
**Software Engineering Department**

**Braude College of Engineering**

# Improvements in Person Identification in Video under constraints

**Project code: 25-1-R-20**

**Capstone Project Phase A – 61998**

**Feb 2025**

**Asad Saffouri**

**Alexandra Simonishvili**

**Git repository link:**

https://github.com/asadsaffouri/phase-A.git

**Supervisor:**

**Mr. Ilya Zeldner**

# 1 Abstract

Today cameras and videos take a significant part in many various aspects of our lives. They already have a wide usage in security, production, culture entertainment and i.e. Along with that arises the need for efficient and precise person identification systems for videos. Our project provides a new approach for person recognition and tracking in video, combining person identification technology and object tracking in video algorithm. We use person identification techniques from previous project (AI-Driven Person Recognition and Identification in Videos [1] by Rom Harell and Itay Benjamin) that combines YOLO (You Only Look Once) algorithm for human detection and FaceNet algorithm for person face recognition and upgrade it with FlowNet algorithm for object tracking. Our approach promises significant improvements in the video person detection ability which is very useful in different institutions.

# 2 Introduction

The rapid growth in the use of video content across various fields has highlighted the need for advanced person recognition systems. These systems are critical in areas such as security, surveillance (e.g., tracking individuals in crowded environments), smart devices vision and media analysis. Existing methods often struggle with accuracy and efficiency due to the inherent challenges of dynamic video environments, including changing lighting, movement, and resolution variability, as well as situations where faces are partially or fully obscured. Enhancing person recognition and tracking in such conditions is crucial to meet the needs of real-world institutions and applications.

Our project aims to develop an innovative system that leverages state-of-the-art deep learning technologies to identify individuals in video footage accurately. The system integrates with some powerful tools: FlowNet, which analyzes motion and tracks changes between consecutive frames [3], YOLO (You only look once) algorithm for real-time object detection [1,2], and FaceNet, which specializes in face recognition by mapping facial features into a Euclidean space [1,2]. By combining motion analysis and facial recognition, the system is designed to perform well even in challenging scenarios where both precision and speed are essential but having to deal with low video quality and different changing conditions.

In our system we combine the findings of the previous project (AI-Driven Person Recognition and Identification in Videos by Rom Harell and Itay Benjamin [1]) and Optical Flow algorithm (FlowNet: Learning Optical Flow with Convolutional Networks by Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., & Brox, T [3], Computer Vision and Deep Learning: From Image to Video Analysis by Jeremy Cohen [9]) to optimize the results and to get the benefits from both worlds. Our system focuses on optimizing computational efficiency by analyzing motion dynamics and analyzing frames for face recognition. Instead of processing and tracking all the objects in a video, our system prioritizes zones at frames where a detection match occurred. This approach minimizes redundant computations and ensures the system can track individuals even when their faces are not always fully visible. By leveraging FlowNet ability to detect movement and FaceNet reliable recognition capabilities.

# 3 Background and Related Work

## 3.1 Person Identification in Videos

Person identification in videos is a tough problem due to several real-world challenges. Lighting in videos can change because of factors like time of day, different environments, or camera angles, making it harder to see faces and objects clearly. In crowded areas, faces might be partially blocked, complicating the recognition process. On top of that, scenes can change rapidly, making it difficult to track movement, identify people, or recognize features consistently. These challenges are further intensified by the huge amount of data in video streams, requiring systems to process many frames in real-time without slowing down. In the past, traditional methods relied on manually designed features, which were not as flexible in adapting to dynamic conditions. However, deep learning, especially Convolutional Neural Networks (CNNs) [9], has made a big difference by automating feature extraction, allowing the system to learn patterns from the data directly and perform better in complex scenarios. Recent advancements have leveraged CNNs and Recurrent Neural Networks (RNNs) to improve identification across video frames, with CNNs excelling at spatial features and RNNs [10], like ConvLSTMs, capturing temporal dynamics. These models have helped systems adapt to variations in appearance, lighting, and posture. Techniques like 3D pose estimation and multi-stream networks have further enhanced performance [11], especially in cases involving multiple perspectives. But there are still considerable obstacles, particularly around real-time processing, scalability, and adapting to different datasets, which continue to limit the effectiveness of person recognition systems in real-world applications.

## 3.2 Uses of Person Detection

Person detection in video streams has a wide range of applications across various industries. It is widely used in surveillance and security systems to monitor public and private spaces, helping detect intruders and raise alarms in case of suspicious activity. In face recognition systems, person detection is the first step for identifying and verifying individuals, used in security systems like access control, smartphone unlocking, or entity

verification at the airport during passport control. Retailers use person detection for retail analytics, tracking customer behavior and foot traffic to optimize store layouts and marketing efforts. It is also crucial in autonomous vehicles, helping detect pedestrians and cyclists to improve safety. In smart cities, person detection supports crowd monitoring, traffic management, and optimizing public transportation. Additionally, sports and performance analysis use person detection to track athletes. These applications highlight the versatility and importance of person detection across many sectors.

## 3.3 CNN

Convolutional neural networks (CNNs) have changed the computer vision and image processing field. CNNs are a class of deep neural network architectures, which get hierarchical features from raw pixel data and allow to achieve high accuracy in solving different computer vision tasks, in image classification, object detection, semantic and instance segmentation [1]. They consist of convolutional layers that apply learnable filters on input images resulting in low-level features like edges, textures, patterns and etc. They capture spatially localized patterns, so the network exploit inherent stationary images, and pooling layers that reduce computational complexity by downsizing the spatial dimension retaining the most informative parts. Nonlinear activation functions, such as ReLUs, enable CNNs to learn complex data patterns. As networks deepen, they combine lower-level features to form high-level, semantic representations. Despite their effectiveness, CNNs face challenges such as the need for large, annotated datasets, vulnerability to adversarial attacks, and difficulty in interpretation. Research is exploring methods like transfer learning and explainable AI to improve their robustness and transparency.

### 3.3.1 CNN architecture

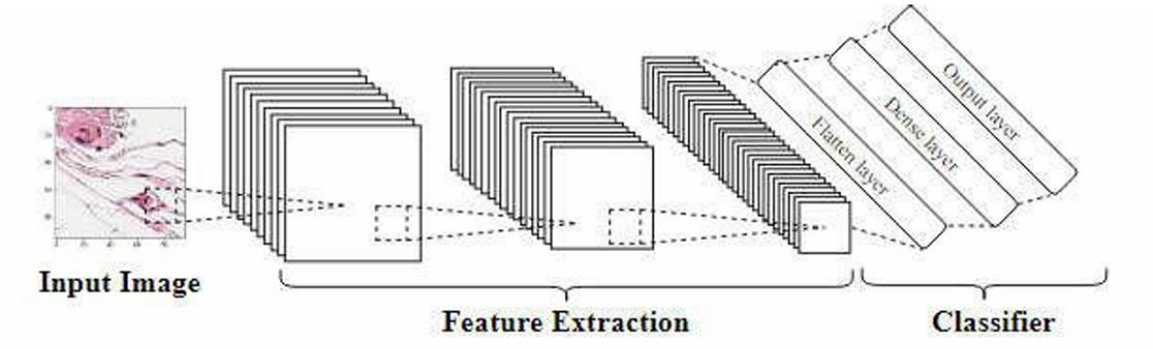



*Figure 1: Convolutional neural network (CNN) diagram [1]*

CNN architecture consists of convolutional layers that extract features from images, pooling layers that reduce spatial dimensions, and fully connected layers that combine features for final predictions. This structure allows CNNs to automatically learn hierarchical features and recognize complex patterns in data.

## 3.4 FlowNet

FlowNet is a deep learning framework made for optical flow estimation, which is an important task in video analysis. Optical flow is a way to measure how objects, surfaces, or edges move between two video frames. It creates a dense map of motion vectors that show the direction and speed of each pixel's movement. FlowNet is unique because it was the first convolutional neural network (CNN) that could learn optical flow directly from data in an end-to-end way [3], without needing traditional methods that used handcrafted features and slow optimization techniques.

Optical flow has many uses. For example, it helps in video stabilization by detecting and correcting unwanted camera movements. It is also used in action recognition to understand behaviors, object tracking to follow objects in a video, and motion analysis to study how things move in a scene. Older methods like Horn-Schunck [4] or Lucas-Kanade [5] were accurate but very slow and not practical for real-time applications.

FlowNet solves these problems by using a neural network that learns how motion works directly from data. It is much faster than the older methods and can be trained on

different datasets to handle many types of scenarios. This makes FlowNet a great tool for tasks that need both speed and accuracy, like real-time video analysis.

## 3.4.1 Architecture of FlowNet

FlowNet takes two consecutive image frames as input and outputs a dense optical flow field. It consists of a deep convolutional network trained to predict motion vectors for each pixel. The network learns from synthetic datasets with ground truth optical flow (such as the Flying Chairs dataset used in FlowNet article [2]) ensuring robust performance.

FlowNet has two main variants: FlowNetSimple and FlowNetCorr [1,2].

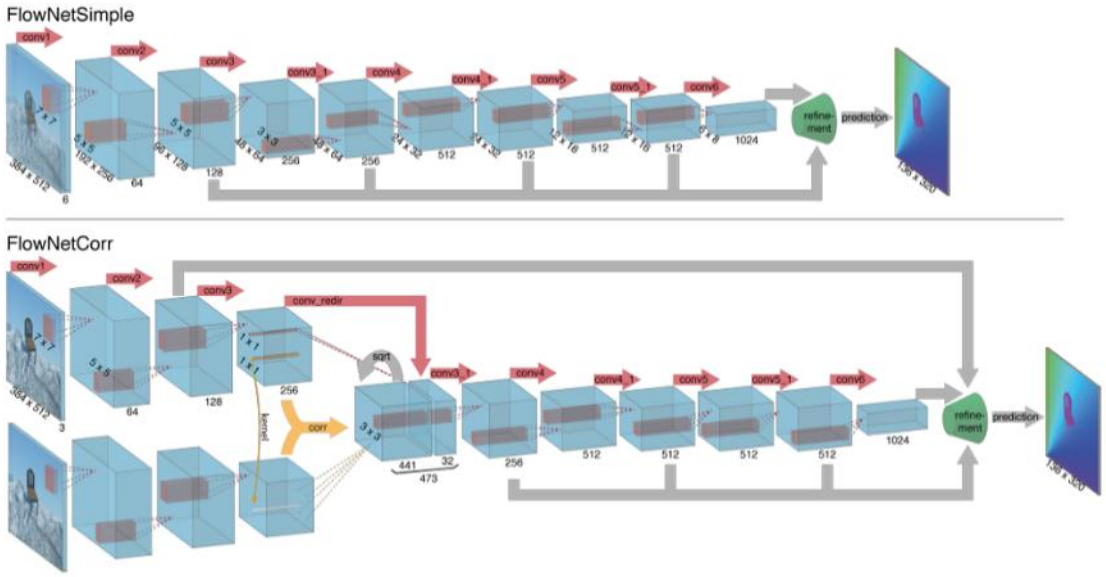



Figure 2. The two network architectures: FlowNetSimple (top) and FlowNetCorr (bottom).

## 3.4.2 Advantages of FlowNet

FlowNet offers several key advantages over traditional optical flow methods. First, it supports end-to-end learning, meaning it is fully trainable without the need for

manually designed features, which simplifies the process and improves flexibility. Additionally, FlowNet is much faster than traditional optimization-based methods, making it well suited for real-time applications where speed is crucial. It also offers excellent scalability, as it can generalize across a variety of scenarios when trained on diverse datasets, making it adaptable to different tasks. Finally, FlowNet adaptability allows for fine-tuning on domain-specific data, significantly improving its performance for specialized applications.

### 3.4.3 Why is FlowNet Unique?

FlowNet is unique for several reasons that set it apart from other systems. First, it performs end-to-end learning, meaning it learns the entire process of motion estimation directly from the data without the need for manual features. Additionally, it operates at high speed, making it suitable for real-time applications where fast processing is required. Its adaptability is another key advantage, as FlowNet can be fine-tuned for different applications by retraining it on custom datasets. Finally, its simplicity allows for quick understanding and easy implementation.

## 3.5 FlowNetSimple

FlowNetSimple is a basic version of a network designed to estimate motion in videos in a simple and effective way [3,13,15]. It works by taking two consecutive frames from a video and combining them into a single tensor with 6 channels (3 for each frame, representing RGB color). The network then uses convolutional layers to detect different patterns of motion and space. The initial layers focus on learning basic features like edges, textures, and motion, while the deeper layers capture features that are more complex. After the network learns these features, the image goes through de-convolution layers to expand it back to its original size, ensuring that the motion is predicted for every pixel in the image. The final output is a 2D optical flow field that shows the direction and strength of the motion for each pixel in the video.

## 3.6 FlowNetCorr

FlowNetCorr is an improved version of FlowNetSimple, adding a correlation layer to boost its performance [3,13,14]. Just like FlowNetSimple, it takes two consecutive frames and combines them into a tensor with 6 channels. The frames are processed separately through convolutional layers to extract features from each one. Then, the correlation layer calculates how similar the features from both frames are, helping the network had better capture long-range motion. After that, the combined features go through up sampling to create a dense optical flow field. The final output is a 2D optical flow field that shows the motion between the two frames.

## 3.7 YOLO

YOLO (You Only Look Once) is a real-time object detection system [1,4,6] developed by Joseph Redmon et al. YOLO uses deep learning to identify specific objects in images and videos. It is a single convolutional neural network that divides the input image into a grid system where each cell is responsible for detecting objects within itself. The network predicts both bounding boxes and class probabilities for these boxes simultaneously.

### 3.7.1 YOLO architecture

YOLO starts with an input image, which is divided into an N x N grid. Each cell processes the image independently to detect objects [16]. After being trained, YOLO can recognize specific object classes (e.g., person, dog, wolf, cow). For each cell, it predicts the probability of each class, stored in an array format (e.g., 50% person, 30% dog). YOLO also predicts several bounding boxes per cell, each defined by its center coordinates, width, height, and confidence score (indicating the likelihood of overlap with a real object).

The algorithm loops through each cell, extracting class probabilities and selecting the bounding boxes with the highest confidence scores. If a class's confidence exceeds a threshold, its prediction is added to the final list. The output is a list of detected objects,

each including the bounding box coordinates and predicted class, which are encoded later into a tensor with dimensions SxSx(B*5+C) [1,6].

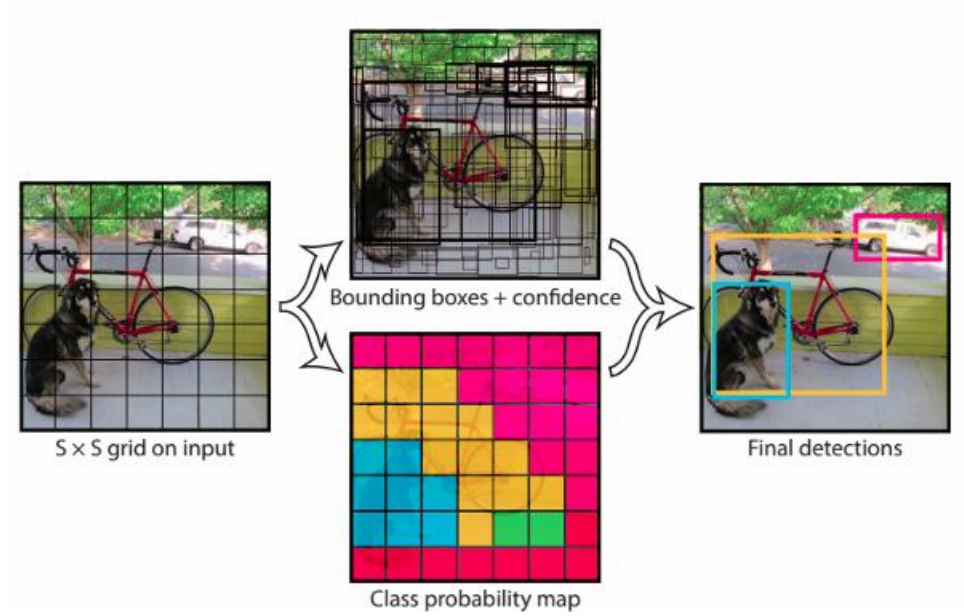



*Figure 3: Yolo bounding box prediction simulation [1, 6]*

### 3.7.2 Why is YOLO unique?

YOLO is unique by its ability to process information in real time. YOLO can handle up to 45 frames per second. The object detection is performed with only one evaluation of the neural network what ensures high efficiency. YOLO also can generalize and detect objects that were not specifically trained on [6]. So due to its speed, YOLO is promising for live streaming videos.

## 3.8 YOLO V8

YOLOv8 represents the latest version of the YOLO (You Only Look Once) object detection algorithm [1]. Over time YOLO architecture has evolved, with each new version introducing innovative updates that improve its performance and accuracy. When compared to YOLOv5, YOLOv8 offers faster and more precise object detection, making it the most accurate version to date and setting a new standard. The

architecture of YOLO consists of three key parts: the backbone, which extracts features; the neck, which combines features from various scales, and the head, which makes the final detection predictions. YOLOv8 keeps this structure intact but introduces several changes to enhance its performance. YOLOv8 uses the CIoU and DFL loss functions for bounding-box loss, along with binary cross-entropy for classification loss, which enhances object detection, especially for smaller objects.

## 3.9 FaceNet

FaceNet (developed by Schroff et al. [2] brings a significant advancement in face recognition technology. FaceNet is a deep learning model which converts facial images into compact Euclidean space embeddings where the distances between embeddings reflect the similarity between faces. The architecture of FaceNet includes three key components: the feature extraction network, the embedding layer, and the triplet loss function.

### 3.9.1 FaceNet Architecture

FaceNet model converts facial images into Euclidean space embeddings for efficient similarity measurement. Its architecture consists of three main components [1,2]:

**Feature Extraction Network**: Using a deep Convolutional Neural Network (CNN) based on the Inception architecture, it processes face images to extract high-dimensional features that are invariant to pose, lighting, and expression, and the network learns to recognize them.

**Embedding Layer**: The extracted features are mapped to a fixed-size vector (typically 128 dimensions), where the Euclidean distances between embeddings represent face similarity. This helps cluster images of the same person closer together and those of different people farther apart. This approach fastens the search through large, labeled faces dataset which has a various spectrum of face images from different subjects taken under many visual variations.

**Triplet Loss Function**: During training, FaceNet uses triplets consisting of an anchor image, a positive image (same person), and a negative image (different person). The

goal is to minimize the distance between the anchor and positive embeddings while maximizing the distance to the negative embeddings, enabling better clustering of faces. During testing, similar images are considered to belong to the same person, while faces of different people are widely separated by the x-y axes.

FaceNet has been highly effective in face recognition tasks, outperforming humans on datasets like LFW [2], and has driven the development of improved datasets like VGGFace2. Its unified embedding approach has significantly impacted both research and practical applications in biometric identification and computer vision.

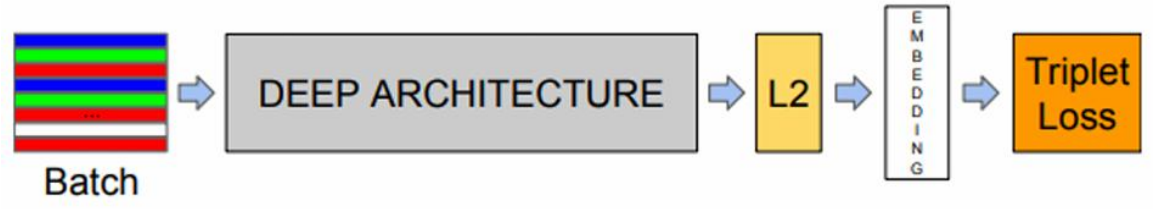



*Figure 4: FaceNet model structure [1, 2]*

The figure illustrates the model structure, where the network includes a batch input layer followed by a deep CNN. After processing through CNN, L2 normalization is applied to produce the face embedding, and during training, the triplet loss function is used to optimize the model.

### 3.9.2 Why is FaceNet unique?

FaceNet's unified embedding approach enables various face recognition tasks, such as verification, identification, and clustering. For example, one-to-one matching (face verification), one-to-many matching (face recognition) and clustering based on facial similarity. It has outperformed humans on benchmark datasets like LFW and influenced the creation of the VGGFace2 dataset [1, 2], which supports better face recognition across variations like pose and aging. FaceNet's focus on learning discriminative embeddings has shaped modern face recognition systems, with ongoing research addressing challenges like variations, fairness, and scalability. Overall, FaceNet has significantly advanced face recognition technology and its practical applications.

# 4 Research / Engineering Process

## 4.1 Research direction

We build on the findings of the last research project, AI-Driven Person Recognition and Identification in Videos [1] by Rom Harell and Itay Benjamin and upgrade it further for better performances. The previous project combines YOLO (You Only Look Once) algorithm for human detection and FaceNet algorithm for person face recognition. In our project we upgraded it with FlowNet algorithm for object tracking to cover up the lack of tracking and identification abilities in different constraints. Our approach promises significant improvements in the video person detection ability.

## 4.2 Development methods

### 4.2.1 Python for AI and Machine Learning as server

We consider using Python as it offers several advantages, including its simplicity and readability. Those make it ideal for rapid development and prototyping. The language is supported by a vast ecosystem of libraries and frameworks, such as OpenCV, Dlib, TensorFlow, and PyTorch, that provide ready-to-use tools for computer vision and machine learning tasks. Python benefits from strong community support and abundant online resources, facilitating problem-solving and collaboration. Python's versatility and easy integration with other technologies make it an efficient choice for developing AI and neural network applications [17].

### 4.2.2 React for the Frontend

We selected React for the frontend due to its flexibility, efficiency, and scalability in creating user interfaces [18]. Its component-based architecture enhances reusability and maintainability, while the virtual DOM optimizes performance by reducing unnecessary updates. React’s unidirectional data flow makes debugging easier, and its hooks simplify state management. Additionally, React's extensive ecosystem of tools and libraries, along with a strong developer community, speeds up development and

boosts productivity. Its seamless integration with various backend technologies further solidifies it as an ideal choice for modern web applications.

## 4.3 Algorithms and Dataset

### 4.3.1 YOLOv8 for object detection

In the previous project - AI-Driven Person Recognition and Identification in Videos (by Rom Harell and Itay Benjamin [1]), YOLOv8 and FaceNet were utilized for object detection and person identification. YOLOv8 was chosen for providing the best solutions among other environments as MobileNet V2 SSD or YOLO older versions (YOLOv5, YOLOv6, YOLOv7) [1] in terms of speed and accuracy.

### 4.3.2 FaceNet for Person Recognition

As we mentioned earlier, FaceNet model was chosen for face comparison and recognition process [1]. Using a triplet loss function, this model learns to distinguish faces with high accuracy, even under challenging conditions like lighting changes and partial occlusion. FaceNet is widely used in identity verification and surveillance systems due to its efficiency and robust performance.

### 4.3.3 Integrating FlowNet for Person tracking

In our updated version, we integrate the FlowNet optical flow algorithm (from the research FlowNet: Learning Optical Flow with Convolutional Networks by Dosovitskiy et al. [3]). We use it to enhance the system's motion analysis capabilities to meet the needs for advanced person recognition, especially in dynamic environments such as security and surveillance. Our system combines YOLO's real-time object detection with FaceNet's facial recognition, and now with FlowNet's motion tracking. Due to that, it efficiently handles challenges like changing lighting, resolution, and partial face occlusions. This integration improves tracking accuracy and computational efficiency by prioritizing areas of interest in video frames, ensuring more reliable identification even in complex scenarios.

### 4.3.4 COCO Dataset

For testing and evaluation, we are using the COCO 2017 Validation Dataset [19] as the last project [1], focusing on images containing people. This dataset offers detailed annotations, including bounding boxes and segmentation masks, which are vital for training advanced models. It contains 2,693 images that include the "person" category [1,19]. Using the COCO API from pycocotools, we filter the dataset to assess the model's performance on person detection with metrics like mean Average Precision (mAP) and Average Recall (AR).

### 4.3.5 FiftyOne for data visualization

We also use FiftyOne [20] to visualize and analyze the dataset, which enabling us to compare model predictions with ground truth annotations. FiftyOne provided tools for plotting precision-recall curves and confusion matrices, helping us refine the model and improve performance in real-world scenarios.

## 4.4 Methods to measure precision

### 4.4.1 Methods to measure precision of object detection

**Recall** - (also known as Sensitivity) measures how many of the actual objects in the dataset are detected by the model. It is defined as the ratio of true positive detections to the total number of ground truth objects (true positives + false negatives).

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

**Precision** - a measure of how many of the objects detected by the model are actually correct (i.e., true positives). It is defined as the ratio of true positive detections to the total number of detections made (true positives + false positives).

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

**Average Precision (AP)** - a measure that combines both precision and recall to provide a single metric for model performance. It is calculated as the area under the precision-recall curve. The precision-recall curve plots precision (y-axis) against recall (x-axis) for different threshold values. AP is usually computed for different Intersection over Union (IoU) thresholds, and the mean of these AP values gives a comprehensive view of the model's performance.

$$AP = \int_0^1 Precision(Recall)\ d(Recall)$$

**Average Recall (AR)** - is the average of recall values computed for different IoU thresholds and object categories. It provides an overall measure of the model's ability to detect objects across all categories and IoU thresholds.

$$AR = \frac{1}{N}\sum_{i=1}^{N} Recall_i$$

**Mean Average Precision (mAP)** - the average of the AP values computed for different IoU thresholds and object categories. It provides an overall measure of the model's performance across all categories and IoU thresholds.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \mid N\ is\ the\ number\ of\ IoU\ thresholds$$

**Intersection over Union (IoU)** - a metric used to evaluate the accuracy of an object detection model. It measures the overlap between two bounding boxes: the predicted bounding box and the ground truth bounding box. The IoU value ranges from 0 to 1 where: 0 indicates no overlap between the bounding boxes, and 1 indicates a perfect overlap between the bounding boxes. IoU is calculated as the ratio of the area of overlap between the predicted bounding box and the ground truth bounding box to the area of their union.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

## 4.4.2 Methods to measure precision of Person Identification

**Precision** - a measure of how many of the objects detected by the model are actually correct (i.e., true positives). It is defined as the ratio of true positive detections to the total number of detections made (true positives + false positives).

This same calculation is useful for both calculating Verification Precision and Identification Precision.

In **Verification Precision** it measures the accuracy of verifying whether two images belong to the same person (i.e., face verification). It is determined by comparing the predicted similarity score against a threshold, with a positive match being a true positive (TP) and a mismatch being a false positive (FP) or false negative (FN).

In **Identification Precision** this metric evaluates how many of the top-ranked candidates in a face recognition task are correct matches for the identified person. For each query face, FaceNet ranks potential matches from a database, and the top-ranked match is considered a true positive if it correctly identifies the person.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

**True Acceptance Rate (TAR) -** a metric measures the percentage of correct matches (true positives) from all attempted comparisons for a particular identity.

$$Tar = \frac{Number\ of\ CorrectMatches}{TotalNumber\ of\ Comparisons}$$

**False Acceptance Rate (FAR)** – a metric measures the proportion of incorrect matches, i.e., the number of times the system falsely accepts an incorrect identity as a match.

$$Far = \frac{FalsePositives}{FalsePositives + TrueNegatives}$$

**Rank-N Precision** - This method measures how many times the correct person appears within the top N ranked candidates provided by FaceNet during an identification task. Typically, N is set to 1, but it can be expanded to higher values to measure the system's performance when a larger set of candidates is considered. This provides an insight into how well FaceNet ranks faces in terms of their similarity to the query face.

$$Rank - N\ Precision = \frac{CorrectMatches\ in\ TopN}{N}$$

**Cumulative Matching Characteristic (CMC) Curve** - a plot of the recognition rate (i.e., the percentage of correct matches) versus the rank of the correct match in a list of candidates. It usually shows the percentage of correct matches as a function of the rank position.

$$CMC@Rank - N = \frac{Number\ of\ CorrectMatches\ in\ Top - N}{TotalNumber\ of\ Queries} \times 100$$

**Equal Error Rate (EER)** - a performance metric that evaluates the trade-off between the False Acceptance Rate (FAR) and the False Rejection Rate (FRR). The EER is the point where the FAR and FRR are equal and is used to measure precision by balancing the model's false positives and false negatives.

$$EER = FAR = FRR\ \ at\ the\ threshold\ where\ both\ are\ equal$$

**False Rejection Rate (FRR)** - this method measures the proportion of actual matches (true positives) that the system incorrectly rejects. It calculates the ratio of false rejections (when the system fails to recognize a correct match) to the total number of positive cases (true positives and false negatives).

$$FRR = \frac{FalseNegatives}{False\ Negatives + TruePositives}$$

### 4.4.3 Methods to measure precision of Person tracking

**End-Point Error (EPE)** – a method measures the difference between the predicted flow vectors and the ground truth flow vectors at the endpoints. It is one of the most used metrics to evaluate the accuracy of optical flow methods.

There $v_i$ is the ground truth flow vector for the pixel i, $\hat{v}_i$ is the predicted flow vector for the pixel i and N is the total number of pixels. A lower EPE indicates that the predicted optical flow vectors are closer to the ground truth, signifying better tracking accuracy.

$$EPE = \frac{1}{N}\sum_{i=1}^{N} \|v_i - \hat{v}_i\|$$

**Average Optical Flow (AOF)** - this metric evaluates the average magnitude of optical flow across all pixels or regions in the frame. It helps in assessing how well FlowNet tracks the motion of objects across frames.

There $v_i$ is the flow vector at pixel i and N is the total number of pixels in the frame. The AOF provides an overall measure of the motion in the video. A higher average optical flow typically corresponds to more significant movement, which is often desirable for tracking objects in dynamic scenes.

$$AOF = \frac{1}{N}\sum_{i=1}^{N} \|v_i\|$$

**Tracking Accuracy** - this metric evaluates how closely the predicted tracking paths of objects align with their ground truth trajectories. It assesses how well FlowNet tracks an object across multiple frames. The method counts the number of successful objects matches across consecutive frames, indicating how well FlowNet tracks an object's movement. A higher tracking accuracy means better tracking performance.

$$TrackingAccuracy = \frac{Number\ of\ CorrectObjectPredictions}{TotalNumber\ of\ ObjectPredictions}$$

**Motion Boundary Accuracy (MBA)** - a metric that evaluates the accuracy of object boundaries by comparing the predicted motion field with the actual motion in the frame. It is particularly useful for tracking objects that have sharp boundaries or undergo significant motion.

There $B_i$ is the true boundary at pixel i and $\hat{B}_i$ is the predicted flow vector for the pixel i. **IoU** stands for Intersection over Union, which measures the overlap between the true and predicted boundaries. A higher Motion Boundary Accuracy indicates that the optical flow method accurately tracks the boundaries of moving objects, which is important in scenarios like object segmentation or detection.

$$MBA = \frac{1}{N}\sum_{i=1}^{N} IoU(B_i, \hat{B}_i)$$

**Object Tracking Success Rate** – this metric measures the percentage of frames where the tracked object is correctly identified and followed over time. It helps evaluate how robust the object tracking is in maintaining the identity of the tracked object across frames. A higher success rate indicates that the object is consistently tracked correctly across frames. Success is typically defined by the object being correctly identified and staying within a predefined tracking region.

$$TrackingSuccess\ Rate = \frac{Number\ of\ Successful\ Frames}{TotalNumber\ of\ frames} \times 100$$

**Intersection over Union (IoU)** - a metric used to evaluate the accuracy of an object detection model. IoU for object tracking evaluates the overlap between the predicted bounding box of the tracked object and the ground truth bounding box at each frame.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

**F-Score for Object Tracking** - is the harmonic mean of precision and recall, used to evaluate the trade-off between the accuracy of tracking and the completeness of tracking across frames. The F-Score balances precision (how many of the predicted

tracks are correct) and recall (how many of the actual object tracks are captured), giving a holistic view of tracking performance.

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

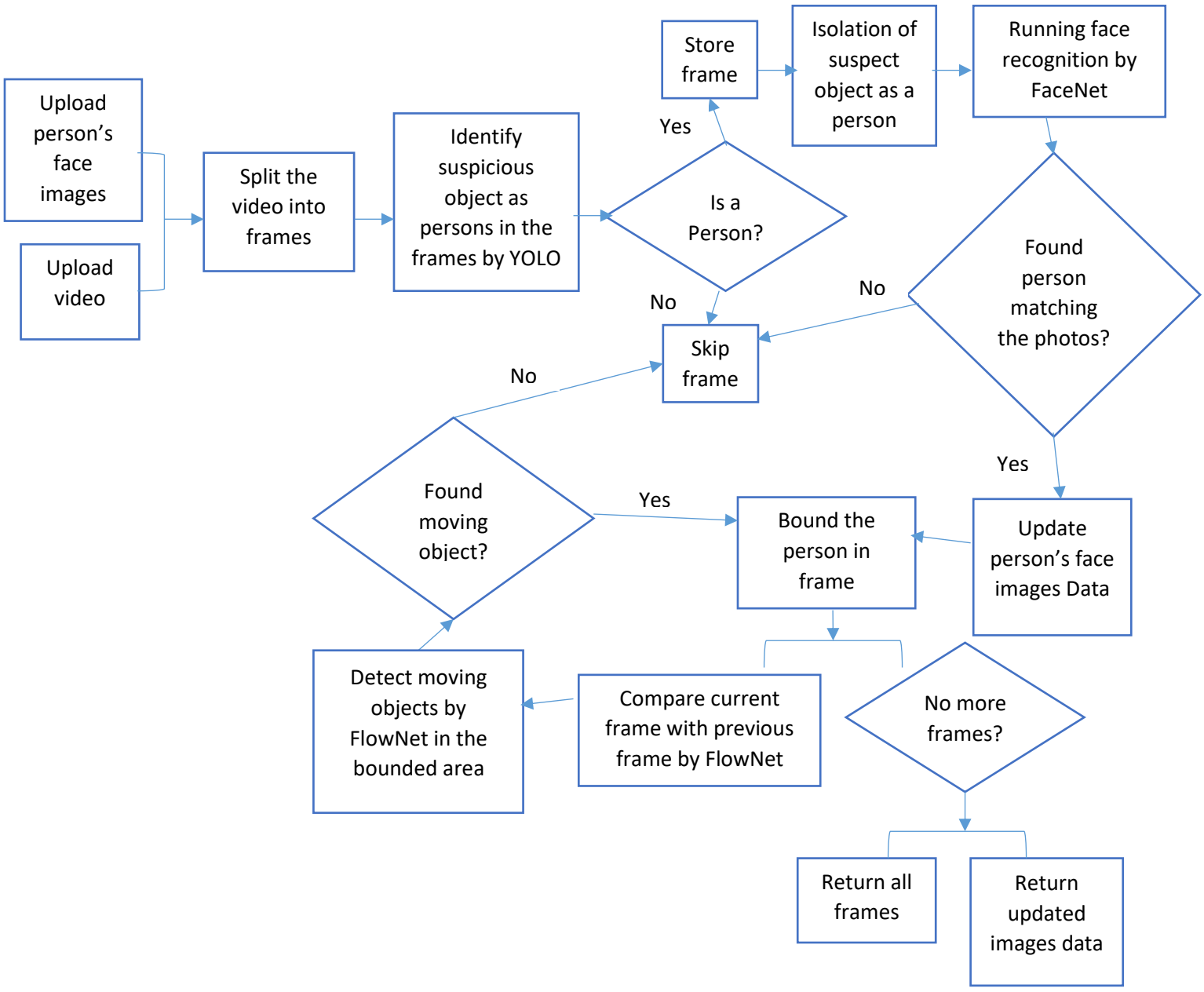
**Tracking Precision** - This evaluates the accuracy of the predicted positions of the object in the tracked frames compared to the ground truth positions.

There $P_i$ is the true position of the object in frame i and $\hat{P}_i$ is the predicted position of the object in frame i. N is the number of frames. These metric measures how close the predicted object positions are to the actual positions across all frames. Lower values of tracking precision indicate better tracking performance.

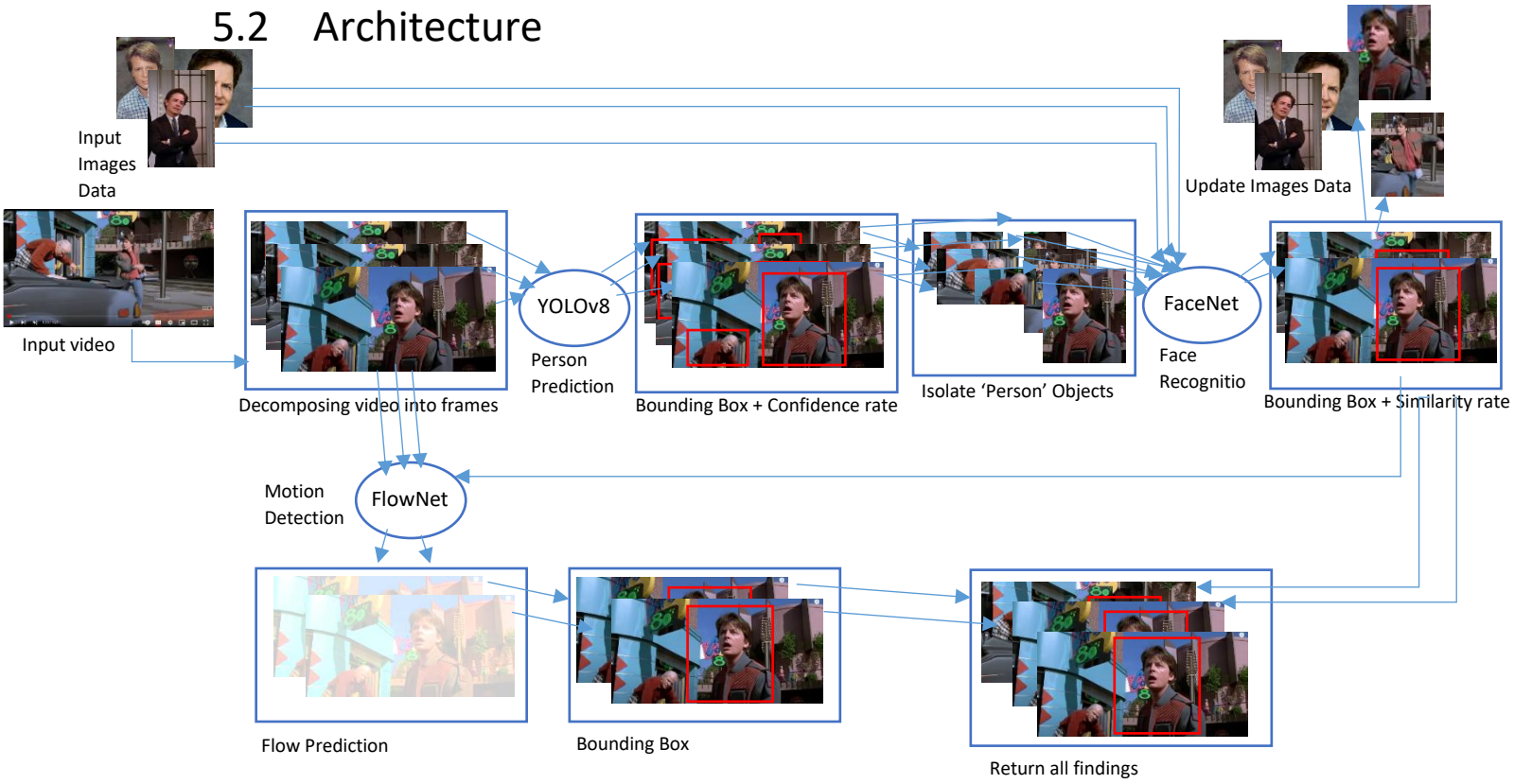
$$TrackingPrecision = \frac{1}{N}\sum_{i=1}^{N} \|P_i - \hat{P}_i\|$$

# 5 Work Artifacts

## 5.1 System Flow



## 5.2 Architecture




### 5.2.1 Algorithm Description –

**System Input:**
The system receives two inputs: a video that needs to be analyzed and some images of a specific person to compare against the video. The goal is to prepare these inputs for further processing and analysis.

**Preprocessing:**
The video is divided into individual frames (images), where each frame is saved and processed separately. This step allows the system to transform the video into a series of frames, which are easier to analyze individually. The output of this stage is a sequence of numbered frames ready for analysis.

**Detection with YOLO:**
For each frame, the YOLO (You Only Look Once) model is used to detect objects, particularly people, within the image. YOLO returns bounding boxes, which are

rectangular areas that indicate the location of detected individuals. This step is crucial as it narrows down the parts of the image that need further analysis, improving efficiency. The output is a list of bounding boxes for each frame, showing where people are located.

**Face Detection with FaceNet:**
The bounding boxes from YOLO are passed to the FaceNet model, which is designed to detect and recognize faces. FaceNet analyzes the faces within the bounding boxes and compares them to the data of input images of the person. This comparison allows the system to determine whether the person in the video matches the one in the image. The output of this step is a list of identified people, including a verification of whether the person from the image is present in the current frame, and his matching rate.

**Motion Analysis with FlowNet:**
FlowNet is used to analyze motion between consecutive frames. It compares the current frame with the previous frame. FlowNet computes the motion only in regions where YOLO detected people, and where FaceNet confirmed that the person matches the target images. This step produces motion data, including direction, speed, and movement path for each identified person.

**Result Integration:**
The system integrates the results from the previous stages: the location data from YOLO, the identity verification from FaceNet, and the motion data from FlowNet. For each person, the system combines and shows their detected location, identity, and movement across frames. The result is a comprehensive list of individuals found to be matching the person from the images provided by the user, showing bounded characters with matching rate, their location and how they move across the video. The system stores matching individuals' images for later comparisons and update image data with high rate findings.

**Final Output:**
The final output provides detailed information about each identified person, including whether they appear in the video based on the target images, where they are located in each frame and how they move overtime. The system also provides updated person's faces images data.

## 5.3 FR & NFR Requirements

### 5.3.1 Functional Requirements

| No. | requirement |
|---|---|
| 1 | The system shall provide a user interface for users to upload videos and images. |
| 2 | The system shall provide upload functionality. |
| 2.1 | The system shall allow users to upload a video. |
| 2.2 | The system shall allow users to upload pictures. |
| 3 | The system shall decompose the uploaded video into individual frames. |
| 3.1 | The system shall perform preprocessing on extracted frames |
| 3.2 | The system shall perform preprocessing on the uploaded pictures. |
| 4 | The system shall detect person in each frame using the YOLO model. |
| 4.1 | The system shall generate bounding boxes around detected persons. |
| 5 | The system shall isolate detected faces within bounding boxes. |
| 6 | The system shall generate embeddings for detected faces using FaceNet. |
| 6.1 | The system shall classify a detected face as a match if the similarity score exceeds a threshold. |
| 7 | The system shall save the matching detected face into FaceData. |
| 8 | The system shall analyze motion between consecutive frames using FlowNet. |
| 8.1 | The system shall correlate motion data with detected persons for tracking. |

| | |
|---|---|
| 9 | The system shall track detected persons across multiple frames. |
| 9.1 | The system shall store frames where the detected person appears. |
| 9.2 | The system shall store timestamps for matched detections. |
| 10 | The system shall display the user the video frames, the bounding boxes, the similarity score and the timestamps where detection occurred. |
| 10.1 | The system shall allow users to download results, including detected frames and motion data. |
| 12 | The system shall display error messages for invalid file uploads. |
| 13 | The system shall notify users if no match is found. |

### 5.3.2 Non-functional Requirements

| **No.** | **requirement** | **type** |
|---|---|---|
| 1 | The system will avoid running unnecessary person detections in frames where no object found. | performance |
| 1.1 | The system will avoid performing unnecessary calculations in frames where no expected match for the person we are looking for. | performance |
| 2 | The system should be capable of processing video streams in real-time, ensuring that motion detection, face recognition, and person identification happen with minimal latency. | performance |
| 2.1 | The system should ensure that each video frame is processed efficiently without causing significant delays. | performance |
| 3 | The system should be able to handle videos of varying lengths without significant performance degradation. | Scalability |

| | | |
|---|---|---|
| 3.1 | The system should be optimized to handle different video resolutions and qualities without compromising processing speed or accuracy. | Scalability |
| 4 | The system should be able to handle multiple images of person for search process. | Scalability |
| 4.1 | The system should be optimized to handle different picture resolution and qualities. | Scalability |
| 5 | The system must provide high accuracy in both face recognition (using FaceNet) and motion detection (using FlowNet) to minimize false positives and negatives. | Accuracy |
| 5.1 | The system should avoid unnecessary analysis by detecting and excluding frames where no people are present. | Accuracy |
| 6 | The system should be highly available, with minimal downtime. | Reliability |
| 7 | The system should provide an intuitive interface for users to upload video files and images, start processing, and view results easily. | Usability |
| 7.1 | The system should allow users to specify the person to track in a video through simple image input. | Usability |
| 8 | The system should be designed for easy maintenance and updates, with clear documentation and modular code. | Maintainability |
| 8.1 | The system should make it easy to find and fix problems quickly. | Maintainability |
| 9 | The system will use advanced machine learning algorithms for facial recognition (FaceNet), person detection (YOLO), and motion analysis (FlowNet) | Technology |
| 10 | The system should be optimized to process data quickly and efficiently. | Optimization |

## 5.4 Pseudo-Code

### 5.4.1 YOLO Pseudo-Code

**Input**:

frame - A single image or video frame for object detection.

**Output**:

List of detected objects with bounding boxes, class labels, and confidence scores.

1. Load YOLO Model:
   - Initialize YOLO with pre-trained weights and configuration.
   - Load predefined class labels (e.g., 'person').

2. Preprocess Frame:
   - Resize the frame to match the input size required by YOLO.
   - Normalize pixel values.

3. Perform Detection:
   - Pass the preprocessed frame through the YOLO model.
   - Extract raw detections (bounding boxes, class probabilities, confidence scores).

4. Post-Process Detections:
   - Filter detections based on confidence threshold.
   - Apply Non-Maximum Suppression (NMS) to reduce overlapping bounding boxes:
     - Retain the box with the highest confidence.
     - Discard overlapping boxes with high IoU.

5. Return Results:
   - For each valid detection, return:
     - Bounding box coordinates.
     - Class label.
     - Confidence score.

## 5.4.2 FaceNet Pseudo-Code

**Input**:

detected_faces - List of face regions detected by YOLO.

Input_image – List of input face images to compare against.

**Output**:

List of matched faces with bounding boxes and similarity scores.

1. Preprocess the input face and detected faces:
   - Resize the Input_image and detected_faces to match FaceNet input size.
   - Normalize pixel values to range [-1, 1].
   - Convert faces to tensor format.

2. Generate embeddings:
   -For each image in Input_image:
      - Pass it through the FaceNet model to obtain its embedding.
   - For each detected_face:
      - Pass it through the FaceNet model to obtain its embedding.

3. Compare embeddings:
   - For each detected_face embedding:
      - Compute similarity using cosine similarity: $\frac{\vec{b} \cdot \vec{a}}{\|\vec{b}\| \|\vec{a}\|} = \text{Cosine Similarity}$

        similarity = dot(embedding_ Input_image, embedding_detected_face) /
                (||embedding_ Input_image || * ||embedding_detected_face||)
      - If similarity > threshold:
        Mark the face as a match.

4. Return matched faces:
   - List of detected faces with bounding boxes and similarity scores above the threshold.

## 5.4.3 FlowNet Pseudo-Code

**Input**:

frame_n - Current frame

frame_n_minus_1 - Previous frame

regions_of_interest - Identified regions by FaceNet

**Output:**

Motion data for detected regions (e.g., motion vectors, direction, speed)

1. Preprocess the frames:
   - Resize frame_n and frame_n_minus_1 to match FlowNet input size.
   - Normalize pixel values to range [0, 1].
   - Convert both frames to blob format.

2. Extract regions of interest:
   - For each region detected by FaceNet in frame_n:
     - Crop the corresponding region in frame_n_minus_1.
     - Crop the same region in frame_n.

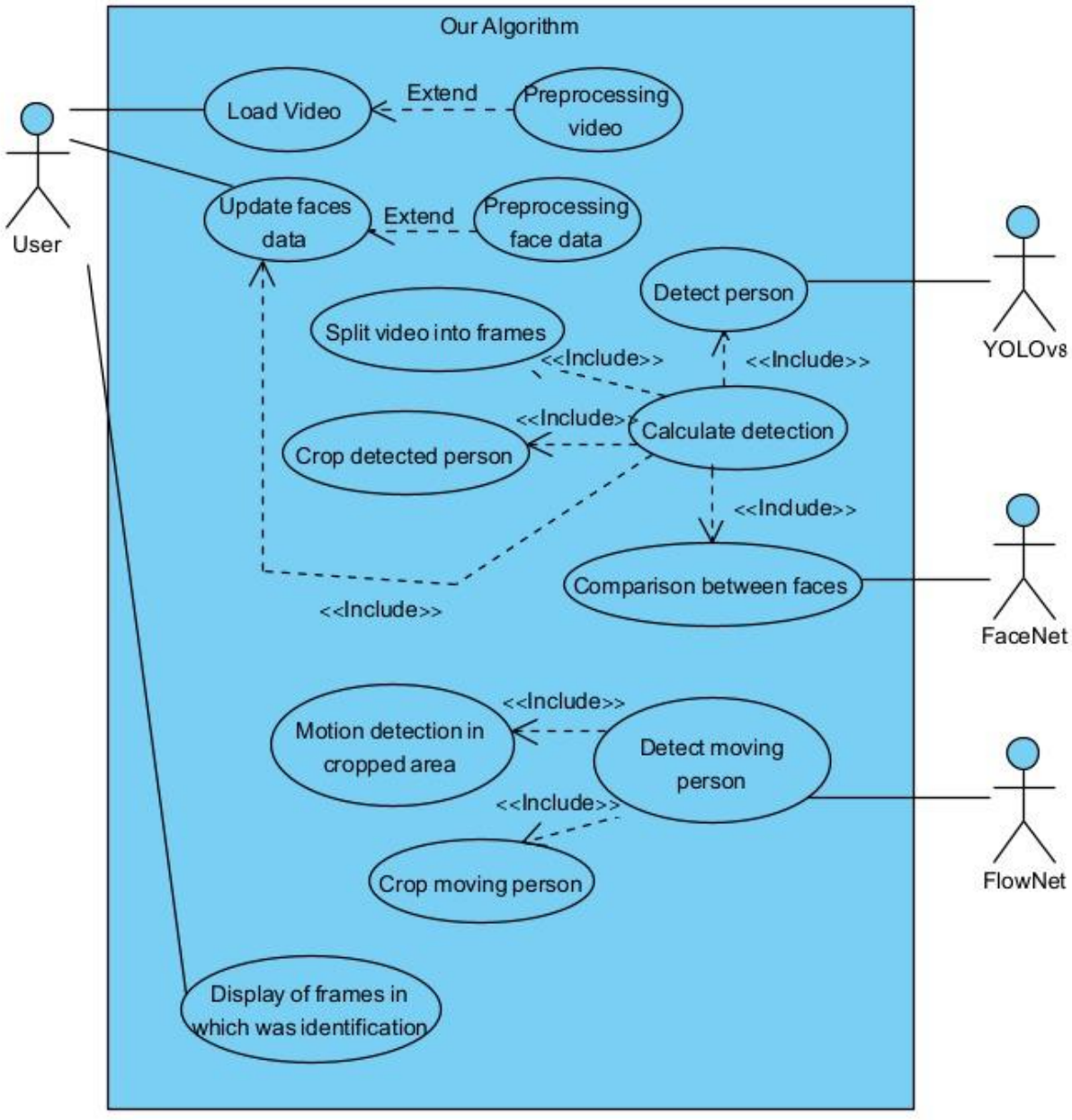
3. Compute optical flow:
   - For each region of interest:
     - Pass the cropped regions (from frame_n and frame_n_minus_1) through the FlowNet model.
     - Obtain motion vectors representing pixel displacements between frames.

4. Post-process motion data:
   - Aggregate motion vectors for each region.
   - Compute:
     - Average motion direction.
     - Average motion speed.

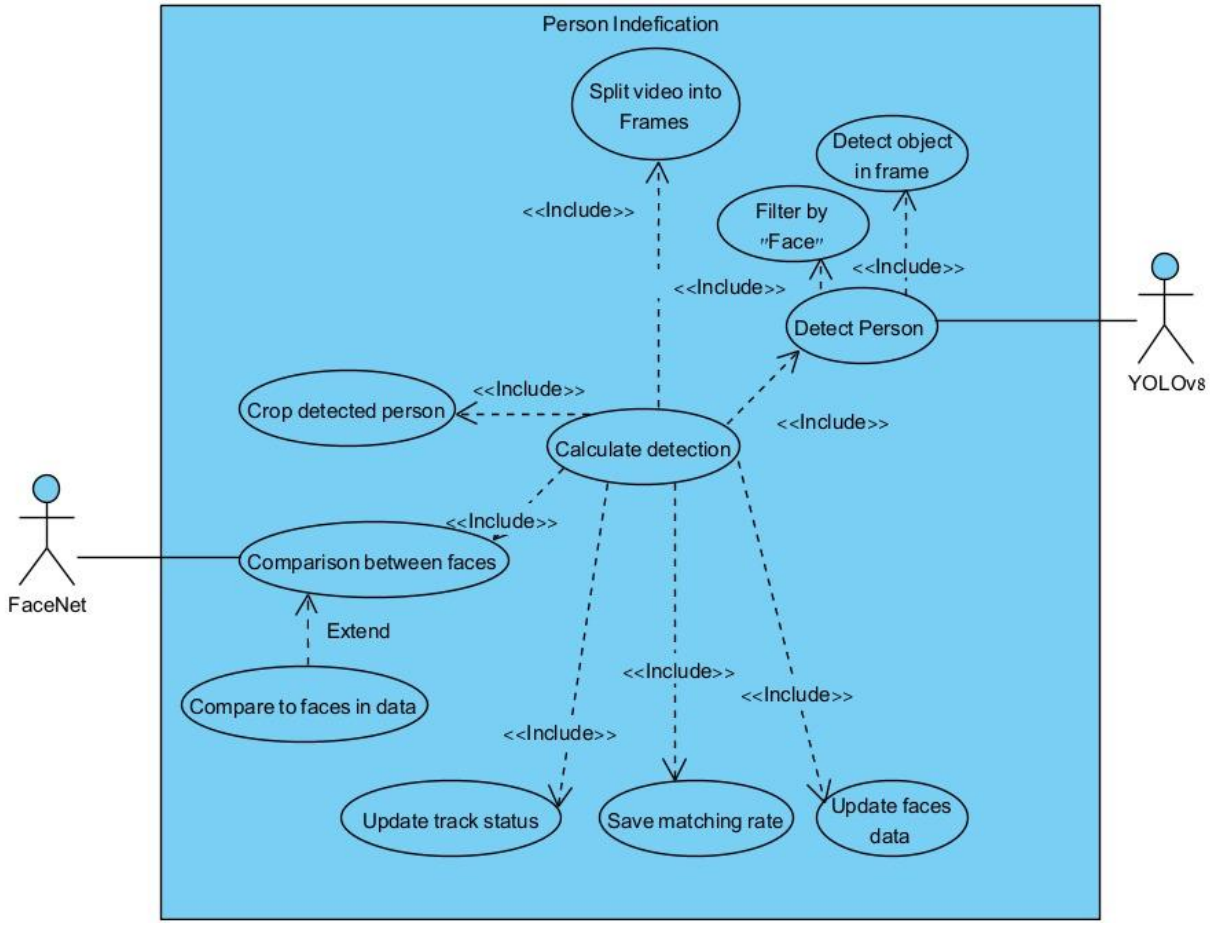
5. Return motion data:
   - For each region:
     - Motion vector (direction and speed).

## 5.5 Use Cases

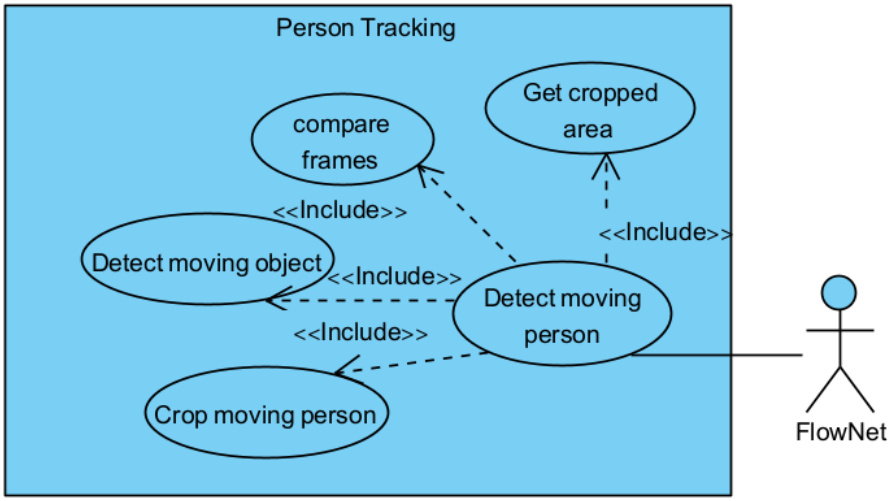
### 5.5.1 System Use Case



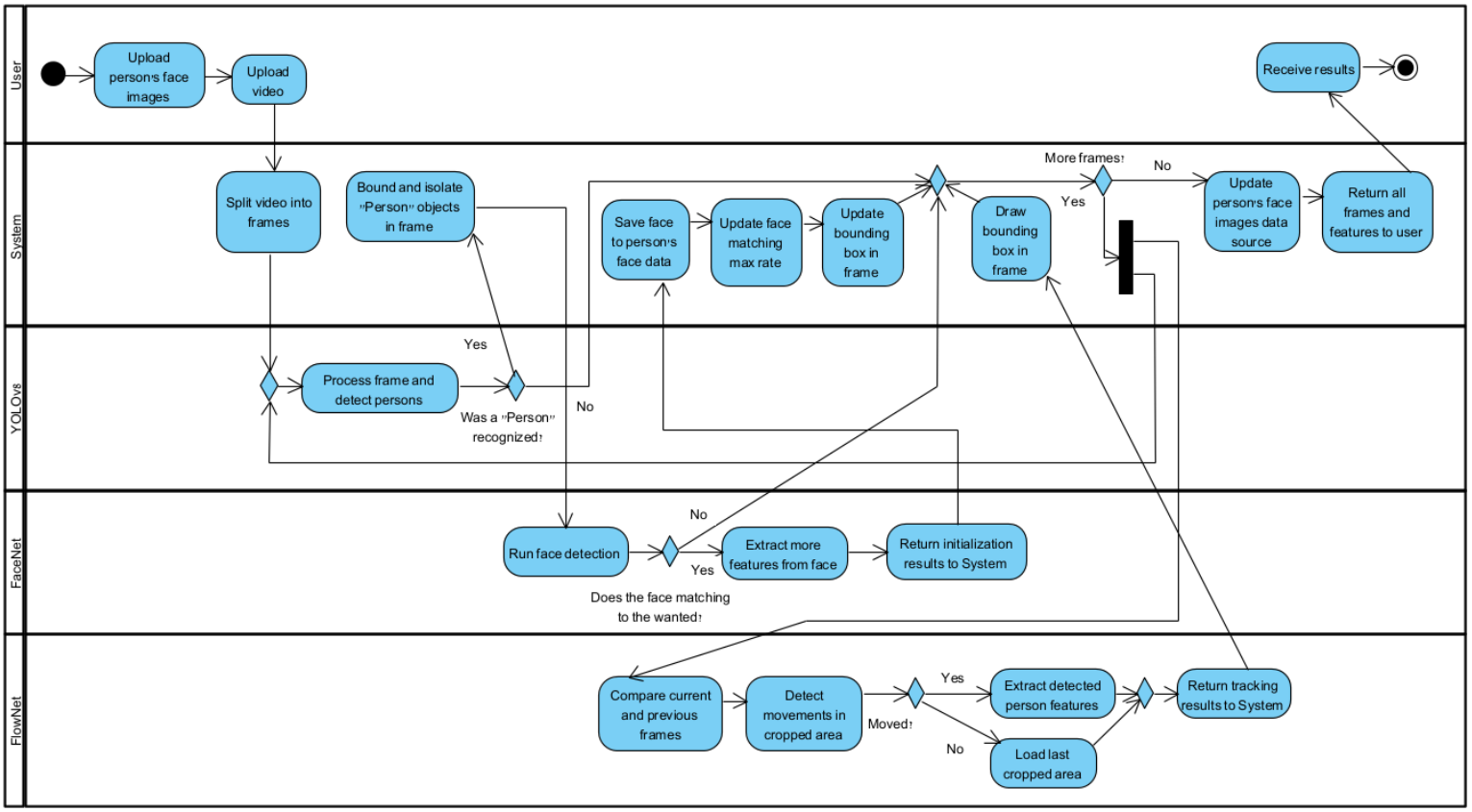
## 5.5.2 Person Initialization Use Case




## 5.5.3 Person Tracking Use Case

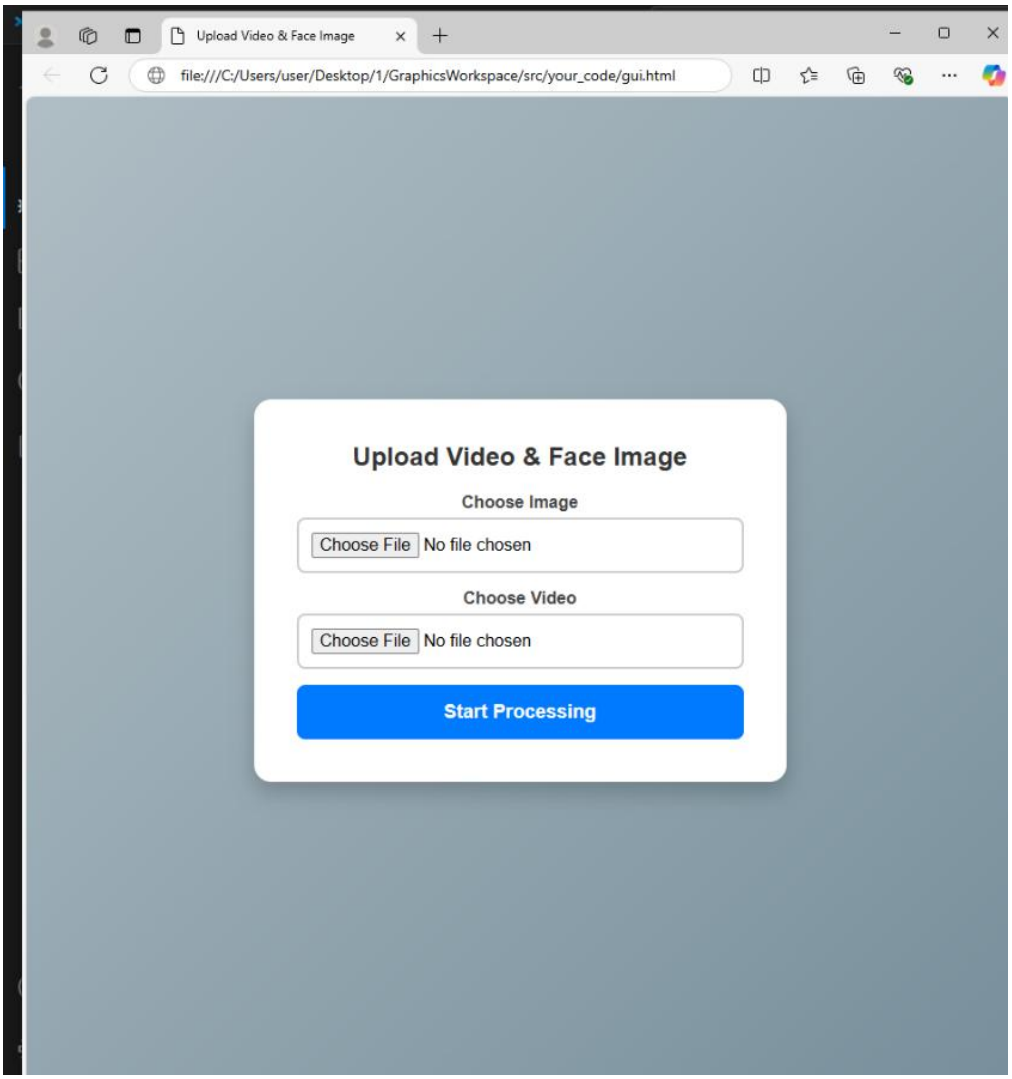


## 5.6 System Activity Diagram



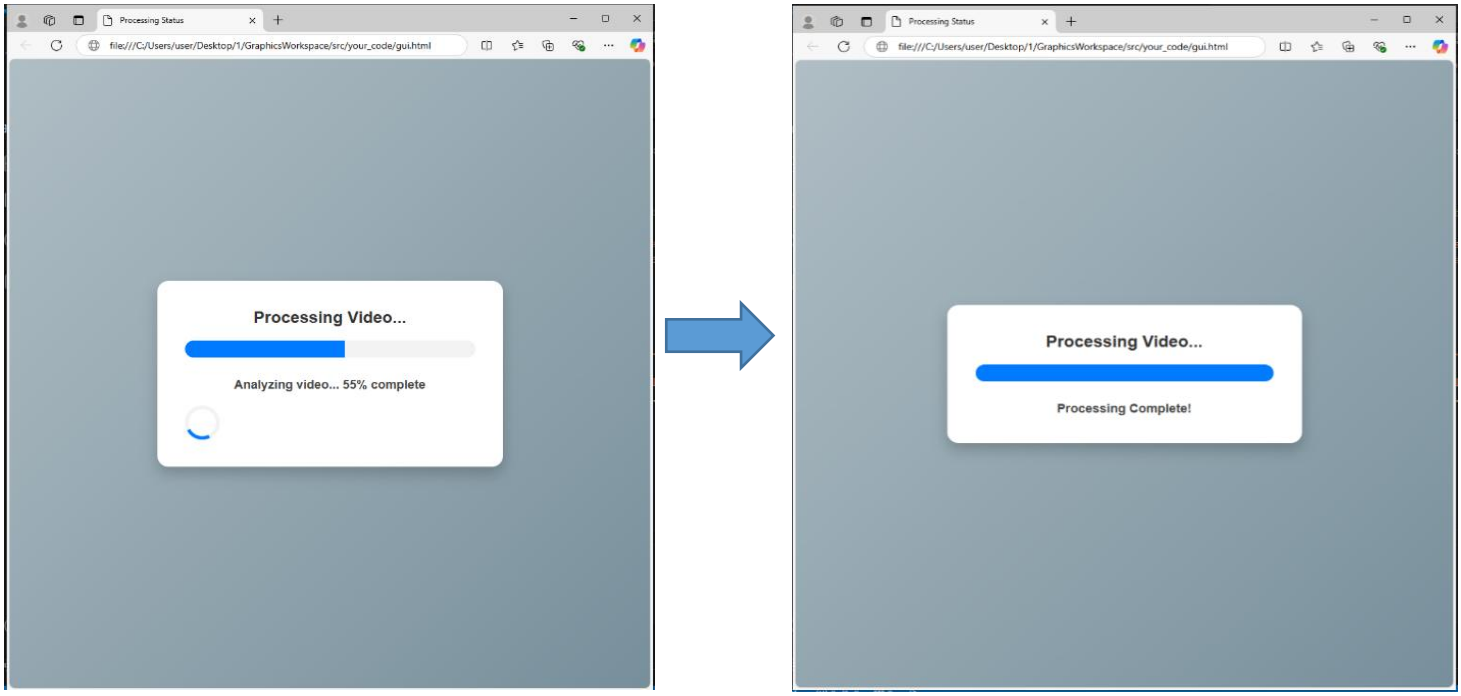
## 5.7 User Interface

**Page1**:
Upload a video file for analysis.
Upload a reference face image for identification.
Start the processing (YOLO + FaceNet + FlowNet).



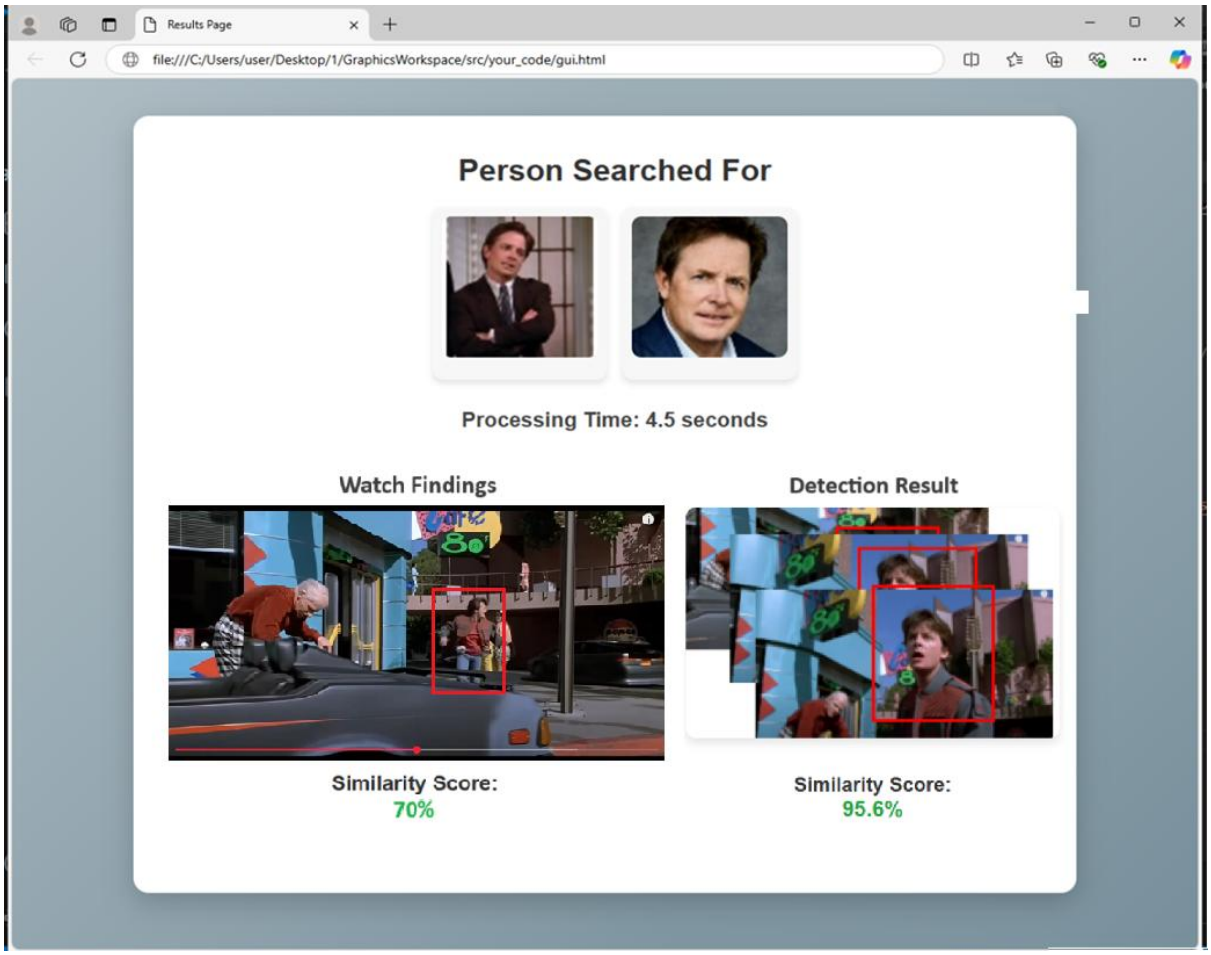
**Page2**:

This page displays a processing status for a video analysis.



**Page 3**:

This page displays the result of a search for a person, Showing the bounding box around the matching person in the video allowing the user to examine the character moves, and showing images of the search and the detection, along with the processing time and similarity score.



# 6 Testing Plan

## 6.2 Test Cases

| No. | Test Case | Test Description | Preconditions | Test Steps | Expected Result | Pass/Fail |
|---|---|---|---|---|---|---|
| 1 | Upload Video and Image | Verify that the user can successfully upload a video and an image. | The system should be up and running. | 1.Upload a video file<br>2.Upload an image of a person. | Both files should be successfully uploaded. | Pass if files are uploaded without errors.<br>Fail if an error occurs. |
| 2 | Decompose Video into Frames | Verify that the system correctly decomposes a video into individual frames. | A valid video has been uploaded. | 1. Click the "Process Video" button.<br>2. Check the output to verify that the video is broken into frames. | Each second of the video should result in a series of frames. | Pass if frames are generated correctly.<br>Fail if frames are missing or incorrect. |
| 3 | Detect Persons Using YOLO | Verify that the system detects persons in each frame using YOLO. | A valid video and image have been uploaded. | 1. Process a frame through the YOLO model.<br>2. Check the output for bounding boxes around detected persons. | Persons should be detected with bounding boxes surrounding them. | Pass if bounding boxes are correctly generated.<br>Fail if no bounding boxes are detected. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | Generate Embedding's Using FaceNet | Verify that the system generates embedding is for detected faces. | A valid frame with a detected face is processed. | 1. Pass a detected face through the FaceNet model.<br>2. Verify the generated embedding for the face. | The system should generate a unique embedding for each detected face. | Pass if embeddings generated.<br>Fail if embeddings are missing or incorrect. |
| 5 | Analyze Motion Using FlowNet | Verify that the system analyzes motion between consecutive frames. | The video is processed, and persons are detected. | 1. Process consecutive frames using FlowNet.<br>2. Check if motion data) is correctly calculated. | Motion data should be generated for detected persons between frames. | Pass if motion data is generated correctly.<br>Fail if no motion data is detected. |
| 6 | Track Detected Persons Across Frames | Verify that the system tracks detected persons across multiple frames. | The system has detected persons in the video. | 1. Track persons from one frame to another.<br>2. Check if the tracking is consistent across frames. | Persons should be consistently tracked across frames. | Pass if tracking is correct.<br>Fail if tracking is inconsistent. |
| 7 | Performance Test for Real-Time Processing | Verify that the system processes video streams in real-time without significant delay. | A valid video and reference image are uploaded. | 1. Start processing the video.<br>2. Measure the processing time for each frame. | The system should process each frame in real-time, with minimal latency. | Pass if the frame processing time is within acceptable limits.<br>Fail if there are significant delays. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | Scalability Test | Verify that the system can handle videos of varying lengths without significant performance worsening. | The system is ready to process video. | 1.Upload videos of different lengths<br>2. Measure the system's performance as the video length increases. | The system should be able to handle longer videos without performance worsening. | Pass if the performance is consistent across video lengths.<br>Fail if performance drops significantly. |
| 9 | Reliability Test for System Availability | Verify that the system is available and performs tasks with minimal downtime. | The system is running. | 1. Start processing a video.<br>2. Check if the system remains available and functional throughout the process. | The system should remain operational with minimal downtime during processing. | Pass if the system runs without interruption. Fail if there are crashes. |
| 10 | Handling Occlusion | Verify that the system can handle scenarios where a person is partially occluded or blocked and can detect them once they become visible again. | The system is ready to process video frames. | 1. Ensure the video contains a person who becomes partially occluded.<br>2. Play the video and wait for the person to reappear after the occlusion.<br>3. Verify if the system detects and resumes tracking the | The system should detect and track the person after they become visible. | Pass if the person is tracked correctly.<br>Fail if the system fails to detect the person after occlusion. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | person once fully visible again. | | |
| 11 | Handling Quick Movements | Verify that the system can track a person even if they move quickly across the frame. | The system is ready, and the video is already processed. | 1. Ensure the video contains a person moving quickly.<br>2. Play the video and check if the system can detect the person during their quick movement.<br>3. Verify that the system continues tracking the person without losing them. | The system should detect and track the person even during quick movements. | Pass if the person is detected and tracked correctly.<br>Fail if the system loses track of the person during fast movements. |
| 12 | Handling Low Lighting Conditions | Verify that the system can still detect and track a person even in low lighting conditions. | The system is ready, and the video is already processed.<br>The video contains low-light scenarios. | 1. Ensure the video contains a person in low lighting.<br>2. Play the video and check if the system detects the person under low-light conditions.<br>3. Verify that the system | The system should detect and track the person in low lighting. | Pass if the person is detected and tracked in low lighting.<br>Fail if the system fails to detect the person in low lighting. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | continues tracking the person despite the low lighting. | | |
| 13 | Handling Partial Occlusion by a Moving Object | Verify that the system can resume tracking a person after partial occlusion by a moving object. | The system is ready, and the video is already processed.<br>The video contains a person being partially occluded by a moving object (another person walking in front of them). | 1. Ensure partial occlusion by a moving object.<br>2. Play the video and check tracking after occlusion. | Tracking resumes once the person is fully visible again. | Pass if tracking resumes; fail if not. |

# 7 Expected Accomplishments

**Accurate Person Detection and Tracking:**

The system will detect people in video frames using YOLO, and even if someone is partially hidden or behind an object, it will pick them up again once they are fully visible.

**Facial Recognition Efficiency:**

The system will match faces using FaceNet, and even if there are some challenges like lighting changes or partial obstructions, it will still identify the person correctly.

**Motion Tracking:**

With FlowNet, the system will track the movement of people between frames. Even if their movement is blocked for a moment or they move intermittently, it will still capture their motion data accurately.

**Real-Time Processing:**

The system will process video streams in real-time, ensuring that everything happens smoothly without any noticeable delay. Motion detection, face recognition, and person identification will all work together seamlessly as the video plays.

## 7.1 Problems

**Achieving real time performance**: Such systems require significant computational power, yet they must also maintain high speeds. Therefore, optimization must always strike a balance between efficiency and accuracy.

**Low video quality**: A major challenge in person identification is dealing with low video quality, where factors like low resolution, noise, or blurriness can make it difficult to accurately identify individuals, especially when they are partially obscured or the video lacks clarity.

**Cross domain generalization:** It has been challenging to generalize models trained on one dataset to perform well on another or in real-world scenarios, yet this is critical for achieving reliable performance across all domains.

## 7.2 Criteria for Success

**Identification Accuracy**: The software should maintain a precision rate of at least 90% when verifying individuals through videos captured in any environment, as demonstrated by routine testing and validation.

**Tracking Success**: The system should consistently track the same identified person with an accuracy rate of at least 70% across consecutive frames, as validated by ongoing testing and evaluation.

**Operational Impact:** The system should lead to a 30% decrease in response time and enhance decision-making efficiency in security and emergency situations.

**User Feedback and Satisfaction**: Structured feedback from specialized teams should reflect a satisfaction rate of at least 80%, demonstrating the system's effectiveness and value to them.

**System Reliability & Performance**: The system must achieve at least 99% uptime, ensuring stable performance even during high load conditions or when exposed to different environments.

**Adoption & Engagement**: The system will be considered successful if it gains adoption from 50% of the target user groups within six months of its application.

# 8 References


[1] Ron Harel & Itay Benjamin (2024). AI-Driven Person Recognition and Identification in Videos.

[2] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

[3] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., & Brox, T. (2015). FlowNet: Learning Optical Flow with Convolutional Networks. IEEE International Conference on Computer Vision (ICCV).

[4] Horn, B. K. P., & Schunck, B. G. (1981). *Determining optical flow*. Artificial Intelligence, 17(1-3), 185–203.

[5] Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI).

[6] 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 (pp. 779-788).

[7] Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018, May). Vggface2: A dataset for recognising faces across pose and age. In 2018 13th IEEE international conference on automatic face &gesture recognition (FG 2018) (pp. 67-74). IEEE.

[8] Wang, M., & Deng, W. (2021). Deep face recognition: A survey. Neurocomputing, 429, 215-244.

[9] Jeremy Cohen. (2023 September). Computer Vision and Deep Learning: From Image to Video Analysis.

[10] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25

[11] Wang, P., Li, W., Gao, Z., Tang, C., & Ogunbona, P. O. (2018). Depth pooling based large-scale 3-d action recognition with convolutional neural networks. IEEE Transactions on Multimedia, 20(5), 1051-1061.

[12] 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 (pp. 779-788).

[13] FlowNet Pseudocode: GitHub - ClementPinard/FlowNetPytorch: Pytorch implementation of FlowNet by Dosovitskiy et al.

[14] FlowNetCorr Pseudocode: FlowNetPytorch/models/FlowNetC.py at master · ClementPinard/FlowNetPytorch · GitHub

[15] FlowNetSimple Pseudocode: FlowNetPytorch/models/FlowNetS.py at master · ClementPinard/FlowNetPytorch · GitHub

[16] YOLO Pseudocode: YOLO Pseudocode.py · GitHub

[17] Python: Welcome to Python.org

[18] React: React

[19] Cocodataset official COCO - Common Objects in Context

[20] FiftyOne Integrations COCO Integration: COCO Integration — FiftyOne 1.3.0 documentation

[21] images of person detection : https://www.ntu.edu.sg/rose/research-focus/deep-learning-video-analytics/person-re-identification