**SENTINEL: INTELLIGENT MULTI CAMERA FACE DETECTION, RECOGNITION AND TRACKING SYSTEM**

## A PROJECT REPORT

***Submitted by,***

**ISRAR AHMED - 20201CAI0107**

**RISHI RAGAV V - 20201CAI0128**

**RAKSHITH M B - 20201CAI0117**

**MOHD FAIZAN USMAN SAIT - 20201CAI0090**

### *Under the guidance of,*

**Mr. SHEIK JAMIL AHMED**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**At**



**PRESIDENCY UNIVERSITY**

**BENGALURU**

**JANUARY 2024**

**PRESIDENCY UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE ENGINEERING**

**CERTIFICATE**

This is to certify that the Project report **“SENTINEL: INTELLIGENT MULTI CAMERA FACE DETECTION, RECOGNITION AND TRACKING SYSTEM”** being submitted by “ISRAR AHMED”, “RISHI RAGAV V”, “RAKSHITH M B”, “MOHD FAIZAN USMAN SAIT” bearing roll number(s) “20201CAI0107”, “20201CAI0128”, “20201CAI0117”, "20201CAI0090” in partial fulfilment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

|  |  |
| --- | --- |
| **Mr. SHEIK JAMIL AHMED**  Assistant Professor  School of CSE&IS  Presidency University | **Dr. ZAFAR ALI KHAN**  Associate Professor & HoD  School of CSE&IS  Presidency University |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Dr. C. KALAIARASAN**  Associate Dean  School of CSE&IS  Presidency University | **Dr. L. SHAKKEERA**  Associate Dean  School of CSE&IS  Presidency University | **Dr. SAMEERUDDIN KHAN** Dean  School of CSE&IS  Presidency University |

**PRESIDENCY UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE ENGINEERING & INFORMATION SCIENCE**

**DECLARATION**

We hereby declare that the work, which is being presented in the project report entitled **SENTINEL: INTELLIGENT MULTI CAMERA FACE DETECTION, RECOGNITION AND TRACKING SYSTEM** in partial fulfilment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Mr. SHEIK JAMIL AHMED, ASSISTANT PROFESSOR,** **School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

**Signature**

**ISRAR AHMED 20201CAI0107**

**RISHI RAGAV V 20201CAI0128**

**RAKSHITH M B 20201CAI0117**

**MOHD FAIZAN USMAN SAIT 20201CAI0090**

**ABSTRACT**

Sentinel is an innovative multi-camera system revolutionizing video surveillance through intelligent face detection, recognition, and tracking. Employing advanced computer vision and deep learning models, the system ensures accurate face detection under challenging conditions, enhancing its reliability in diverse environments. By integrating facial recognition technology with a comprehensive database, Sentinel enables rapid identification and alerts for known individuals of interest. The system's intelligent tracking algorithms allow for seamless monitoring and tracking of individuals across multiple camera feeds, mitigating the limitations of conventional surveillance. Sentinel's scalability ensures compatibility with various surveillance camera systems, promoting widespread adoption. Additionally, the system prioritizes privacy by adhering to ethical data handling practices, securely managing facial data in compliance with privacy regulations. Sentinel's cutting-edge architecture marks a significant leap forward in video surveillance capabilities, offering a comprehensive solution for real-time face detection, recognition, and tracking. Its applicability spans across sectors such as public safety, law enforcement, and critical infrastructure protection, making it a vital tool in bolstering security measures and ensuring efficient surveillance operations.

**ACKNOWLEDGEMENT**

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. Kalaiarasan C and Dr. Shakkeera L,** School of Computer Science Engineering & Information Science, Presidency University and **Dr.** **ZAFAR ALI KHAN,** Head of the Department, School of Computer Science Engineering & Information Science, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Mr. SHEIK JAMIL AHMED, ASSISTANT PROFESSOR**, School of Computer Science Engineering & Information Science, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators **Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS** and also the department Project Coordinators “**Dr. MULARI PAREMESWARAN**”.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

**ISRAR AHMED**

**RISHI RAGAV V**

**RAKSHITH M B**

**MOHD FAIZAN USMAN SAIT**

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Figure Name** | **Caption** | **Page No.** |
| 1  2  3  4  5  6  7  8  9  10 | Fig 5.1  Fig 5.2  Fig 7.1  Fig 8.1  *Fig 9.1*  *Fig 9.2*  *Fig 9.3*  *Fig 9.4*  *Fig 9.5*  *Fig 9.6* | Architecture of YOLOv8 in 3D  Architecture of YOLOv8 in 2D  Architecture of the system  Gantt Chart  Results of the model  Labels Correlogram  Labels  Normalized Confusion Matrix  Precision-Recall Curve  Training information graph | 14  14  31  40  42  42  43  43  44  44 |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **ACKNOWLEDGEMENT**  **LIST OF FIGURES** | **iv**  **v**  **vi** |
| **1.** | **INTRODUCTION** | 1 |
|  | 1.1 Overview | 1 |
|  | 1.2 Problem Statement | 2 |
|  | 1.3 Existing System | 5 |
|  |  | 8 |
|  | 1.2.2.2 . . . . . . . . | 10 |
|  | 1.2.2 | 12 |
|  | 1.3 . . .. . . . . . .. . . . . . . | 13 |
|  | 1.4 . . . . . . . . . . . . . . . . | 15 |
| **2.** | **LITERATURE REVIEW** | **16** |
|  | 2.1 GENERAL | 17 |
|  | 2.2 . . . . .. | 19 |
|  | 2.2. | 20 |

**CHAPTER-1**

**INTRODUCTION**

* 1. **Overview**

Video surveillance has undergone a remarkable transformation, evolving from rudimentary closed-circuit systems into an essential pillar of modern security and monitoring technologies. This progression reflects the increasing importance of visual data capture and analysis in today's world. Video surveillance involves the deployment of cameras, sensors, and recording devices to monitor specific locations continuously, generating a wealth of visual data for various applications. This discussion delves into the multifaceted world of video surveillance, exploring its historical evolution, technological advancements, and the diverse range of sectors that rely on this crucial tool. The concept of video surveillance can trace its roots back to the mid-20th century when closed-circuit television (CCTV) systems were initially introduced. These early systems, often associated with banks and governmental facilities, consisted of analog cameras connected to monitors and recording devices. While these systems had limited capabilities, they marked the inception of a technology that would go on to reshape the landscape of security and monitoring. As technology advanced, so did video surveillance. The transition from analog to digital systems in the late 20th century was a pivotal moment in the field. Digital video surveillance introduced many benefits, including higher resolution, increased storage capacity, and more flexible data management. These systems allowed for remote monitoring, enabling users to access live or recorded footage from virtually anywhere with an internet connection. Moreover, digital cameras could be integrated with other technologies, such as facial recognition software and motion detection, enhancing their effectiveness for various applications. The 21st century ushered in an era of rapid innovation in video surveillance technology. High-definition (HD) and ultra-high-definition (UHD) cameras became the new standard, providing exceptional image clarity and detail. These cameras are now capable of capturing images in low-light conditions and adverse weather, further expanding the range of scenarios where video surveillance can be applied. The advent of artificial intelligence (AI) and machine learning algorithms brought a new dimension to video surveillance. These technologies enable automated analysis of video data, allowing systems to detect anomalies, recognize faces, and track objects in real-time.

Such capabilities have increased the accuracy and efficiency of video surveillance systems. The applications of video surveillance have also proliferated over time. Initially, it was primarily used in high-security environments like government facilities, banks, and casinos. However, as technology became more accessible and affordable, its use expanded into various sectors. Today, video surveillance is integral to public safety, urban planning, transportation, retail, residential security, and much more. In law enforcement and public safety, video surveillance is pivotal in crime prevention, investigation, and community protection

* 1. **Problem Statement**

Video surveillance aims to gather information, to prevent crime, protect property, person, or object and to inspect the scene of crime. The participants are required to build a pipeline that acquires image from multiple CCTV cameras and carry out face detection, face recognition and tracking of selected individuals.

1. Acquisition: Multiple static CCTV cameras are considered.

2. Face detection & Recognition: detect the faces and recognize the individuals

3. Multiple Person Tracking: Out of the recognized individuals, track target individuals across multiple cameras. The pipeline must have list of recognized individuals' details, from which the user can select target individuals.

* 1. **Existing System**

[1] ‘Probabilistic recognition of human faces from video by q Shaohua Zhou,\* Volker Krueger, and Rama Chellappa’ provides an existing solution. The research explores advanced methodologies for human face recognition in video surveillance, emphasizing a probabilistic framework. Investigating still-to-video and video-to-video scenarios, a novel time series state space model is introduced to integrate temporal information. This model simultaneously characterizes kinematics and identity using motion vectors and identity variables. For still-to-video recognition, a tracking-and-recognition approach is proposed, addressing challenges such as poor video quality and pose variations. In video-to-video recognition, the model generalizes still templates to video sequences, employing exemplar-based learning. The methodology is validated through experiments on datasets with pose/illumination variations, demonstrating its efficacy in dynamic surveillance environments.

**Drawbacks**

Despite its advancements, the proposed approach may face challenges in scalability and computational complexity, particularly when dealing with extensive video datasets. The integration of exemplar-based learning introduces a dependency on the quality and representativeness of selected video representatives, which might affect recognition accuracy. Additionally, the generalizability of the model across diverse surveillance scenarios requires careful consideration of image representations and transformations. Balancing computational efficiency and model adaptability remains an ongoing concern, necessitating further optimization for real-world applications.

**CHAPTER-2**

**LITERATURE SURVEY**

**2.1 Literature Survey**

[2] Liu, Z., Luo, D., Wang, Y., Wang, L., Tai, Y., Wang, C., Li, J., Huang, F., & Lu, T. (2020). "TEINet: Towards an Efficient Architecture for Video Recognition."

In recent years, video recognition architectures, particularly in the context of action recognition, have been a focal point in computer vision research. While 3D Convolutional Neural Networks (CNNs) have shown remarkable progress in this domain, their computational cost remains a challenge compared to their 2D counterparts. The proposed TEINet (Temporal Enhancement-and-Interaction Network) addresses this issue by introducing a novel Temporal Enhancement-and-Interaction (TEI) module, which is seamlessly integrated into existing 2D CNNs. This module employs a two-stage approach, incorporating a Motion Enhanced Module (MEM) to enhance motion-related features and a Temporal Interaction Module (TIM) to capture contextual information efficiently. This flexible and principled design not only captures temporal structures effectively but also maintains high inference efficiency. Through experiments on benchmark datasets like Something-Something and Kinetics, TEINet demonstrates superior performance over previous approaches while ensuring fast inference speeds, highlighting its potential as an efficient architecture for video recognition.

[1] Zhou, S., Krueger, V., & Chellappa, R. (2003). "Probabilistic recognition of human faces from video." The paper explores probabilistic video analysis for human face recognition, focusing on a comprehensive investigation using a gallery of still or video images and a probe set of videos. The authors propose a time series state space model for still-to-video recognition, employing a motion vector and identity variable to fuse temporal information. The model, enhanced by sequential importance sampling (SIS) algorithm, achieves robust recognition by propagating the identity variable over time. The study also extends to video-to-video recognition, introducing exemplar-based learning for adaptability. While face recognition has been extensively studied, the integration of temporal information in a unified probabilistic framework distinguishes this work. Empirical results demonstrate the effectiveness in challenging scenarios, emphasizing the significance of appropriate model choices for still-to-video and video-to-video recognition.

[3] Mohana and Dr. H. V. Ravish Aradhya. Design and Implementation of Object Detection, Tracking, Counting and Classification Algorithms using Artificial Intelligence for Automated Video Surveillance Applications. The paper addresses the critical domain of automated video surveillance through the design and implementation of object detection, tracking, counting, and classification algorithms using artificial intelligence (AI). Video surveillance is pivotal for safety and security, gaining increased importance due to global security concerns. The study emphasizes the shift from traditional surveillance to automated systems, reducing manual intervention and enhancing efficiency. The proposed algorithms leverage technologies such as MATLAB, DSP, FPGA, and AI methods, including convolutional neural networks (CNN) and YOLO-based models. The implementation explores real-time detection, tracking, and classification, highlighting applications in various scenarios. The integration of DSP, FPGA, and AI technologies demonstrates a comprehensive approach. The literature survey underscores the significance of intelligent surveillance systems for threat detection, activity analysis, and real-world applications, positioning the research within the evolving landscape of video surveillance.

[4] Dhaya, R. "CCTV Surveillance for Unprecedented Violence and Traffic Monitoring.” The paper addresses the growing need for monitoring traffic and unforeseen violence in both urban and rural areas through the development of a CCTV surveillance system. Emphasizing the importance of understanding events in real-time, the proposed method focuses on video synchronization and alignment using motion detection and contour filtering algorithms. Motion detection identifies object movements, including vehicles and unexpected activities, while contour filtering helps identify objects based on their colour. The synchronization and alignment process aims to provide detailed information about each object in the scenario. The algorithm, implemented in Java, utilizes open-source libraries. Validation of the model using real-time datasets demonstrated its efficiency compared to existing methods, with a notably faster outcome. The paper contributes to user-friendly video uploading interfaces and efficient low-dimensional frame matching for video ordering.

[5] Davies, A. C., & Velastin, S. A. (2005). "A Progress Review of Intelligent CCTV Surveillance Systems." Paper presented at the IDAACS’05 Workshop, Sofia, September 2005. The paper presents a comprehensive review of the historical development and current state of Intelligent Closed-Circuit Television (CCTV) Surveillance Systems. Addressing the ubiquitous deployment of CCTV cameras in urban environments, the study explores public attitudes categorized into concerns over privacy invasion and authoritarian control versus the appreciation for increased safety and reduced antisocial behaviour. The evolution of CCTV surveillance from simple camera systems to sophisticated multi-camera setups with computational intelligence for object recognition and scene analysis is discussed. Emphasizing the need for increased automation, the paper highlights the role of computer intelligence in detecting and analysing significant events, alerting human observers when necessary. The challenges of managing increasing camera numbers, video recording, and ensuring legal admissibility of CCTV footage are also examined. The literature provides valuable insights into the evolution, challenges, and applications of intelligent CCTV surveillance systems.

[6] Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2023). "Real-Time Flying Object Detection with YOLOv8.” This paper addresses the crucial challenge of real-time flying object detection, emphasizing the increasing threat posed by drones in various scenarios, including security and surveillance. The authors introduce a generalized model for detecting flying objects and a refined model suitable for real-world implementation. Leveraging transfer learning and the YOLOv8 single-shot detector, the study encompasses a comprehensive analysis of the architecture and functionality of YOLOv8. With a focus on the challenges associated with flying objects, the paper explores variances in spatial sizes, aspect ratios, and inference speed. The proposed models exhibit promising results, with the refined model achieving an improved mean average precision (mAP50-95) of 0.835 at an impressive inference speed. The significance of visual detection in countering the stealth capabilities of drones is emphasized, urging the integration of reliable detection technology.

[7] Thaler, M., & Bailer, W. (2013). Real-time person detection and tracking in panoramic video. The document is a research paper that proposes a novel method for multi-person tracking in basketball scenarios using single-camera video sequences. The method is based on multi-scale detection and pose estimation techniques. The paper also presents a dataset of 22 basketball sequences from European games, with a total of 1019 frames and 11339 ground truth bounding boxes. The dataset covers different basketball offensive plays, jersey colors, and skin tones1. The paper evaluates the detection and tracking performance using standard metrics such as precision, recall, and F1 score, and shows that the proposed method outperforms state-of-the-art methods in terms of these metrics. The paper also discusses the limitations of the proposed method and suggests directions for future work.

[8] Arbués-Sangüesa, A., Haro, G., & Ballester, C. (2019). Multi-Person tracking by multi-scale detection in Basketball scenarios. The objective is to present a machine learning approach for visual object detection that is fast and accurate. The approach is demonstrated on the task of face detection. A new image representation that allows for very fast feature evaluation. The features are like Haar basis functions and can be computed at any scale or location in constant time. AdaBoost is a learning algorithm that selects a small number of critical features from a large set and yields efficient classifiers. The features are chosen by minimizing a weighted error function. Cascade of Classifiers is defined as method for combining increasingly more complex classifiers in a cascade structure that discards non-face regions quickly while spending more computation on promising regions4. The cascade provides statistical guarantees that discarded regions are unlikely to contain faces. Results of the face detection system achieves high detection rates and low false positive rates on a difficult dataset. It can process images extremely rapidly (15 frames per second) without using any auxiliary information. It is also robust to various conditions such as illumination, scale, pose, and camera variation.

[9] Paul Viola and Michael Jones,Rapid Object Detection using a Boosted Cascade of Simple Features. This paper primarily discusses a machine learning approach for visual object detection that is both rapid and accurate. This approach is significant in the field as it offers a balance between speed and precision, which are often seen as trade-offs in this domain. A key concept introduced in the document is that of an ‘Integral Image’. This is a novel image representation that allows for the quick computation of features that resemble Haar basis functions. This representation is crucial for the efficiency of the proposed object detection method. The document also explains a learning algorithm known as ‘AdaBoost’. This algorithm is designed to select a small number of critical features from a larger set, resulting in efficient classifiers. The use of AdaBoost is integral to the object detection method proposed in the document. Furthermore, the document proposes a method for combining classifiers of increasing complexity in a ‘Cascade’ structure. This structure is designed to discard most of the negative sub-windows while retaining the positive ones, thereby improving the efficiency of the object detection process. Finally, the document demonstrates the effectiveness and efficiency of the proposed framework through an application in face detection. It shows that the method achieves high detection rates and low false positive rates on a challenging dataset, thereby validating the proposed approach. This practical application serves as a strong testament to the potential of the proposed object detection method.

[10] Oluwatoyin P. Popoola and Kejun Wang Video-Based Abnormal Human Behavior Recognition—A Review .The document under review is a comprehensive study on video-based abnormal human behaviour recognition, a significant research area in computer vision and machine learning. The need for intelligent visual surveillance systems that can automatically detect and analyse abnormal events in various contexts such as security, healthcare, and multimedia is emphasized. The document provides an extensive overview of previous surveys and research directions in human motion analysis and behaviour recognition. It proposes a taxonomy based on the learning framework, scene density, and contextual types of anomalies. The definition and properties of abnormal events, which are rare, unexpected, atypical, and out-of-the-ordinary, are discussed. These properties depend on the scene context and semantics. The main approaches and techniques for feature extraction, behaviour representation, and behaviour modelling are described. These include rule-based, statistical, clustering, Markov, and topic models. The document compares the advantages and disadvantages of supervised, unsupervised, and semi supervised methods for learning models of normal and abnormal behaviors from labeled or unlabeled data. The impact of the number and complexity of moving objects in the scene on the choice of methods for behavior characterization and anomaly detection is analyzed. The document distinguishes between uncrowded and crowded scenes. It summarizes different types of anomalies and datasets that are available for different contexts, such as public places, private environments, single-person or group behaviors, and spatial or temporal anomalies. This document serves as a valuable resource for researchers and practitioners in the field of video-based abnormal human behavior recognition.

**CHAPTER-3**

**RESEARCH GAPS OF EXISTING METHODS**

The current research methods are not focused on tacking the issues of real time data. As this process is too time-consuming. Some environmental factors might also come into play. These factors might be low light conditions, lower quality of camera, distance between the camera and the subject. Multi camera capabilities require the system to have synchronized data with continuous stream to have proper concurrency capabilities.

The research gaps in the current methods:

1. Scalability and Computational Complexity:

- Issue: Dealing with extensive video datasets may pose challenges in terms of scalability and computational complexity.

- Research Gap: There is a need for research focused on developing more scalable and computationally efficient algorithms that can handle large-scale video datasets, ensuring real-time processing and analysis.

2. Dependency on Exemplar-Based Learning:

- Issue: The integration of exemplar-based learning introduces a dependency on the quality and representativeness of selected video representatives.

- Research Gap: Future research could explore alternative learning methods or improvements to exemplar-based learning to enhance the robustness and generalizability of the model, especially under varying environmental conditions.

3. Generalizability Across Diverse Surveillance Scenarios:

- Issue: The generalizability of the model across diverse surveillance scenarios requires careful consideration of image representations and transformations.

- Research Gap: There is a need for research that focuses on developing models that are more adaptable and robust to variations in environmental conditions, such as low light, lower camera quality, and varying distances between the camera and subjects.

4. Real-Time Data Processing:

- Issue: The current research methods are not focused on addressing real-time data processing challenges, and the process is deemed time-consuming.

- Research Gap: Future research should prioritize the development of real-time face recognition algorithms, considering factors such as efficient data processing, low-latency responses, and adaptability to dynamic environments.

5. Environmental Factors:

- Issue: Environmental factors like low light conditions, lower camera quality, and varying distances between the camera and subjects can impact recognition accuracy.

- Research Gap: Research efforts should be directed towards designing algorithms that are resilient to environmental challenges, incorporating features or techniques that can enhance performance in adverse conditions.

6. Multi-Camera Capabilities:

- Issue: Multi-camera capabilities require synchronized data with continuous streams to ensure proper concurrency capabilities.

- Research Gap: There is a need for research focused on developing efficient methods for handling multi-camera systems, ensuring synchronization, and addressing challenges related to data concurrency to enable effective surveillance across different viewpoints.

In summary, addressing these research gaps could lead to the development of more robust, scalable, and adaptable face recognition systems for video surveillance, better suited for real-world applications with diverse and challenging conditions.

**Chapter 4**

**SYSTEM REQUIREMENTS SPECIFICATIONS**

The core document in the software development process is the Software Requirement Specification, commonly referred to as SRS. Acting as the foundational phase, the SRS outlines the system's requirements and provides a detailed description of each feature within those requirements. It serves as a comprehensive representation of the organization's understanding of the customer's system requirements, emphasizing the interrelationships among all system components. The SRS fosters mutual understanding between the client and the organization, nurturing a healthy and transparent relationship. Functioning as a blueprint for implementation, it serves as a guide for the allocation of organizational resources, including workforce and time, offering insights into costs. The SRS acts as the guardian throughout the software development phases, providing accurate guidance for actions in areas such as design specifications, documentation, software architecture specification, and testing and validation plans.

**4.1 Functional Requirements**

**1.** **Enhanced Face Detection:** Develop a highly accurate face detection algorithm that can efficiently identify and locate faces in real-time, regardless of variations in lighting conditions, angles, and facial expressions.

**2. Robust Face Recognition:** Create a robust face recognition system that can match detected faces to a pre-defined database of individuals, allowing for the positive identification of persons of interest and generating alerts as necessary.

**3. Multi-Camera Integration:** Implement the capability to seamlessly integrate with multiple cameras across a surveillance network, enabling simultaneous monitoring and tracking of individuals across different camera views.

**4. Real-Time Tracking:** Develop an intelligent tracking system that can monitor and track recognized faces as they move within the camera network, providing real-time information on their whereabouts and activities.

**5. Alert and Reporting System:** Design a comprehensive alert and reporting system that can automatically generate alerts for suspicious activities, unauthorized access, or individuals on watchlists, and provide detailed reports for post-incident analysis and evidence gathering.

**4.2 Non-Functional Requirements**

**1. Behavioral Analysis:** Incorporate behavioral analysis algorithms to detect and alert on unusual or suspicious activities, such as loitering, sudden crowd dispersal, or unusual gestures, enhancing the system's overall security capabilities.

**2. User-Friendly Interface:** Develop an intuitive and user-friendly interface for security personnel to interact with the system, allowing for easy configuration, monitoring, and response to incidents, and potentially enabling remote access and control for authorized personnel.

**4.2.1 Resource Requirements**

A dataset is a structured and organized collection of data that is systematically arranged for a specific purpose, often in a digital format. It serves as a foundational component in various fields, including machine learning, statistics, and scientific research. Typically, a dataset comprises individual data points or instances, each representing a unique observation, and is organized into variables or features that describe specific characteristics of those instances. Datasets can be diverse, ranging from numerical and textual information to images, audio, or video files. They play a crucial role in training and evaluating machine learning models, conducting statistical analyses, and deriving meaningful insights from large volumes of information. Properly curated datasets are essential for the accuracy and reliability of analyses and model outcomes in diverse domains.

Roboflow is a comprehensive platform designed to simplify and streamline the process of working with computer vision datasets. It offers a range of tools and services that assist researchers, developers, and data scientists in managing, augmenting, and preparing image datasets for machine learning projects. Roboflow provides features such as dataset annotation, image preprocessing, and transformation capabilities, making it easier to ensure data quality and suitability for training models. The platform supports various image formats and annotations, facilitating compatibility with popular machine learning frameworks. With Roboflow, users can efficiently organize and enhance their datasets, accelerating the development and deployment of computer vision applications by automating essential steps in the data preparation pipeline. The platform contributes to advancing the accessibility and effectiveness of computer vision by offering a user-friendly environment for dataset management.

**4.2.2 Software Requirements**

Google Colab : Model Training

Roboflow: Model creation and Instantiation

Visual Studio Code: Code Editor

Github: Collaborative Coding Platform

OS: Windows 11

**4.2.3 Hardware Requirements**

Processor: i5-7400

Graphical Processing unit: 8GB RTX 4060

Mouse: 3 Button

Keyboard: Standard 102 Keys

Monitor: 18’’

**Summary**

This chapter gives information about the requirements of the system such as functional and non-functional requirements, hardware, and software requirements etc.

**CHAPTER-5**

**PROPOSED METHODOLOGY**

**5.1 Proposed Methodology**

It was found that computer vision is a data heavy and computationally expensive process that requires a powerful system and high quality as well extensive dataset. Hence models were trained in the server side so that the user can focus on other parts of their project. The application of edge computing where the use of training data locally on the device itself has become popular. To facilitate this process, there have been improvements made in computer vision and machine learning algorithms. One such example is YOLOv8 whose highlighting features are transferring learning, improved loss function and improved speed for data crunching of large datasets.

A diagram of a block diagram

Description automatically generated

*Fig 5.1 Architecture of YOLOv8 in 3D*

A screenshot of a computer

Description automatically generated

*Fig 5.2 Architecture of YOLOv8 in 2D*

The algorithm used for this project is YOLOv8. YOLOv8 is a state-of-the-art object detection algorithm that uses a single convolutional neural network to predict bounding boxes and class probabilities for multiple objects in an image. YOLOv8 is faster and more accurate than previous versions of YOLO, and it also supports other tasks such as segmentation, classification, and pose estimation.

The YOLOv8 algorithm consists of four main components: the backbone network, the neck network, the head network, and the loss function. Let’s look at each of them in detail.

1. The backbone network is the base feature extractor that takes an input image and produces a feature map that encodes the semantic and spatial information of the image. The backbone network used by YOLOv8 is CSPNet, which is a modified version of ResNet that uses cross-stage partial connections to reduce the number of parameters and increase the efficiency of the network.
2. The neck network is the intermediate layer that connects the backbone network and the head network. The neck network performs feature fusion and context aggregation to enhance the feature map and make it suitable for the detection task. The neck network used by YOLOv8 is PANet, which is a bottom-up and top-down pathway that combines low-level and high-level features and applies spatial attention to highlight the regions of interest.
3. The head network is the final layer that outputs the predictions for the detection task. The head network used by YOLOv8 is YOLOF, which is an anchor-free detection head that does not rely on predefined anchor boxes to generate bounding boxes. Instead, it uses a dense sampling strategy to assign each pixel in the feature map to a potential object center, and then regresses the bounding box coordinates and class probabilities from the center point.
4. The loss function is the objective function that measures the difference between the predicted outputs and the ground truth labels. The loss function used by YOLOv8 is GIoU loss, which is a generalized version of IoU loss that considers not only the overlap between the predicted and ground truth bounding boxes, but also the smallest enclosing box that contains both of them. This way, the loss function penalizes the cases where the predicted bounding box is far away from the ground truth bounding box, or where the predicted bounding box is much larger or smaller than the ground truth bounding box.

To illustrate how the YOLOv8 algorithm works, let’s consider a simple example of detecting a person and a dog in an image. Suppose the input image has a size of 640 x 640 pixels, and the backbone network produces a feature map of size 80 x 80 x 256. The neck network then fuses the feature map with other feature maps from different scales and applies spatial attention to obtain a refined feature map of size 80 x 80 x 256. The head network then samples 16 x 16 pixels from the feature map and assigns each pixel to a potential object center. For each center point, the head network predicts four values for the bounding box coordinates, one value for the objectness score, and 80 values for the class probabilities (assuming there are 80 classes in total). The bounding box coordinates are normalized by the image size, and the objectness score and class probabilities are passed through a sigmoid function to obtain values between 0 and 1. The loss function then compares the predicted outputs with the ground truth labels and computes the GIoU loss for the bounding box coordinates, the binary cross-entropy loss for the objectness score, and the focal loss for the class probabilities. The total loss is the weighted sum of these three losses.

Here is a mathematical example of how the YOLOv8 algorithm works for the person and the dog detection:

1. Suppose the ground truth labels for the person and the dog are as follows:
   * Person: bounding box coordinates = (0.4, 0.3, 0.6, 0.7), objectness score = 1, class probability = 1 for class 0 (person) and 0 for the rest of the classes.
   * Dog: bounding box coordinates = (0.7, 0.4, 0.9, 0.6), objectness score = 1, class probability = 1 for class 16 (dog) and 0 for the rest of the classes.
2. Suppose the predicted outputs for the person and the dog are as follows:
   * Person: bounding box coordinates = (0.42, 0.28, 0.58, 0.72), objectness score = 0.95, class probability = 0.98 for class 0 (person) and 0.01 for the rest of the classes.
   * Dog: bounding box coordinates = (0.68, 0.38, 0.92, 0.62), objectness score = 0.9, class probability = 0.96 for class 16 (dog) and 0.02 for the rest of the classes.
3. The GIoU loss for the person bounding box is calculated as follows:
   * The IoU (intersection over union) between the predicted and ground truth bounding boxes is:
   * The smallest enclosing box that contains both bounding boxes has coordinates (0.4, 0.28, 0.9, 0.72), and its area is:A black screen with red dots

     Description automatically generated
   * The GIoU (generalized intersection over union) between the predicted and ground truth bounding boxes is:
   * The GIoU loss is the negative of the GIoU value:  
      GIoU loss = −GIoU ≈ −0.63
4. The GIoU loss for the dog bounding box is calculated similarly and is approximately -0.67.
5. The binary cross-entropy loss for the person objectness score is calculated as follows:
   * The binary cross-entropy loss between the predicted and ground truth objectness scores is:  
     BCE loss = −ylog⁡(p) − (1 − y)log⁡(1 − p)
   * Where y is the ground truth objectness score and p is the predicted objectness score.
   * For the person, y = 1 and p = 0.95, so the BCE loss is:  
     BCE loss = −1log(0.95) − (1 − 1)log(1 − 0.95) ≈ 0.051
6. The binary cross-entropy loss for the dog objectness score is calculated similarly and is approximately 0.105.
7. The focal loss for the person class probability is calculated as follows:
   * The focal loss between the predicted and ground truth class probabilities is:  
     
   * Where y is the ground truth class probability, p is the predicted class probability, α and β are scaling factors, and γ is a focusing parameter.
   * For the person, y = 1 for class 0 (person) and 0 for the rest of the classes, p = 0.98 for class 0 (person) and 0.01 for the rest of the classes, α = 0.25, β = 1.5, and γ = 2.
   * The focal loss for class 0 (person) is:  
     Focal loss = −0.25(1 − 0.98)21log(0.98) − 1.50.982(1 − 1)log(1 − 0.98) ≈ 0.001
   * The focal loss for the rest of the classes is:  
     Focal loss = −0.25(1 − 0.01)20log⁡(0.01) − 1.50.012(1 − 0)log⁡(1 − 0.01) ≈ 0.0002
   * The total focal loss for the person is the sum of the focal losses for all the classes, which is approximately 0.002.
8. The focal loss for the dog class probability is calculated similarly and is approximately 0.003.
9. The total loss for the person and the dog is the sum of the GIoU losses, the BCE losses, and the focal losses for each object, which is approximately -0.63 + -0.67 + 0.051 + 0.105 + 0.002 + 0.003 = -1.136.

This is a simplified example of how the YOLOv8 algorithm works.

**CHAPTER-6**

**OBJECTIVES**

**6.1 Enhanced Face Detection**

Objective:

Develop a highly accurate face detection algorithm capable of efficiently identifying and locating faces in real-time, while accommodating variations in lighting conditions, angles, and facial expressions.

Key Components:

1. Real-Time Processing:

Implement a system that can process video streams in real-time, ensuring immediate detection of faces as they appear in the camera feed.

2. Lighting Condition Handling:

Develop algorithms that adapt to varying lighting conditions to maintain accurate face detection even in low-light or harsh lighting scenarios.

3. Angle and Expression Tolerance:

Enhance the algorithm to accurately detect faces at different angles and under various facial expressions, ensuring robust performance in diverse situations.

**6.2 Robust Face Recognition**

Objective:

Create a robust face recognition system capable of matching detected faces to a pre-defined database of individuals, facilitating positive identification and generating alerts when necessary.

Key Components:

1. Database Integration:

Integrate a database of individuals with corresponding facial features for accurate recognition and identification.

2. Matching Algorithms:

Implement advanced face matching algorithms that account for variations in facial features, allowing for reliable identification even with changes in appearance.

3. Alert Generation:

Design a system that generates alerts upon recognizing a face and positively identifying individuals on watchlists or persons of interest.

**6.3 Multi-Camera Integration**

Objective:

Implement the capability to seamlessly integrate with multiple cameras across a surveillance network, enabling simultaneous monitoring and tracking of individuals across different camera views.

Key Components:

1. Network Compatibility:

Ensure compatibility with various camera types and surveillance systems to establish a unified network.

2. Synchronization:

Implement synchronization mechanisms to align timestamps and maintain consistency in data across multiple cameras.

3. Seamless Transition:

Enable smooth transitioning of tracking from one camera to another, ensuring uninterrupted monitoring as individuals move within the surveillance network.

**6.4 Real-Time Tracking**

Objective:

Develop an intelligent tracking system that can monitor and track recognized faces as they move within the camera network, providing real-time information on their whereabouts and activities.

Key Components:

1. Continuous Tracking:

Implement continuous tracking mechanisms to monitor the movement of recognized faces across frames and cameras.

2. Behavior Analysis:

Integrate behavior analysis algorithms to understand and predict the trajectory of tracked individuals based on their movements.

3. Real-Time Updates:

Provide real-time updates on the location and activities of tracked individuals, enabling immediate response to potential security threats.

**6.5 Alert and Reporting System**

Objective:

Design a comprehensive alert and reporting system that can automatically generate alerts for suspicious activities, unauthorized access, or individuals on watchlists, and provide detailed reports for post-incident analysis and evidence gathering.

Key Components:

1. Automated Alerting:

Develop a system that triggers alerts in real-time for identified threats, unauthorized access, or individuals matching watchlists.

2. Reporting Mechanism:

Create a reporting mechanism that compiles detailed reports on incidents, including timestamps, images, and contextual information for post-incident analysis.

3. Integration with External Systems:

Ensure compatibility with external security systems and authorities for seamless sharing of alert information and reports.

**CHAPTER-7**

**SYSTEM DESIGN & IMPLEMENTATION**

**7.1 System Design**

System design is a fundamental process in software engineering, where a system’s architecture is planned and designed to meet specific requirements. It’s a complex task that requires a deep understanding of the problem domain, technical knowledge, and design skills.

**7.1.1 Concepts of System Design**

The concepts of system design are:

**Understanding Requirements**

The first step in system design is understanding the system’s requirements. This involves identifying the system’s users, understanding their needs, and defining the system’s functionality. This is often done through use cases or user stories. The requirements are documented in a Product Requirements Document (PRD), which serves as a contract specifying what the system should do.

**High-Level Design**

The high-level design phase involves defining the system’s architecture. This includes deciding on the technology stack, database design, and the system’s overall structure. The system is usually broken down into smaller, manageable modules. The high-level design provides an overview of the solution, identifies the components and their interaction, and sets the direction for the detailed design.

**Detailed Design**

In the detailed design phase, each module’s functionality is defined in detail. This includes defining algorithms, interface designs, and error handling mechanisms. The goal is to design modules that are robust, scalable, and maintainable. Detailed design is about how to implement the high-level design in code. It’s the blueprint for coding.

**Non-Functional Requirements**

The system’s design should also consider non-functional requirements like performance, security, and usability. For example, the system may need to support a certain number of concurrent users, have low latency, or have a user-friendly interface. These requirements are just as important as functional requirements and can significantly impact the system’s design.

**Design Principles and Patterns**

Design principles and patterns guide the design process. Principles like SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation, and Dependency inversion) help ensure that the system is modular, maintainable, and scalable. Design patterns provide reusable solutions to common problems and can help improve the efficiency of the design process.

**Prototyping**

Prototyping is an optional step where a working model of the system is built with some functionalities. This prototype is used to get early feedback and refine the requirements and design. It helps in understanding how the final product will look and feel.

**Validation and Iteration**

Finally, the system design is reviewed and validated against the requirements. Any issues or gaps are addressed, and the design is iterated until it meets the requirements. This ensures that the system design accurately represents what needs to be built.

**Documentation**

Documentation is a critical part of system design. It ensures that the design is accurately communicated to the stakeholders and the development team. It also serves as a reference for future maintenance and enhancement.

In summary, system design is a critical phase in the software development lifecycle that ensures the system meets its intended purpose effectively and efficiently. It’s a complex task, but with careful planning and design, it can lead to successful software systems. It’s important to note that system design is not a one-time activity but an ongoing process that evolves as the system and its requirements change. It’s a balance between meeting the current requirements and anticipating future needs.

**7.1.2 Features Of System Design**

The features of system design are:

**Designing Business Processes**

This is like drawing a map of how different tasks in a business connect and flow into each other. For example, in an online store, this could be the process from adding items to the cart, to checking out and shipping.

**Creating Data Models**

This is deciding how all the information in a system is linked and stored. Like in a library system, you’d have information about books, authors, and who borrowed which book.

**Defining Events and Procedures**

This is about identifying what can happen in the system and how the system should respond. Like in a bank system, an event could be someone trying to withdraw money, and the procedure would be to check if they have enough balance, then give them the money and update their account.

**Designing Applications**

This is about creating the different parts of the system that users interact with. Like in a hospital system, there could be different applications for registering patients, scheduling appointments, and keeping track of medical records.

**Designing System Interactions**

This is planning how different parts of the system communicate with each other. Like in a ride-hailing app, the rider’s app needs to talk to the driver’s app and the main server to book a ride, track the ride, and handle payment.

**Security**

This is about protecting the system and its data from threats. This could involve things like encrypting data, making sure users are who they say they are, and controlling who has access to what.

**Scalability and Performance**

This is about making sure the system can handle growth and still perform well. Like a social media platform needs to be designed to handle millions of users and a huge amount of data.

**Testing the Design**

This is about checking that the system works as expected. This could involve trying out different scenarios to make sure the system handles them correctly.

**Deploying and Maintaining the System**

Once the system is up and running, it needs to be kept up-to-date and running smoothly. This could involve fixing problems, updating parts of the system, and adding new features.

**Iterative Process**

System design isn’t a one-time thing. It’s a cycle where you keep learning more, updating the design, and making improvements.

In short, system design is like creating a blueprint for a building. It’s about planning out all the details of how the system should work, from the big picture down to the tiny details. It’s a complex task, but it’s crucial for creating a system that works well and meets the needs of its users. It’s a mix of creativity and logical thinking, and it’s what turns a list of requirements into a working system.

**7.1.3 Waterfall Method**

The Waterfall model is a sequential design process in which progress is seen as flowing steadily downwards (like a waterfall) through several phases.

The Waterfall model is a classic and linear approach to software development, emphasizing a structured and sequential progression through distinct phases. The model is characterized by its rigid, step-by-step nature, where each phase must be completed before moving on to the next.

The steps involved in the Waterfall method, are:

**Requirements Gathering:**

The Waterfall model commences with the Requirements Gathering phase, where project stakeholders collaborate to gather and document the project's requirements. This involves interactions with clients, end-users, and other relevant parties to understand the system's functionalities, constraints, and objectives thoroughly. The outcome of this phase is a comprehensive requirements document that serves as the foundation for subsequent stages.

**System Design:**

Once the requirements are well-defined, the development process proceeds to the System Design phase. In this stage, the focus is on transforming the gathered requirements into a detailed system design. System architects and designers create blueprints, specifying the overall system structure, modules, interfaces, and data flow. This phase is critical in laying the groundwork for the subsequent implementation phase and requires careful consideration of technical feasibility, scalability, and maintainability.

**Implementation:**

Building on the detailed design specifications, the Implementation phase involves translating the design into actual code. Developers write the source code based on the design documents, following coding standards and best practices. This step requires a high level of precision, as the code serves as the backbone of the system. The output of this phase is the executable code that will eventually form the working software.

**Integration and Testing:**

Following the completion of the implementation phase, the next step is Integration and Testing. This involves combining individual components or modules and testing their interactions to ensure they function seamlessly as a unified system. The testing phase includes various levels such as unit testing, integration testing, and system testing. This step is crucial for identifying and rectifying defects or inconsistencies in the code, ensuring the overall reliability and correctness of the system.

**Deployment:**

Once the system has undergone rigorous testing and validation, it is ready for Deployment. In this phase, the finalized software is released and installed in the live environment for end-users. The deployment process may involve activities such as data migration, training for end-users, and the establishment of support mechanisms. The goal is to transition the software from the development environment to the operational environment smoothly.

**Maintenance:**

The final phase of the Waterfall model is Maintenance. This phase extends throughout the software's lifecycle and involves addressing issues identified post-deployment, incorporating updates or enhancements, and ensuring ongoing system performance. Maintenance activities may include bug fixes, security patches, and modifications to accommodate evolving user needs. The iterative nature of maintenance ensures that the software remains relevant and effective over time.

Characteristics of the Waterfall Model:

**Sequential Progression:**

The Waterfall model follows a strict, sequential flow, with each phase building upon the outcomes of the previous one. This linear progression provides a clear roadmap for development.

**Document-Driven:**

Documentation is a central aspect of the Waterfall model. Each phase requires comprehensive documentation, from initial requirements to detailed design specifications. This documentation serves as a reference and guide throughout the project.

**Well-Defined Phases:**

Each phase in the Waterfall model has well-defined entry and exit criteria, ensuring that progress is measurable and controlled. This characteristic helps in managing project timelines and expectations.

**Rigidity:**

The Waterfall model is known for its rigidity, as it may not easily accommodate changes once a phase is completed. This can be both an advantage and a limitation, depending on the stability of project requirements.

Implications and Criticisms:

Limited Flexibility:

The sequential nature of the Waterfall model can be a limitation in dynamic environments where requirements may evolve. The model may struggle to accommodate changes once the project is underway.

**Late Feedback:**

Stakeholder involvement and feedback occur primarily at the beginning and end of the project. This limited interaction can result in late identification of issues, potentially leading to higher correction costs.

**Long Delivery Time:**

Due to its sequential structure, the Waterfall model may result in longer delivery times, especially when compared to more iterative and agile methodologies.

**Risk Management Challenges:**

Risks may not be identified until the testing phase, making it challenging to address potential issues early in the development process.

**Client Involvement:**

Client involvement is primarily concentrated at the beginning and end of the project, potentially leading to misalignments between client expectations and the final product.

In conclusion, the Waterfall model, with its structured and systematic approach, has been a longstanding methodology in software development. It may be suitable for projects with stable and well-understood requirements, its limitations in adapting to change have led to the emergence of more flexible and iterative methodologies in contemporary software development practices. Understanding the steps, characteristics, and implications of the Waterfall model provides valuable insights into its application and relevance in diverse project scenarios.

**7.2 System Architecture**

System architecture is like the blueprint of a building but for software. It defines the structure and organization of a complex system, detailing how different components or modules interact to achieve overall functionality. Just as an architect plans the layout, materials, and connections in a building, system architecture involves designing the high-level structure of a software system.

**7.2.1 Key aspects of system architecture**

**Components and Modules:**

System architecture breaks down a software system into manageable components or modules. These could be individual features, functions, or services that collectively form the entire system. Each module has a specific role and interacts with others to accomplish tasks.

**Interactions and Interfaces:**

It defines how the various components communicate and interact with each other. This involves specifying the interfaces, or points of connection, where different parts of the system exchange information. This communication is crucial for seamless operation and data flow.

**Data Management:**

System architecture addresses how data is stored, retrieved, and managed within the system. This includes decisions about databases, data structures, and algorithms for efficient data processing. It ensures that information is handled accurately and securely.

**Technology Stack:**

System architecture considers the technologies and tools that will be used to build the system. This could involve choices like programming languages, frameworks, databases, and third-party services. Selecting the right technology stack is vital for achieving the desired system performance and scalability.

**Scalability and Performance:**

Scalability is a crucial aspect of system architecture, especially for applications that may need to handle growing amounts of users or data. It involves designing the system to gracefully handle increased load without sacrificing performance. This might include considerations for distributed computing, load balancing, and efficient resource utilization.

**Security:**

System architecture incorporates security measures to protect the system from unauthorized access, data breaches, and other threats. This involves implementing encryption, authentication, and authorization mechanisms to ensure the confidentiality and integrity of sensitive information.

**Flexibility and Adaptability:**

A well-designed system architecture allows for flexibility and adaptability to changes. It should be able to accommodate new features or modifications without requiring a complete overhaul of the entire system. This is especially important in dynamic environments where requirements may evolve over time.

**User Experience:**

While system architecture primarily deals with the technical aspects, it also considers the user experience. The way components are designed to interact impacts how users will interact with the system. A user-friendly architecture ensures a smoother and more intuitive experience for the end-users.

In essence, system architecture is the conceptual foundation that guides the development of a software system. It's the result of careful planning and decision-making to ensure that the system meets its objectives efficiently, is robust, secure, and can adapt to changing needs. Much like a well-designed building that stands the test of time, a thoughtfully crafted system architecture forms the basis for successful software solutions

.

This phase comprises:

* Understanding and creating system architecture.
* Choosing programming language/s, platform/s utilized in the implementation.
* Choosing the correct design of modules.
* Implementation of the application and services.

A black background with white rectangular boxes

Description automatically generated

*Fig 7.1 Architecture of the system*

The above system follows a comprehensive architecture that encompasses data handling, model training, real-time processing, evaluation, and continuous refinement. Each step contributes to the overall effectiveness and accuracy of the system.

**1. Start State: Create/Update Dataset**

Objective: Acquiring and Maintaining Data

Data Collection Module:

- Utilizes multiple static CCTV cameras to capture facial images.

- Captured images are stored in the dataset, ensuring a diverse and representative sample.

- Periodically updates the dataset to include new facial data.

Data Preprocessing Module:

- Cleans and preprocesses the collected data to enhance image quality.

- Augments the dataset with variations in lighting, angles, and expressions to improve model robustness.

**2. Train the Model on the Dataset**

Objective: Model Learning and Enhancement

Model Training Module:

- Utilizes the preprocessed dataset for training machine learning and deep learning models.

- Chooses appropriate algorithms such as Convolutional Neural Networks (CNNs),etc for facial feature extraction.

- Fine-tunes model parameters to optimize performance.

Model Evaluation Module:

- Assesses the trained model's performance using validation datasets.

- Metrics like accuracy, precision, recall, and F1 score are calculated.

- Adjusts hyperparameters and model architecture based on evaluation results.

**3. Face Detection Module**

Objective: Detecting Faces in Images

Image Acquisition Module:

- Obtains images from the CCTV cameras in real-time.

- Utilizes static images or video frames as input for face detection.

Face Detection Algorithm:

- Applies machine learning and deep learning techniques, potentially using algorithms like Haar cascades or deep learning-based detectors (e.g., Single Shot Multibox Detector - SSD).

- Identifies and localizes faces within the images.

**4. Face Recognition Module**

Objective: Identifying Individuals from Detected Faces

Feature Extraction Module:

- Extracts facial features from the detected faces using the trained model.

- Focuses on key facial landmarks and characteristics.

Face Recognition Algorithm:

- Applies machine learning and deep learning techniques for matching facial features against the dataset.

- Utilizes algorithms like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or deep neural networks for recognition.

**5. Continuous Tracking Module**

Objective: Real-Time Monitoring of Recognized Individuals

Person Tracking Algorithm:

- Utilizes real-time data to continuously track the movements and activities of recognized individuals across multiple cameras.

- Dynamic tracking ensures an up-to-date understanding of the person's location and behavior.

**6. End State: Apply Evaluation Methods**

Objective: Assessing System Performance

Performance Evaluation Module:

- Applies evaluation metrics to measure the accuracy, precision, and recall of the face detection and recognition system.

- Analyzes the real-time tracking data for consistency and reliability.

- Generates reports on system performance.

**7. Refinement and Iteration**

Objective: Enhancing Model Effectiveness

- Feedback Loop Module:

- Gathers user feedback, performance metrics, and real-world data.

- Refines the model based on identified shortcomings or areas for improvement.

- Iteratively retrains the model with updated datasets to achieve a satisfactory result.

**8. Start from the Start State Again**

Objective: Continuous Improvement

- The system loops back to the initial state, initiating the process of dataset creation or update.

- The refined model, based on previous evaluations, is used as a starting point for subsequent iterations.

- The continuous improvement cycle ensures that the system adapts to evolving conditions, maintaining optimal performance.

Conclusion:

This holistic system architecture seamlessly integrates data acquisition, model training, real-time processing, and continuous refinement. By iteratively incorporating user feedback and performance evaluations, the system strives for continuous improvement, ensuring adaptability and effectiveness in dynamic surveillance environments.

**7.3 Implementation**

Implementation is the stage in a project where the ideas and plans are turned into reality. It’s the phase where all the tasks and activities outlined in the project plan are carried out. This phase also includes monitoring the project’s progress to ensure it stays on track and any issues are addressed promptly. The success of the project largely depends on how well this phase is executed. It’s the bridge between the planning and the final outcome, and it’s where the project manager takes control, guiding the team towards the project goals.

The implementation is the stage in the software development process where the software is coded, tested, and deployed. Some of the tasks involved in this phase are:

1. **Dataset Creation:** Dataset creation is the process of collecting, organizing, and labeling data for a specific purpose, such as machine learning, data analysis, or visualization.
2. **Coding:** This is the task of writing the source code of the software using a programming language such as Python. The code should follow the design specifications and coding standards of the project. The code should also be well-documented and commented on for readability and maintainability.
3. **Testing:** This is the task of verifying and validating the functionality, performance, and quality of the software. Testing can be done at different levels, such as unit testing, integration testing, system testing, and acceptance testing. Testing can also involve various techniques, such as black-box testing, white-box testing, regression testing, and stress testing. Testing should aim to find and fix any errors, bugs, or defects in the software before deployment.
4. **Deployment:** This is the task of installing and launching the software in the target environment, such as a server, a device, or a cloud platform. Deployment can also involve configuration, customization, and optimization of the software according to the user’s needs and preferences. Deployment should ensure that the software is accessible, usable, and secure for the end-users.

**7.3.1 DATASET CREATION**

We are creating our own dataset for this project using Roboflow. Roboflow is a platform that lets you create, train, and deploy computer vision models using various annotation and inference tools. You can use foundation models from OpenAI, Meta AI, and other sources, or train your own models with text-based search, CLIP vectors, and Segment Anything. Roboflow also provides a Python package, roboflow-python, that enables you to interact with models, datasets, and projects hosted on Roboflow. Some of the features and benefits of Roboflow are:

1. **Dataset management**: Roboflow lets you search, curate, and organize visual data from various sources and formats. You can also filter, tag, segment, preprocess, and augment your data to make it more suitable for your purpose.
2. **Annotation tools**: Roboflow lets you label your data manually, semi-automatically, or automatically, using different methods and sources. You can also use foundation models, such as CLIP and Segment Anything, to generate labels without hand-labeling images

.

1. **Model training**: Roboflow lets you use foundation models from OpenAI, Meta AI, and thousands of open source repositories, or train your own models using different frameworks, such as TensorFlow, PyTorch, YOLO, etc. You can also use text-based semantic search and CLIP vectors to find similar data and anomalies.
2. **Model deployment**: Roboflow lets you deploy your models at scale, on-device or in the cloud, using different platforms, such as NVIDIA Jetson, Raspberry Pi, SageMaker, Azure, etc. You can also use Roboflow’s inference server, Roboflow Deploy, to run production models reliably and without friction.
3. **Collaboration tools**: Roboflow lets you collaborate with other developers and share your datasets, models, and projects. You can also access a collection of open source Jupyter notebooks, utilities, and forums to learn and work with the latest computer vision models.

**7.3.2 CODING**

The language used for backend development is Python. Python is a popular and versatile programming language that can be used for various purposes, such as web development, data analysis, machine learning, artificial intelligence, and more. Python has a simple and readable syntax that makes it easy to learn and use. Python also has a rich and powerful standard library that provides many built-in modules and functions for common tasks. Python is an interpreted language, which means it does not need to be compiled before running. Python is also a dynamic and object-oriented language, which means it supports multiple programming paradigms and allows changing the type and behavior of objects at runtime. Some of the features and benefits of Python are:

1. **High-level**: Python is a high-level language, which means it abstracts away the low-level details of the computer hardware and operating system, and allows the programmer to focus on the logic and functionality of the program.
2. **General-purpose**: Python is a general-purpose language, which means it can be used to create a wide range of applications, from web development to data science, from scripting to game development, and more.
3. **Interpreted**: Python is an interpreted language, which means it does not need to be compiled before running. The Python interpreter reads and executes the source code line by line, which makes it easier to debug and test the program.
4. **Dynamic**: Python is a dynamic language, which means it allows changing the type and behavior of objects at runtime. Python does not require declaring the type of variables or parameters, and it automatically infers the type based on the value assigned. Python also supports dynamic binding, which means it allows assigning new attributes and methods to existing objects.
5. **Object-oriented**: Python is an object-oriented language, which means it supports the concept of objects, classes, inheritance, polymorphism, and encapsulation. Python allows defining custom classes and creating instances of them, as well as inheriting from existing classes and overriding their methods. Python also supports multiple inheritance, which means a class can inherit from more than one parent class.
6. **Easy to learn and use**: Python has a simple and readable syntax that follows the principle of readability counts. Python uses indentation to define blocks of code, and avoids the use of curly braces or semicolons. Python also has a clear and consistent naming convention for variables, functions, and modules. Python is designed to be easy to learn and use, and has a large and active community of developers who provide tutorials, documentation, and support.
7. **Powerful standard library**: Python has a rich and powerful standard library that provides many built-in modules and functions for common tasks, such as file handling, string manipulation, math operations, networking, database access, and more. The standard library also includes modules for specific domains, such as web development, data analysis, machine learning, and more. The standard library is also known as the batteries included philosophy, which means Python comes with everything you need to get started.
8. **Open source and cross-platform**: Python is an open source and cross-platform language, which means it is free to use and distribute, and it can run on any operating system, such as Windows, Linux, Mac OS, and more.

**7.3.3 TESTING**

The technique used is black-box testing. Black box testing is a software testing method where the internal workings or code structure of the system being tested are not known to the tester. The tester focuses solely on the external behavior of the software, without having access to its internal source code. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues, and reliability issues.

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. It is also known as Behavioral Testing.

Black box testing can be done in the following ways:

1. **Syntax-Driven Testing**: This type of testing is applied to systems that can be syntactically represented by some language.
2. **Equivalence partitioning**: The idea is to partition the input domain of the system into several equivalence classes such that each member of the class works similarly.
3. **Boundary value analysis**: Boundaries are very good places for errors to occur. Hence, if test cases are designed for boundary values of the input domain then the efficiency of testing improves and the probability of finding errors also increases.
4. **Cause effect graphing**: This technique establishes a relationship between logical input called causes with corresponding actions called the effect.
5. **Requirement-based testing**: It includes validating the requirements given in the SRS of a software system.

Black box testing is not just limited to functional testing, it also includes non-functional testing such as performance, scalability, usability.

**7.3.4 DEPLOYMENT**

Streamlit is an open-source tool designed to build and deploy data applications with less coding compared to traditional front-end technologies like HTML, CSS, and JavaScript. It’s a low-code tool specifically designed for building data science applications.

Streamlit was created by Adrien Treuille, Thiago Teixeira, and Amanda Kelly with the aim of simplifying the deployment of machine learning models and Python projects. The tool is user-friendly and doesn’t require knowledge of HTML, CSS, or JavaScript.

Streamlit is a single Python package that you install through pip. It provides a set of functions that can be interleaved into an existing ML code script, making the ML code parameterizable. It also does a bit of layout and magically turns your ML code into a beautiful app.

Streamlit supports many Python frameworks such as scikit-learn, spaCy, Pandas, and various visualization frameworks such as Altair, Plotly, and Matplotlib. All of the data is held and processed on the server hosting Streamlit. What is sent to the front end are the visuals for display.

**CHAPTER-8**

**TIMELINE FOR EXECUTION OF PROJECT**

**(GANTT CHART)**

**A graph with black lines

Description automatically generated**

Fig 8.1 Gantt Chart

**CHAPTER-9**

**OUTCOMES**

The development of the Sentinel System focuses on creating a video surveillance system called the Sentinel system. This intelligent system utilizes cameras to detect, recognize and track faces effectively.

One of the goals of the Sentinel system is to ensure face detection, under challenging conditions. It should be able to detect faces in situations with different lighting conditions, obstructions and other factors that can impact face detection performance.

Additionally, the system aims to incorporate facial recognition technology for identifying individuals. Integrating with a database will enable identification and alerts for known individuals of interest.

To ensure surveillance coverage intelligent tracking algorithms will be integrated into the Sentinel system. These algorithms allow monitoring and tracking of individuals across camera feeds. The system can track individuals as they move between camera views.

Compatibility and scalability are aspects of the Sentinel system. It is designed to be compatible with a range of surveillance camera systems to facilitate adoption. Furthermore, it can handle a number of cameras effectively. Process data from multiple sources efficiently.

Privacy and Ethics: The main goal is for the Sentinel system to prioritize privacy by following practices when handling data. It should securely handle data in accordance with privacy regulations to safeguard the privacy rights of individuals.

The objective of this project is to create a solution for detecting, recognizing and tracking faces in time. Our goal is to develop the Sentinel system, which will efficiently and effectively perform these tasks to support surveillance operations.

This project has ranging applications, across sectors, including public safety, law enforcement and critical infrastructure protection. The expected result is a tool that enhances security measures and enables surveillance in these sectors.

**CHAPTER-10**

**RESULTS AND DISCUSSIONS**

**A group of graphs showing the size of a line

Description automatically generated with medium confidence**

*Fig 9.1 Results of the model*

**A group of blue graphs

Description automatically generated with medium confidence**

*Fig 9.2 labels Correlogram*

**A collage of different colored squares

Description automatically generated**

*Fig 9.3 Labels*

**A blue square with white text

Description automatically generated**

*Fig 9.4 Normalized Confusion Matrix*

**A graph with a line

Description automatically generated with medium confidence**

*Fig 9.5 Precision-Recall Curve*

**A group of graphs showing the size of a line

Description automatically generated with medium confidence**

*Fig 9.6 Training information graph*

**CHAPTER-11**

**CONCLUSION**

Accomplishments: In the conclusion we can summarize the development and implementation of the Sentinel system. We would highlight how it has achieved face detection, recognition and tracking capabilities. Additionally, we would mention its integration, with a database that enables identification of known individuals.

Advancements in Video Surveillance; The conclusion will underscore the significant progress represented by the Sentinel system in video surveillance capabilities. We will emphasize how it incorporates computer vision and deep learning models well as intelligent tracking algorithms to enhance its effectiveness.

Potential Applications: In the conclusion we will explore the sectors where the Sentinel system can find applications. These sectors include safety, law enforcement and critical infrastructure protection. We will emphasize that this system can significantly contribute to strengthening security measures and improving surveillance operations in these areas.

Scalability and Compatibility; Within the conclusion we will highlight the scalability and compatibility features of the Sentinel system. Specifically, we will mention its ability to seamlessly work with surveillance camera systems. This compatibility opens possibilities for adoption and seamless integration into existing surveillance infrastructures.

Privacy and Ethical Considerations; In addressing privacy concerns within our conclusion, we will emphasize how committed our system is to data handling practices while ensuring compliance with privacy regulations. It is crucial to highlight our management of data and our dedication to protecting individuals’ privacy rights.

The final section can consider the possibilities and potential improvements for the Sentinel system. It may delve into avenues for research and development like enhancing real time processing abilities broadening the systems functionalities or exploring applications, in connected domains.

**REFERENCES**

[1] Probabilistic recognition of human faces from video by q Shaohua Zhou,\* Volker Krueger, and Rama Chellappa

[2] Liu, Z., Luo, D., Wang, Y., Wang, L., Tai, Y., Wang, C., Li, J., Huang, F., & Lu, T. (2020). TEINet: Towards an Efficient Architecture for Video Recognition.

3] Mohana and Dr. H. V. Ravish Aradhya. Design and Implementation of Object Detection, Tracking, Counting and Classification Algorithms using Artificial Intelligence for Automated Video Surveillance Applications

[4] Dhaya, R. CCTV Surveillance for Unprecedented Violence and Traffic Monitoring.

[5] Davies, A. C., & Velastin, S. A. (2005). A Progress Review of Intelligent CCTV Surveillance Systems.

[6] Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2023). "Real-Time Flying Object Detection with YOLOv8.”

[7] Thaler, M., & Bailer, W. (2013). Real-time person detection and tracking in panoramic video.

[8] Arbués-Sangüesa, A., Haro, G., & Ballester, C. (2019). Multi-Person tracking by multi-scale detection in Basketball scenarios

[9] Paul Viola and Michael Jones,Rapid Object Detection using a Boosted Cascade of Simple Features.

[10] Oluwatoyin P. Popoola and Kejun Wang Video-Based Abnormal Human Behavior

Recognition—A Review

**APPENDIX-A**

**PSUEDOCODE**

!yolo task=detect mode=train model=yolov8n.pt data={dataset.location}/data.yaml seed=14 epochs=250 imgsz=640 cos\_lr=True optimizer='SGD' verbose=True  cache=True patience=50  single\_cls=True amp=True batch=30 weight\_decay=0.00075 warmup\_epochs=2.5

!yolo task=detect mode=val model=/content/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml plots=True

!yolo task=detect mode=predict model=/content/runs/detect/train/weights/best.pt conf=0.5 source={dataset.location}/test/images save\_txt=true save\_conf=true visualize=True augment=True

**APPENDIX-B**

**SCREENSHOTS**

**A person walking in a room

Description automatically generated**

**A person in a black shirt

Description automatically generated**

**APPENDIX-C**

**ENCLOSURES**

**1. Conference Paper Presented Certificates of all students.**

**2. Include certificate(s) of any Achievement/Award won in any project related event.**

**3. Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need of page-wise explanation.**