Mini Project Report

on

CCTV Footage Review Optimization

Submitted by

20BCS011 - Amit Appanna Talawar

20BCS014 - Anant Terkar

20BCS061 - Udit Jain

20BCS106 - Rahul Pentamsetty

Under the guidance of

Dr. Prabhu Prasad B M

Asst. Prof., Dept. of Computer Science and Engineering

and

Dr. Manjunath K V

Asst. Prof., Dept. of Data Science and Intelligent Systems

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

07/11/2023

Contents

List of Figures

List of Tables

1 Introduction

In today's world, surveillance plays an increasingly important role in terms of safety and security. Closed-circuit television (CCTV) provides an efficient way to monitor many different environments and are very easy to install and setup. Due to the fact that CCTV's run 24/7, the footage stored stores contains a lot of idle frames where there is no movement or activity and consequently reviewing CCTV camera footage for specific incidents is time-consuming and labor-intensive.

The process involves manually watching hours of footage to identify a small interval of movement related to the incident. Even when speeding up video playback, this approach does not significantly reduce the time and effort required. Even if we have an idea of the timeframe during which the incident has occurred, it is still an arduous process to go through the footage.

Taking all the above factors into consideration, there is an imminent need for a more efficient method to review and retrieve relevant CCTV footage for specific incidents. This project aims to introduce a system that aids in improving efficiency and reducing the time required for reviewing CCTV footage. The process of identifying specific incidents within the footage is automated and relevant video segments related to these incidents are efficiently extracted and presented. This enables quicker incident response and reduced manual effort.

2 Related Work

A survey that was conducted by IHS technologies in 2021 [1] estimates that there are over 1 billion CCTV cameras in the world. Due to the vast number of cameras, a huge amount of footage is stored. In 2012, Seagate presented a paper [2] that showed that compression techniques result in loss of video quality without much affecting the size of the video. Another method was introduced in 2016 [3] that proposes an algorithm that optimizes the storage of CCTV footage which is done by using Mean Squared Error(MSE) between the adjacent frames of the video clip and then deleting redundant frames by comparing the adjacent frames using the MSE.

Majority of the previous work fails to extract the actual segments during which motion is detected. They also fail to identify and recognize the objects present in the video. We also make use of a method called adaptive thresholding, that adjusts the thresholds in order to identify objects in any environment, something that other papers fail to perform.

3 Procedure

3.1 Steps

To achieve the goal of optimizing footage review we need to first split apart our problem into multiple parts:

1. Use motion detection to first highlight instances of motion
2. Save these instances onto a table
3. Using this table and the video, extract the snippets of the video with motion
4. Use object detection on these snippets to describe each of the video snippets with motion, and the objects identified in these instances,

4 Motion Detection

To achieve motion detection, one of the simplest and widely used methods is the frame difference technique. This approach involves computing the pixel-wise difference between two consecutive frames and analyzing whether there is a significant variation. If such a difference is detected, it's indicative of motion occurring within the video sequence. However, there is an important consideration when applying this method: the initial frame, which serves as a reference, cannot be assumed to represent a static background. Consequently, running a sequence-by-sequence frame differencing approach is often not suitable. A more effective approach is to establish a background model for the video, allowing it to serve as a reference static frame against which current frames can be compared. This background model can improve the accuracy of motion detection. To do this we use a method called Background Estimation.

4.1 Background Estimation

We employed median frames technique for estimating the background of an image. While this method may involve a slightly higher computational cost than calculating the mean, it offers greater resilience to outliers and transient objects in the video sequence.

In this approach, we rely on the median to create a robust representation of the background. This choice is made to reduce the impact of any temporary variations or moving objects that may appear in the video stream. By calculating the median of 50 randomly selected frames from the video, we ensure that the background model accurately reflects the stationary elements within the scene. For example, if more than 50% of the frames contain a car at a specific pixel, that car is considered part of the background in our estimation.

This median-based technique, combined with a sufficient number of frames, enhances the accuracy of motion detection by reducing sensitivity to temporary fluctuations and moving objects.

4.2 Thresholding

After using the background estimation, once we run the video, using grayscale images and gaussian blur we clearly see which part of the video has movement and not. To add bounding boxes to the video, we calculate the contours in the differential frame after thresholding and then if we do have considerable difference we approve the motion for the current frame.
Since our CCTV cameras did not all have the same characteristics of distance between objects and lighting, we needed to make sure that our code works for all cases.
We use image thresholding, to create a binary image from a grayscale image, to do image segmentation. We have implemented various types of thresholding to determine which method is ideal in our case of CCTV footage.

4.2.1 Global Thresholding

Often referred to as single-level thresholding; The primary goal of global thresholding is to classify each pixel in the image into one of two categories: foreground or background. This classification is based on the comparison of the pixel's intensity value to a single, predetermined threshold value.

The process of global thresholding can be broken down into the following key steps:

1. Threshold Value Selection: The first crucial step is to select an appropriate threshold value. This threshold value is typically determined based on the specific characteristics of the image and the requirements of the application. It is essentially a grayscale intensity level, often denoted as "T."

2. Binarization: Once the threshold value is established, every pixel in the grayscale image is examined. Each pixel's intensity value is compared to the selected threshold value. The pixel is then assigned to one of two categories based on this comparison:

i If the pixel intensity is greater than or equal to the threshold value (I ¿= T), it is classified as part of the foreground.

ii If the pixel intensity is less than the threshold value (I ¡ T), it is categorized as part of the background.

3. Binary Image Creation: As a result of this process, the grayscale image is transformed into a binary image, where the foreground is usually represented as white, and the background as black. The binary image is a visual representation of the segmented objects or regions of interest in the original grayscale image.

4.2.2 Otsu Binarization

Instead of relying on a fixed, predetermined threshold, Otsu's method automatically calculates the threshold that maximizes the separation of the two classes of pixels (foreground and background) in terms of their grayscale intensity values.
The following steps outline the functioning of Otsu's Binarization:

1. Histogram Analysis: The process begins with the generation of a histogram of the grayscale image. The histogram displays the frequency of occurrence of each intensity level, effectively providing a representation of the distribution of pixel intensities in the image.

2. Threshold Optimization: Otsu's method calculates the optimal threshold by iteratively evaluating every possible threshold value. At each iteration, it calculates the between-class variance and within-class variance. The between-class variance measures the separation

between the foreground and background classes, while the within-class variance quantifies the variance within each class.

3. Maximizing Variance: Otsu's Binarization aims to find the threshold value that maximizes the ratio of between-class variance to within-class variance. This is achieved by maximizing the formula for between-class variance divided by within-class variance.

4. Binarization: The threshold value that maximizes this ratio is then applied to the grayscale image. Pixels with intensities greater than or equal to the threshold are classified as part of the foreground, while pixels with intensities below the threshold are categorized as part of the background.

5. Binary Image Creation: The result is a binary image, where the foreground is usually represented as white, and the background as black.

4.2.3 Adaptive Thresholding

It is an image processing technique that dynamically determines local threshold values for different regions of an image. It contrasts with global thresholding, which applies a single threshold to the entire image. Adaptive thresholding takes into account the varying characteristics of different image regions, allowing for more accurate segmentation.
The process of adaptive thresholding involves the following key steps:

1. Local Region Division: The image is divided into smaller, non-overlapping regions or windows. Each region is analyzed independently.

2. Threshold Calculation: For each local region, a threshold value is computed based on the pixel intensities within that region. Common methods include finding the mean or median intensity of the region or using statistical measures like standard deviation.

3. Binarization: With individual threshold values determined for each local region, the image is binarized using these specific thresholds. This results in a binary image where each pixel's classification (foreground or background) is based on the threshold value specific to its local region.

4.2.4 Adaptive Mean Thresholding

The process of adaptive mean thresholding involves the following key steps:

1. Local Region Division: The image is divided into non-overlapping regions or windows, each treated independently.
2. Local Mean Calculation: Within each region, the mean (average) pixel intensity is computed. This local mean serves as the threshold value for that specific region.
3. Binarization: The image is binarized using the local threshold values. Pixels with intensities greater than or equal to the local mean are classified as foreground, while those with intensities below the local mean are designated as background.

4.2.5 Adaptive Gaussian Thresholding

The process of adaptive gaussian thresholding involves the following key steps:

1. Local Region Partition: The image is divided into non-overlapping regions or windows, each treated as an independent entity.
2. Local Gaussian Distribution: Within each region, a Gaussian distribution is fitted to the pixel intensities. The parameters of this distribution, such as the mean and standard deviation, are used to calculate a local threshold.
3. Binarization: The image is binarized using the local thresholds. Pixels with intensities greater than or equal to the threshold are assigned to the foreground, while those with intensities below the threshold are assigned to the background.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Figure 1. Gaussian Distribution

Where:

1. f(x) represents the probability density function.
2. mu is the mean (average) value of the distribution.
3. sigma is the standard deviation, which characterizes the spread or dispersion of the data.
4. x represents the pixel intensity values within the region.

In the context of adaptive Gaussian thresholding, the mean (mu) and standard deviation (sigma) are computed for the pixel intensities within a local region or window. These parameters are used to define the Gaussian distribution that best fits the pixel intensity data within that region.

Once the Gaussian distribution is characterized by its mean and standard deviation, you can set a threshold value based on these parameters. For instance, a common practice is to set the threshold as a multiple of the standard deviation (e.g., T=mu+ksigma), where T is the threshold value, mu is the mean, sigma is the standard deviation, and k is a user-defined constant that determines how many standard deviations you want to move from the mean to set the threshold.

4.3 Contours, Bounding Boxes and Timestamping

After thresholding we use the contours identified using our thresholding to create bounding boxes onto our video to highlight instances of motion. We maintain a variable 'status list' which is a 2 value list which signifies either motion or no motion. We declare start motion when there was no motion on the previous frame and motion on the current frame, and vice versa for end motion. On either of these instances we save the timestamp of the video. Once the video is completed processing, we deal with various exceptions :

1. Video starts with a motion and that single motion does not end
2. Video starts with motion and stops before video ends, but motion does not restart
3. Motion begins somewhere and does not end
4. Video begins with motion and there were multiple instances of start and stop motion

5. Video ends with motion

To further clean our table, we remove small instances of no motion between two instances of motion, this is done to reduce the number of times we have to run our object detection code. For example a 20-25 second video will be broken down to 2-3 instances of motion of approximately 5 seconds each, which reduces the footage review by 40 percent.

In a video which is 24 hours long, our table will be much longer and the percentage of content we have to review will be much lesser.

Instead of a user manually having to review the entire video within these timestamps, we can provide the user with the video snippets with only the highlighted motion. To do this we use the moviepy library to clip the videos according to the timestamps we saved earlier. We create a folder containing all the video snippets with motion.

5 Object Detection

5.1 MobileNet SSD

MobileNet SSD[4] is a combination of the MobileNet-v2 model and SSD approach. MobileNet-v2[5] is a convolutional neural network that is 53 layers deep. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. Also, MobileNet uses depth-wise separable convolution layers, which reduces computation complexity. Single shot multibox detection algorithm only takes a single shot at detecting multiple objects in an image. It uses a single deep neural network to achieve this with a non-max suppression layer to ensure that there's only one bounded box around an object. Since SSD is independent of its base network, MobileNet was used as the base network for SSD to tackle the problem of computation complexity. The setback of MobileNet SSD was that it performs poorly in detecting distant objects while showing considerable accuracy in detecting near objects.

5.2 YOLO (yolov8)

YOLOv8[6] is a new state-of-the-art computer vision model built by Ultralytics, the creators of YOLOv5. Building upon previous YOLO versions, YOLOv8 introduces new features and

optimizations for various object detection tasks in a wide range of applications.
YOLOv8 is an anchor-free detection system, unlike SSD-based models. Though YOLO is a state-of-the-art model, it is less accurate in detecting near objects than SSD-based models, but faster and less tolling on the processing unit.

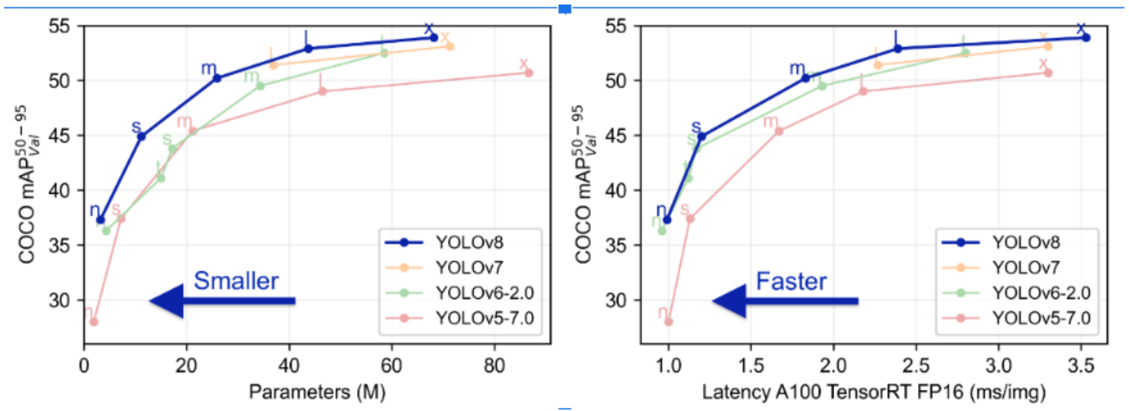


Figure 2. YOLOv8

6 Results and Discussions

6.1 MotionDetection

As we discussed various methods for thresholding during Motion Detection namely, Global Thresholding, Adaptive Thresholding. As we see in the Figure 3, 4, people moving far away from the camera are not being detected by the program. For global thresholding, we need to set a certain value for threshold, since the program, camera can be situated anywhere indoor, outdoor and far or close to moving objects, to configure a single value for each case, proves to be challenging and does not make sense for our use case. We use Adaptive Thresholding to fix this issue of setting a certain fixed value for the same.

We can see in the images below Figure 8, it is identifying the motion more accurately than the mean threshold 6, however due to the contouring and motion thresholds that we use Gaussian Thresholding is not able to capture all movements.

6.1.1 Global Threshold

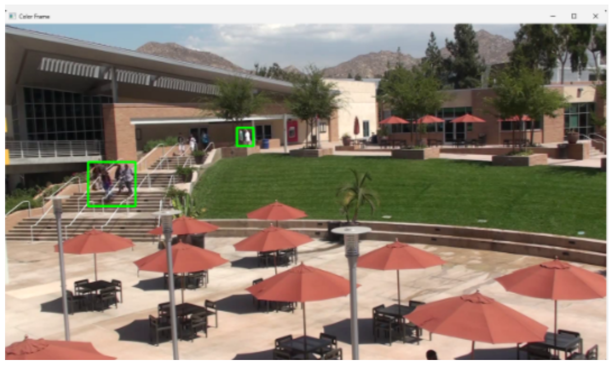


Figure 3. Global Frame

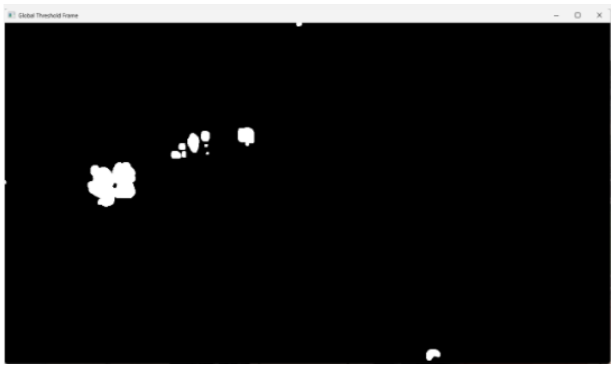


Figure 4. Global Threshold

6.1.2 Adaptive Mean Threshold

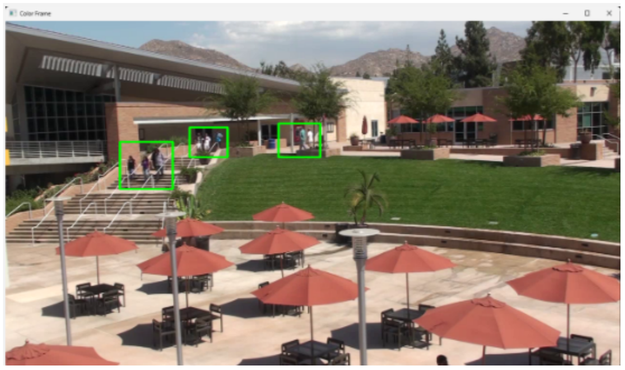


Figure 5. Mean Frame

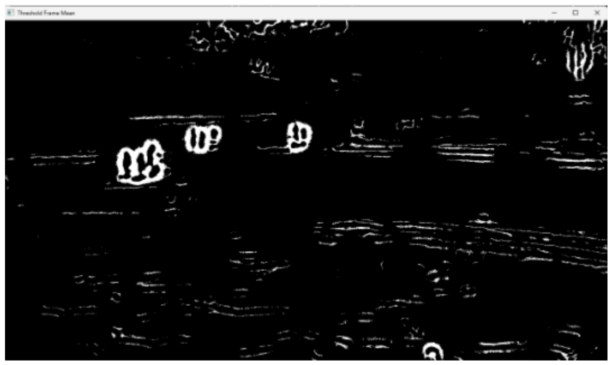


Figure 6. Mean Threshold

6.1.3 Adaptive Gaussian Threshold

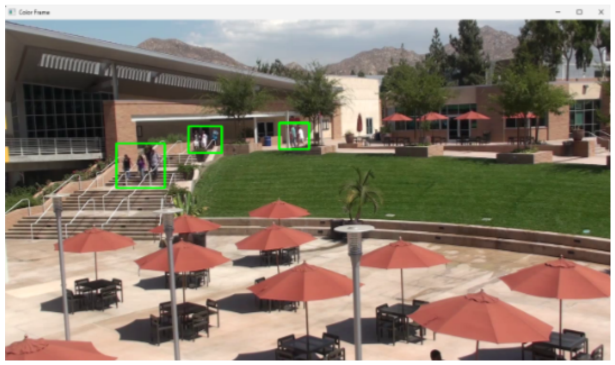


Figure 7. Gaussian Frame

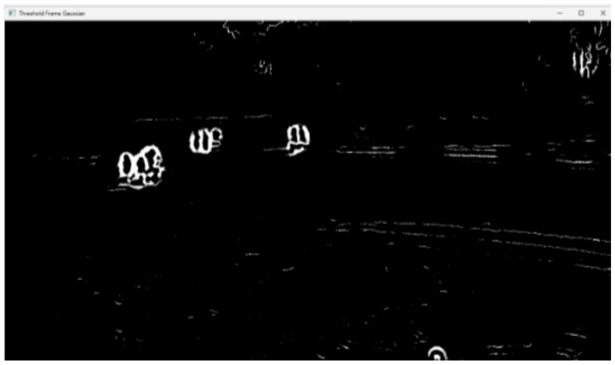


Figure 8. Gaussian Threshold

6.2 Object Detection Results

The dataset that we have used is taken from "https://viratdata.org/" and certain night-time CCTV clips. As the CCTV footage was of open areas, MobileNet SSD couldn't identify objects. This made us shift towards YOLOv8. After clipping the videos based on timestamps generated during which motion was detected, we ran yolov8 pre-trained on the VIRAT Ground video dataset. The objects detected and their timestamps:

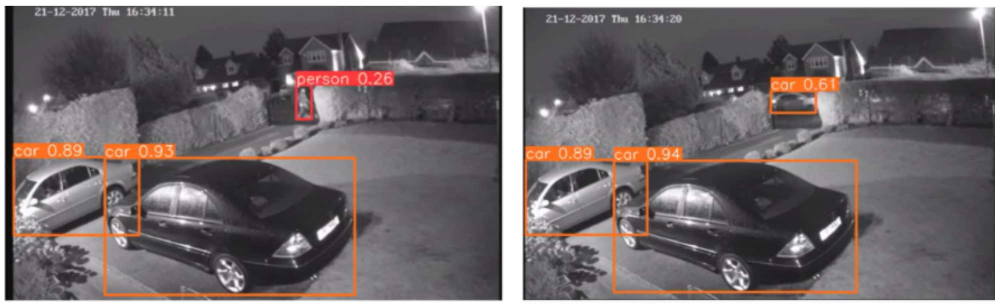


Figure 9. Night-time CCTV Snipet

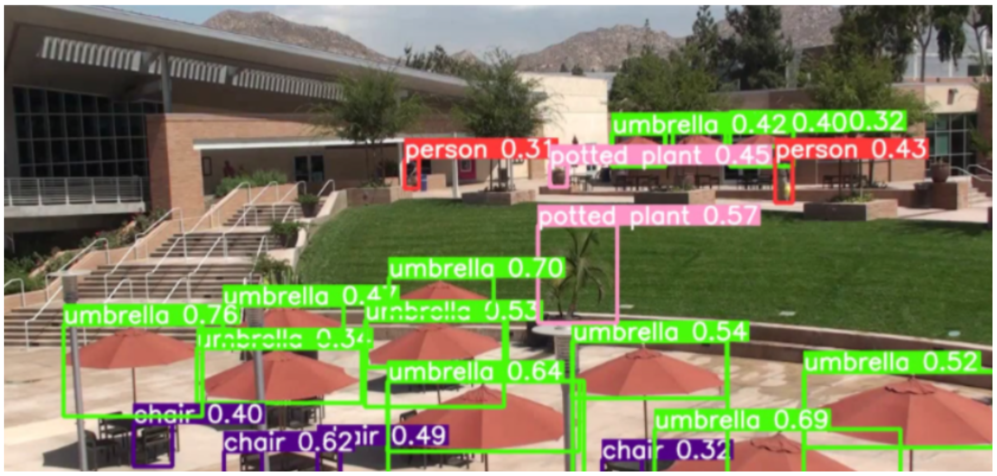
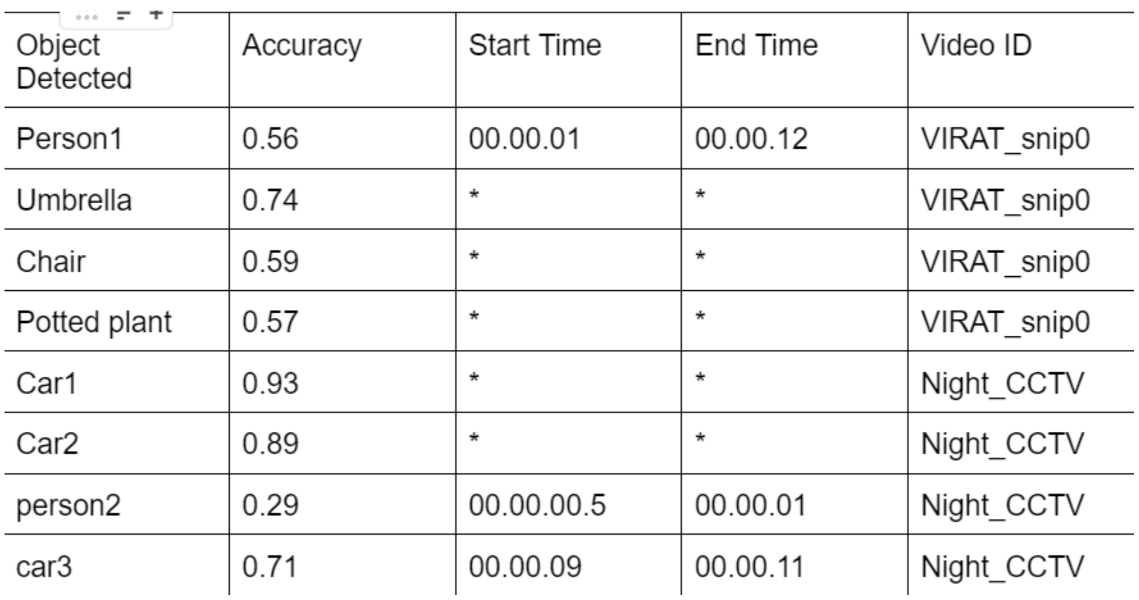


Figure 10. VIRAT Ground dataset with boxes



Object Detected	Accuracy	Start Time	End Time	Video ID
Person1	0.56	00.00.01	00.00.12	VIRAT_snip0
Umbrella	0.74	*	*	VIRAT_snip0
Chair	0.59	*	*	VIRAT_snip0
Potted plant	0.57	*	*	VIRAT_snip0
Car1	0.93	*	*	Night_CCTV
Car2	0.89	*	*	Night_CCTV
person2	0.29	00.00.00.5	00.00.01	Night_CCTV
car3	0.71	00.00.09	00.00.11	Night_CCTV

Figure 11. Timestamps With Object Detection

7 Conclusion

With the growing number of CCTV's all around the world, and also taking into factor the quality improvement, storage constraints and effort and time put into analyzing footages, this system will definitely help to improve efficiency vastly. Using the power of neural network models and the OpenCV library we are able to alleviate a majority of the issues faced during footage review and optimize the whole process and all while retaining the original quality of the footage, reducing size of the footage, and recognizing objects in the footage.

While performing motion detection in CCTV systems, we were presented with a nuanced challenge of dealing with the demands of calibration of varying thresholds. Specifically, objects in close proximity to the camera necessitate a higher threshold, while objects situated farther away call for a lower threshold to ensure accurate detection. Another problem that we faced was that while the substantial separation between the objects and the camera did not pose an issue in defining the bounding box around the moving object, the performance of the object detection models tends to be less accurate in the scenarios when the object is located in considerable distances as seen while making use of the Mobilenet SSD model

Looking at the future scope for the project, we can train the YOLO model on many other objects in order to enable detection in any environment. As the current system takes in an input and outputs video segments, we can further apply this onto live footage through the use of a buffer, that can enable us to optimize the storage of footage in real-time, rather than after. The system can be converted onto an application that can be seamlessly be executed with a single click, and the output would be a singular table with all instances of motion with objects detected in them. Another future scope of this project could be the inclusion of action-recognition that could be used to describe exactly what was happening in the video, further reducing footage review time.

References

[1] https://www.visualcapitalist.com/ranked-the-worlds-most-surveilled-cities/: :text=IHS

[2] Seagate, Video Surveillance Storage: How Much is Enough? White Paper, February 2012 http://www.seagate.com/files/staticfiles/docs/pdf/whitepaper/video-survstorage-tp571-3-1202-us.pdf

[3] S. Arora, K. Bhatia and V. Amit, ”Storage optimization of video surveillance from CCTV camera,” 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 2016, pp. 710-713, doi: 10.1109/NGCT.2016.7877503.

[4] S. Kumar, R. Kumar and Saad, ”Real-Time Detection of Road-Based Objects using SSD MobileNet-v2 FPNlite with a new Benchmark Dataset,” 2023 4th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2023, pp. 1-5, doi: 10.1109/iCoMET57998.2023.10099364.

[5] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C. ”MobileNetV2: Inverted Residuals and Linear Bottlenecks.” In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4510-4520). IEEE.

[6] https://docs.ultralytics.com/models/yolov8/.