophisticated, there is a growing need for advanced systems that can intelligently detect and respond to potential security breaches. One such innovation lies in the integration of Machine Learning (ML) techniques to develop Trespasser Detection Systems.

A Trespasser Detection System using machine learning aims to automate the identification of unauthorized individuals entering restricted areas. By leveraging algorithms capable of learning from data, this system can analyze real-time video feeds, sensor inputs, and environmental data to accurately detect potential intruders. These systems improve upon traditional security methods by providing proactive and automated threat identification, reducing the dependency on human intervention.

Machine learning techniques, particularly Computer Vision, Object Detection, and Anomaly Detection, play a pivotal role in identifying human movement and behavior patterns in real-time. These systems can be trained on large datasets to recognize typical and atypical activities, minimizing false alarms and optimizing the response time. Moreover, the use of deep learning algorithms like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) has made it possible to detect intruders with high accuracy, even under challenging environmental conditions such as poor lighting or low-resolution footage.

The purpose of this project is to design and implement a Trespasser Detection System that utilizes machine learning models to detect and alert authorities about potential trespassing incidents. This thesis will explore the various techniques used in training the models, the data collection process, system architecture, and the overall effectiveness of the proposed system in real-world applications. By automating the detection process and reducing human oversight, this system represents a significant step forward in the evolution of security systems, offering improved reliability, scalability, and efficiency.

1.2 Problem Statement

The security of restricted areas and valuable assets remains a growing challenge with the limitations of traditional security systems, such as CCTV cameras and alarms. While these systems are commonly used, they often fail to effectively detect and respond to real security threats in a timely manner. One key issue is their inability to distinguish between different types of movement, which leads to false alarms and unnecessary interventions, wasting time and resources. Additionally, many systems have a slow response time and do not provide real-time alerts, allowing trespassers to cause damage or theft before security personnel can react.

Another significant problem is that traditional systems struggle with monitoring remote or unmanned locations, which lack the necessary human presence and infrastructure. These areas are often more vulnerable to unauthorized access, as current systems cannot provide adequate surveillance in such environments.

These limitations highlight the shortcomings of current security systems, which fail to provide the accuracy, speed, and coverage needed to effectively protect valuable assets and ensure safety.

1.3 Motivation

Traditional security systems face several key challenges in detecting unauthorized access to private and restricted areas. One of the main problems is the reliance on human monitoring, which can be inefficient and prone to oversight. Surveillance footage requires constant attention, and even then, intrusions may not be detected quickly. Additionally, these systems often have slow response times, leading to delayed actions when a trespassing event occurs.

This project addresses these issues by using machine learning to automate the detection of trespassers in real-time. By analyzing video feeds instantly, the system can identify intruders without the need for constant human monitoring. It provides immediate alerts to security teams, improving response times and minimizing the chances of missing important events. Through this automated detection, the system enhances both the accuracy and efficiency of security operations, overcoming the limitations of traditional surveillance methods.

1.4 Proposed System

The proposed system is designed to provide real-time detection of trespassers, aiming to improve security by identifying unauthorized individuals in restricted areas. Utilizing advanced object detection techniques such as YOLO (You Only Look Once) for real-time object recognition, the system continuously analyzes video feeds from surveillance cameras to detect potential trespassers. The system is equipped with motion detection algorithms and deep learning models to accurately identify individuals and distinguish between authorized and unauthorized presence. Once a trespasser is detected, an alert is triggered, notifying security personnel for immediate action.

CHAPTER 2
LITEERATURE REVIEW

2.1 Existing Systems

[1] Deshpande, H., Singh, A., & Herunde, H. (2020). Comparative analysis on YOLO object detection with OpenCV. ***International journal of research in industrial engineering,*** **9(1), 46-64.**

The survey paper "Comparative Analysis of Object Detection Models using YOLO, Faster R-CNN, and Fast R-CNN" compares different object detection models—YOLO, Faster R-CNN, and Fast R-CNN—using OpenCV. The COCO dataset, which contains a variety of annotated images, was used for training and testing these models.The paper highlights the drawbacks of each model. R-CNN is slow and uses a lot of memory, while Fast R-CNN is faster but struggles with small objects. Faster R-CNN improves speed with Region Proposal Networks (RPNs), but it is still too slow for real-time use. YOLO is the fastest, designed for real-time detection, but it faces challenges with small or hidden objects.In terms of performance, YOLO achieved a detection time of 1.93 seconds, much faster than Faster R-CNN (38.53 seconds) and Fast R-CNN (41.64 seconds). While YOLO is ideal for real-time applications, it struggles with small or occluded objects. Overall, YOLO is the best option for speed, while Faster R-CNN and Fast R-CNN are more accurate but not suitable for real-time detection.

[2] **Arumugam,K.,& Rajgure,Y.(2021). Railway Trespassing Detection and Alert System using Deep Learning, CNN, YOLO.** ***International Research Journal of Engineering and Technology.***

The survey paper "Railway Trespassing Detection and Alert System using Deep Learning, CNN, YOLO" presents a system designed to detect and prevent unauthorized trespassing on railway tracks using advanced computer vision techniques. The methodology involves capturing live video footage, filtering frames for relevant content, and utilizing YOLO (You Only Look Once) CNN for efficient object detection and real-time alert generation to prevent accidents. The system is trained using video surveillance data from railway tracks, ensuring its effectiveness in real-world scenarios. Despite its high performance, achieving an accuracy of 90-95% in detecting trespassing activities, the system faces challenges such

as high computational costs, expensive installation requirements, and concerns about the reliability of cameras under varying environmental conditions.

[3] Varma, B., & Reddy, T. (2022). SMART CCTV CAMERA MONITORING SYSTEM USING IOT. *International Research Journal of Modernization in Engineering Technology and Science*.

The survey paper "Smart CCTV Camera Monitoring System Using IoT" presents a security system that integrates USB cameras for live monitoring, face recognition, and motion detection. It utilizes PIR/IR sensors to detect motion and sends real-time alerts via email or SMS, ensuring timely notifications for users. The system is tested using datasets that include face recognition images, motion detection data, and environmental footage to validate its effectiveness. Despite its benefits, the system faces challenges such as privacy concerns, high setup costs, vulnerability to hacking, and the need for significant storage capacity for recorded footage. Overall, the system enhances property security by providing real-time alerts, reducing false alarms, and improving safety measures.

[4] Rahmaniar, W., & Hernawan, A. (2021). Real-time human detection using deep learning on embedded platforms: A review. *Journal of Robotics and Control (JRC)*, 2(6), 462-468.

The survey paper "Real-Time Human Detection Using Deep Learning on Embedded Platforms" presents a project designed to develop an efficient real-time human detection system on embedded platforms like the NVIDIA Jetson TX2 and Nano, evaluating the performance of deep learning models including PedNet, Multiped, SSD MobileNet V1/V2, and SSD Inception V2 using datasets such as PASCAL VOC, COCO, and ILSVRC. The methodology involved assessing each model's accuracy and speed under different conditions, with drawbacks including PedNet's struggle with clutter, Multiped's occlusion issues, SSD MobileNet V1's poor performance with blurred images, and SSD Inception V2's failure in low-light conditions. The result showed that SSD MobileNet V2 outperformed all models with the highest accuracy and speed, processing at 17.32 FPS, making it the most effective model for real-time human detection.

[5]Ma, B. (2020, October). Design of intelligent burglar alarm system in laboratory based on embedded system. In 2020 13th *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)* (pp. 392-395). IEEE.

The survey paper "Design of Intelligent Burglar Alarm System in Laboratory Based on Embedded System" presents a project designed to create an anti-theft alarm system using the STC89CS2 microcontroller and pyroelectric infrared sensors, with sound and light alarms triggered by motion detection. The system was evaluated through simulations and practical tests, as no specific datasets were used. The methodology focused on detecting motion and activating alarms upon presence detection. The drawbacks included a limited range, the potential for false alarms, and the system's reliance on continuous power supply. Despite these limitations, the result demonstrated successful detection of presence and reliable activation of alarms, effectively meeting the design objectives.

[6] Goel, P. (2021). Realtime object detection using tensorflow an application of ml. *International Journal of Sustainable Development in Computing Science*, 3(3), 11-20.

The survey paper titled *"Real-Time Object Detection Using TensorFlow: An Application of Machine Learning"* explores the implementation of the SSD (Single Shot Multibox Detector) architecture for real-time object detection. The methodology used involves training the SSD model on the PASCAL VOC dataset, which contains approximately 10,000 images across 20 classes, with over 25,000 annotations. The model employs multi-box classification loss to achieve accurate object detection in real-time. However, the paper highlights several drawbacks, including variable output dimensions, the trade-off between speed and accuracy, challenges with occlusion, and the dominance of larger objects in the detection process. Despite these challenges, the model achieves a mean Average Precision (mAP) of 0.633, with the highest precision for detecting cats (0.909) and the lowest for bottles (0.272). Additionally, the model demonstrated robustness to illumination variations, making it effective in varied lighting conditions.

[7] Keat, L. H., & Wen, C. C. (2018). Smart indoor home surveillance monitoring system using Raspberry Pi. JOIV: *International Journal on Informatics Visualization*, 2(4-2), 299-308.

The survey paper titled "Smart Indoor Home Surveillance Monitoring System Using Raspberry Pi" discusses the development of a home surveillance system utilizing Object-Oriented Analysis and Design (OOAD). The methodology involves implementing several modules, including motion detection, image capturing, and notifications. The system operates using real-time data collected from sensors and cameras, without relying on specific datasets. However, the paper identifies several drawbacks, including being limited to a single user, the potential for false positives, reliance on a stable internet connection, and power dependency. Despite these limitations, the system successfully captures intruder images, provides timely notifications (SMS within 10 seconds and email within 30 seconds), and achieves successful user acceptance.

[8] **Schwartz, W. R., Kembhavi, A., Harwood, D., & Davis, L. S. (2009, September). Human detection using partial least squares analysis. In *2009 IEEE 12th international conference on computer vision* (pp. 24-31). IEEE.**

The survey paper titled "Human Detection Using Partial Least Squares Analysis" presents a human detection method combining Histogram of Oriented Gradients (HOG), color, and texture features. The methodology employs Partial Least Squares (PLS) for dimensionality reduction and Quadratic Discriminant Analysis (QDA) for classification after PLS reduction. A two-stage detection approach is used to enhance speed: initial filtering with a small feature set, followed by detailed analysis of the remaining windows. The system is tested on several datasets, including the INRIA Person Dataset (2416 positive samples), the ETHZ Pedestrian Dataset (evaluated in crowded scenes), and the DaimlerChrysler Dataset (tested at low resolutions). However, the paper highlights some drawbacks, such as high dimensionality, false positives from HOG features alone, and accuracy issues due to variability in clothing colors. Despite these challenges, the approach achieves low false positive rates (10^{-5} to 10^{-6} FPPW) and outperforms state-of-the-art methods, proving effective in detecting humans even at low resolutions.

[9] Aboyomi, D. D., & Daniel, C. (2023). A Comparative Analysis of Modern Object Detection Algorithms: YOLO vs. SSD vs. Faster R-CNN. *ITEJ (Information Technology Engineering Journals)*, 8(2), 96-106.

The survey paper titled "A Comparative Analysis of Modern Object Detection Algorithms: YOLO vs. SSD vs. Faster R-CNN" provides a comprehensive comparison of three popular object detection algorithms—YOLO, SSD, and Faster R-CNN—using the COCO dataset. The methodology involves evaluating metrics such as mean Average Precision (mAP), inference time, FPS, and GPU memory usage after fine-tuning pre-trained models. The COCO dataset, which features a diverse range of object categories and challenging scenarios, was used for testing. The paper highlights several drawbacks: YOLO struggles with detecting small objects, SSD has moderate accuracy in crowded scenes, and Faster R-CNN suffers from high computational cost and slower inference times. The results show that YOLOv4 achieves a mAP@0.5 of 54.30%, SSD achieves 56.80%, and Faster R-CNN performs the best with 61.20%. In terms of inference time, YOLOv4 processes images in 25 ms, SSD takes 45 ms, and Faster R-CNN takes 120 ms. For GPU memory usage, YOLOv4 consumes 2800 MB, SSD uses 3200 MB, and Faster R-CNN uses 5400 MB.

[10] Parveen, S., & Shah, J. (2021, February). A motion detection system in python and opencv. In *2021 third international conference on intelligent communication technologies and virtual mobile networks (ICICV)* (pp. 1378-1382). IEEE.

The survey paper titled "A Motion Detection System in Python and OpenCV" outlines a motion detection system implemented using Python and OpenCV. The methodology involves capturing initial frames from a webcam, converting them to grayscale, calculating the phase difference (Delta_frame), and applying image processing techniques such as thresholding, dilation, and contouring. The system also records timestamps for object entry and exit and visualizes data using Bokeh. It operates in real-time, relying on live input from the webcam, rather than predefined datasets, for testing and analysis. The paper highlights several limitations, including sensitivity to lighting changes, varying performance based on camera quality, and the restriction to detecting only large moving objects due to area thresholding. The system outputs visual frames of detected motion, logs timestamps in a CSV file, and provides graphs to show the duration of object presence.

[11] Singla, N. (2014). Motion detection based on frame difference method. *International Journal of Information & Computation Technology*, 4(15), 1559-1565.

The survey paper titled "Motion Detection Based on Frame Difference Method" describes the methodology employed in detecting motion through the frame difference technique. The system begins by capturing sequences of images from a static camera. The core of the motion detection process involves calculating the absolute difference between consecutive frames, which highlights areas of motion. To enhance the accuracy of the detection, morphological operations such as dilation and erosion are applied to the resulting difference images to reduce noise and improve object visibility. After identifying motion, the system produces binary images where areas of detected motion are shown in white, while regions with no movement are black. While this method effectively detects moving objects, it faces limitations such as false motion detection caused by factors like air movement and the creation of holes in the binary image when no object is present. Despite these challenges, the frame difference method demonstrates reliable performance for detecting moving objects in various scenarios.

[12] Iqbal, M. J., Iqbal, M. M., Ahmad, I., Alassafi, M. O., Alfakeeh, A. S., & Alhomoud, A. (2021). *Real-Time Surveillance Using Deep Learning. Security and Communication Networks*, 2021(1), 6184756.

The survey paper titled *"Real-Time Surveillance Using Deep Learning"* discusses the use of the frame difference method for detecting moving objects in a real-time surveillance system. The methodology involves transforming frames to grayscale, applying Gaussian low-pass filtering to reduce noise, and using binarization to identify moving pixels. The system operates on sequences of images captured from a static camera under stable lighting conditions, with different scenarios tested for object detection. However, the system is sensitive to noise, lighting changes, shadows, and unintended motion, such as air movement, which may lead to false positives in detection. Despite these challenges, the system effectively identifies moving objects with good performance and efficiency, although it can occasionally mistakenly detect motion due to environmental factors.

[13] Redmon, J. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition.*

The survey paper "You Only Look Once: Unified, Real-Time Object Detection" presents a novel object detection methodology called YOLO, which employs a unified regression approach. This method predicts bounding boxes and class probabilities directly from full images using a single convolutional neural network (CNN). The paper evaluates YOLO on well-established datasets such as the PASCAL VOC 2007 and 2012 datasets, and further tests its generalizability on the Picasso and People-Art datasets. Despite its innovative design, YOLO faces several drawbacks, including difficulty in detecting small objects, limited generalization to unusual aspect ratios, and reliance on coarse features resulting from downsampling, which can negatively impact localization accuracy. The results demonstrate that YOLO achieves a 57.9% mean Average Precision (mAP) on the PASCAL VOC 2012 dataset, showing competitive performance, especially in detecting certain classes, but struggles with the accurate detection of small objects.

[14] Bathija, A., & Sharma, G. (2019). Visual object detection and tracking using yolo and sort. *International Journal of Engineering Research Technology*, 8(11), 345-355.

The survey paper "Visual Object Detection and Tracking using YOLO and SORT" explores the use of YOLO for object detection and the SORT (Simple Online and Realtime Tracking) algorithm for object tracking in Python. The methodology involves the application of YOLO for detecting objects in video frames and SORT for tracking them across frames, utilizing a custom dataset of 800 annotated images. The dataset consists of six classes: Person, Car, Truck, Bus, Bicycle, and Motorbike, and was manually annotated using LabelImg. Despite its strengths, the approach faces potential inaccuracies in object detection due to issues such as occlusions and the similarity between different objects, which can result in misidentification and tracking failures. The results show varying accuracy and precision across five video sequences, with overall accuracy ranging from 44.4% to 85.1%, indicating that while the system performs well in certain scenarios, it also has limitations in others.

2.2 Survey Table

Sl No	Title	Methodology	Dataset	Drawbacks	Result
1	Comparative Analysis on YOLO Object Detection with OpenCV	Compare YOLO, Faster R-CNN, Fast R-CNN using OpenCV; Steps include grid creation, object detection and visualization	COCO Dataset: Diverse images for training and testing	R-CNN: Slow, high memory, Fast R-CNN:Limited on small objects, Faster R-CNN: Time consuming, YOLO: Fast but struggles with small/ obscured objects	YOLO achieved a detection time of approximately 1.93 seconds, while Faster RCNN took 38.53 seconds and Fast R-CNN took 41.64 seconds. YOLO proved effective for real-time applications.
2	Railway Trespassing Detection and Alert System using Deep Learnig,CNN ,YOLO	Live video capture, frame filtering, YOLO CNN for detection, alert generation.	Video surveillance data from Railway tracks.	High computational costs, expensive insta llation,doubts about c amera effectiveness.	90-95% accuracy in Trespassing detection, real-time alerts.
3	Smart CCTV Camera Moni toring System Using IOT	Utilizes USB cameras for live monitoring and face recognition. PIR/IR sensors detect motion; alerts via email /SMS.	Face recognition images, motion detection data and environment al footage for system testing.	Privacy concerns, highsetup costs, and vulnerability to hacking. Requires significant storage for footage.	Enhanced security with real-time alerts,reduced false alarms and improved safety for properties.
4	Real Time Human Detection Using Deep Learning on Embedded Platforms	Evaluated deep learning models (PedNet,multiped, SSD MobileNetV1/V2, S SD inception V2) on NVIDIA Jetson TX2 and Nano using datasets for real-time human detection.	PASCAL VOC, COCO (Common Objects in Context), ILSVRC(Image Net Large Scale Visual Recogn-ition Challenge)	PedNet struggled with clutter, multiped faced occlusion issues, SSD MobileNet V1 was poor with blurred Images and SSD Inception V2 performed badly in low light.	SSD MobileNet V2 achieved the highest accuracy and speed, processing at 17.32 FPS, making it the most effective model for real-time human detection .
5	Design of Intelligent Burglar Alarm Syste min Laborato ry Based on Embedded System	Designed an anti-theft alarm system using STC89CS2 microcontro ller and pyroelectric infrared sensors. Implemented sound and light alarms triggered by motion detection.	No specific datasets were used. The system was evaluated through simulations and practical tests	Limited range, potential for false Alarms and reliance on power.	Successful detection of presen ce and activation of alarms, meeting the design objectives.

6	Realtime Object Detection Using Tensor Flow an application of ML	SSD architecture; trained on PASCAL VOC dataset;employing multibox classification loss for accurate object detection in real-time.	PASCAL VOC: 10,000 images,20classes, 25,000 annotations.	Variable output dimensions, speed-accuracy trade-off, occlusion issues, larger object dominance.	Mean Average Precision (MAP) of 0.633, with the highest precision for Cat (0.909) and lowest for Bottle (0 .272), and showed robustness to illumination variations.
7	Smart Indoor Home Surveillance Monitoring System Using Raspberry Pi	Object-Oriented Analysis and Design (OOAD), Implementation of Modules (motion detection,image capturing, notifications)	No specific datasets. Relies on real-time data from sensors and cameras.	Limited to single user, potential false positives, reliance on stable internet, and power dependency.	Captured intruder images, timely notifications (SMS under 10s, email under 30s), and successful user acceptance.
8	Human Detection Using Partial Least Squares Analysis	Combined HOG, color, and texture features; used PLS for dimensionality reduction and QDA for classification after PLSreduction; two-stage detection approach to enhance speed: initial filtering with a small feature set, followed by detailed analysis on remaining windows.	INRIA Person Dataset: 2416 positive sample. ETHZ Pedestrian Dataset:Tested in crowded scenes. DaimlerChrysler Dataset: Evaluated on low resolutions.	High dimensionality, false positives from HOG features alone, variability in clothing colors may affect accuracy.	Low false positive rates (10^{-5}, 10^{-6} FP PW), outperformed state-of-the-art methods, effective detection at low resolutions.
9.	A Comparative Analysis of Modern Object Detection Algorithms: YOLO vs SSD vs Faster R-CNN	Compared YOLO, SSD, and Faster R-CNN using the COCO dataset, evaluating metrics like mAP, inference time,FPS, and GPU memory usage after fine-tuning pre-trained models.	COCO: Diverse range of object categories and challenging scenarios.	YOLO difficulty in detecting small objects, SSD moderate accuracy in crowded scenes, and Faster R-CNN high computational cost and slower inference times.	YOLOv4 is best suited for application requiring real-time detection with moderate accuracy. SSD provides a good balance of speed and accuracy. Faster R-CNN should be used when high precision is crucial, despite its slower processing speed.
10.	A Motion Detection System in Python and OpenCV	Initial frame capture via webcam, Convert to grayscale, Calculate phase difference (Delta frame), Apply thresholding and image processing (dilation, contouring), Capture timestamps for object entry/exit, Plot data using Bokeh.	Images captured in real-time from webcam. No predefined data sets; relies on live input for testing and analysis.	Sensitive to lighting changes, performance varies by camera quality, limited to detecting large moving objects due to area thresholding.	Visual frames of detected motion, a CSV file logging timestamps, and graphs show duration of object presence in front of the camera.

11	Motion Detection Based on Frame Difference Method	Frame difference method: detects moving objects by calculating the absolute difference between consecutive frames and applying morphological operations.	Sequences of images captured from a static camera, comparing two frames to detect movement.	False motion detection due to air movement, holes in binary images when no actual object is present.	Binary images show white for detected motion and black for no movement, effective detection of moving objects demonstrated.
12	Real-Time Surveillance Using Deep Learning	Frame difference method for moving object detection,Transformation of frames to gray images, Gaussian low-pass filtering to reduce noise, Binarization to identify moving pixels.	Sequences of images captured from a static camera under stable lighting conditions, different scenarios tested for object detection.	Sensitive to noise, light ing changes, shadows, and unintended motion, such as air movement, which can lead to false positives in detection.	Effectively identifies moving objects with good performance and efficiency, though it can also mistakenly detect motion due to environmental factors .
13.	You Only Look Once: Unified, Real-Time Object Detection	YOLO employs a unified regression approach that predicts bounding boxes and class probabilities directly from full images using a single convolutional neural network.	PASCAL VOC dataset (2007 and 2012) and tested for generalizability on the Picasso and People-Art datasets.	Difficulty in detecting small objects, limited generalization to unusual aspect ratios, and reliance on coarse features from down sampling, which can affect localization accuracy.	YOLO achieved 57.9% mAP on the PASCAL VOC 2012 dataset, demonstrating competitive performa nce with strengths in certain classes but lower accuracy on small objects.
14.	Visual Object Detection and Tracking using YOLO and SORT	YOLO for object detection and SORT for tracking in Python with a custom dataset of 800 annotated images.	Custom dataset of 800 images with 6 classes: Person, Car, Truck, Bus, Bicycle,Motorbike manually annotated using Labelling.	Potential inaccuracies in object detection due to occlusions and similarities between objects, which can lead to misidentification and tracking failures.	Varying accuracy and precision across five video sequences, with overall accuracy ranging from 44.4% to 85.1%, highlightin g strengths in some scenarios but also limitations in others.

2.3 Research Gap

Through an extensive review of the literature papers, it become clear that current security systems face several important challenges. Many existing systems lack advanced detection technologies, limiting their ability to effectively monitor vulnerable areas and detect intruders accurately. Additionally, the studies highlight the absence of proactive alert systems, which are essential for notifying individuals when an intrusion is detected. Without such alert systems, timely responses to security threats are compromised, leaving room for potential breaches. These observations suggest that improving detection

capabilities and integrating alert mechanisms is crucial for enhancing the effectiveness of security systems.

2.3 Survey Outcome

Our examination of the "Real-Time Trespasser Detection System" using advanced technologies such as computer vision and machine learning demonstrated promising results. The analysis focused on evaluating object detection techniques and their efficiency in real-time trespasser detection. The findings revealed that traditional security methods, such as manual surveillance and static sensors, consume more time and resources while offering limited accuracy and responsiveness. This project aims to improve the detection speed, accuracy, and automation of trespasser identification using cutting-edge real-time tracking and detection technologies**.**

[2] This paper presents a real-time trespassing detection system for railway tracks that leverages YOLO for object detection and CNN for feature extraction. The study highlights the ability of the system to detect and identify individuals on tracks with high accuracy, providing real-time alerts to authorities for prompt action.

[10] This paper presents a motion detection system that leverages Python and OpenCV for real-time object detection. It uses webcam capture and various image processing techniques such as frame differencing and contour detection to detect movement efficiently. The study highlights how the system can log timestamps for the duration of motion, providing valuable insights.

[6] This paper presents the application of the SSD architecture, trained on the PASCAL VOC dataset, to achieve real-time object detection. The study demonstrates the system's capability to accurately detect and classify objects, even in challenging conditions, by employing multibox classification loss for precise localization.

[9] This paper presents a comparative analysis of YOLO, SSD, and Faster R-CNN on the COCO dataset, emphasizing key performance metrics such as mAP and inference speed. The study discusses the strengths and weaknesses of each model, providing insights into their suitability for various use cases.

CHAPTER 3
METHODOLOGY

3.1 Feasibility Study

A feasibility study for the Real-Time Trespasser Detection System evaluates the project's viability by analyzing five key areas: the frame of definition (project goals and objectives), frame of contextual risks (external risks like environmental and legal factors), frame of potentiality (future scalability), parametric frame (system hardware and software requirements), and the frame of dominant and contingency strategies (implementation and backup plans). The study also considers the four Ps: Plan (project road map), Processes (integration procedures), People (team skills and coordination), and Power (stakeholders and decision-makers). Risks such as technological, environmental, and legal issues are identified, alongside Points of Vulnerability (internal risks). Constraints on calendar (timeline), costs (budget), and norms of quality (technical and regulatory standards) are also assessed. The study involves reviewing relevant research to define system requirements and guide the implementation process. This helps ensure that the project is feasible, efficient, and meets security objectives. For this system, the feasibility study is conducted by reviewing various research papers and then a rough idea is generated about the System Requirements i.e. hardware and software requirements. When all the given requirements were available, implementation of the system got start.

3.1.1 MINIMUM SYSTEM REQUIREMENTS

3.1.1.1 Hardware Requirements:

- 8GB RAM or more preferred for optimal performance.
- Intel Core i5 or AMD Ryzen 7 with at least 6 cores for efficient processing.
- NVIDIA GeForce GTX 1650 or higher for faster image processing and machine learning tasks.
- 250GB SSD (Solid State Drive) for faster data retrieval and system operations.
- Preferably an 8-core processor with support for AVX (Advanced Vector Extensions) for high-performance computation.
- High-definition cameras (720p or higher) for real-time video surveillance.

3.1.1.2 Software Requirements:

- 64 bit operating system, Windows 10/11 preferred
- Visual Studio 2022 or equivalent modern IDE with Python support
- TensorFlow (CPU version preferred)
- OpenCV

3.2 Methodology

The methodology for a Real-Time Trespasser Detection System involves several key components to ensure effective monitoring and security in restricted areas. The system uses ML-based video surveillance to detect and track intruders in real-time analyzing footage from high-resolution cameras to identify any unauthorized presence. Machine learning algorithms process the video data to distinguish between normal movement and potential security threats enabling the detection of trespassers with high accuracy. The system's ability to analyze video in real-time ensures immediate identification allowing security personnel to take swift action.Upon detecting a trespasser, the system sends real-time alerts to security teams and video feed for verification. The system can also trigger automated security responses, such as activating alarms, to deter further trespassing.

Additionally, the system is scalable allowing for easy expansion by adding more cameras or covering additional areas without compromising performance. The system integrates seamlessly with existing security infrastructure, offering a modern, reliable and efficient solution for trespasser detection and area protection, while addressing issues such as human error and response time delays in traditional security systems.

3.2.1 Algorithm

1. Start
2. Collect images and video frames .
3. Next, apply data augmentation techniques to increase the dataset size and annotate the data for training.
4. Divide the dataset into three parts: training data, testing data, and validation data.
5. Train the model on the training data and evaluate it using the validation data.
6. Capture the real-time video feed from the surveillance camera.
7. Analyze the video feed frame by frame to detect a trespasser.
8. Check the detection score for each frame; if it exceeds a predefined threshold, trigger an alarm.
9. If a trespasser is detected, alert security.
10. If no trespasser is detected, continue analyzing the next frame in the video feed.
11. End

3.2.2 Flowchart

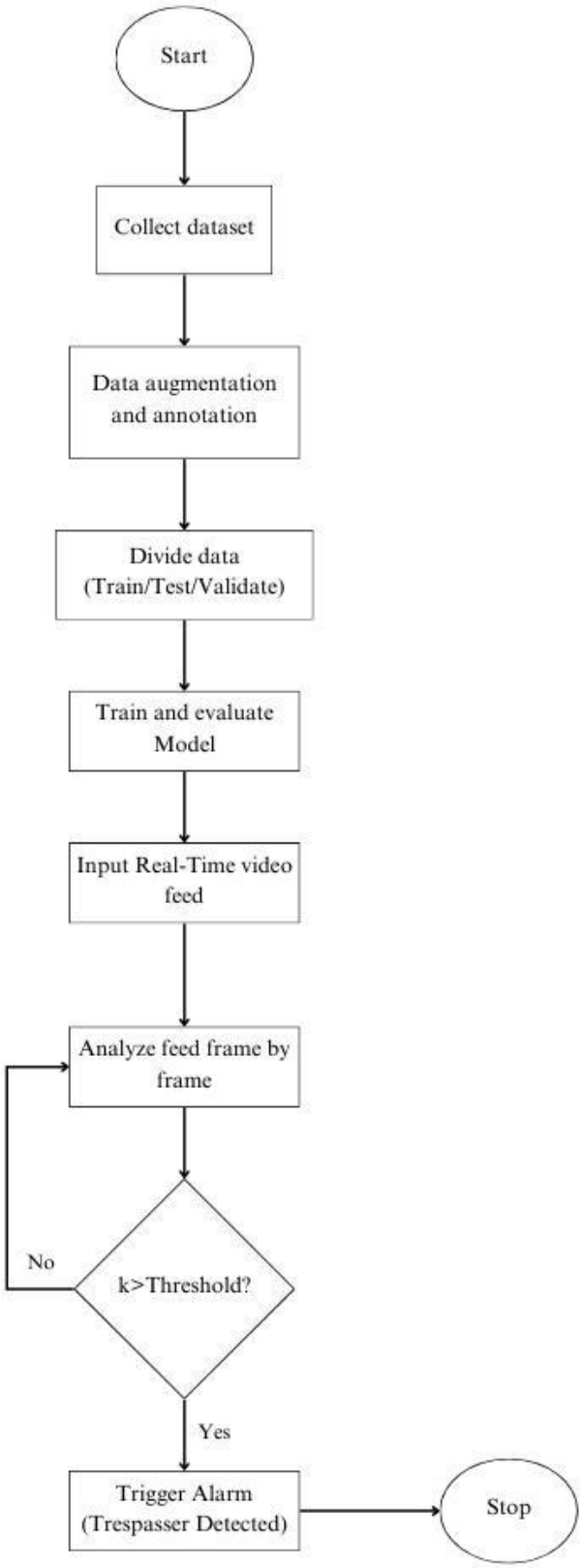

Fig 3.1 Flowchart

CHAPTER 4
DESIGN

4.1 Low Level Design

Low-Level Design is the process of defining the detailed internal structure of a system, focusing on how each component or module will function. It translates the high-level design into a more concrete and implementable set of specifications, including data structures, algorithms, database schema, and specific logic to be used in the development process. Low Level Design focuses on the specifics of how the system's components will interact with each other, how data will be processed and how various tasks will be executed. It provides clear and precise instructions for developers to implement the system efficiently, ensuring the system's components are properly integrated and meet the intended functionality.

4.1.1 SDLC Model

The Software Development Life Cycle (SDLC) is a systematic process that software development teams follow to create software applications. It provides a well-defined approach to guide the development process from initial concept to deployment and maintenance. The SDLC ensures that the final product is of high quality, meets user requirements and is delivered within the set time frame and budget. It encompasses a series of stages, including planning, design, coding, testing, deployment and maintenance, each of which serves a specific purpose. The planning phase involves understanding and defining the requirements, while the design phase focuses on creating the architecture of the software. During the coding phase, developers write the actual code based on the design specifications. The testing phase ensures that the software works as intended and is free of defects. After deployment, the maintenance phase ensures that the software continues to perform well and any issues or bugs are addressed promptly. The SDLC not only provides a structured framework for software development but also helps mitigate risks, improves collaboration among teams, ensures that all aspects of the project are considered and enhances the efficiency of the development process.

4.1.1.1 Iterative Waterfall Model

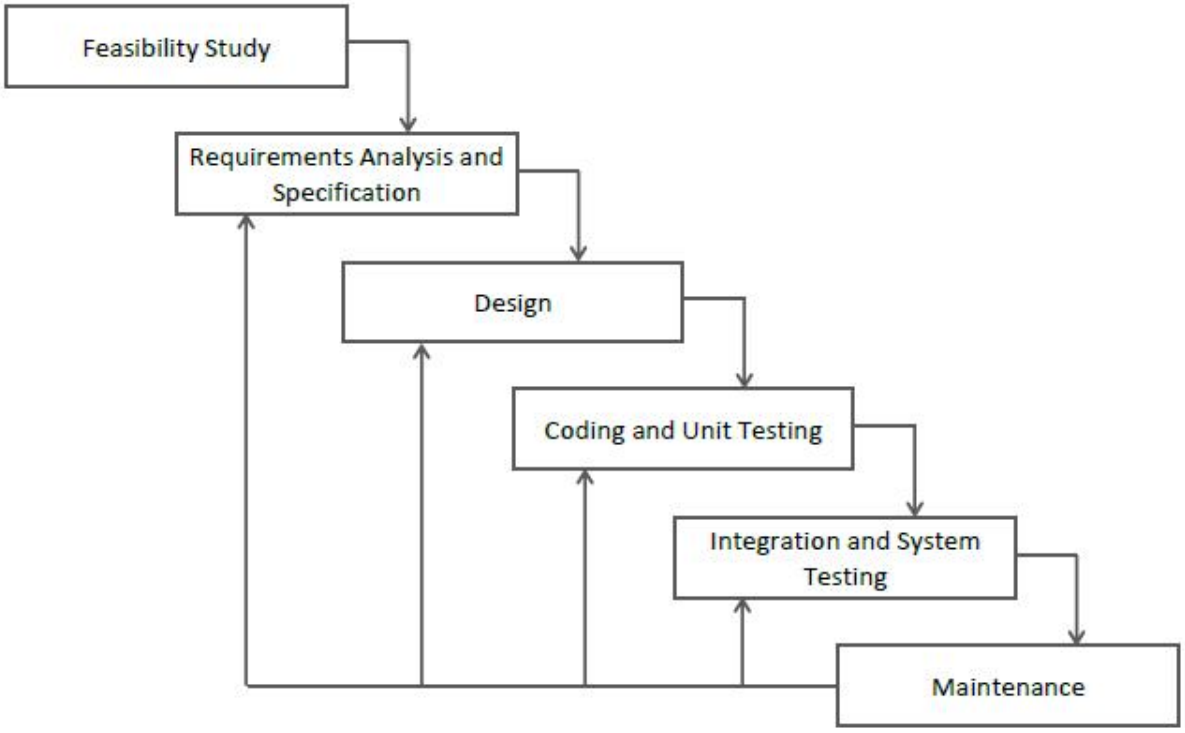

Fig 4.1: Iterative Waterfall Model

The Iterative Waterfall Model consists of the following stages:

1. Feasibility Study:

The Feasibility Study phase is the initial step where the project's viability is assessed. This includes evaluating the technical, operational, and financial feasibility of implementing a real-time trespasser detection system. The study helps determine if the system can be built within the available resources, budget, and time. Key factors like technology choices (e.g., cameras, sensors), compatibility with existing security infrastructure, environmental challenges (e.g., outdoor deployments), and the overall goal of enhancing security are analyzed. The result of this phase is a clear understanding of the project's feasibility and the decision whether to proceed with the next stages or revisit the plan.

2. Requirement Analysis and Specification:

After the feasibility study, the Requirement Analysis and Specification phase begins. In this stage, the stakeholders, including security personnel and system users, define the system's functional and non-functional requirements. Functional requirements could include real-time trespasser detection, the ability to trigger alerts, and integration with existing security infrastructure. Non-functional requirements may include system performance (e.g., real-time alerts), reliability, and scalability. System interfaces are also defined, such as integration with cameras, motion detectors, and alarms. Constraints such as budget, time, and regulatory requirements are identified, which influence the design and implementation of the system. The outcome of this stage is a detailed set of specifications that will guide the design and development phases.

3. Design:

The Design phase involves creating the architecture and system design based on the requirements gathered in the previous stage. During this phase, the development team outlines the technical specifications, including the selection of hardware (e.g., cameras, sensors) and software (e.g., backend systems, databases). The system's overall architecture is broken down into smaller modules, with each component's functionality defined clearly. Design documents are prepared, including data flow diagrams, use case diagram and other relevant artifacts to guide the development process and ensure the system meets the specified requirements.

4. Coding and Unit Testing:

In the Coding and Unit Testing phase, the development team begins coding the real-time trespasser detection system according to the design specifications. This involves implementing various modules, such as the motion detection algorithm, the alert notification system, and the database for storing detected events. Unit testing is performed simultaneously to verify the functionality of each individual component to ensure they perform as expected. Developers write test cases for each function to identify bugs or issues early in the process. This stage ensures that each unit of the system is reliable and free of critical errors before moving on to the next stage.

5. Integration and System Testing:

The Integration and System Testing phase involves combining the individual components or modules into a cohesive system and testing them as a whole. During this stage, the different parts of the real-time trespasser detection system are integrated such as the camera feeds and notification system. System testing is conducted to ensure that the integrated system functions as expected and meets all the requirements outlined in the Requirement Analysis phase. Test cases are run to check the system's overall performance, handling of real-time data and response to detected trespassing events. This phase aims to identify and resolve any integration issues or discrepancies between system components.

6. Maintenance:

The Maintenance phase focuses on the ongoing support and improvement of the real-time trespasser detection system after it is deployed. During this phase, the system is monitored to ensure its continuous operation and reliability in the field. Any issues, such as performance degradation or security breaches, are addressed promptly. Additionally, updates or enhancements may be made to the system based on user feedback, technological advancements, or changes in security protocols. The maintenance phase is crucial to keep the system running smoothly, ensuring that it continues to meet the evolving needs of security personnel and adapts to changing environments or technologies.

4.2 High Level Design

High-Level Design is the process of planning the overall structure of a system or software. It gives a broad view of how the system will work, focusing on the main components and how they will interact. It helps define the key parts of the system, like databases, user interfaces, and backend services, and how they communicate with each other. High-Level Design also considers the tools and technologies needed like programming languages and frameworks. The goal is to create a clear, flexible plan that shows how the system will be built without getting into too many details, ensuring everyone involved understands the big picture. This design serves as a guide for developers, helping them understand how to implement each part of the system. It also allows stakeholders to verify that the system

will meet their requirements. High-Level Design ensures that the system's architecture is scalable, secure, and efficient.

4.2.1 Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a graphical representation of how data flows through a system. It visually maps out the flow of information, illustrating how data is input, processed, stored, and output within a system. DFDs are used to show the relationships between different system components, such as processes, data stores, and external entities (users, other systems, etc.). It helps to identify how data moves between these components and how it is transformed along the way. DFDs are often created at different levels of detail, starting with a high-level diagram that shows the overall system flow and then breaking it down into more specific, detailed diagrams for individual processes. DFDs are helpful in understanding the system's functionality, analyzing its design, and identifying potential improvements.

Level 0

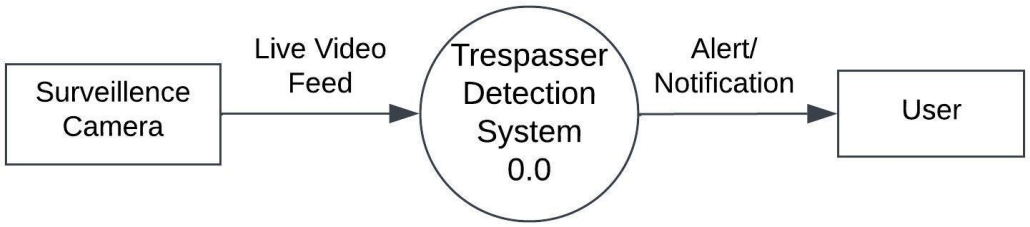

Fig 4.2 Trespasser Detetcion System DFD Level 0

Level 1

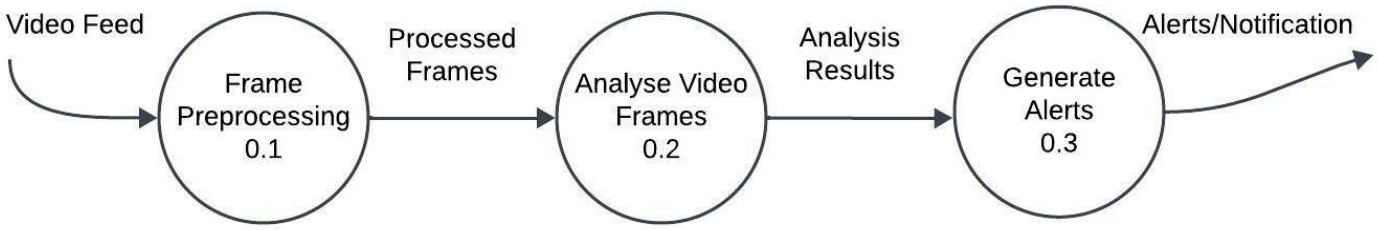

Fig 4.3 Trespasser Detetcion System DFD Level 1

Level 2

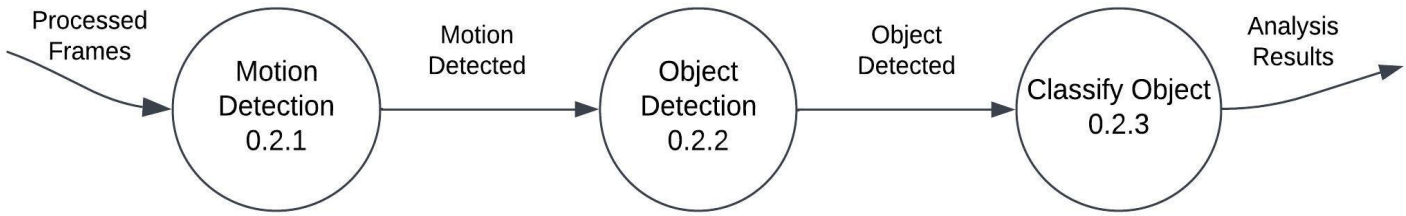

Fig 4.4 Trespasser Detetcion System DFD Level 2

4.2.2 Use Case Diagram

A Use Case Diagram is a visual representation that illustrates the interactions between users (or "actors") and a system, showing the system's functionalities (use cases) from the user's perspective. It helps to capture the requirements and functionalities of a system by depicting how different types of users will interact with it. The diagram typically includes actors, which can be human users or other systems and use cases, which represent specific actions or processes the system will perform in response to user requests. Use Case Diagrams help clarify the system's behavior, identify key requirements, and define the scope of the system.

Actor

Actor in a use case diagram is any external entity that interacts with the system, typically representing a user, another system, or a device. Actors are responsible for initiating or engaging in specific actions or use cases within the system, helping to define how the system will be used and what external roles are involved. An actor in a use case diagram is drawn as a simple stick figure or a rectangle with a label as shown in figure 4.5.

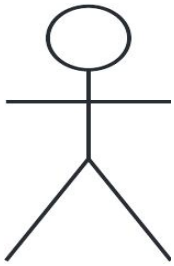

Fig 4.5 Actor

Use Case

A use case in a use case diagram represents a specific functionality or action that a system performs in response to an actor's interaction. It describes the behavior or process the system will carry out to achieve a goal for the actor. A use case is drawn as an oval or ellipse with the name of the functionality written inside it as shown in figure 4.6. The use case is connected to actors with lines to show the relationship between the actor and the use case, indicating which actors interact with which functionalities.

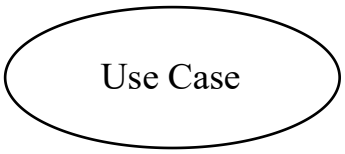

Fig 4.6 Use Case

System

The system in a use case diagram represents the boundary of the software or application being modeled. System is drawn as a rectangle that contains all the use cases (ovals) within it, indicating the scope of the system as shown in figure 4.7. The actors interact with the system's use cases but the system itself is defined by this boundary, which separates the internal functionality (use cases) from the external entities (actors).

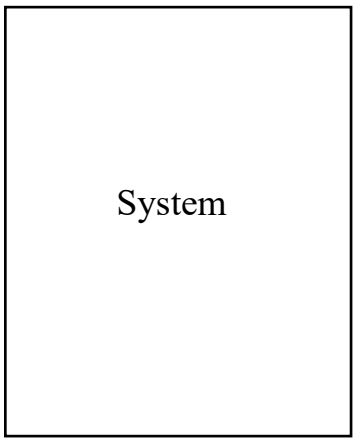

Fig 4.7 System

Here is the use case of our system,

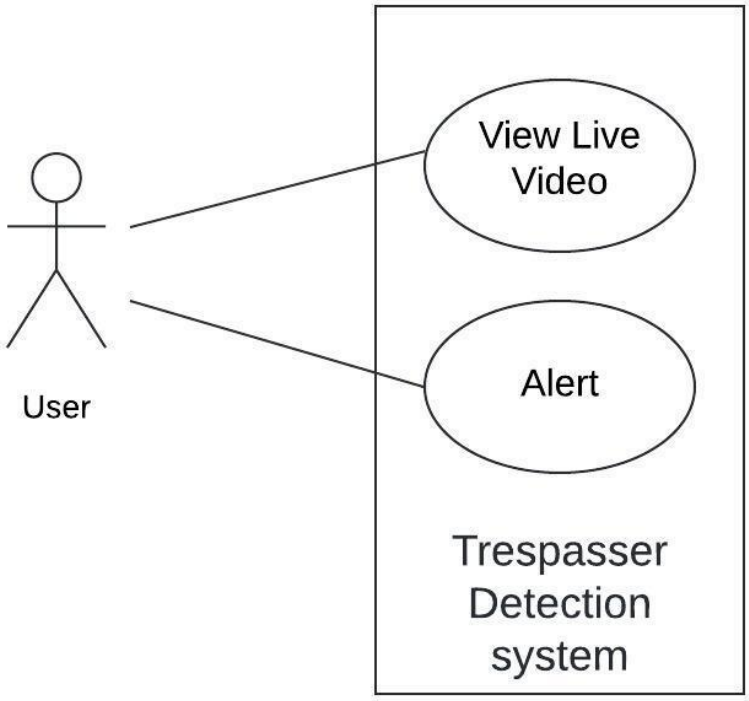

Fig 4.8 Use Case Diagram

Text Description:

I.U1: View Live Video: Using this use case, the user can view live streaming video through the system. The user accesses real-time video feeds, which could be from cameras.

1. Scenario 1: Mainline Sequence

    a. User: Selects the "View Live Video" option on the system's interface.
    b. System: Displays a list of available live video feeds (e.g., cameras or streaming sources).
    c. User: Chooses a specific video feed from the available list.
    d. System: Starts streaming the selected video feed. The live video appears on the screen in real-time.
    e. User: Watches the live video.
    f. System: Continues to display the video feed until the user decides to exit or change the feed.

2. Scenario 2: Alternate Flow (Video Feed Unavailable) Mainline Sequence from step c.

    c. User: Chooses a specific video feed.
    d. System: Attempts to stream the selected video but detects an issue (e.g., camera offline or connection error).
    e. System: Displays an error message: "The video feed is unavailable. Please try again later."
    f. User: Exits the video viewing screen or selects a different feed.

3. Scenario 3: Alternate Flow (Interruption in Video Feed) from Mainline Sequence step c.

    c. User: Chooses a specific video feed.
    d. System: Starts streaming the selected video.
    e. System: Detects an interruption in the video feed (e.g., poor internet connection, technical error).
    f. System: Pauses the video and displays a message: "Video feed interrupted. Please wait while we reconnect."
    g. User: Waits for the feed to resume or exits the video viewing screen.

II.U2: Alert: Using this use case, the user can receive alerts or notifications about important events, such as unauthorized access or security breaches. When the system identifies an trespasser, it triggers an alert and notifies the user.

1. Scenario 1: Mainline Sequence

a. System: Detects a trespasser via surveillance
b. System: Generates an alert for the trespassing event.
c. System: Sends the alert to the user (or security team) via notification
d. User: Receives the alert and reviews the details.
e. User: Takes appropriate action, such as viewing the live video feed or notifying security personnel.

CHAPTER 5
IMPLEMENTATION

CHAPTER 6

TESTING

6.1 Introduction

Testing is a critical phase in the software development lifecycle, aiming to identify and resolve defects within the system. For the Real-Time Trespasser Detection System, testing was performed at various stages to ensure the system's reliability, accuracy, and efficiency. Unit testing was first conducted on individual modules, such as the pre-processing module, detection model, alarm system, video feed handler, and post-processing module. Each module was tested independently to ensure it functioned correctly. After unit testing, integration testing was carried out incrementally, ensuring smooth interaction between the modules and minimizing error propagation. Once the system was fully integrated, system testing was performed to verify the functionality of the entire system. This stage tested the system’s ability to detect trespassers under real-world conditions, assess its performance, and ensure its robustness and accuracy.

6.2 Test Environment

All modules of the “Trespasser Detection System Using Machine Learning” were tested out under the following test environment:

Operating System: Windows 11 with 64bit
Vscode interpreter: Version 3.7 64bit
Integrated Development Environment :
Major Libraries :

- Node.js
- Node Package manager(npm)
- NVidia CUDA version 12.7
-

Hardware Environment:

- Intel core i5
- 16GB RAM
- NVidia GTX 1650
- Micron SSD 512GB 2210 NVMe

6.3 Testing Methodologies

Testing methodologies are systematic approaches used to evaluate and ensure the functionality, reliability, and performance of a system. These methodologies are essential to ensure that a product, such as a real-time trespasser detection system, works as intended and meets its requirements. The most common testing methodologies include unit testing, integration testing, and system testing each serving a specific purpose in the software development and validation lifecycle.

6.3.1 Unit Testing

Unit testing is the process of testing individual components or units of a system in isolation to verify that each part functions correctly. In unit testing, the smallest testable parts of an application, such as individual functions, methods, or classes, are tested independently to ensure they behave as expected. In a trespasser detection system, unit testing involve testing the functionality of individual sensors (like motion detectors or cameras), algorithms used to analyze sensor data, or alerting mechanisms. Unit testing typically involves creating mock objects or stubs to simulate the behavior of dependent components. The goal of unit testing is to catch bugs early in the development process by validating each unit's functionality before moving on to more complex integration scenarios.

6.3.2 Integration Testing

Integration testing focuses on testing how different components or systems interact with each other. It verifies that the individual units, which have been tested in isolation through unit testing, work together as a whole. In integration testing, the goal is to check the interfaces between different modules and ensure that data flows correctly between them.In a trespasser detection system, integration testing verify that the camera correctly activates to capture footage and that the data processing unit analyzes the data without delay. Additionally, integration testing checks that the system triggers alerts appropriately based on the data received from the sensors.

6.3.2 System Testing

System testing is a comprehensive testing methodology that evaluates the entire system's functionality as a whole. It involves testing the complete, integrated system in a controlled

environment to ensure that all components work together as expected and meet the defined requirements. For a trespasser detection system, system testing would simulate real-world scenarios, such as detecting intrusions, processing data, triggering alarms, and notifying security teams in real time. System testing checks that all individual components, as well as their integrations, function correctly within the broader context of the system.

CHAPTER 7
RESULT AND ANALYSIS

CHAPTER 8
CONCLUSION

8.1 Conclusion

In the duration of developing this project, we have successfully designed and implemented a Real-Time Trespasser Detection System aimed at enhancing security through advanced detection and user alert systems. The system integrates multiple components, including video cameras, data processing algorithms, and alert mechanisms, to create a comprehensive and reliable solution for detecting unauthorized intruders in real time. By using machine learning-based image recognition, the system is able to detect and classify potential threats accurately reducing the risk of false positives and improving overall security. Through the various stages of system development, including design, implementation, and testing, we have demonstrated the effectiveness of the system in providing timely alerts and activating automated responses. Furthermore, the system's ability to scale and adapt to different security environments adds to its flexibility and potential for widespread deployment.

In conclusion, the Real-Time Trespasser Detection System advances security technology with proactive, automated threat detection. Its successful implementation paves the way for future enhancements, including additional sensors and improved machine learning, contributing to safer environments for individuals and organizations.

8.2 Future Work

The system can be further improved by exploring the following areas:

- Advanced Detection Capabilities: Incorporating additional sensor types, such as infrared or thermal cameras, to improve detection in low-light or challenging environmental conditions.
- Cloud Integration: Implementing cloud-based analytics and storage to enable remote monitoring and improve scalability, offering flexibility in accessing data and alerts from any location.

CHAPTER 9

REFERENCES

[1] Deshpande, H., Singh, A., & Herunde, H. (2020). Comparative analysis on YOLO object detection with OpenCV. *International journal of research in industrial engineering*, 9(1), 46-64.
[2] Arumugam,K.,& Rajgure,Y.(2021). Railway Trespassing Detection and Alert System using Deep Learning, CNN, YOLO. *International Research Journal of Engineering and Technology*.
[3] Varma, B., & Reddy, T. (2022). SMART CCTV CAMERA MONITORING SYSTEM USING IOT. International Research Journal of Modernization in Engineering Technology and Science.
[4] Rahmaniar, W., & Hernawan, A. (2021). Real-time human detection using deep learning on embedded platforms: A review. Journal of Robotics and Control (JRC), 2(6), 462-468.
[5] Ma, B. (2020, October). Design of intelligent burglar alarm system in laboratory based on embedded system. In 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI) (pp. 392-395). IEEE.
[6] Goel, P. (2021). Realtime object detection using tensorflow an application of ml. International Journal of Sustainable Development in Computing Science, 3(3), 11-20.
[7] Keat, L. H., & Wen, C. C. (2018). Smart indoor home surveillance monitoring system using Raspberry Pi. JOIV: International Journal on Informatics Visualization, 2(4-2), 299-308.
[8] Schwartz, W. R., Kembhavi, A., Harwood, D., & Davis, L. S. (2009, September). Human detection using partial least squares analysis. In 2009 IEEE 12th international conference on computer vision (pp. 24-31). IEEE.
[9] Aboyomi, D. D., & Daniel, C. (2023). A Comparative Analysis of Modern Object Detection Algorithms: YOLO vs. SSD vs. Faster R-CNN. ITEJ (Information Technology Engineering Journals), 8(2), 96-106.
[10] Parveen, S., & Shah, J. (2021, February). A motion detection system in python and opencv. In 2021 third international conference on intelligent communication technologies and virtual mobile networks (ICICV) (pp. 1378-1382). IEEE.
[11] Singla, N. (2014). Motion detection based on frame difference method. International Journal of Information & Computation Technology, 4(15), 1559-1565.
[12] Iqbal, M. J., Iqbal, M. M., Ahmad, I., Alassafi, M. O., Alfakeeh, A. S., & Alhomoud, A. (2021). Real-Time Surveillance Using Deep Learning. Security and Communication Networks, 2021(1), 6184756.
[13] Redmon, J. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition.

[14] Bathija, A., & Sharma, G. (2019). Visual object detection and tracking using yolo and sort. International Journal of Engineering Research Technology, 8(11), 345-355.

Table of Contents