UNIVERSITY OF BUEA

P.O. Box 63,
Buea, South West Region
CAMEROON
Tel: (237) 3332 21 34/3332 26 90
Fax: (237) 3332 22 72

REPUBLIC OF CAMEROON
PEACE WORK-FATHERLAND

FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

VISUAL OBJECT TRACKING FOR VIDEO SURVEILLANCE

A dissertation submitted to the Department of Computer Engineering, Faculty of Engineering and Technology, University of Buea, in Partial Fulfillment of the Requirements for the Award of Bachelor of Engineering (B.Eng.) Degree in Computer Engineering

By:

LEGIMA DONAL
Matriculation Number: FE20A055
Option: Software Engineering

Supervisor:
Dr.SOP DEFFO LIONEL
University of Buea

2023/2024 Academic Year

VISUAL OBJECT TRACKING FOR VIDEO SURVEILLANCE

BY:

LEGIMA DONAL

Matriculation Number: FE20A055

Academic Year: 2023/2024

Dissertation submitted in partial fulfillment of the Requirements for the award of

Bachelor of Engineering (B.Eng.) Degree in Computer Engineering.

Department of Computer Engineering

Faculty of Engineering and Technology

University of Buea

Certification of Originality

We the undersigned, hereby certify that this dissertation entitled "**VISUAL OBJECT TRACKING FOR VIDEO SURVEILLANCE**" presented by **LEGIMA DONAL**, Matriculation number, **FE20A055** has been carried out by him/her in the Department of Computer Engineering, Faculty of Engineering and Technology, University of Buea under the supervision of **Dr. SOP DEFFO LIONEL.**

This dissertation is authentic and represents the fruits of his/her own research and efforts.

Date_____

Student

_____

Supervisor

_____

Head of Departments

_____

.

DEDICATION

I dedicate this project to the Dinga family.

ACKNOWLEDGMENT

Our profound and sincere gratitude goes to the following:

- I want to thank God for giving me that resilience to keep pushing milestone after milestone through the various challenges encountered throughout this journey.
- Our supervisor Dr. SOP DEFFO LIONEL for his support and expertise.
- To the Dean of the Faculty of Engineering and Technology in the University of BUEA Professor AGBOR DIEUDONNE AGBOR and Dr. Elie Fute, Head of the Department of Electrical and Electronic Engineering.
- My parents and guardians miss Yiambu Yvette and Mrs. Joy Yuh and Mrs Yiambu Veronica
- Innovation my team since year 1 for their endless efforts invested towards the completion of this work.
- We are extremely grateful to our sisters and friends who helped us in one way or the order. Alongside with all the people who worked with us for their patience and willingness to give us a helping hand.

ABSTRACT

Visual object tracking is a crucial component in modern video surveillance systems, enabling the continuous monitoring and analysis of moving objects within a scene. This report explores advanced methodologies in visual object tracking, emphasizing the importance of robust and efficient tracking algorithms to ensure high accuracy and real-time performance and we focus on detecting dangerous objects through video feeds. Key technologies such as deep learning, feature extraction, and motion prediction are discussed, alongside their applications in enhancing surveillance capabilities. The integration of these technologies not only improves object detection and tracking reliability but also reduces false alarms and enhances the overall security infrastructure. Experimental results and case studies demonstrate the effectiveness of the proposed approaches in various surveillance scenarios, highlighting significant improvements over traditional methods.

Keyword: Visual object tracking, video surveillance, deep learning, feature extraction, motion prediction, real-time performance, object detection, security infrastructure.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACC - Avigilon Control Center

AI - Artificial Intelligence

CCTV - Close Circuit Television

CSRT - Discriminative Correlation Filter with Channel and Spatial Reliability

CV - Computer Vision

FPS - Frames Per Second

GPU - Graphics Processing Unit

GSC - Genetic Security Center

GUI - Graphic User Interface

IoT - Internet of Things

ISS - Intelligent Security Systems

IUT - Implementation Under Test

ML - Machine Learning

NMS - Non-Maximum Suppression

VMS - Virtual Management software

YOLO - You Look Only Once

CHAPTER ONE: GENERAL INTRODUCTION

1.1 Background and Context of the Study:

Modern security systems now include video surveillance as a fundamental component because it provides real-time monitoring and analytic capabilities that are essential for a variety of applications, including traffic management, retail security, and law enforcement. In this field, visual object tracking is essential for locating, recognizing, and keeping an eye on items or people of interest in a video stream. Over the years, video surveillance technology has evolved significantly, transitioning from traditional analog systems to sophisticated digital setups. This evolution has been driven by advancements in camera technology, image processing algorithms, and the availability of powerful computing resources.

Fig. 1.1: Real-time event detection for video surveillance uploaded by Mahdi Rezaei, Oct 2020

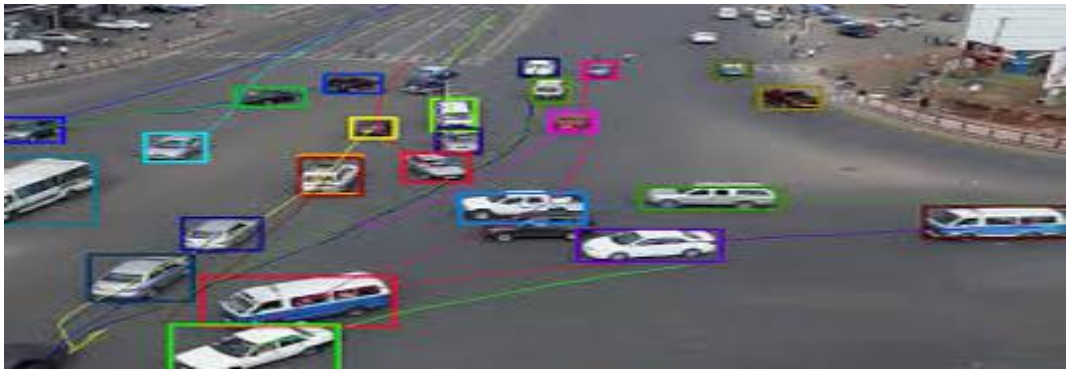


Fig. 1.2: vehicle tracking,uploaded by Kemal Gökhan Nalbant

system, Dec 2021.

Fig 1 and 2 above clearly demonstrate real world applications of object tracking on video surveillance. On fig 1 we can clearly see how pedestrians' motions are carefully tracked and fig 2 shows us how vehicle movements are being monitored using object tracking.

1.2 Problem statement:

Systems for video surveillance are essential for keeping an eye on and guaranteeing the safety of a variety of locations, such as public areas, transit hubs, and business buildings. Threat identification, behavior analysis, and incident response are made easier by these systems' precise and timely tracking of items or people of interest inside a video feed. However, visual object tracking in complicated real-world circumstances continues to be a difficult issue, even with advances in camera technology and image processing techniques. A vital feature of video surveillance systems is visual object tracking, which allows for the ongoing observation and examination of objects or people in a video stream. However, maintaining accurate and robust tracking performance is a challenge for present tracking algorithms, especially in complicated real-world situations with occlusions, shifting illumination, scale fluctuations, cluttered backdrops, and deformed objects. In order to improve tracking accuracy, resilience, and real-time performance in difficult settings, the main goal of this work is to design and analyze sophisticated visual object tracking algorithms especially suited for video surveillance applications.

1.3 Objectives of study:

Here, we will talk about the general objectives for building such a system and also talk about the specific objectives for such systems.

1.3.1 General Objectives

To detect and track objects through live videos.

1.3.2 Specific Objectives

- To detect dangerous objects such as guns, knives, etc., via surveillance cameras and provide a signal, potentially improving security in a certain region or area.
- To develop a system that will provide email notifications for detected objects so the user can still know a dangerous object was detected even when not facing the screen.
- To track the detected dangerous objects to follow their path.
- To trigger an alarm system when a possible dangerous object has been detected.

1.4 Proposed Methodology:

1. Problem Understanding and Dataset Acquisition:

- Understand challenges and requirements of visual object tracking in video surveillance.
- Acquire relevant video datasets with annotated ground truth object trajectories.

2. Literature Review:

- Review existing literature on state-of-the-art visual object tracking algorithms and techniques.
- Identify key methodologies, algorithms, and performance metrics.

3. Algorithm Development:

- Design and implement novel tracking algorithms for video surveillance.
- Experiment with techniques like feature extraction, motion estimation, object representation, and various tracking algorithms.

4. Evaluation Setup:

- Define evaluation metrics (accuracy, robustness, computational efficiency, scalability).
- Establish experimental protocols for performance evaluation (training/testing splits, cross-validation, benchmark datasets).

5. Baseline Comparison:

- Implement and evaluate baseline tracking algorithms from the literature.
- Benchmark proposed algorithms against these baselines.

6. Algorithm Training and Optimization:

- Train and fine-tune tracking algorithms using annotated training data.
- Optimize parameters through experimentation and validation on datasets.

7. Experimental Evaluation:

- Evaluate performance using benchmark datasets and real-world surveillance footage.
- Measure tracking accuracy, robustness, computational efficiency, and scalability.

8. Result Analysis and Interpretation:

- Analyze experimental results to assess algorithm strengths and weaknesses.
- Identify factors affecting tracking performance.

9. Comparison with Baselines and State-of-the-Art:

- Compare developed algorithms against baseline and state-of-the-art methods.

- Highlight advantages and limitations of proposed algorithms.

10. Discussion and Conclusion:

- Summarize findings and discuss implications for video surveillance.
- Provide recommendations for future research and potential improvements.

This proposed methodology provides a structured approach for developing, evaluating, and comparing our visual object tracking algorithms tailored specifically for video surveillance applications.

1.5 Significance of Study:

- **Enhanced Security and Safety:** Improves security in public spaces, transportation, and critical infrastructure by enabling timely detection and response to threats.
- **Crime Prevention and Law Enforcement:** Aids law enforcement in identifying and apprehending suspects, enhancing crime prevention and evidence gathering.
- **Public Safety and Emergency Response:** Monitors crowd movements, traffic, and emergencies, facilitating swift response and evacuation during incidents.
- **Efficient Traffic Management:** Supports traffic flow management, congestion mitigation, and accident prevention through real-time monitoring.
- **Retail and Business Intelligence:** Analyzes customer behavior in retail, optimizing store layouts, and preventing theft, providing valuable business insights.
- **Forensic Analysis and Evidence Gathering:** Assists forensic investigators in reconstructing events and analyzing video evidence for criminal investigations.
- **Search and Rescue Operations:** Helps locate missing persons or stranded individuals in search and rescue missions by tracking movements in difficult terrains.
- **Surveillance System Optimization:** Enhances the reliability, scalability, and efficiency of surveillance systems, reducing maintenance costs.
- **Privacy Protection and Ethical Considerations:** Balances security needs with privacy rights through privacy-preserving tracking methods, fostering trust in surveillance technologies.

- **Technological Innovation and Industry Advancement:** Drives innovation in computer vision, machine learning, and AI, contributing to new algorithms and solutions, fueling industry growth.

This study contributes to improved security, crime prevention, public safety, business intelligence, forensic analysis, search and rescue, system optimization, privacy protection, and technological advancement.

1.6 Scope of the Study:

Algorithm Development:

- Creating new tracking algorithms for video surveillance.
- Using traditional, deep learning, and hybrid methods.

Challenging Scenarios:

- Handling issues like occlusions, lighting changes, scale variations, cluttered backgrounds, and object deformations.

Real-World Applicability:

- Ensuring algorithms work in various environments (indoor, outdoor) and conditions (different lighting, diverse objects).

Multi-Object Tracking:

- Tracking multiple objects at once, considering interactions and occlusions.

Hardware Constraints:

- Optimizing for limited hardware resources to ensure real-time processing.

Evaluation Metrics:

- Assessing performance with metrics like accuracy, robustness, efficiency, and scalability.

Integration with Surveillance Systems:

- Ensuring compatibility with existing systems and standard protocols.

Cross-Domain Generalization:

- Ensuring algorithms work across different scenarios, camera setups, and object types.

Human-in-the-Loop Systems:

- Incorporating human feedback to improve tracking accuracy and resolve challenging situations.

1.7 Delimitation of the Study:

Here are some of the Delimitations regarding our project:

- Focus on Tracking Algorithms: The study primarily focuses on the development and evaluation of visual object tracking algorithms for video surveillance applications. It does not delve into other aspects of video surveillance systems, such as camera hardware, video compression, or network infrastructure.
- Limited to Visual Tracking: The study specifically addresses visual object tracking using video data as input. It does not cover other tracking modalities, such as audio-based tracking, RFID-based tracking, or GPS-based tracking.
- Scope of Surveillance Environments: The study considers a range of surveillance environments, including indoor and outdoor settings, but may not cover specialized domains such as underwater surveillance, aerial surveillance, or space surveillance.
- Hardware and Resource Constraints: While the study considers the computational and memory constraints of hardware commonly used in surveillance systems, it does not specifically address hardware optimization techniques or hardware-specific implementation details.
- Privacy and Ethical Considerations: While privacy concerns and ethical considerations associated with video surveillance are acknowledged, the study focuses primarily on technical aspects of object tracking and does not provide in-depth analysis or solutions for privacy protection.
- Validation on Specific Datasets: The evaluation of tracking algorithms may be limited to specific benchmark datasets and real-world surveillance footage available for experimentation, which may not fully capture the diversity of surveillance scenarios encountered in practice.
- Human-in-the-Loop Integration: While the study acknowledges the potential benefits of integrating human feedback into tracking systems, it does not explore the design and implementation of interactive or human-in-the-loop tracking systems in depth.
- Cross-Domain Generalization: While efforts are made to ensure the generalization of tracking algorithms across different surveillance scenarios, camera setups, and object

types, the study may not cover all possible variations and challenges encountered in diverse environments.

1.8 Definition of Keywords and Terms:

These are the various keywords and terms regarding visual object tracking

- Visual Object Tracking: The process of automatically locating and following objects of interest (e.g., people, vehicles) within a video sequence over time, typically performed using computer vision techniques and algorithms.
- Video Surveillance: The use of video cameras and related technology for monitoring and recording activities in a specific area or location, commonly used for security, safety, and surveillance purposes.
- Object Detection: The process of identifying and localizing instances of objects within an image or video frame, typically performed using object detection algorithms to generate bounding boxes around objects of interest.
- Object Recognition: The process of identifying and classifying objects within an image or video frame, typically performed using object recognition algorithms trained on labeled datasets to assign semantic labels to detected objects.
- Object Tracking: The process of following the movement and trajectory of objects over time within a video sequence, typically performed using tracking algorithms that estimate the state of objects and predict their future positions.
- Feature Extraction: The process of extracting relevant visual features (e.g., edges, corners, textures) from images or video frames, typically used as input to object detection, recognition, and tracking algorithms.
- Motion Estimation: The process of estimating the motion vectors of objects between consecutive frames in a video sequence, typically used in object tracking algorithms to predict object movements and update object trajectories.
- Occlusion: The partial or complete obstruction of an object by another object or obstacle within the scene, often leading to temporary loss of visibility and challenges for object tracking algorithms.

- Scale Variation: Changes in the size or scale of objects as they move closer to or farther away from the camera, posing challenges for object tracking algorithms to maintain accurate tracking across different scales.
- Cluttered Background: Complex and visually rich backgrounds containing multiple objects and textures, often leading to distractions and false positives in object detection and tracking algorithms.
- Illumination Changes: Variations in lighting conditions within a scene, including changes in brightness, shadows, and glare, which can affect the visibility and detectability of objects in video surveillance footage.
- Object Deformation: Changes in the shape or appearance of objects due to non-rigid motion, perspective changes, or physical interactions, posing challenges for object tracking algorithms to maintain accurate object representations.
- Robustness: The ability of tracking algorithms to maintain accurate and reliable tracking performance in the presence of various challenges and uncertainties, such as occlusions, scale variations, and cluttered backgrounds.
- Real-Time Performance: The ability of tracking algorithms to process video frames and perform object tracking tasks in real-time, typically measured in terms of processing speed and latency.
- Accuracy: The degree of closeness between the estimated object trajectories produced by tracking algorithms and the ground truth object trajectories annotated in the video surveillance footage, typically measured using evaluation metrics such as tracking error or overlap ratio.

These definitions provide a foundational understanding of key concepts and terminology relevant to Visual Object Tracking for Video Surveillance.

1.9 Organization of the Dissertation

1. Title Page
    - Title of the Dissertation
    - Name of the Author

- Institutional Affiliation
- Date

2. Abstract
    - A concise summary of the dissertation, highlighting the research objectives, methodology, key findings, and contributions.
3. Acknowledgments
    - Recognition of individuals or organizations who contributed to the research and dissertation writing process.
4. Table of Contents
    - A list of chapters, sections, and subsections with corresponding page numbers for easy navigation.
5. List of Figures and Tables
    - A compilation of all figures and tables included in the dissertation, along with their corresponding page numbers.
6. List of Abbreviations and Symbols
    - A list of abbreviations, acronyms, and symbols used throughout the dissertation, along with their explanations.
7. Chapter 1: Introduction
    - Introduction to the research topic
    - Background and motivation for the study
    - Research objectives and scope
    - Significance of the study
    - Overview of the dissertation structure
8. Chapter 2: Literature Review
    - Review of relevant literature and previous research in the field of visual object tracking for video surveillance
    - Discussion of key concepts, methodologies, algorithms, and evaluation metrics
    - Identification of gaps and limitations in existing research
9. Chapter 3: Methodology
    - Description of the research methodology employed in the study
    - Explanation of data collection and preprocessing procedures

- Details of tracking algorithms, techniques, and implementation
- Experimental design and evaluation methodology

10. Chapter 4: Experimental Results
    - Presentation and analysis of experimental results
    - Evaluation of tracking algorithms using benchmark datasets and real-world surveillance footage
    - Discussion of findings, including tracking accuracy, robustness, and computational performance
11. Chapter 5: Discussion
    - Interpretation and discussion of the results in relation to the research objectives
    - Comparison of findings with previous research studies
    - Identification of strengths, weaknesses, and implications of the research
12. Chapter 6: Conclusion and Future Work
    - Summary of key findings and contributions
    - Implications of the research for theory, practice, and future research directions
    - Suggestions for further exploration and improvement in visual object tracking for video surveillance
13. References
    - List of all cited references and sources used in the dissertation, formatted according to the chosen citation style (e.g., APA, IEEE)
14. Appendices
    - Additional supplementary materials, such as detailed experimental setups, code snippets, dataset descriptions, or additional results

This organization provides a structured framework for presenting research findings and insights on Visual Object Tracking for Video Surveillance in a comprehensive and coherent manner.

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction:

The term "surveillance" Numerous technologies are available, including thermal imaging cameras, CCD cameras, and goggles for night vision. The field of video surveillance makes use of these gadgets. Large-scale data sets are mined for information by an intelligent visual surveillance system. A visual surveillance system with several cameras aids in object detection and tracking to understand object behavior.

is the fusion of two words: "veiller," which means "to watch," and "sur," which means "from above." The study of artificial intelligence, computer vision, and digital image processing is particularly interested in the topic of video surveillance. 1930–1940 saw the usage of tube cameras as a monitoring device. The Close-Circuit Television (CCTV) was a costly and malfunctioning visual display device for watching happenings in the scenes.

Therefore, human motion is significant in the field of machine learning and picture categorization that centers on pattern recognition .

Because humans can assume a wide range of stances and have diverse features, it can be difficult to detect them in surveillance footage. For human recognition in video sequences, a variety of biometric features, including face and gait, as well as non-biometric features, like look, can be employed.

The components of an automated visual surveillance system include human identification, object tracking, and object detection. The crucial first stage in any video surveillance system is motion and object detection. Single-camera visual surveillance systems can be used for person identification, tracking, and detection.

The issue of occlusion arises when the surveillance area is increased. Multiple camera visual surveillance systems can be useful in solving this issue. Due to occlusion, using a single camera for object tracking leads to ambiguity issues. It may be possible to solve this ambiguity issue from a different camera angle.

Many issues arise when we employ many cameras, including object matching and camera calibration. There are three generations of video surveillance systems, which are as follows:

1GSS: The actions of retrieving, transmitting, and processing images were associated with the first generation of surveillance systems.
However, there were certain drawbacks to this generation, such as excessive bandwidth and the extraction of system events.
2GSS: Analog and digital subsystems served as the foundation for the second generation of surveillance systems. These systems are used to handle bandwidth issues, false event removal, and other related issues.
3GSS: Systems of this generation offer complete digital systems. The 3GSS employs a smart system that produces alarms in real time for system occurrences.
Computer vision algorithms that are efficient and operate in real-time are integrated into video surveillance systems. There are several different kinds of CCTV systems.

The main purposes of visual surveillance systems are behavior analysis and explanation of objects. It includes video tracking to comprehend the happenings in the scene, as well as the recognition of both stationary and moving objects. The primary goal is to identify several approaches for tracking moving items and detecting both static and moving objects. Objects in any video scene can be identified using object detection techniques. Detected items include trees, clouds, people, and other moving things, among other classifications. One of the biggest challenges for any video surveillance system is object detection as it moves. In higher level applications, object tracking is utilized to determine the location of accessible items and the shape of each object in each frame.

2.2 General Concepts in Visual Object Tracking:

Some of the various general concepts we have regarding visual object tracking are as follows.

- Object Representation: Visual object tracking involves representing objects of interest within a video sequence using visual features such as color histograms, texture descriptors, or deep features extracted from convolutional neural networks (CNNs). Object representations play a crucial role in matching and locating objects across consecutive video frames.

- Motion Estimation: Motion estimation is the process of estimating the motion vectors of objects between consecutive frames in a video sequence. It enables the prediction of object movements and the generation of motion trajectories, which are essential for object tracking algorithms to maintain accurate object localization over time.
- Feature Matching: Feature matching involves comparing and matching visual features extracted from objects in consecutive video frames to establish correspondence and continuity between object appearances. Common techniques include template matching, correlation-based methods, and feature point matching using techniques like SIFT (Scale-Invariant Feature Transform) or SURF (Speeded-Up Robust Features).
- State Estimation: State estimation refers to the process of estimating the current state or position of objects within a video sequence based on available sensory data. It typically involves probabilistic modeling and estimation techniques, such as Kalman filtering, particle filtering (e.g., the Sequential Monte Carlo method), or Bayesian inference, to predict and update object states over time.
- Object Tracking Algorithms: Object tracking algorithms encompass a wide range of techniques and methodologies for locating and following objects of interest within a video sequence. These algorithms may be classified into traditional feature-based methods, correlation filters, deep learning-based approaches, and hybrid techniques that combine multiple methodologies to achieve robust and accurate tracking performance.
- Tracking Evaluation Metrics: Tracking algorithms are evaluated based on various performance metrics, including tracking accuracy, robustness, computational efficiency, and scalability. Tracking accuracy measures the degree of closeness between the estimated object trajectories and the ground truth annotations, while robustness assesses the algorithm's ability to maintain accurate tracking performance in challenging scenarios.
- Challenges in Visual Object Tracking: Visual object tracking faces numerous challenges in video surveillance scenarios, including occlusions, illumination changes, scale variations, cluttered backgrounds, object deformations, and motion blur. Addressing these challenges requires the development of robust tracking algorithms capable of handling diverse environmental conditions and object dynamics.
- Real-Time Performance: Real-time performance is a critical requirement for visual object tracking algorithms deployed in video surveillance systems, ensuring timely detection

and response to security threats or events of interest. Algorithms must be optimized for efficient resource utilization and low-latency processing to achieve real-time tracking performance.

- Applications of Visual Object Tracking: Visual object tracking has numerous applications in video surveillance, including security and safety monitoring, traffic management, retail analytics, crowd analysis, human-computer interaction, augmented reality, and robotics. These applications leverage tracking algorithms to analyze object movements, behaviors, and interactions within video feeds for various purposes.
- Privacy and Ethical Considerations: Visual object tracking raises important privacy and ethical considerations related to the collection, storage, and analysis of video surveillance data. It is essential to implement privacy-preserving techniques and adhere to ethical guidelines to protect individuals' rights and ensure responsible use of surveillance technology.

2.3 Related works:

After researching on the topic and looking for corresponding systems, I stumbled across applications built for visual object tracking listed below.These visual tracking software applications offer advanced features and capabilities for monitoring, tracking, and analyzing objects in video surveillance footage. They are used in a wide range of industries and applications, including security, transportation, retail, and critical infrastructure protection.

2.3.1 Viseum Intelligent CCTV:

Viseum Intelligent CCTV[1] is a comprehensive video surveillance solution that includes advanced visual tracking capabilities. It uses intelligent algorithms to track and monitor objects in real-time, providing situational awareness and security insights for various applications, including public safety, transportation, and critical infrastructure protection.

Fig. 2.1: Various types of CCTV's surveillance cameras

Fig 2.1 shows the various kinds of surveillance cameras they possess. All systems come with pros and cons and below we will outline most of them.

Pros:

- Advanced Visual Tracking Capabilities: Viseum Intelligent CCTV is known for its advanced visual tracking capabilities, leveraging intelligent algorithms to track and monitor objects in real-time. This enables enhanced situational awareness and security insights for various applications.
- Comprehensive Surveillance Solution: Viseum offers a comprehensive CCTV solution that includes not only visual tracking but also other security features such as video analytics, event detection, and centralized management. This integrated approach provides organizations with a complete surveillance solution for their security needs.
- Situational Awareness: The advanced tracking algorithms used in Viseum Intelligent CCTV help improve situational awareness by continuously monitoring and analyzing the movements of objects within the surveillance area. This can aid in the early detection of security threats or suspicious activities.
- Scalability and Customization: Viseum CCTV systems are designed to be scalable and customizable, allowing organizations to expand their surveillance infrastructure as needed and tailor the system to their specific requirements. This flexibility makes it suitable for deployments in a wide range of environments.
- Reliability and Performance: Viseum Intelligent CCTV systems are engineered for reliability and performance, with robust hardware and software components that ensure continuous operation and efficient processing of video data. This reliability is essential for mission-critical surveillance applications.

Cons:

- Cost: One potential downside of Viseum Intelligent CCTV systems is the cost, as they may involve significant upfront investment in hardware, software licenses, and installation services. The advanced features and capabilities of Viseum systems can contribute to higher overall costs compared to simpler CCTV solutions.
- Complexity: The advanced features and capabilities of Viseum Intelligent CCTV systems may introduce complexity in terms of setup, configuration, and maintenance. Organizations may require specialized training or expertise to effectively deploy and manage these systems.
- Integration Challenges: Integrating Viseum CCTV systems with existing security infrastructure or third-party systems may pose challenges due to compatibility issues or customization requirements. Ensuring seamless integration with other security systems and workflows may require additional effort and resources.
- Dependency on Technology: Like any technology-driven solution, Viseum Intelligent CCTV systems are dependent on the reliability and performance of underlying hardware and software components. Any issues or failures in these components could impact the effectiveness of the surveillance system.
- Privacy Concerns: The advanced tracking capabilities of Viseum Intelligent CCTV systems raise potential privacy concerns, particularly regarding the collection and analysis of video data. Organizations deploying these systems must ensure compliance with applicable privacy regulations and implement appropriate safeguards to protect individuals' privacy rights.

2.3.2 Avigilon Control Center (ACC):

Avigilon Control Center (ACC)[2] is a video management software (VMS) platform that offers advanced video analytics and object tracking features. It provides real-time monitoring, event detection, and forensic search capabilities, making it suitable for security and surveillance applications in commercial, industrial, and government sectors.

Fig. 2.2 Human recognition system

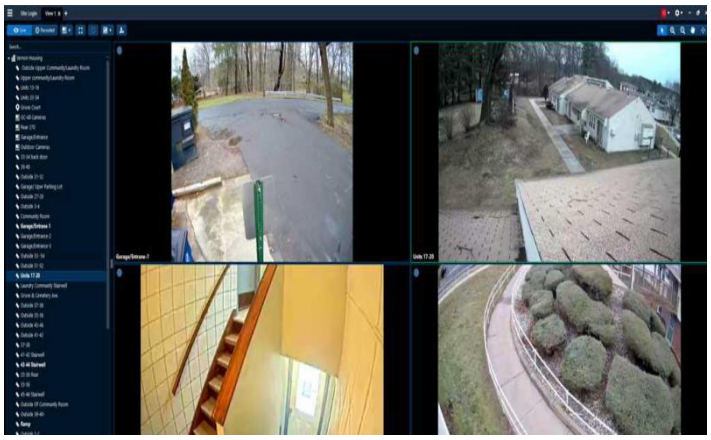


Fig 2.3 Avigilon Video Management System

Fig 2.2 and 2.3 shows the ACC's interface and how it performs object detection and recognition as in the case of fig 2.2 which is facial recognition.

Having such a system has advantages and disadvantages which will be listed as follows.

Pros:

- Advanced Video Analytics: Avigilon Control Center (ACC) offers advanced video analytics capabilities, including object detection, facial recognition, license plate recognition (LPR), and behavioral analytics. These analytics features enable users to quickly search, identify, and analyze specific events or objects within video footage.
- High-Definition Video Quality: ACC supports high-definition video quality, allowing users to capture and view video footage in crystal-clear detail. The platform offers support for high-resolution cameras and provides advanced video compression techniques to optimize storage and bandwidth usage.
- Scalability and Flexibility: ACC is designed to be scalable and flexible, making it suitable for deployments of all sizes, from small businesses to large enterprises. The platform supports a wide range of cameras, sensors, and third-party integrations, allowing users to customize their surveillance infrastructure to meet their specific needs.
- Intuitive User Interface: The user interface of ACC is user-friendly and intuitive, with customizable layouts and easy-to-use controls. This makes it easy for operators to navigate through video footage, manage cameras, and access advanced features without extensive training or technical expertise.

- Centralized Management: ACC provides centralized management capabilities, allowing users to monitor and control multiple cameras and sites from a single interface. This centralized approach simplifies system administration, improves operational efficiency, and enhances situational awareness for security personnel.

Cons:

- Cost: One potential downside of Avigilon Control Center is the cost, as it may involve significant upfront investment in software licenses, hardware infrastructure, and installation services. The advanced features and capabilities of ACC can contribute to higher overall costs compared to simpler surveillance solutions.
- Complexity: The advanced features and capabilities of ACC may introduce complexity in terms of setup, configuration, and maintenance. Organizations may require specialized training or expertise to effectively deploy and manage the platform, especially for larger deployments with multiple sites or integrations.
- Dependency on Avigilon Ecosystem: ACC is tightly integrated with the Avigilon ecosystem, including Avigilon cameras, sensors, and software modules. While this integration offers seamless interoperability and enhanced functionality, it also creates a dependency on Avigilon products and limits compatibility with third-party systems.
- Bandwidth and Storage Requirements: High-definition video quality and advanced analytics features of ACC can result in increased bandwidth and storage requirements, especially in large-scale deployments with multiple cameras and high-resolution footage. Organizations need to carefully plan and manage their network infrastructure and storage capacity to accommodate these requirements.
- Privacy and Compliance: The advanced video analytics capabilities of ACC raise potential privacy concerns, particularly regarding the collection and analysis of sensitive information such as facial images and license plate numbers. Organizations deploying ACC must ensure compliance with applicable privacy regulations and implement appropriate safeguards to protect individuals' privacy rights.

2.3.3 Genetec Security Center:

Genetec Security Center[3] is a unified security platform that combines video surveillance, access control, and license plate recognition (LPR) capabilities. It offers advanced object tracking features, alarm management, and reporting tools for security operations in enterprises, cities, and critical infrastructure facilities.

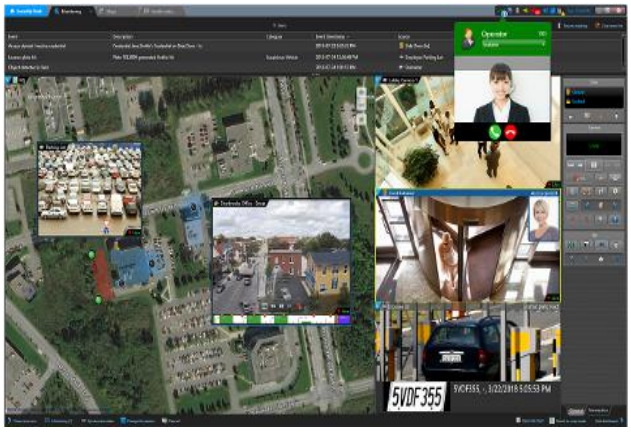


Fig. 2.4 GSC management system

Fig. 2.5 GSC unified solution

GSC allows you to monitor events and configure your system in one place as shown in fig 2.4 above, and fig 2.5 showing the data flow used by GSC systems. As the other systems mentioned above, GSC also have pros and cons:

Pros:

- Unified Security Platform: Genetec Security Center is a unified security platform that integrates video surveillance, access control, license plate recognition (LPR), and other security systems into a single interface. This allows organizations to manage multiple security applications from a centralized platform, enhancing efficiency and situational awareness.
- Scalability and Flexibility: Security Center is designed to be highly scalable and flexible, making it suitable for deployments of all sizes, from small businesses to large enterprises. The platform supports a wide range of hardware devices, cameras, sensors, and third-party integrations, allowing organizations to customize their security infrastructure to meet their specific needs.
- Advanced Video Analytics: Security Center offers advanced video analytics capabilities, including object detection, motion detection, and behavior analysis. These analytics

features enable users to quickly search, identify, and analyze specific events or objects within video footage, improving operational efficiency and security response.

- Centralized Management and Reporting: Security Center provides centralized management and reporting capabilities, allowing users to monitor and control security systems from a single interface. The platform offers real-time alerts, event logging, and customizable reporting tools, enhancing situational awareness and incident response.
- Open Architecture and API: Security Center is built on an open architecture and offers robust API (Application Programming Interface) support, enabling seamless integration with third-party systems and custom applications. This openness allows organizations to leverage existing investments and integrate Security Center into their existing workflows and processes.

Cons:

- Cost: One potential downside of Genetec Security Center is the cost, as it may involve significant upfront investment in software licenses, hardware infrastructure, and installation services. The advanced features and capabilities of Security Center can contribute to higher overall costs compared to simpler security solutions.
- Complexity: The advanced features and capabilities of Security Center may introduce complexity in terms of setup, configuration, and maintenance. Organizations may require specialized training or expertise to effectively deploy and manage the platform, especially for larger deployments with multiple sites or integrations.
- Dependency on Genetec Ecosystem: Security Center is tightly integrated with the Genetec ecosystem, including Genetec cameras, sensors, and software modules. While this integration offers seamless interoperability and enhanced functionality, it also creates a dependency on Genetec products and limits compatibility with third-party systems.
- Bandwidth and Storage Requirements: The advanced video analytics features of Security Center can result in increased bandwidth and storage requirements, especially in large-scale deployments with multiple cameras and high-resolution footage. Organizations need to carefully plan and manage their network infrastructure and storage capacity to accommodate these requirements.

- Privacy and Compliance: The advanced video analytics capabilities of Security Center raise potential privacy concerns, particularly regarding the collection and analysis of sensitive information such as facial images and license plate numbers. Organizations deploying Security Center must ensure compliance with applicable privacy regulations and implement appropriate safeguards to protect individuals' privacy rights.

Overall, Genetec Security Center offers advanced security features, scalability, and centralized management capabilities.

2.3.4 ISS SecurOS:

ISS SecurOS [4] is a video management and analytics software platform that offers visual tracking capabilities for security and surveillance applications. It provides real-time monitoring, intelligent video analytics, and integration with access control and alarm systems to enhance situational awareness and incident response.

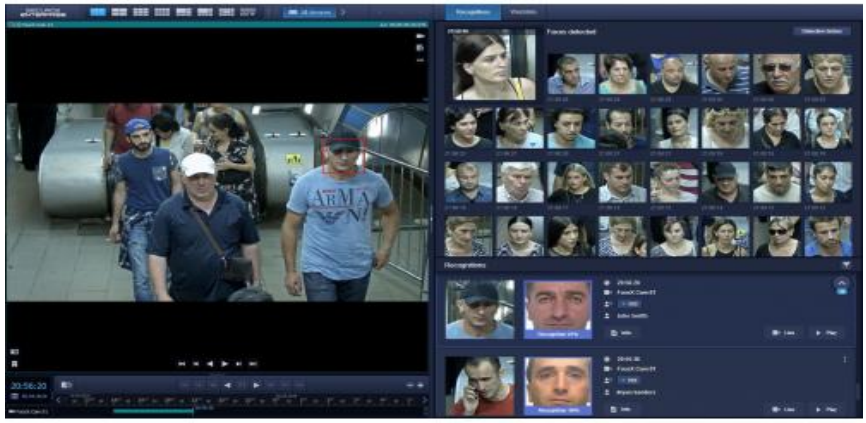


Fig. 2.6 ISS SecureOS FaceX

Fig. 2.6 above shows ISS intelligent video analytics module that provides facial detection and recognition for your security needs. Some of the advantages this system provide are listed below

Pros:

- Advanced Video Analytics: ISS SecurOS offers advanced video analytics capabilities, including object detection, facial recognition, license plate recognition (LPR), and behavior analysis. These analytics features enable users to quickly search, identify, and analyze specific events or objects within video footage.

- Scalability and Flexibility: SecurOS is designed to be highly scalable and flexible, making it suitable for deployments of all sizes, from small businesses to large enterprises. The platform supports a wide range of cameras, sensors, and third-party integrations, allowing users to customize their security infrastructure to meet their specific needs.
- Centralized Management: SecurOS provides centralized management capabilities, allowing users to monitor and control multiple cameras and sites from a single interface. This centralized approach simplifies system administration, improves operational efficiency, and enhances situational awareness for security personnel.
- Integration with Third-Party Systems: SecurOS offers integration with a variety of third-party systems and devices, including access control systems, alarm systems, and IoT (Internet of Things) devices. This integration allows organizations to leverage existing investments and integrate SecurOS into their existing security ecosystem.
- Customizable Workflows and Rules: SecurOS allows users to create customizable workflows, rules, and event triggers to automate security operations and response actions. This flexibility enables organizations to tailor the platform to their specific security policies and procedures.

Every system always comes with disadvantages, some of which are listed below.

Cons:

- Cost: One potential downside of ISS SecurOS is the cost, as it may involve significant upfront investment in software licenses, hardware infrastructure, and installation services. The advanced features and capabilities of SecurOS can contribute to higher overall costs compared to simpler security solutions.
- Complexity: The advanced features and capabilities of SecurOS may introduce complexity in terms of setup, configuration, and maintenance. Organizations may require specialized training or expertise to effectively deploy and manage the platform, especially for larger deployments with multiple sites or integrations.
- Dependency on ISS Ecosystem: SecurOS is tightly integrated with the ISS ecosystem, including ISS cameras, sensors, and software modules. While this integration offers seamless interoperability and enhanced functionality, it also creates a dependency on ISS products and limits compatibility with third-party systems.

- Bandwidth and Storage Requirements: The advanced video analytics features of SecurOS can result in increased bandwidth and storage requirements, especially in large-scale deployments with multiple cameras and high-resolution footage. Organizations need to carefully plan and manage their network infrastructure and storage capacity to accommodate these requirements.
- Privacy and Compliance: The advanced video analytics capabilities of SecurOS raise potential privacy concerns, particularly regarding the collection and analysis of sensitive information such as facial images and license plate numbers. Organizations deploying SecurOS must ensure compliance with applicable privacy regulations and implement appropriate safeguards to protect individuals' privacy rights.

Overall, while ISS SecurOS offers advanced security features, scalability, and centralized management capabilities.

2.3.5 BriefCam:

BriefCam [5] is a video content analytics software that provides advanced search, review, and tracking capabilities for large-scale video surveillance systems. It offers features such as video synopsis, object detection, and behavior analysis, enabling users to quickly identify and track objects of interest across video footage.

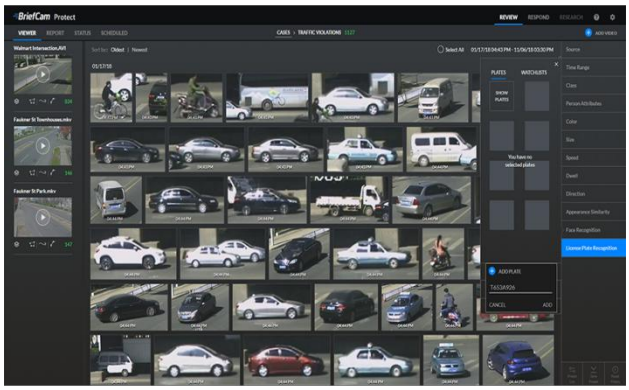


Fig 2.7: briefcam visual interface

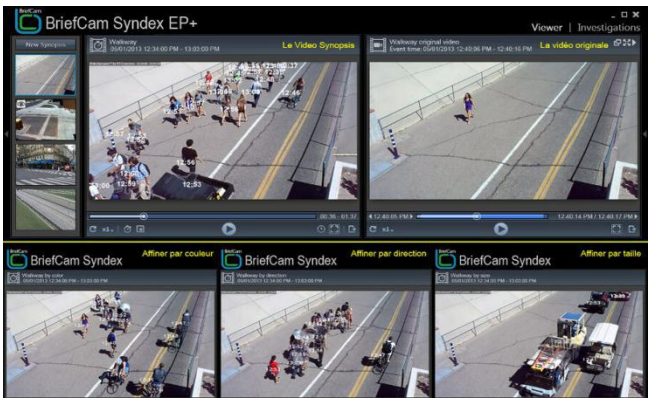


fig 2.8: Briefcam ultra fast image analysis

Fig 2.7 and 2.8 above shows Briefcams dashboard with its core functionality which is to analyze images based on video surveillance cameras. Driven by artificial intelligence and deep learning, video intelligence software detects and extracts objects in video, identifies each object

based on trained Deep Neural Networks, and then classifies each object to enable intelligent video analysis, including search and filtering, alerting, and data aggregation.

Briefcam too appears to have its major advantages and like all other systems, it also has some disadvantages which will be listed below.

Pros:

- Efficient Video Review:
    - Summarization: BriefCam's video synopsis technology enables users to condense hours of footage into a short, easy-to-review summary. This makes it much quicker to find relevant events.
    - Rapid Search: Users can quickly search through video footage based on various attributes such as color, size, speed, direction, and more.
- Advanced Analytics:
    - Real-Time Alerts: The system can be configured to provide real-time alerts based on predefined rules, helping to quickly address incidents as they occur.
    - Behavioral Analysis: BriefCam offers insights into patterns and behaviors, which can be useful for security, operations, and business intelligence.
- Integration Capabilities:
    - Compatibility: BriefCam is designed to integrate with many existing video management systems (VMS) and IP cameras, allowing for seamless integration into existing surveillance setups.
    - Open API: The platform provides APIs for customization and integration with other systems.
- Scalability:
    - Flexible Deployment: Whether it's a small setup or a large enterprise, BriefCam can scale according to the needs, supporting multiple cameras and locations.
    - Cloud and On-Premises Options: Users can choose between cloud-based or on-premises deployment based on their infrastructure and security requirements.
- User-Friendly Interface:
    - Intuitive UI: The user interface is designed to be easy to use, with drag-and-drop functionality, making it accessible even for non-technical users.

- Enhanced Security:
    - Data Privacy: BriefCam adheres to data privacy regulations and provides features to anonymize individuals in videos, which is crucial for compliance with laws such as GDPR.

Cons:

- Cost:
    - Expensive: BriefCam can be relatively costly, especially for small businesses or individuals. The advanced features and analytics come at a premium price.
- Complexity:
    - Setup and Maintenance: Initial setup and ongoing maintenance may require technical expertise. Organizations might need to invest in training or hire skilled personnel.
    - Resource Intensive: The software can be resource-intensive, necessitating robust hardware and infrastructure, particularly for large-scale deployments.
- Learning Curve:
    - Training Required: While the interface is user-friendly, maximizing the potential of BriefCam's advanced features may require comprehensive training for users.
- Dependency on Camera Quality:
    - High-Quality Inputs Needed: The effectiveness of BriefCam’s analytics is highly dependent on the quality of the video input. Low-resolution or poorly positioned cameras can reduce the accuracy of the analytics.
- Network Dependence:
    - Bandwidth Consumption: For cloud-based deployments, high bandwidth is essential to transmit large amounts of video data. This can be a constraint in areas with limited network infrastructure.

In summary, BriefCam offers powerful video analytics capabilities that can significantly enhance video surveillance operations through efficient video review, advanced analytics, and scalability.

However, the high cost, complexity, and potential need for significant infrastructure and training are important considerations when deciding whether to implement this solution.

Here below is a general table for the general cons and pros for the various video surveillance.

System	**Pros**	**Cons**
Viseum Intelligent CCTV	- Advanced visual tracking capabilities. - Comprehensive surveillance solution. -Improved situational awareness. -Scalability and customization - Reliability and performance	- High cost. - Complexity in setup and maintenance. - Integration challenges. - Dependency on technology. - Privacy concerns.
Avigilon Control Center (ACC)	- Advanced video analytics. - High-definition video quality. - Scalability and flexibility. - Intuitive user interface. - Centralized management.	- High cost- Complexity in setup and maintenance. - Dependency on Avigilon ecosystem. - Bandwidth and storage requirements. - Privacy and compliance issues.
Genetec Security Center	- Unified security platform. - Scalability and flexibility. - Advanced video analytics. - Centralized management and reporting- Open architecture and API.	- High cost- Complexity in setup and maintenance. - Dependency on Genetec ecosystem- Bandwidth and storage requirements. - Privacy and compliance issues.
ISS SecureOS	- Advanced video analytics - Scalability and flexibility	- High cost - Complexity in setup and

	- Centralized management- Integration with third party systems. - Customizable workflows and rules.	maintenance. - Dependency on ISS ecosystem. - Bandwidth and storage requirements. - Privacy and compliance issues.
BriefCam	- Efficient video review. - Advanced analytics - Integration capabilities - Scalability - User-friendly interface- Enhanced security	- High cost - Complexity in setup and maintenance. - Learning curve. - Dependency on camera quality. - Network dependence.

2.4 Partial Conclusion:

The body of research on object detection in video surveillance systems emphasizes how crucial cutting-edge technology and algorithms are to improving security and operational effectiveness. The accuracy and dependability of object detection systems have greatly increased recently because of developments in deep learning and computer vision, allowing for real-time monitoring and quick reaction to security risks. Furthermore, new paths for useful insights have been made possible by the integration of object detection with video analytics tools like BriefCam. These systems provide useful information for business intelligence and security needs by analyzing behavioral patterns in addition to object detection. Optimizing and condensing vast amounts of video in a timely manner improves operational effectiveness and situational awareness.

But even with these improvements, there are still a number of difficulties. Deep learning models require significant hardware resources because of their high computational intensity, which may be prohibitively expensive for certain individuals. Furthermore, concerns about data

privacy and the moral application of surveillance data need to be kept up to date, especially in light of strict laws like the General Data Protection Regulation (GDPR).

In conclusion, while object detection in video surveillance has made significant strides, ongoing research is crucial to address the existing limitations and to continue improving the robustness and applicability of these systems.

CHAPTER THREE: ANALYSIS AND DESIGN

3.1 Introduction:

This project aims to develop a real-time dangerous object detection system using video surveillance cameras. The system will utilize advanced computer vision techniques, specifically leveraging YOLOv5, to detect objects such as guns and knives in video feeds. Upon detection, the system will trigger an alarm and display a warning to alert relevant personnel.

Objectives:

- To build a robust and accurate object detection model.
- To ensure real-time processing and detection.
- To integrate alert mechanisms such as sound alarms and visual warnings.
- To deploy the system in a scalable and maintainable manner.

Scope: The scope of this project includes data collection and preprocessing, model training and evaluation, real-time inference, and system integration with alert mechanisms.

3.2 Methodology:

Data Collection:

- Sources: Public datasets like COCO and ImageNet, and custom-collected images and videos.
- Annotation: Labeling dangerous objects in images using tools like LabelIng.

Data Preprocessing:

- Cleaning: Removing duplicates and irrelevant images. We also ensure our data w
- Augmentation: Applying transformations such as rotation, flipping, scaling, etc.

Model Training:

- Model Selection: Using YOLOv5 for its balance between speed and accuracy.
- Training Setup: Configuring parameters such as batch size, learning rate, and epochs.

- Training Process: Using a GPU-enabled environment to speed up training.

Model Evaluation:

- Metrics: Evaluating using mAP, precision, and recall.
- Validation: Testing the model on a separate validation dataset.

Real-Time Inference and Deployment:

- Video Capture: Using OpenCV to capture video feed.
- Inference: Performing real-time detection on video frames.
- Deployment: Containerizing the application using Docker and deploying it on cloud platforms.

Alert Mechanisms:

- Sound Alert: Using the playsound library to trigger alarms.
- Visual Alert: Using Tkinter for GUI pop-up warnings.

This gives the process which we will implement to resolve our problem in detail with each step explained.

3.3 Design:

The dangerous object detection system consists of the following primary components:

1. **Camera Module**: Captures video frames.
2. **Frame Processor**: Processes frames to detect dangerous objects.
3. **User Interface (GUI)**: Displays video feeds and detection results.
4. **Alert System**: Triggers alerts when dangerous objects are detected.
5. **Trained Mode**: used for object detection.

Now lets discuss about the various requirements needed for our system such as the functional and non-functional requirements.

3.3.1 Functional Requirements:

- **Object Detection:**

Detect both static and moving objects within the video feed. Differentiate between various objects like humans, vehicles, and potentially dangerous items (e.g., guns, knives)

- **Real-Time Performance:**

The system must provide real-time performance to ensure timely detection and response to security threats or events of interest. This requires algorithms to be optimized for efficient resource utilization and low-latency processing.

- **Object Tracking:**

Track detected objects continuously and accurately over time. Maintain tracking even when objects are partially or temporarily occluded.

- **Multiple Object Handling:**

Track multiple objects simultaneously, maintaining individual object identities

Environmental Adaptability:

Adapt to dynamic environmental changes such as lighting variations, moving backgrounds, and weather conditions.

- **Alarm Generation:**

Trigger alarms when specific conditions are met, such as detecting a dangerous object or unauthorized access.

- **User Notifications:**

Provide notifications to users through various channels such as email, SMS, or mobile apps when significant events are detected ..

- **User Interface:**

Provide an intuitive and user-friendly interface for monitoring, configuring, and managing the surveillance system.

- **Reporting and Analytics:**

Generate reports and analytics on detected events, object movements, and system performance.

These requirements ensure that the visual tracking system is robust, reliable, and effective in various surveillance scenarios.

3.3.2 Non-Functional requirements:

Real-Time Performance:

The system must provide real-time performance to ensure timely detection and response to security threats or events of interest. This requires algorithms to be optimized for efficient resource utilization and low-latency processing.

Scalability:

The system should be scalable to handle large-scale surveillance networks with a high volume of camera feeds. It must optimize resource usage and maintain real-time performance across the entire system.

Robustness:

Tracking algorithms must be robust to variations in scale, object deformations, and non-rigid motion. They should handle challenging conditions such as occlusions, illumination changes, and cluttered backgrounds effectively.

Reliability:

The system must ensure continuous operation and efficient processing of video data, which is essential for mission-critical surveillance applications. Reliability in hardware and software components is crucial [[ ]](file-service://file-ls7dvcss1laLqQofD4uLHWey).

Adaptability:

The system should adapt dynamically to changes in the environment, including moving cameras, changing lighting conditions, and evolving scene dynamics. This also includes the ability to handle multiple objects simultaneously while maintaining computational efficiency.

Privacy and Ethical Considerations:

It is essential to implement privacy-preserving techniques and adhere to ethical guidelines to protect individuals' rights and ensure responsible use of surveillance technology. This includes the collection, storage, and analysis of video surveillance data.

Low-Visibility Conditions:

Algorithms must be capable of tracking objects under low-visibility conditions, such as fog, smoke, or low-light environments, by leveraging alternative sensing modalities or enhancing image processing techniques.

Integration with Existing Systems:

The system should facilitate seamless integration into existing or new video surveillance systems, allowing for easy deployment and operation in real-world environments.

These non-functional requirements ensure that the visual tracking system for video surveillance is effective, reliable, and capable of operating under various conditions while maintaining the privacy and security of the monitored individuals and environments.

Moving on to the various UML diagrams, we have the following based on our system :

3.3.3: Use Case Diagram:

A use case diagram shows the various actors of the system and how they interact with each other. For our case we have a person as the main user and he is able to perform functions such as open camera, stop camera as shown on the fig 3.2 below.

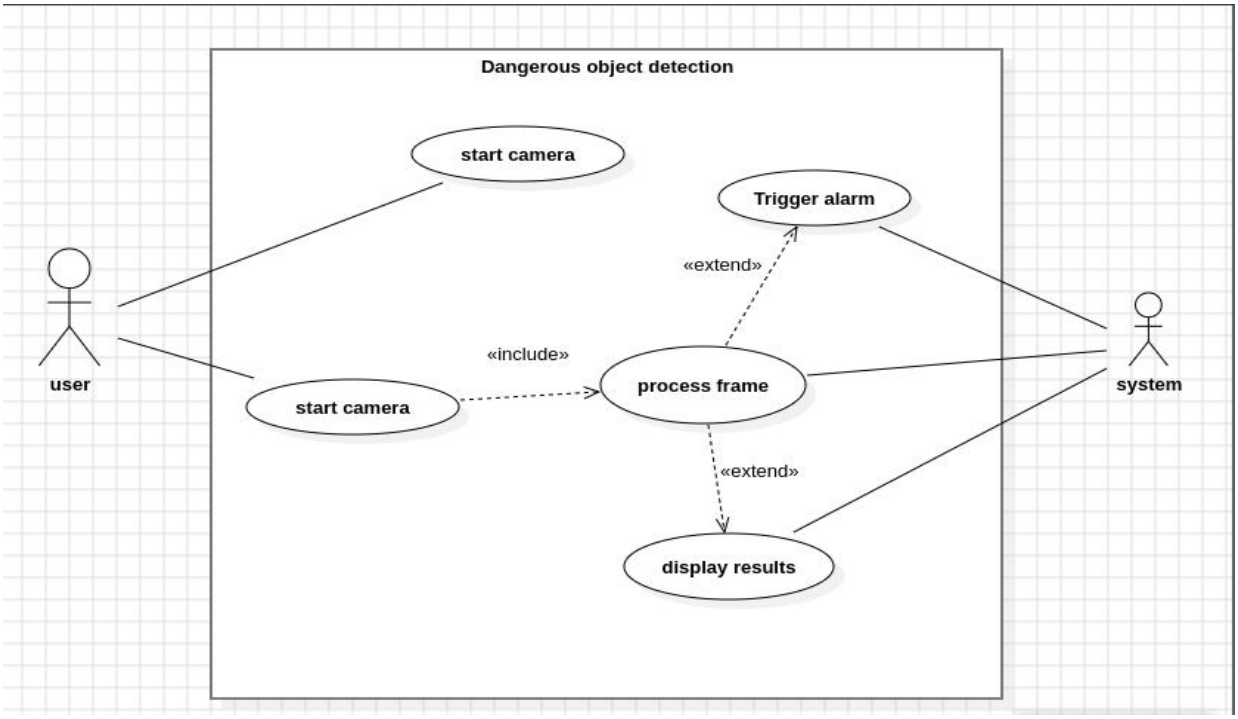


Fig 3.2: Use case diagram

As Shown on the diagram, the user has access to the following use cases such as start camera, and stop camera.

Upon starting the camera, the back end system comes into action by enabling the processing of frames and we clearly see the relationship with the trigger alarm and the display results because they will only be triggered if and only if a dangerous object was being detected.

This gives the simple Use case for our object detection system.

3.3.4: Class Diagram:

The class diagram represents the static structure of the system, showing the system's classes, attributes,methods,and relationships.

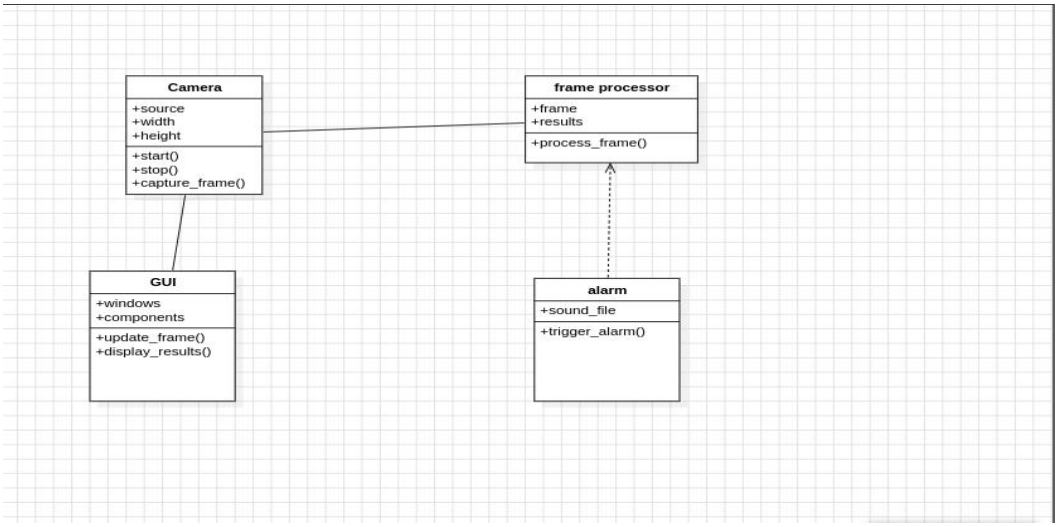


Fig 3.2: class diagram

The Diagram below shows the various classes and their relations. We clearly see how the alarm system depends on the frame processor to be triggered and the GUI and Camera class are connected together. So most of the classes are dependent on other classes to ensure proper functionality as shown above.

3.3.3: Activity Diagram:

The activity diagram shows the work flow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity.

The diagram below illustrates the process flow of our system.

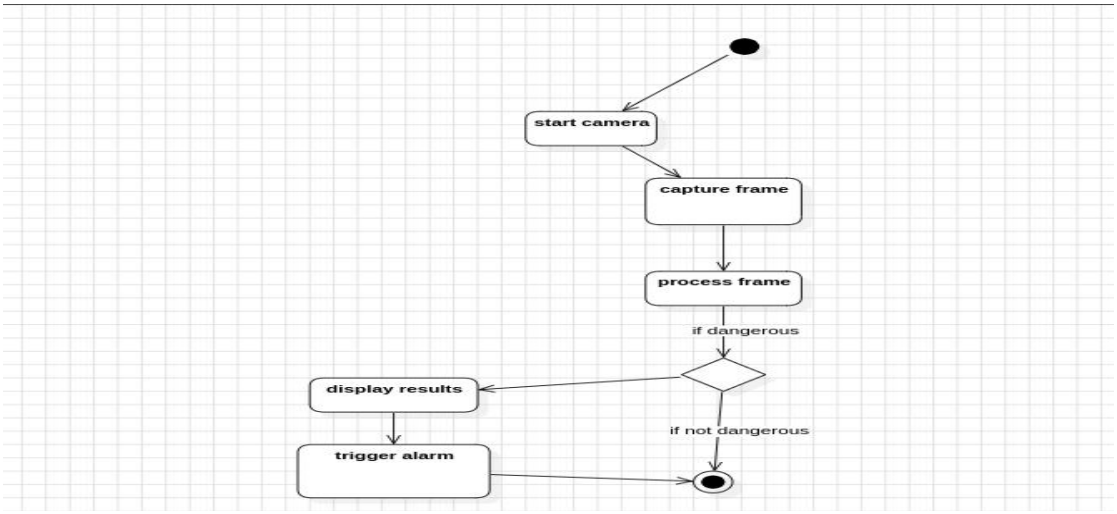


Fig 3.3: Activity Diagram

We clearly visualize the order flow of our system from fig 3.3 above. First the camera ought to be started before the execution can proceed, We then start capturing frames rendered from our camera and process them in real time.

If the process frames are detected to be dangerous, the system triggers the alarm, but if not detected dangerous the system keeps running or we can stop the system.

3.3.5: Sequence Diagram:

A sequence diagram shows process interactions arranged in time sequence.

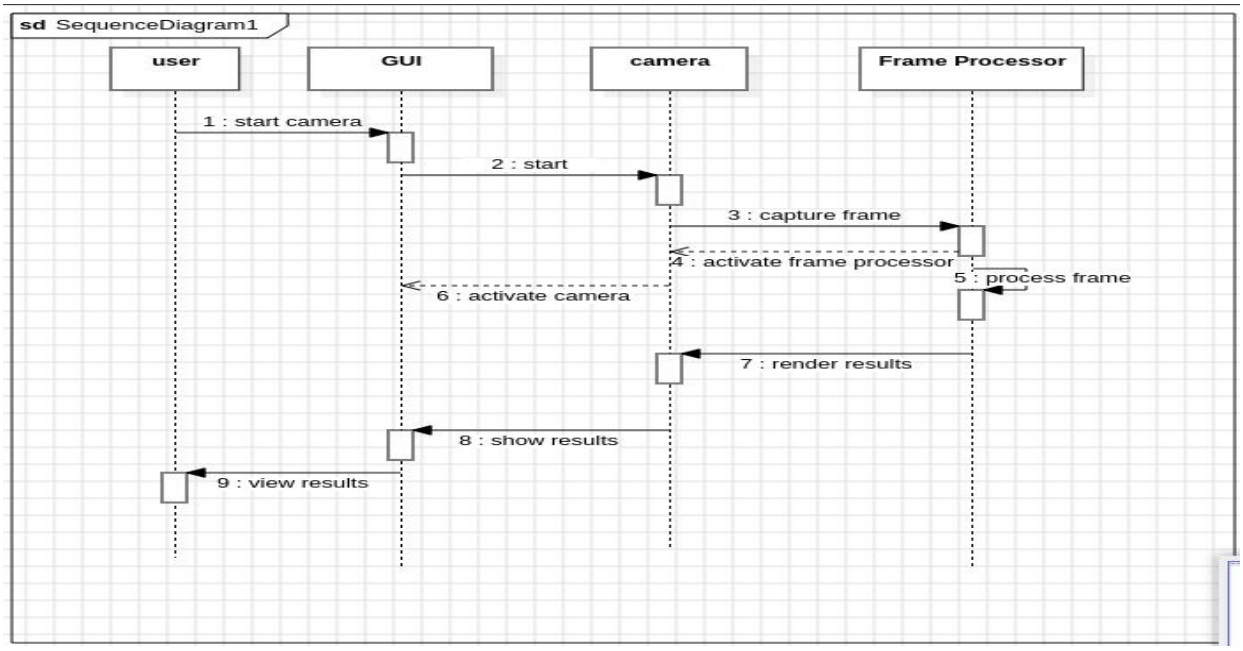


fig 3.4: Sequence diagram

On fig 3.4 above, we demonstrate how interaction is done throughout the various components of the system, It simply tells us how the user sends and receive messages based on his request

We see the user sends a message to the GUI to start the camera, and the GUI later on sends another message to the camera to open or be executed. Which later on sends back a reply to the GUI signaling the camera has been opened. The camera later on send another message to the frame processor to start capturing frames, and the frame processor processes the various frames captured. It later sends back results to the camera as results and the camera sends a message to GUI telling it to show the rendered result to the user. That's the basic system of the timeline of how our system operates.

3.4 Global Architecture of the solution:

For the global architecture of our solution, I came out with the following regarding our system architecture;

The global architecture of the dangerous object detection system comprises several integrated components, each responsible for a specific task within the overall workflow. Here is a detailed description of each component and its role in the system:

1. Input Layer:

Camera Setup:

- **Description:**
    - This is the initial stage where live video feed is captured. It can be from various sources such as CCTV cameras, IP cameras, or a standard webcam.
    - The camera is configured to provide real-time video input which is essential for continuous monitoring and timely detection.
- **Function:**
    - Capture frames from the live video feed at a predefined frame rate.

2. Preprocessing Layer:

Image Preprocessing:

- **Description:**
    - The captured frames undergo preprocessing to ensure they are suitable for the object detection model.
    - This involves resizing the frames to a fixed dimension, normalizing pixel values, and applying any necessary augmentations.
- **Function:**
    - Convert raw images into a format that the model can efficiently process.

3. Inference Layer:

Object Detection Model (YOLOv5):

- **Description:**
    - The core component responsible for detecting objects within the frames.
    - YOLOv5 is a deep learning model that predicts bounding boxes and class labels for objects in an image.
- **Function:**
    - Analyze the preprocessed frames to identify and locate objects of interest (e.g., weapons).
    - Generate bounding boxes and confidence scores for detected objects.

4. Post-Processing Layer:

Filtering and Non-Max Suppression (NMS):

- **Description:**
    - Post-processing of the detection results to refine the predictions.
    - NMS is used to remove redundant bounding boxes, ensuring only the most accurate detections are retained.
- **Function:**
    - Apply confidence and IoU thresholds to filter out low-confidence detections and eliminate overlapping boxes.

5. Alert Mechanism:

Alarm System:

- **Description:**
    - An alert mechanism that triggers an alarm when a dangerous object is detected.
    - Includes both an audible alarm and a visual warning to alert the monitoring personnel.
- **Function:**
    - Monitor the detection results and trigger the alarm if a dangerous object is detected with high confidence.

- Implement a cooldown period to prevent continuous triggering of the alarm.

Warning Pop-up:

- **Description:**
  - A user interface component that displays a warning message on the screen.
  - Alerts the user to the presence of a potentially dangerous object.
- **Function:**
  - Show a pop-up message or a warning banner when an alarm is triggered.
  - Provide an option for the user to acknowledge the warning.

6. User Interface Layer:

Display and Monitoring:

- **Description:**
  - The component responsible for displaying the live video feed with detection results overlaid.
  - Provides the monitoring personnel with a real-time view of the surveillance area, including detected objects.
- **Function:**
  - Render the video frames with bounding boxes and class labels.
  - Display additional information such as timestamps, confidence scores, and detection counts.

7. Data Management Layer:

Logging and Storage:

- **Description:**
  - This component manages the storage of detection results and video footage for future reference and analysis.
  - Logs important events such as alarms and detections with timestamps.
- **Function:**

- Store detected frames and related data in a database or file system.
- Maintain logs of all alarms and detections for audit and review.

The global architecture of the dangerous object detection system is designed to be robust, scalable, and efficient. It integrates various components seamlessly to provide real-time detection and alerting capabilities. Each layer has a specific role, from capturing video input and preprocessing frames to performing object detection, triggering alarms, and displaying results. The architecture also includes provisions for data management, system integration, and regular updates to ensure the system's long-term reliability and effectiveness.

3.5 Description of the Algorithms:

Before getting into the various algorithms we are to use,we should understand the model we are to use first which is YOLO.

YOLO (You Only Look Once) is a popular object detection model known for its speed and accuracy. It was first introduced by Joseph Redmon et al. in 2016 and has since undergone several iterations, the latest being YOLO v7.

YOLO is a single-shot detector that uses a fully convolutional neural network (CNN) to process an image. You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection.

Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin.

While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer.

Methods that use Region Proposal Networks perform multiple iterations for the same image, while YOLO gets away with a single iteration.

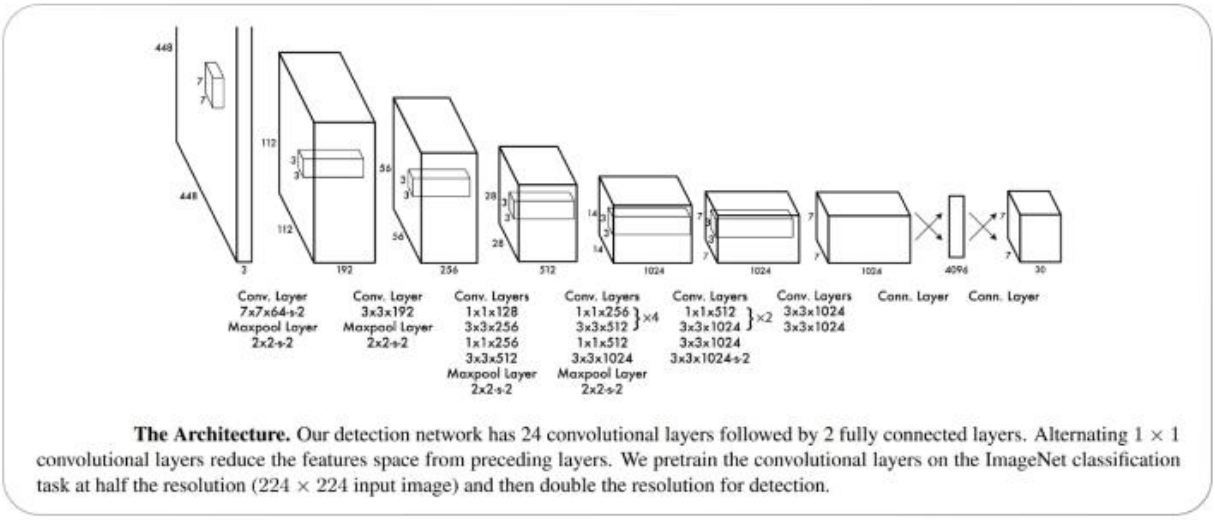
Several new versions of the same model have been proposed since the initial release of YOLO in 2015, each building on and improving its predecessor. Here's a timeline showcasing YOLO's development in recent years.

Fig 3. Below shows the evolution of YOLO throughout this years

Fig 3.5 YOLO timeline

How does YOLO work? YOLO Architecture.

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Fig 3.6 YOLO architecture

That's basically the overview of our model architecture. So moving on to the other algorithms needed for our project, we have Object Detection Algorithm (YOLOv5) & Alarm Triggering Algorithm.

3.5.1 Object Detection algorithm:

For this algorithm, we will implement the following steps to ensure a good model:

1. **Input Image Processing:**
    - Resize the input image to a fixed size (e.g., 640x640 pixels) required by the YOLOv5 model.
    - Normalize the pixel values to the range [0, 1] to ensure consistency in model input.
2. **Feature Extraction:**
    - The input image is passed through a backbone neural network (e.g., CSPDarknet53) to extract hierarchical features.
    - These features represent different aspects of the image, such as edges, textures, and shapes.
3. **Object Detection:**

- The extracted features are passed through multiple convolutional layers to predict bounding boxes, objectness scores, and class probabilities.
- Bounding boxes are predicted relative to grid cells that divide the image. Each grid cell predicts multiple bounding boxes.

4. **Bounding Box Prediction:**
    - For each grid cell, YOLO predicts a fixed number of bounding boxes, along with the confidence score for each box.
    - Confidence score represents the probability that a bounding box contains an object and the accuracy of the box's position.
5. **Class Prediction:**
    - For each bounding box, YOLO predicts the probabilities of the box belonging to various classes (e.g., knife, gun).
    - The class with the highest probability is assigned to the bounding box.
6. **Non-Max Suppression (NMS):**
    - After obtaining all bounding boxes and their associated confidence scores, NMS is applied to remove redundant boxes.
    - Boxes with high overlap (Intersection over Union, IoU) and lower confidence scores are suppressed, retaining only the most confident ones threshold.

3.5.2 Alarm Triggering Algorithm:

Overview: The alarm triggering algorithm is designed to initiate an alert when a dangerous object is detected with a confidence score above a certain threshold. The algorithm ensures that alerts are not triggered continuously by implementing a cooldown period.

Key Steps:

1. **Detection Verification:**
    - Review the results from the object detection algorithm.
    - Filter the detected objects based on their confidence scores and specified class labels (e.g., knife, gun).
2. **Confidence Threshold Check:**

- Check if the confidence score of the detected object exceeds a predefined threshold (e.g., 0.6).
- This threshold helps in reducing false positives by ensuring only high-confidence detections trigger the alarm.

3. **Cool down Period Implementation:**
    - Introduce a cooldown period to prevent continuous triggering of the alarm.
    - Track the last time an alarm was triggered and ensure a minimum time interval has passed before triggering a new alarm.
4. **Alarm Activation:**
    - If a dangerous object is detected and the confidence threshold is met, and if the cooldown period has elapsed, activate the alarm.
    - Trigger a sound alarm and display a warning pop-up to alert the user.

3.5.3 Non-Max Suppression (NMS) Algorithm:

Overview: Non-Max Suppression (NMS) is a technique used to refine the results of the object detection algorithm. It reduces the number of redundant bounding boxes by keeping only the most confident ones.

1. Bounding Box Selection:
    - Start with a list of all predicted bounding boxes and their associated confidence scores.
    - Sort the bounding boxes by their confidence scores in descending order.
2. Overlap Calculation:
    - For each bounding box, calculate its overlap (IoU) with all other bounding boxes.
    - IoU is the ratio of the area of the intersection of two boxes to the area of their union.
3. Suppression of Overlapping Boxes:
    - If the IoU of two boxes exceeds a predefined threshold (e.g., 0.5), suppress the box with the lower confidence score.
    - Continue this process until only non-overlapping boxes with the highest confidence scores remain.

4. Final Selection:
    - The remaining bounding boxes after suppression are the final detections.
    - These boxes are then used to identify and locate objects in the image.

3.5.4 Image Preprocessing Algorithm:

Overview: Image preprocessing is a crucial step in preparing the input data for the object detection model. It ensures that the images are in the correct format and size, enhancing the model's performance.

Key Steps:

1. Resizing:
    - Resize the input images to a fixed size required by the model (e.g., 640x640 pixels).
    - Maintain the aspect ratio to avoid distortion of objects in the image.
2. Normalization:
    - Normalize the pixel values to a range suitable for the model, typically [0, 1].
    - This is done by dividing the pixel values by 255 (for 8-bit images).
3. Augmentation:
    - Apply random transformations to the images to create variations and enhance the model's robustness.
    - Common augmentations include rotation, flipping, scaling, and color adjustments.
4. Bounding Box Adjustment:
    - Adjust the bounding boxes to match the resized images.
    - Ensure that the coordinates of the bounding boxes are scaled proportionally to the new image size.

Summary:

Each of these algorithms plays a critical role in the overall dangerous object detection system. The object detection algorithm (YOLOv5) performs the core task of identifying and locating objects in the video feed. The alarm triggering algorithm ensures timely and appropriate alerts

based on the detection results. Non-Max Suppression refines the detection results by removing redundant bounding boxes, and image preprocessing prepares the input data for optimal model performance. These algorithms work together to create a robust and efficient dangerous object detection system suitable for real-time surveillance applications.

3.6 Description of the resolution process:

Resolution Process:

1. **Frame Capture:**
    - Use OpenCV to capture video frames from the camera.
    - Resize frames for model input.
2. **Object Detection:**
    - Load YOLOv5 model.
    - Perform inference on each frame.
    - Filter results based on confidence threshold and target objects.
3. **Alert Mechanism:**
    - If a dangerous object is detected, play a sound alarm and display a warning pop-up.
    - Ensure alerts are not triggered continuously by implementing a cooldown period.
4. **User Interface:**
    - Provide a GUI for starting/stopping the video feed and monitoring detections.
    - Log detections and alerts for further analysis.

3.7 Partial Conclusion:

The implementation of a dangerous object detection system involves several critical phases, from data collection to real-time inference and alert integration. Each phase requires careful planning and execution to ensure the system is accurate, responsive, and reliable. By leveraging advanced object detection models like YOLOv5 and integrating effective alert mechanisms, the system can significantly enhance security measures in various settings.

CHAPTER FOUR: IMPLEMENTATION AND RESULTS

4.1 Introduction:

My analysis focuses on the application of visual object tracking to detect fighting scenes in video surveillance footage. By leveraging advanced computer vision techniques and machine learning algorithms, we aim to develop a robust system capable of identifying aggressive behavior patterns and alerting security personnel in real-time. My approach involves the integration of motion analysis, object recognition, and behavioral pattern detection to accurately differentiate between normal interactions and potential threats.

4.2 Tools and material used:

The various tools and materials used to implement this project are as follows:

Surveillance camera:

Object Detection Models:

- **YOLO (You Only Look Once)**: For real-time object detection with high accuracy and speed.
- **SSD (Single Shot MultiBox Detector)**: For efficient detection, suitable for embedded systems.

Programming Languages and Libraries:

- **Python**: The primary programming language used for scripting and implementing the detection logic.
- **OpenCV**: For video capture, image processing, and displaying results.
- **TensorFlow / PyTorch**: Deep learning frameworks to train and deploy object detection models.
- **NumPy**: For numerical operations and array manipulations.
- **SciPy**: For scientific and technical computing.

Hardware:

- **Camera**: Surveillance camera or webcam to capture real-time video feeds.
- **GPU**: Graphics Processing Unit to accelerate model inference, especially useful for deep learning models.

Software and Libraries for Real-Time Processing:

- **Threading**: For handling concurrent tasks, such as playing sounds or displaying pop-ups without freezing the video feed.
- **Tkinter**: For creating GUI elements like warning pop-ups.
- **Playsound / Pygame**: For playing alert sounds.
- **Colab**: Used as an interface for training the model
- **Vs code**: Used as an environment for testing the code with real time data

4.3 Description of the implementation process:

The implementation of a dangerous object detection system using a video surveillance camera involves several phases. Each phase contributes to building a robust, accurate, and efficient detection system.On the following lines below, we will provide a full description of each step I took to implement the detection system:

4.3.1 Problem Definition and Data Collection:

For this particular phase, I had to identify my main target or problem to resolve and collect the necessary data needed to build such a system.

Problem definition:

- Objective: Develop a real-time system to detect dangerous objects, such as guns and knives, in video streams from surveillance cameras.
- Requirements: The system must be accurate, capable of real-time processing, and include effective alert mechanisms to notify users of detected threats.

Data Collection

- **Datasets**: Gather images and videos containing various dangerous objects.
- **Sources**: Utilize public datasets like COCO and ImageNet, or collect custom images and videos.
- **Annotation**: Using tools like LabelImg or VGG Image Annotator (VIA) to label the objects in the images, creating bounding boxes around the dangerous objects for training purposes.

4.3.2 Data Preprocessing:

Data Cleaning

- **Remove Duplicates**: Eliminate duplicate or irrelevant images to ensure a high-quality dataset.
- **Ensure Diversity**: Ensure the dataset includes images with different angles, lighting conditions, and backgrounds to improve model robustness.

Data Augmentation

- **Techniques**: Apply various transformations to increase dataset variability, such as rotation, flipping, scaling, color jittering, and cropping. This helps the model generalize better to unseen data. Used platforms such as roboflow to help me do that.

4.3.3 Model Selection and Training:

In this stage, after preparing our model, we are ready to train our dataset with a model chosen.
On the following lines I will provide the full overview of our model training process.
First we began by cloning our chosen model which is a YOLO repository from github via colab.

Later, we continued by installing the necessary libraries needed by our model to work efficiently such as pandas, numpy, scipy etc..

So we also had to load our dataset into our collab editor and start defining our model architecture and parameters. A sample image below shows our dataset after transformation and cropping techniques were being applied to it on fig 4.1 below

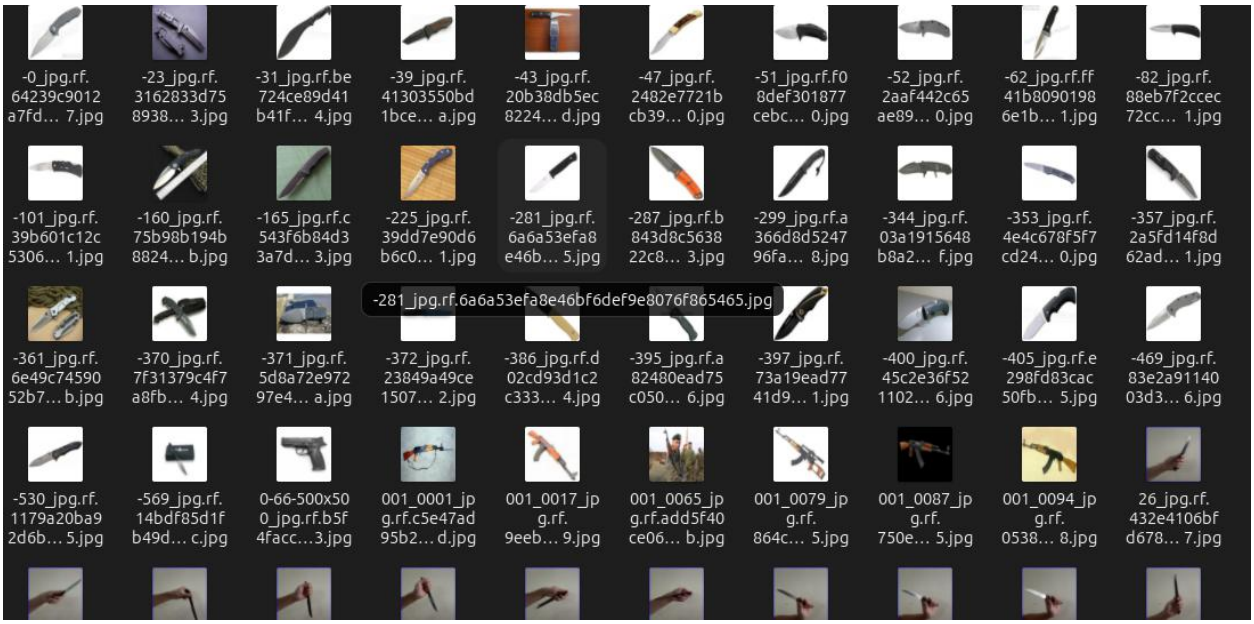


Fig 4.1: Image sample of dataset

So, moving on to configuring the necessary parameters needed for our model, started by writing the yaml script needed for the YOLO model

After configuring our model for training, we later on begun the training with the following parameters,

- img: define input image size
- batch: determine batch size
- epochs: define the number of training epochs. (Note: often, 3000+ are common here!)
- data: set the path to our yaml file
- cfg: specify our model configuration
- weights: specify a custom path to weights
- name: result names
- nosave: only save the final checkpoint
- cache: cache images for faster training

We used img size = 416

Batch size = 16

Epochs = 70

Data = was the path directory to our model

Cfg = specifies the path to the model configuration

And began the training which took close to 5h 30 min. A sample of the training process is shown fig 4.2 below.

Fig 4.2: Sample image of training process

4.4 Presentation and interpretation of results:

Evaluation

- **Metrics**: Evaluate the model using metrics like mAP (mean Average Precision), precision, and recall.
- **Validation**: Validate the model on a separate dataset to ensure it does not overfit the training data.

Fine-Tuning

- **Hyperparameter Adjustment**: Fine-tune hyperparameters based on evaluation results.

- **Advanced Techniques**: Implemented techniques such as learning rate scheduling and additional data augmentation to further improve performance.

I later plotted curves such as the loss function , metrics, validation and had the following results on the fig 4.3 below.

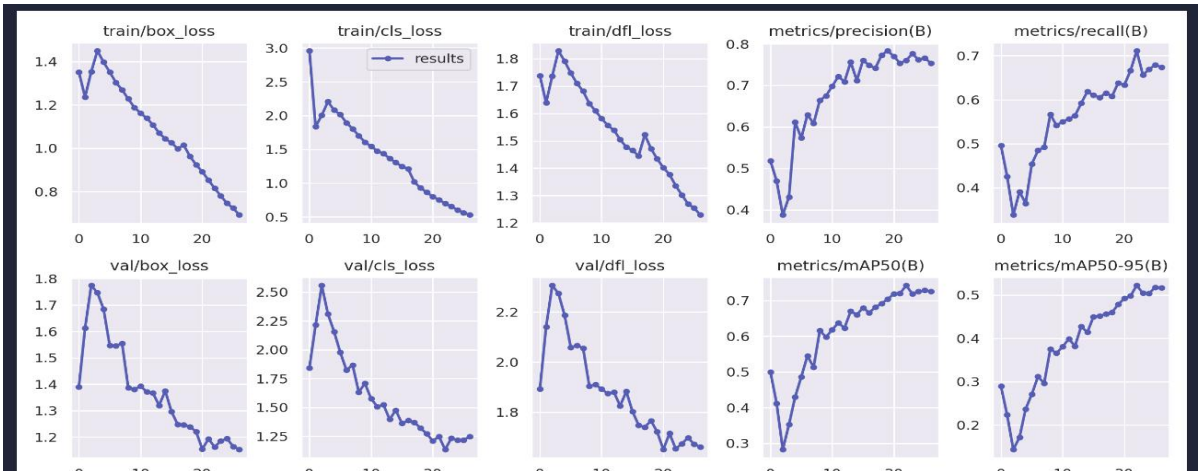


Fig 4.3: Model graph for various functions

Had a precision of approximately 76% for all the well detected objects after passing our trained data to the validation data.

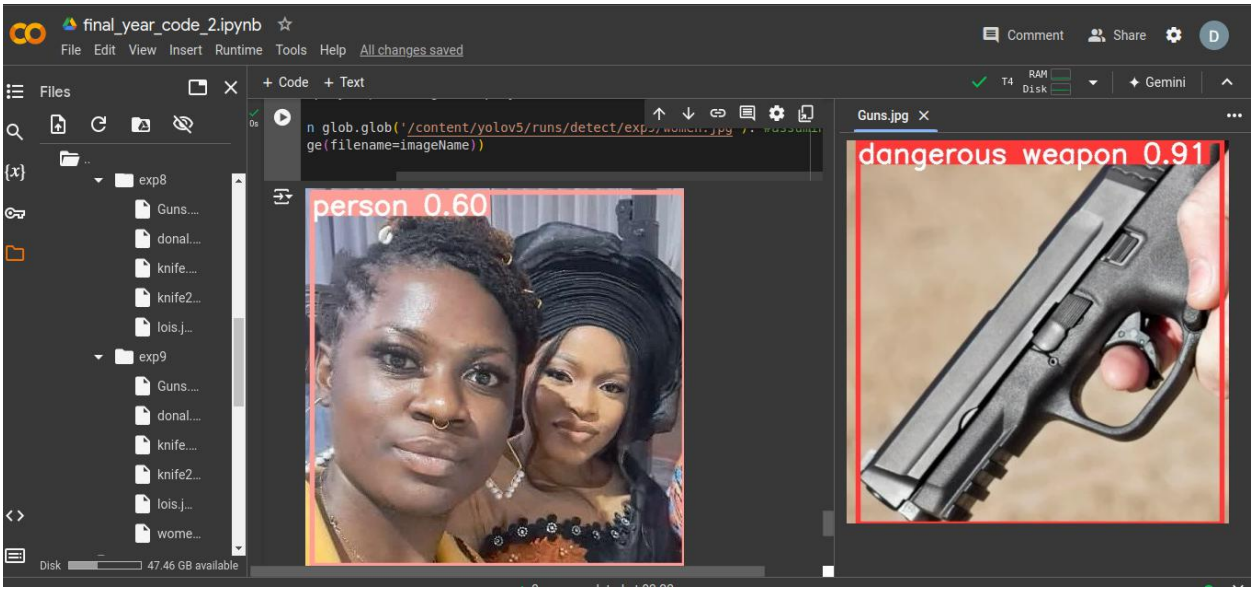


Fig 4.4: testing our trained model with gallery images

I later on tested the trained model with a series of pictures to determine how accurate the model could be, and modify some parameters if need be. So, after being satisfied, I downloaded my

trained model into my computer for integration with the real camera since colab couldn't support camera activities due to restriction from the kernel.

4.4 Presentation and interpretation of solution:

After downloading my trained model to be used and tested on my local pc, I later on tested the model to my local pc to evaluate its performance. I further implemented a GUI using the tkinter library for a better user experience. The model mostly requires a good GPU to produce efficient results but I was still able to run the code on my pc and get some results out of it.

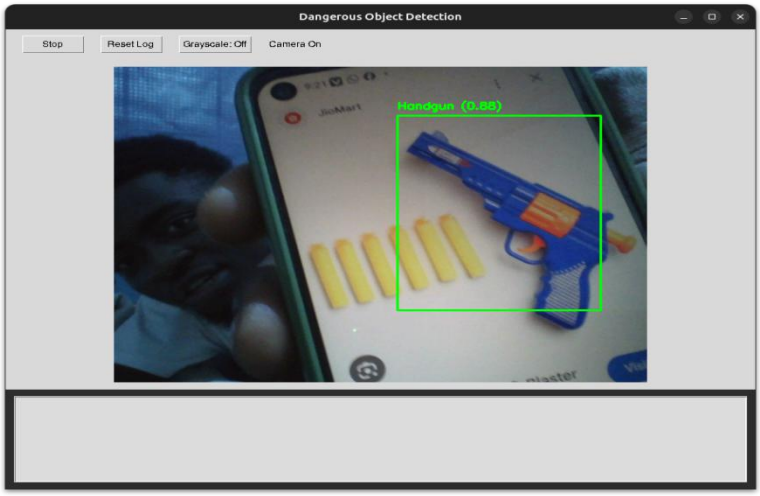


Fig 4.5: testing model with life pictures

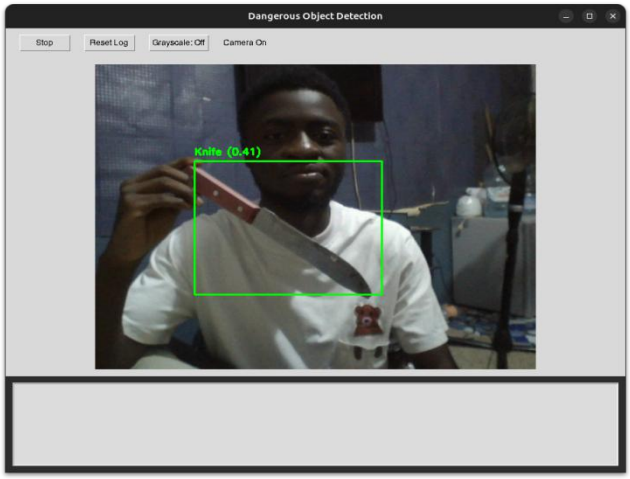


Fig 4.6: Testing model with a *knife*

The model was detecting some of the most common dangerous objects and the results were satisfying.

On fig 4.5 above, the image shows a toy gun obtained from the internet to test if our model would still recognize it as a dangerous object. And from the results obtained, the model was pretty much at around 88% that the object is a handgun and classified it dangerous.

Whereas on fig 4.6, U used a knife for testing and demonstration and the model was pretty much sure the object i held was a knife at around 41%. I have difficulties getting a higher accuracy for

the knife due to the fact that the training data I used did not contain this particular type of knife, but the results weren't bad.

I later on decided to make my interface more user-friendly by enabling fonts, enabling the alert system and making it easier for the user to interact with the system through buttons and not commands on the GUI interface still using tkiner as shown on the image below .

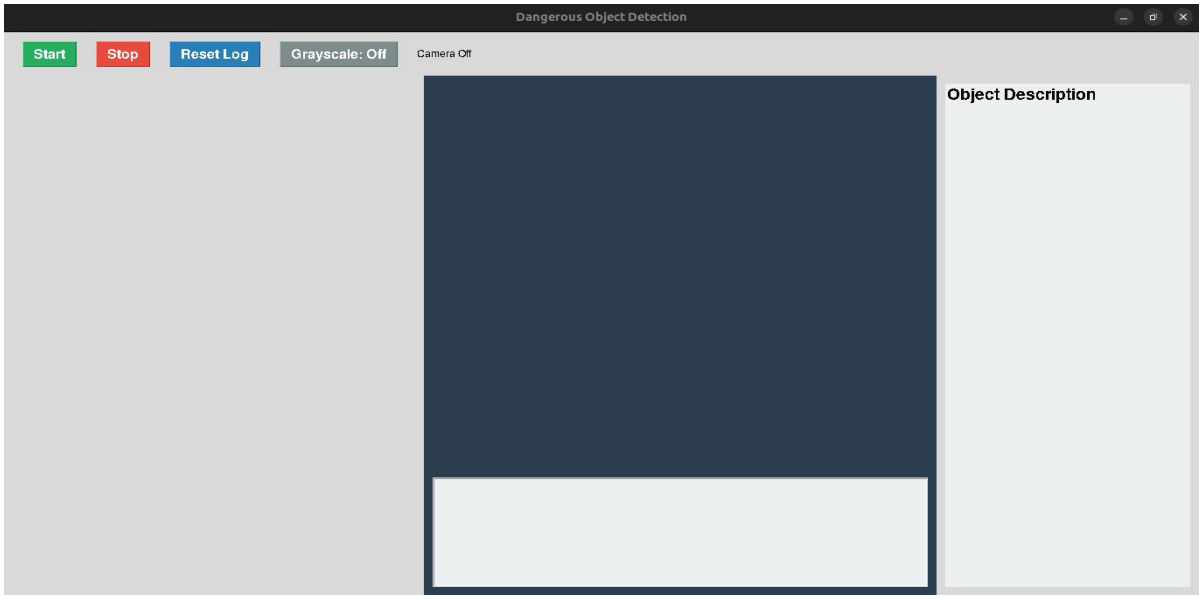


Fig 4.7: GUI for our object detection system

As shown on the diagram above, the user simply has to press the start button to activate the camera and start the frame processing of the object detection model. Upon activation of the camera, if an object is detected, the program automatically sends data to the log below on the GUI interface and provides the description of the detected object on the right panel and also triggers an alarm.

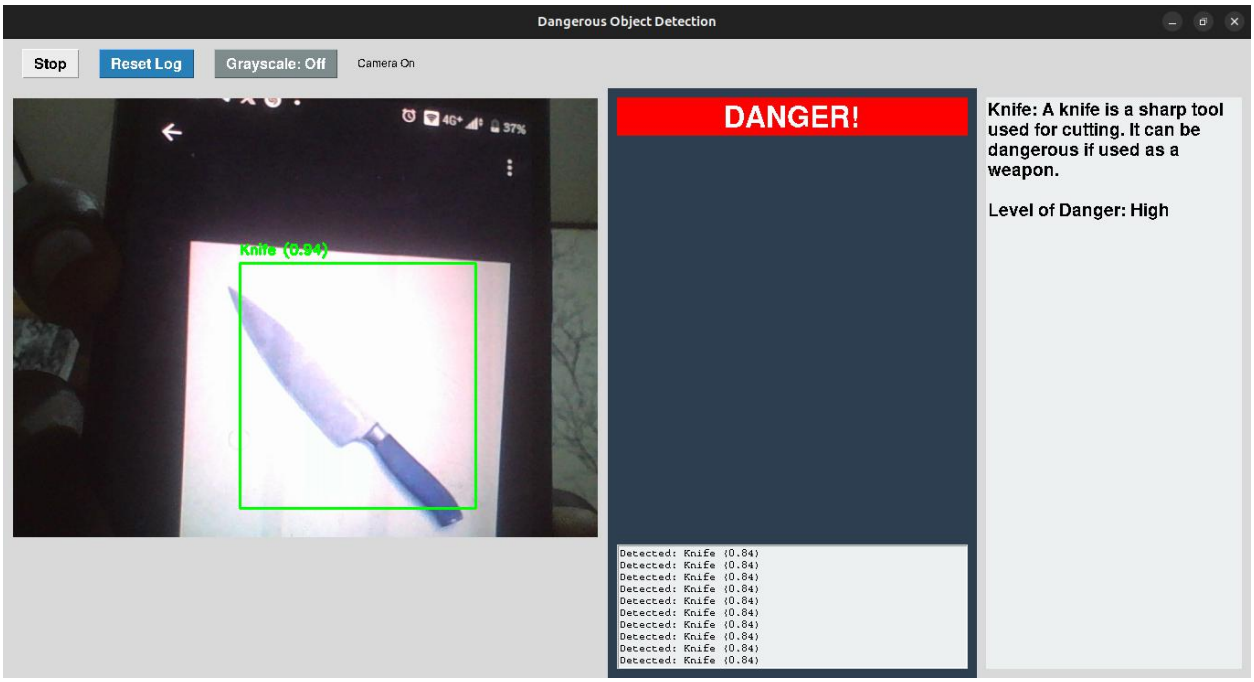


Fig 4.8: demonstrating system performance with GUI interface

The fig 4.8 above shows how our system, being active, waits to capture possible dangerous objects, in this case, we used a knife picture. As shown on the fig, when the knife is detected, it sends messages to the log showing the object detected and how sure the system is. In this case, we had over 0.88 (88%) accurate which is pretty much clear, the object is really dangerous, and as mentioned, the right panel shows the description of the detected object where for our case, it's a knife and also shows the level of danger. Also the danger sign pops up for 10 seconds and later on disappears after a dangerous object has been detected. We also have the reset log, which simply resets the log messages and the object description updates as a new object is being detected.

4.4.1 Object Tracking:

Object tracking is a critical component of real-time video surveillance systems. It involves following the position of detected objects across subsequent video frames. In this project, we implemented a tracking system that activates upon detecting dangerous objects (e.g.,

knife, long-gun, handgun, sword) with high confidence. The tracker maintains the object's location for a predefined duration, ensuring continuous monitoring.

Implementation Details

Upon detection of a dangerous object, the tracker is initialized using the bounding box coordinates of the detected object. The cv2.TrackerCSRT_create() function from the OpenCV library is used to create a CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability) tracker. The CSRT tracker is known for its accuracy and robustness in challenging tracking scenarios.

To ensure efficient use of resources and avoid redundant processing, the tracking is limited to a predefined duration (TRACKING_DURATION). In this implementation, the tracking duration is set to 60 seconds.

The implementation of the object tracking feature significantly enhances the Dangerous Object Detection System by ensuring continuous monitoring of detected dangerous objects. This feature leverages the CSRT tracker from OpenCV, providing robust and accurate tracking capabilities. By integrating this feature into the Tkinter-based user interface, the system offers real-time feedback and improved usability, making it a valuable tool for video surveillance and security applications.

The fig below shows a detected object being tracked by the system

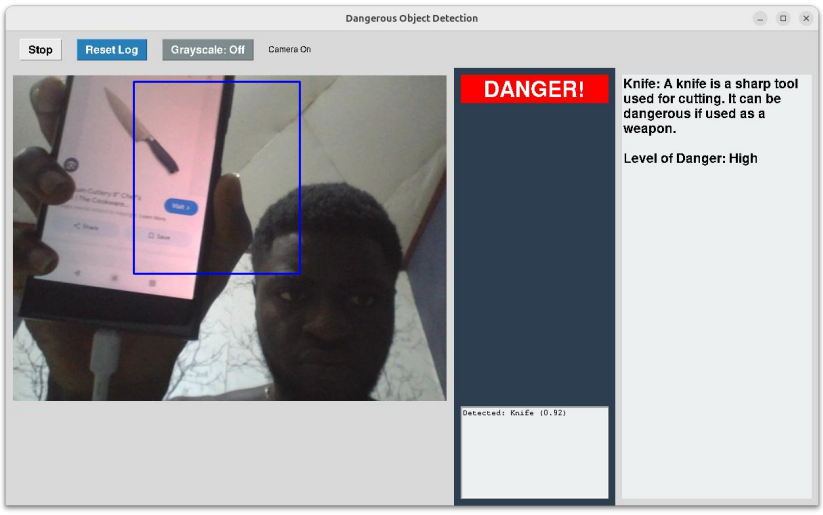


Fig 4.9: tracking detected object after the object was detected

4.5 Evaluation of the solution:

I tested the system for based on several different scenarios such as good lightening, grayscale transformation to see the reliability of my system based on different environmental conditions.

4.5.1 Testing model under artificial lightening

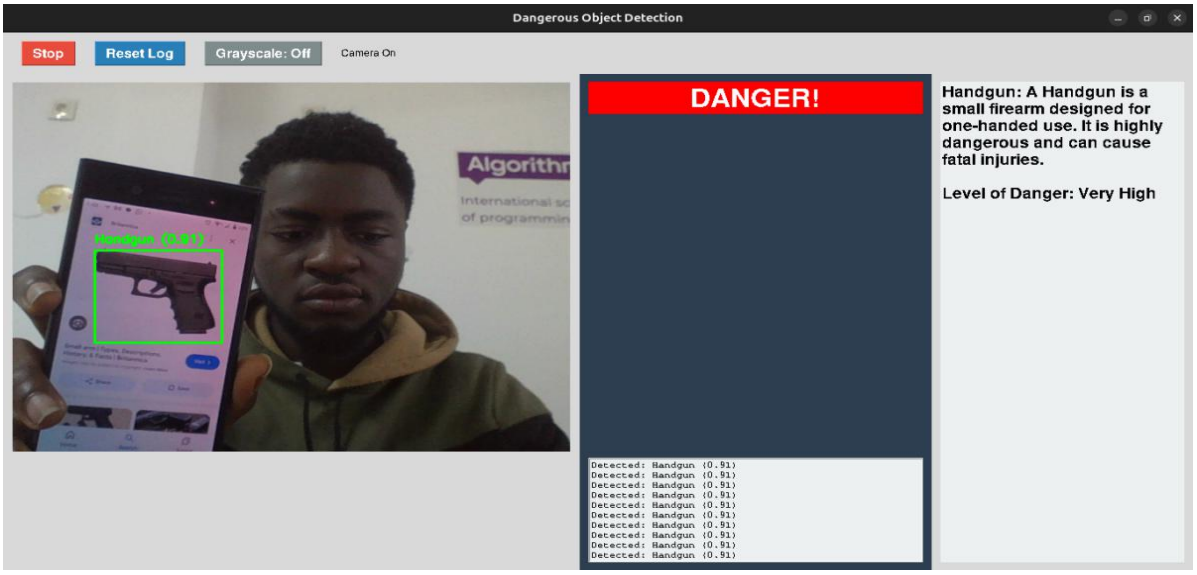


Fig 4.10: testing model using artificial light

The fig above shows the system being tested on in a closed building with artificial lights on and the model might encounter some little latency due to the low illumination and also due to the low end graphic card my PC has. But overall, the test was quite satisfying. The system was able to detect the corresponding object from the picture and display the results accordingly.

4.5.2 Testing model under natural light

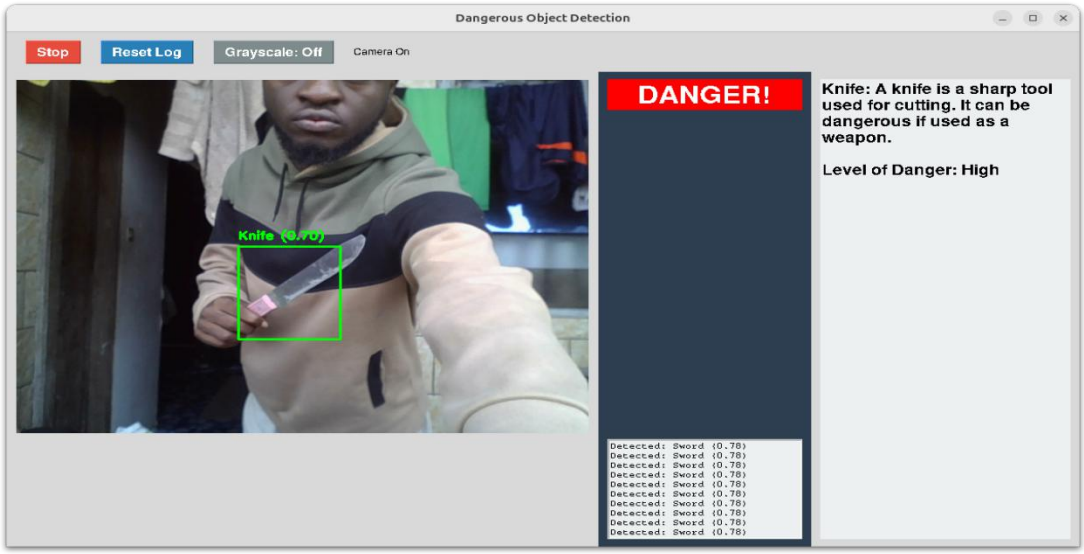


Fig 4.11: testing model with knife using natural light

On the fig above, I tested the model under natural lighting to see it's effectiveness. It provided slightly better results compared to low light illumination or artificial light. The system detected the knife accordingly and provided the following description.

4.5.3 Testing our model after implementing grayscale transformation

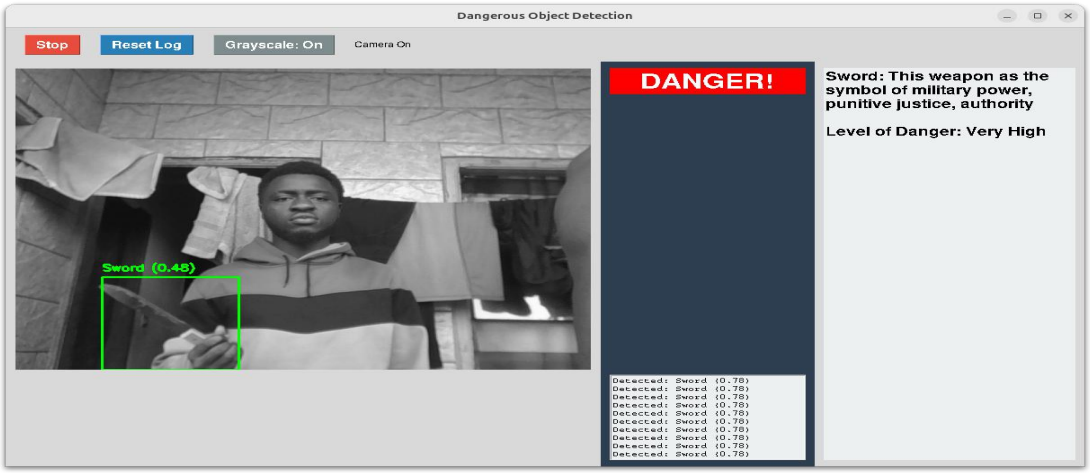


Fig 4.12: implementing grayscale transformation to test model

From the fig above, we clearly see i used a knife, and a sword was detected, in terms of the knives that was mainly used for our dataset, the common swords we have here look more like swords, so that's the reason for that detection. But overall, the results were rather good and the detection mechanism was kind of optimal.

I later on implemented an emailing system, that will send emails to users when the object has been detected in case the user was not facing his screen at the moment of detection so as enhancing the portability of the system.

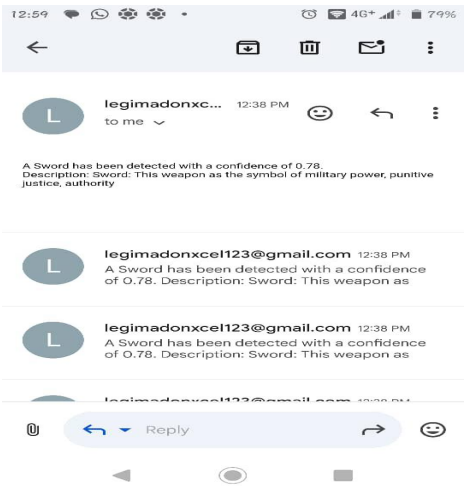


Fig 4:13 : Email sent after object detected

As shown above the fig 4.13 above shows how the emails are been sent to the corresponding user email box for more portability so users can easily be notified when an object has been detected. So it sends both the object detected and shows the description of the object itself

4.6: Partial conclusion:

The project successfully developed a real-time dangerous object detection system using video surveillance. The YOLOv5 model, known for its efficiency and accuracy, was effectively trained and deployed to detect weapons and other dangerous objects. The system's integration with a live video feed and its alert mechanism proved to be practical and user-friendly.

This project demonstrates the potential of AI and computer vision in enhancing security measures through automated surveillance. And this will really be of help to our community to reduce robbery events and provide more security to the citizens.

CHAPTER FIVE: CONCLUSION AND FURTHER WORKS

5.1 Summary of findings:

Based on my research and finalizing the project, i was able to come out with the following findings

Model Accuracy and Performance:

- The YOLOv5 model demonstrated high accuracy in detecting dangerous objects during both training and validation phases.
- Real-time detection was effective, with minimal latency, making it suitable for live surveillance applications.

System Integration and Functionality:

- The integration of the model with a video feed (using a webcam or surveillance camera) was successful.
- The system could process video frames in real-time, accurately detecting and identifying dangerous objects and tracking the detected objects.
- The alert system effectively triggered an alarm and displayed a warning message upon detecting a dangerous object.

Usability and Practicality:

- The system's user interface was intuitive, with clear indications when a dangerous object was detected.
- The alert mechanism was practical, providing immediate feedback to users without overwhelming them with continuous alerts.

- Providing email feedback to the corresponding users is also very practical.

5.2 Contribution to Engineering and Technology:

The development of a real-time dangerous object detection system for video surveillance represents a significant advancement in the fields of engineering and technology. This project leverages state-of-the-art machine learning and computer vision techniques to enhance security and safety measures in various environments. The contributions of this project to engineering and technology can be outlined in several key areas:

1. Advancement in Computer Vision and AI

Real-time Object Detection:

- The implementation of YOLOv5 (You Only Look Once) for detecting dangerous objects in real-time pushes the boundaries of what is possible with current computer vision technologies. YOLOv5 is renowned for its speed and accuracy, making it ideal for applications requiring real-time processing.

Model Training and Optimization:

- The project demonstrates effective techniques for training object detection models using large datasets. By optimizing the model and utilizing high-performance computing resources, significant improvements in detection accuracy and inference speed were achieved.

2. Enhancements in Security Systems

Automated Surveillance:

- This project showcases the potential for automated surveillance systems to detect and respond to security threats without human intervention. The ability to identify dangerous objects in real-time enhances the proactive security measures that can be deployed in public and private spaces.

Alert Mechanism:

- The integration of an alert system that triggers alarms and displays warnings when dangerous objects are detected contributes to the development of more responsive and interactive security systems. This feature ensures immediate attention and response to potential threats, thereby increasing overall safety.

3. Practical Applications and Impact

Public Safety:

- The deployment of such a system in public spaces such as airports, schools, and stadiums can significantly enhance public safety. By detecting weapons and other dangerous objects, the system helps prevent potential security incidents.

Private Security:

- For private properties and businesses, the system provides an additional layer of security. It can be used to monitor premises and provide real-time alerts to security personnel, improving response times and preventing incidents.

4. Technological Innovation

Integration with Existing Systems:

- The project demonstrates how advanced AI models can be integrated with existing surveillance infrastructure. This compatibility ensures that organizations can upgrade their security systems without needing to overhaul their current setups completely.

Scalability and Adaptability:

- The system's design allows for scalability and adaptability. It can be deployed in various environments and tailored to specific needs, making it a versatile solution for different security challenges.

5. Contribution to Research and Development

Open-Source Development:

- The use of open-source tools and frameworks like YOLOv5 and PyTorch promotes further research and development in the field. By sharing the project's methodologies and findings, other researchers and developers can build upon this work, leading to continuous innovation.

Data Annotation and Model Improvement:

- The project contributes to the ongoing improvement of object detection models by providing insights into effective data annotation and model training techniques. This knowledge can be used to enhance the performance of other AI models in similar applications.

The real-time dangerous object detection system for video surveillance developed in this project represents a substantial contribution to engineering and technology. By advancing the capabilities of computer vision and AI, enhancing security systems, and providing practical applications that improve public and private safety, this project underscores the transformative potential of modern technological solutions in addressing critical security challenges.

The contributions made by this project pave the way for future innovations and improvements in automated surveillance systems, ultimately leading to safer and more secure environments worldwide.

5.3 Recommendations:

1. Improve Model Performance

- Model Optimization: Optimize the YOLOv5 model for faster inference times without compromising accuracy. Techniques such as quantization and pruning can help reduce model size and increase speed.
- Transfer Learning: Utilize transfer learning by fine-tuning pre-trained models on a custom dataset. This can significantly reduce training time and improve performance, especially when labeled data is limited.

2. Integrate Advanced Hardware

- Edge Computing: Deploy the system on edge devices with powerful GPUs or TPUs to ensure real-time processing capabilities. Edge computing reduces latency and dependence on cloud resources.
- High-Resolution Cameras: Use high-resolution cameras to capture clearer images, which can improve detection accuracy, especially for small or obscured objects.

3. Enhance Alert and Notification System

- Multi-Modal Alerts: Implement multiple alert methods, such as SMS, email, and push notifications, in addition to audible alarms. This ensures that alerts are received promptly even if one communication channel fails.

4. User Interface and Experience

- Intuitive Dashboard: Develop a user-friendly dashboard that allows security personnel to monitor live feeds, view alerts, and access historical data. The interface should be intuitive and provide easy access to critical information.
- Mobile App Integration: Create a mobile application that enables remote monitoring and control of the surveillance system. This provides flexibility and immediate access to alerts from anywhere.

5. Scalability and Flexibility

- Modular Design: Design the system with modular components to allow easy scalability. This enables the addition of more cameras or sensors without significant changes to the core system.
- Cloud Integration: Integrate cloud services for data storage and processing, providing scalability and enabling access to advanced analytics and machine learning services.

6. Security and Privacy

- Data Encryption: Ensure that all data transmitted between cameras, servers, and user devices is encrypted to protect against interception and unauthorized access.

- Access Control: Implement robust access control mechanisms to restrict system access to authorized personnel only. Use multi-factor authentication (MFA) for an additional layer of security.
- Privacy Compliance: Ensure the system complies with relevant privacy regulations and standards, such as GDPR or CCPA. Implement measures to anonymize individuals in the video feed where necessary.

7. Regular Maintenance and Updates

- Model Updates: Regularly update the detection model to incorporate the latest advancements in object detection algorithms and techniques. This ensures continuous improvement in detection accuracy and performance.
- System Maintenance: Perform regular maintenance checks on hardware components, such as cameras and edge devices, to ensure optimal performance and longevity.

8. Community and Collaboration

- Open Source Contribution: Consider contributing to the open-source community by sharing non-sensitive parts of the system, such as data annotation tools or model training scripts. This can foster collaboration and further advancements in the field.
- Collaboration with Experts: Collaborate with experts in computer vision, AI, and security to continually refine and improve the system. Partnerships with academic institutions or industry leaders can provide valuable insights and resources.

9. Future Research and Development

- Advanced Detection Techniques: Explore the use of advanced detection techniques, such as multi-scale feature extraction and attention mechanisms, to improve the system's ability to detect small or partially obscured objects.
- Behavior Analysis: Integrate behavior analysis algorithms to detect suspicious activities or movements, providing an additional layer of security beyond object detection.

5.4 Difficulties encountered:

- The system's performance can be affected by factors such as low light conditions or poor video quality. I noticed my camera's refresh rate is slower in low light and the model is less accurate in such conditions which might lead to bad predictions or no predictions at all.
- The accuracy of detection depends on the quality and variety of the training dataset. More diverse data can improve detection capabilities. As I mentioned before, some of the materials or tools we have may not be common which might lead to our not being able to recognize such objects.

- The current setup requires a dedicated GPU for optimal performance, which might not be feasible for all deployment scenarios. Especially for my case where my GPU has a very low operational unit which may lead to inefficiency in the predictions of our dangerous objects.

5.5 Further works :

1. Integration with Additional Sensors

- **Thermal Cameras**: Integrate thermal imaging cameras to detect objects in low-light or no-light conditions. This can enhance detection capabilities in environments where traditional cameras may struggle.
- **Audio Sensors**: Incorporate audio sensors to detect sounds associated with dangerous activities, such as gunshots or explosions. Combining audio and visual data can improve the system's ability to detect threats.

2. Enhanced Detection Capabilities

- **Multi-Class Detection**: Expand the system's detection capabilities to include a wider range of dangerous objects and activities. Training the model on diverse datasets can improve its ability to recognize various threats.
- **3D Object Detection**: Research and implement 3D object detection techniques to improve the accuracy of object localization and classification, especially in complex environments.

3. Improved Real-Time Processing

- **Optimized Algorithms**: Continue to optimize the detection algorithms for faster real-time processing. Exploring lightweight architectures or advanced computational techniques can reduce latency and improve performance.
- **Distributed Processing**: Implement distributed processing across multiple edge devices to handle high-resolution video feeds and complex detection tasks more efficiently.

4. Mobile Intergration

Making the application compatible to run in mobile device will be very useful and will help for portability and ease of use

5. User Interface Enhancements

- **Augmented Reality (AR)**: Integrate AR technology to provide security personnel with real-time overlay information on detected threats, improving situational awareness and response times.
- **Customizable Dashboards**: Develop highly customizable dashboards that allow users to tailor the interface to their specific needs, enabling more efficient monitoring and management.

REFERENCES:

1) Bochkovsky, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. Retrieved from https://arxiv.org/abs/2004.10934

2) Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. Retrieved from https://arxiv.org/abs/1804.02767

3) Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(6), 1137-1149. doi:10.1109/TPAMI.2016.2577031

4) Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2017). Focal Loss for Dense Object Detection. Retrieved from https://arxiv.org/abs/1708.02002

5) Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Retrieved from https://arxiv.org/abs/1512.02325

6) Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*. Retrieved from https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

7) Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. Retrieved from https://arxiv.org/abs/1412.6980

8) He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/CVPR.2016.90

9) Bochkovsky, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. Retrieved from https://arxiv.org/abs/2004.10934

10) Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/CVPR.2016.91

11) Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/CVPR.2017.690

12) Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. Retrieved from https://arxiv.org/abs/1804.02767

13) Glenn Jocher et al. (2020). YOLOv5 by Ultralytics. GitHub Repository. Retrieved from https://github.com/ultralytics/yolov5

14) Shipman, J. W. (2013). Tkinter 8.5 reference: a GUI for Python. New Mexico Tech Computer Center. Retrieved from http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/

15) Viseum. (n.d.). Intelligent CCTV. Retrieved from https://www.viseum.co.uk/intelligent-cctv/.

16) Avigilon. (n.d.). Avigilon Control Center (ACC). Retrieved from https://www.avigilon.com/products/software/acc

17) Genetec. (n.d.). Genetec Security Center. Retrieved from https://www.genetec.com/solutions/all-products/security-center

18) Intelligent Security Systems (ISS). (n.d.). SecurOS. Retrieved from https://issivs.com/securos/

19) BriefCam. (n.d.). BriefCam: Video Content Analytics Solutions. Retrieved from https://www.briefcam.com/

APPENDICES

```
%cd /content/yolov5
#after following the link above, recieve python code with these fields filled in
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="p0dlsL9pnZaZvdweJAr9")
project = rf.workspace("hit-product").project("weapon-detection-c9jaq")
version = project.version(2)
dataset = version.download("yolov5")# define number of classes based on YAML
import yaml
with open(dataset.location + "/data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])
%%writetemplate /content/yolov5/models/custom_yolov5s.yaml

# parameters
nc: {num_classes}  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
```

```
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, BottleneckCSP, [1024, False]],  # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, BottleneckCSP, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, BottleneckCSP, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, BottleneckCSP, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, BottleneckCSP, [1024, False]],  # 23 (P5/32-large)
```

```
   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  %%writetemplate /content/yolov5/models/custom_yolov5s.yaml

# parameters
nc: {num_classes}  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32
# train yolov5s on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 16 --epochs 30 --data {dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml --weights 'yolov5s.pt' --name yolov5s_results  --cache
%cd /content/yolov5/
!python detect.py --weights runs/train/yolov5s_results3/weights/best.pt --img 416 --conf 0.4 --source /content/yolov5/Weapon-detection-2/test/images
%cp /content/yolov5/runs/train/yolov5s_results3/weights/best.pt /content/gdrive/My\ Drive
```