Edge Computing-Based Indoor Video Surveillance System for Intruder Detection with Motion-Triggered Recording, Metadata Generation, and Remote Monitoring

Template Used
Camera Surveillance System from CM3065 – Intelligent signal processing

Student Name: **Ogbogolo Victor**
Student Number: **190125590**

Contents

1 Keywords

Indoor video surveillance system, intruder detection, edge computing, remote monitoring, motion triggered recording, metadata generation, motion detection, object detection

2 Literature Review

2.1 Motion detection for triggering indoor video surveillance recording and alert

In order to detect intrusions efficiently and accurately using a Video Surveillance System (VSS), it is essential for the system to recognize alterations in a scene and pinpoint areas of movement or activity within it. To accomplish this task, recognized methods in the field of computer vision, like Background Subtraction and Frame Differencing are typically utilized.

Frame differencing compares pixels of consecutive frames, applying thresholding to the absolute differences to obtain a binary image that distinguishes moving objects (appearing as white) from the static background (appearing as black).

Background subtraction, on the other hand, involves creating a background model presumed to be free of foreground objects and comparing it with incoming frames. By computing the pixel-wise differences between the background model and each incoming frame, thresholding is applied to these differences: pixels with differences exceeding a predetermined threshold are classified as foreground (indicating motion or change), while those below the threshold are classified as background.

While these methods are computationally efficient, they face challenges in dynamic environments due to their sensitivity to illumination changes, shadows, and noise.

2.1.1 Challenges in Background Subtraction Algorithms

Brutzer and Heidemann [1] identified seven key challenges that traditional motion detection methods must overcome in video surveillance applications:

1. Gradual Illumination Changes: Lighting conditions can change slowly over time, requiring the background model to adapt dynamically.
2. Sudden Illumination Changes: Abrupt changes, such as lights switching on or off, can lead to false positives.
3. Dynamic Background: Background objects exhibiting motion, such as rotating fans in indoor environments, can be mistakenly identified as foreground objects.
4. Camouflage: Objects that blend with the background pose classification difficulties.
5. Shadows: Shadows cast by moving objects can interfere with accurate detection.
6. Bootstrapping: The ability to initialize and update the background model even when initial frames contain foreground objects.
7. Video Noise: Sensor noise and compression artifacts can degrade performance.

Addressing these challenges is crucial for developing robust motion detection systems.

2.1.2 Advanced Background Subtraction Techniques

2.1.2.1 Visual Background Extractor (ViBe) [2]

Brutzer and Heidemann further evaluated nine different background subtraction methods employing various model representations, features, and update mechanisms. Among these, the Visual Background Extractor (ViBe) stood out due to its promising performance. They highlighted several strengths of ViBe:

- **Simplicity and Efficiency**: Minimal parameters make it suitable for real-time applications.
- **Adaptability**: Effective in dynamic scenes and gradual illumination changes.
- **Noise Handling**: Performs well in conditions with significant sensor noise.

ViBe [2] employs a pixel-based approach for background modeling in video sequences. For each pixel, ViBe initializes a background model using a sample set of pixel intensities, which may include values from neighboring pixels to incorporate spatial context. When processing a new frame, each pixel is compared against its corresponding sample set in the background model. If the pixel's intensity matches a sufficient number of samples—based on a predefined threshold—it is classified as background; otherwise, it is classified as foreground.

The background model is updated probabilistically: pixels from the current frame are occasionally added to the sample set to adapt to scene changes like lighting variations or gradual movements. To maintain spatial consistency and reduce noise, ViBe also occasionally updates the sample sets of neighboring pixels.

Brutzer and Heidemann highlighted that ViBe's strengths include its simplicity, computational speed, and adaptability to dynamic scenes, making it well-suited for real-time applications. However, they also noted that ViBe may struggle with challenges such as shadows or sudden, significant changes in the background.

2.1.2.2 Improved ViBe Techniques

Tang et al. [3] enhanced the ViBe algorithm to better handle challenges like dynamic backgrounds, shadows, and the ghost area problem (bootstrapping). Instead of relying on a single frame for the initial background model, they employed a mode background modeling technique that analyzes the first few frames to create a more robust initial model. This approach mitigates the ghost area problem by allowing the background model to gradually incorporate ghost regions—areas where foreground objects were initially present—into the background over time.

Furthermore, unlike the classic ViBe that uses a fixed detection radius, their improved ViBe dynamically updates key parameters such as the detection radius and the update rate based on differences between consecutive frames. By adjusting these thresholds in response to scene changes, the algorithm better adapts to environmental complexities, enhancing its ability to handle dynamic backgrounds and improving overall detection accuracy.

2.1.3 Advanced frame differencing Technique

Khadse and Pardeshi [4] proposed a motion detection system capable of distinguishing actual motion from lighting changes, thereby ensuring robust detection. The authors enhance the traditional frame differencing method by computing histogram differences between consecutive frames instead of performing pixel-wise subtraction. This approach helps to account for global illumination variations that might otherwise be mistaken for motion. Subsequently, they apply the Sobel edge detection algorithm to enhance edges and transitions in the image, improving the clarity of detected motion areas. Upon detecting motion, the system triggers an alarm, initiates video recording for storage, and sends an email notification to users. The authors assert that their algorithm effectively distinguishes motion from variations in lighting conditions.

While the paper presents a promising approach, it has certain limitations:

- Lack of Quantitative Analysis: The paper does not provide detailed quantitative results or analyses to support the claim that the system performs well under various lighting conditions.
- Limited Comparative Evaluation: It lacks a detailed comparison with other contemporary techniques such as Gaussian Mixture Models (GMM), ViBe, and other advanced motion detection algorithms, many of which have ready-to-use implementations in OpenCV commonly used in video surveillance.

2.1.4 OpenCV Implementations of Advanced Motion Detection Algorithms

- It is worth noting that OpenCV [5] provides several advanced background subtraction implementations capable of addressing key challenges faced by video surveillance systems (VSS) in the computer vision domain. These algorithms are readily available and suitable for use in developing robust motion detection systems:
- BackgroundSubtractorMOG2: Based on an adaptive Gaussian mixture model, this algorithm estimates the background of a scene by modeling each pixel as a mixture of Gaussians. It can handle illumination changes and incorporates shadow detection, enhancing its robustness in varying lighting conditions.
- BackgroundSubtractorKNN: This algorithm uses a K-nearest neighbors approach for background/foreground segmentation. It models each pixel by a set of recently observed samples and classifies a pixel based on whether its intensity is close to that of its neighbors. This method is effective in handling noise and dynamic backgrounds.
- BackgroundSubtractorGMG: A probabilistic foreground segmentation algorithm that identifies possible foreground objects using Bayesian inference with an estimated time-varying background model [6]. It combines statistical background image estimation and per-pixel Bayesian segmentation, allowing it to adapt quickly to changes in the scene.
- BackgroundSubtractorGSOC: An algorithm that outperformed others in OpenCV on the CDNET 2014 dataset. GSOC (Global SVD and Optical Flow Clustering) incorporates spatial consistency and temporal information, improving robustness to dynamic backgrounds and noise. It effectively handles shadows and illumination changes.
- BackgroundSubtractorLSBP: This algorithm utilizes Local Binary Pattern (LBP) based features with Local Singular Value Decomposition (SVD) to gain robustness to illumination changes and noise. The authors claim it outperforms other popular advanced background subtraction techniques such as ViBe and GMM in terms of precision, recall, and F1-score on the CDNet2012 dataset.
- By leveraging these advanced algorithms available in OpenCV, developers can implement effective motion detection systems that address many of the challenges associated with traditional methods. These implementations offer a balance between computational efficiency and accuracy, making them suitable for real-time applications in video surveillance.

2.1.5 Post-Processing Techniques

Brutzer and Heidemann [1] also emphasized the significant role of post-processing techniques in enhancing the performance of background subtraction methods. They asserted that post-processing can bridge the performance gap between weaker and stronger background subtraction algorithms, making even less sophisticated methods more competitive. They discussed key techniques that may be useful for this project:

1. **Median Filtering:** Applying a median filter to the foreground masks proved highly beneficial, particularly with mask sizes ranging from 14 to 20 pixels. This technique effectively smooths out noise and spurious pixel classifications, leading to a more refined foreground segmentation.
2. **Morphological Operations:** Combinations of morphological operations, specifically opening and closing, demonstrated notable performance improvements. The authors found that applying a small opening followed by a large closing operation yielded the best results. These operations help remove small, isolated foreground regions (noise) and fill in gaps within larger foreground objects, respectively.

2.1.6 Evaluation – Key Takeaways Applicable to the Project

1. General Evaluation of Background Subtraction Techniques:
   The majority of advanced background subtraction techniques build on the core idea of comparing a background model with incoming frames to segment foreground objects. However, they differ from each other based on the techniques they employ to handle the following key areas:

   - **Background Model Update Mechanisms:** Techniques like ViBe consider spatial context while updating the background, which helps address slow-moving background objects and effectively handle small camera movements.
   - **Adaptive Learning Rates:** By adjusting the learning rates controlling the background update speed, these algorithms become more robust to variations in background motion, reducing the likelihood of incorrectly classifying background motion as foreground objects.
   - **Dynamic Threshold Adaptation:** Instead of relying on predetermined thresholds, some methods dynamically adapt thresholds for foreground classification, primarily addressing the challenge of achieving robust foreground detection under various lighting conditions and scene complexities.
   - **Combination of Approaches:** Some techniques combine the above-mentioned approaches to improve accuracy and minimize false alarms.

   Project Benefit: This knowledge can be leveraged in the project to ensure the robustness of the selected algorithm, specifically by focusing on dynamic updates and adaptability to environmental changes.

2. Emphasis on Excluding Shadow Elimination Mechanisms:
   Motion detection will be used to trigger video recording and alerts, so shadows are not considered problematic since they follow the intruder's movement. Ignoring shadows helps simplify processing, leading to faster computation and enabling more efficient real-time detection. Therefore, algorithms

with built-in shadow elimination mechanisms will be avoided if these mechanisms cannot be disabled. Instead, the focus will be on handling dynamic backgrounds, illumination changes, and bootstrapping.

3. Testing Multiple Algorithms in OpenCV:
   Since there are numerous advanced motion detection algorithms, including those already implemented in OpenCV, these can be tested on indoor video datasets to determine their performance in conditions similar to the target environment. Emphasis should be placed on adaptability, accuracy, and computational efficiency. Specific evaluation metrics like precision, recall, F1-score, and latency should be used to comprehensively evaluate each algorithm. The best-performing algorithm will be selected as the main motion detection core of the project.

4. Hybrid Approach to Adapt Algorithms to Conditions:
   Consider a hybrid approach instead of relying on a single method. For instance, if Algorithm A performs better at night and Algorithm B performs better during the day, then the project design will consider using both, switching between them based on the time of day when they are most effective. Implementation can be managed using a time-based logic to automate the switching process and optimize performance under different lighting conditions.

5. Applying Post-Processing Techniques:
   The project design can benefit from applying post-processing techniques to achieve better results. Techniques such as median filtering and morphological operations (e.g., opening and closing) can significantly enhance foreground detection accuracy by smoothing out noise, removing small isolated regions, and filling in gaps within larger detected objects. Incorporating these techniques in the motion detection pipeline will help refine detection and reduce false positives, ensuring more reliable system performance.

2.2 Object detection for metadata extraction

To extract metadata from a video clip and provide a text description of the main events, the system should be capable of detecting a range of object classes in the scene. It should differentiate a human from an animal, and advanced systems should distinguish different types of animals or identify whether a person is carrying a dangerous object like a gun or a knife. Techniques like background subtraction and frame differencing are ineffective and not suitable for this kind of task.

Cob-Parro et al. [7] designed a system capable of detecting, tracking, and counting people movement in a surveillance area. The system utilized the MobileNet-SSD, a popular deep learning model for object detection.

To enhance tracking capabilities, the researchers integrated Kalman filters, a statistical technique used to estimate the state of a dynamic system over time, into the MobileNet-SSD architecture.

The study referred to state-of-the-art methods like ACF (Aggregate Channel Features), PCL-MUNARO, and YOLOv3 (You Only Look Once) and used them for performance comparison. The authors also mentioned HOG (Histogram of Oriented Gradients) and SVM (Support Vector Machine) as popular classical approaches for people detection in video surveillance.

A limitation of their work, with respect to the needs of this project, is that they only considered a single object class (people) without addressing a broader range of object classes. However, the discussion of various existing object detection techniques, broadened my understanding of the object detection landscape and provided options to consider based on project requirements.

2.2.1 Traditional object detection methods based on feature extraction and machine learning classification

Methods like HOG-SVM, PCL-MUNARO, and ACF mentioned in Cob-Parro et al. [7] work all utilizes feature extraction followed by classification using a trained machine learning model.

Feature Extraction:

- **HOG-SVM:** Applies gradient calculation to compute intensity gradients in an image using filters like Sobel to capture edge information. Gradients are grouped into bins based on their angles, emphasizing shape and direction. The image is divided into cells for histogram computation and blocks for gradient normalization, improving robustness to lighting variations.
- **ACF**: Applies downsampling optimization on 10 channel features. These include Gradient Magnitude (captures edge strength), LUV Color Channels (more perceptually uniform than RGB and robust to lighting changes), and Histogram of Oriented Gradients (HOG) (captures the distribution of edge orientations in localized regions).
- **PCL-MUNARO**: Uses color histogram features representing the distribution of colors within an image region.

Model Used for Classification:

• **HOG-SVM:** Extracted features are fed into an SVM classifier. SVM is a supervised learning model that finds the optimal boundary to separate data points into distinct classes, making it effective for binary classification tasks.

• **ACF:** Extracted features are fed into an AdaBoost classifier. AdaBoost is an ensemble learning technique that combines multiple weak classifiers to create a strong classifier, improving overall accuracy.

• **PCL-MUNARO:** Uses the AdaBoost algorithm to create an online appearance classifier. The term "online" signifies that the classifier learns and updates its model of each person's appearance during the tracking process, unlike "offline" learning where models are pre-trained.

2.2.2 Deep Learning based object detection methods

YOLO is a deep neural network-based object detection method employing end-to-end learning. Features are learned hierarchically through convolutional layers.

Lema and Garcia [8] showcased two more popular deep learning models: MobileNet-SSD and EfficientDet. The study conducted a comprehensive evaluation of object detection models on edge devices, considering factors like device hardware, model complexity, framework optimization, and input image size.

Key insights from studies:

- YOLO vs. MobileNet-SSD: YOLO has a good trade-off between accuracy and inference speed, making it suitable for real-time applications. YOLOv5 consistently outperforms SSD-MobileNet v2 in terms of accuracy.
- YOLO vs. EfficientDet: EfficientDet aims to improve accuracy without significantly impacting efficiency. On Google Coral TPU Dev Board, EfficientDet-Lite 1 demonstrates faster inference times and higher Average Precision (AP) than YOLOv5.

While the authors address deploying computationally demanding deep learning models on resource-constrained edge devices, one limitation is that they considered only GPU hardware, neglecting devices with only CPUs. GPU development boards are significantly more expensive than CPU-based ones, potentially negating the cost advantage if used for this project.

2.2.3 Traditional vs deep learning method of object detection

Erabati and Araújo [9] compared traditional and deep learning methods of object detection.

2.2.3.1 Advantages of Traditional Methods

1. **Lower Computational Requirements:** require less computational power for training and inference and as such suitable for devices with limited processing capabilities.
2. **Reduced Data Dependency:** May not require extensive datasets especially for tasks with well-defined features and relatively simple object classes.

2.2.3.2 Limitations of Traditional Methods

1. **Challenges Associated with Detecting a Wide Range of Objects:**

    - Scalability Issues: As the number of object classes or the complexity of the detection task increases, designing and extracting handcrafted features becomes increasingly challenging and often does not scale well. Traditional classifiers like SVMs, although effective for a moderate number of classes, may struggle with scalability when dealing with a large and diverse set of object categories.
    - Cumbersome Feature Engineering: Traditional object detection relies heavily on handcrafted features. As the number of object classes increases, designing effective features for each class becomes increasingly complex and time-consuming.

2. **Limited Accuracy:** Traditional methods generally lag behind deep learning approaches in accuracy, especially in complex detection scenarios or when dealing with a wide variety of object classes.

2.2.3.3 Advantages of Deep Learning Models

- **End-to-End Learning:** Deep learning models facilitate end-to-end learning, where the entire object detection pipeline, from feature extraction to classification and localization, is trained as a single, unified system. This integrated approach optimizes all components of the pipeline simultaneously, leading to better overall performance compared to traditional methods, where feature extraction and classification are often handled as separate, independent stages.
- **Adaptability and Generalization:** Deep learning models can adapt to a wide range of object classes and detection scenarios due to their data-driven nature. By training on large and diverse datasets, deep

neural networks can learn to detect a wide variety of objects, even those not explicitly encountered during training.

2.2.3.4 Limitations of Deep Learning Models

- **Computational Demands:** Training deep learning models requires significant computational power. The complexity of these networks and the vast amounts of data used for training necessitate powerful hardware, making deep learning approaches less accessible for developers with limited resources.
- **Data Dependency:** Deep learning's effectiveness heavily relies on the availability of large, labeled datasets. The quality and quantity of training data directly influence the model's performance. Insufficient or biased training data can lead to poor generalization and inaccurate detections. Gathering and annotating large datasets can be a time-consuming and expensive process, posing a significant challenge for deep learning applications.
- **Training Time:** Deep learning models often require extensive training times, potentially lasting for hours or even days, even with powerful GPUs. This prolonged training process can hinder experimentation and rapid development cycles.

2.2.4 Evaluation – Key Takeaways Applicable to the Project

- **Using Pre-Trained Models:** Pre-trained models can mitigate the limitations of deep learning with respect to training time and computational power requirements. However, inference can still require significant resources, which cost still impact project costs.

- **OpenCV Model Zoo Benchmarks:** The OpenCV [10] Model Zoo offers pre-trained models with benchmark data for inference times across different hardware, including CPUs and GPUs. For instance, yolo_x and Nanodet models show impressive inference times of 1614.13 ms for 640x640 image input and 214.59 ms for 416x416 image input on a Raspberry Pi 4 CPU. This benchmark data supports the viability of deploying a cost-effective solution capable of achieving the proposed design goals, aligning with claim made Erabati and Araújo [9] that deep learning approaches like SSD and YOLO can achieve real-time or near real-time processing speeds.

- **Benefits of Deep Learning for This Project:**
  1. **Ease of Integration:** OpenCV Zoo provides straightforward integration of pre-trained models.
  2. **Better Accuracy:** Deep learning approaches offer higher accuracy compared to traditional methods.
  3. **Wide Range of Object Classes:** Deep learning models can detect multiple object classes, improving metadata quality.
  4. **Reasonable Inference Time:** Proven achievable on CPU-based hardware like the Raspberry Pi 4.
  5. **Superior Performance:** Despite slower training compared to traditional methods, deep learning delivers better inference accuracy and speed [7].

2.3 Secured End-to-End Remote Monitoring

2.3.1 Challenges with Cloud-Based Solutions

A critical aspect of this project is enabling users to remotely monitor events, especially during intrusion alerts. While traditional video surveillance systems (VSS) often depend on cloud computing for this purpose, Deligiannidis [11] highlights concerns regarding privacy and the reliance on third-party providers despite the cloud's popularity due to its simplicity.

A real-world example of these concerns is illustrated by the Federal Trade Commission's (FTC) action against Ring, a home security camera company owned by Amazon, due to significant privacy and security failures. The FTC reported that Ring failed to adequately protect user data, leading to incidents of employee spying and allowing hackers to gain control of user cameras. As a result, Ring was fined $5.8 million in consumer refunds [12].

Lema and García [8] further emphasize the limitations of cloud computing, particularly with regard to latency, connectivity, and privacy, advocating for the advantages of edge computing as an alternative.

2.3.2 Edge Computing Challenges for Remote Monitoring

Implementing remote monitoring using edge computing involves turning the edge device into a video streaming server—a process fraught with its own challenges. One major limitation is the prevalence of private IP addresses due to the scarcity of IPv4 addresses [13]. Most edge devices are behind Network Address Translation (NAT), meaning they share a public IP address managed by their internet service provider (ISP). This makes direct discovery of edge devices by other computers on the internet difficult, complicating remote access.

Under typical NAT scenarios, edge devices can only communicate if they first initiate a request. The NAT gateway—often managed by the ISP—maps the private IP and port of the edge device (e.g., 192.168.1.5:12345) to a shared public IP and a unique port. The ISP then routes the request to the web server, which responds via the public address, ultimately directing the response back to the edge device through the NAT table. This means that without an initial outgoing request, the edge device cannot receive incoming data. For effective remote monitoring, however, the edge device needs to be able to respond to requests from remote clients without having initiated communication [14].

While some ISP-provided routers offer port forwarding options to address this limitation, this solution is not universally available. For instance, devices connected through Carrier-Grade NAT (CGNAT) complicate matters further. In CGNAT scenarios, devices share an internet connection via hotspots, and the use of CGNAT prevents port forwarding.

Another challenge is ensuring the secure transmission of video streams without compromising data privacy. The question is how an edge device can act as a streaming server while maintaining end-to-end encryption for secure remote monitoring.

2.3.3 WebRTC as a Solution

Baretto et al. [15] propose a promising approach to addressing these challenges using WebRTC, an open-source web standard designed for peer-to-peer data transfer. Their mobile surveillance system employs WebRTC to create direct, real-time communication links between devices, utilizing UDP for data transport.

To address NAT-related connectivity issues, the system uses STUN (Session Traversal Utilities for NAT) to help peers discover their public IP addresses and NAT type. If a direct connection cannot be established, TURN (Traversal Using Relays around NAT) servers are used to relay traffic between peers.

Regarding data transmission security, WebRTC ensures that all data is encrypted using Datagram Transport Layer Security (DTLS), which is based on Transport Layer Security (TLS). Mozilla [16] explains that the encryption used in WebRTC is as secure as any HTTPS connection, and since WebRTC is a peer-to-peer protocol, data does not pass through central servers, reducing the risk of interception.

Deligiannidis [11] also suggests Virtual Private Networks (VPNs) as an alternative, noting that they provide a secure method for transmitting video streams remotely. However, VPNs can require compatible hardware and software on both client and router sides, increasing costs and configuration complexity. To further enhance security, Deligiannidis recommends allowing camera owners to retain control over encryption keys, ensuring that edge devices serve only as relays for encrypted streams without the ability to decrypt the data.

2.3.4 Evaluation – Key Takeaways Applicable to the Project

Remote monitoring functionality can be integrated into the project leveraging WebRTC for its low latency, peer-to-peer communication, and strong encryption capabilities, making it well-suited for secure, real-time video streaming.

3 Reference List

[1] S. Brutzer, B. Höferlin, and G. Heidemann, “Evaluation of background subtraction techniques for video surveillance,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1937–1944, 2011, doi: 10.1109/CVPR.2011.5995508.

[2] O. Barnich and M. Van Droogenbroeck, “ViBE: A powerful random technique to estimate the background in video sequences,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 945–948, 2009, doi: 10.1109/ICASSP.2009.4959741.

[3] G. Tang, J. Ni, P. Shi, Y. Li, and J. Zhu, “An improved ViBe-based approach for moving object detection,” *Intell. Robot.*, vol. 2, no. 2, pp. 130–144, 2022, doi: 10.20517/ir.2022.07.

[4] M. V Khadse and Y. D. Pardeshi, “An Effective Object Detection Video Surveillance andAlert System,” *Int. J. Comput. Appl.*, pp. 975–8887, 2016.

[5] OpenCV, “cv::Background Segmentation Namespace Reference.” Accessed: Nov. 20, 2024. [Online]. Available: https://docs.opencv.org/4.x/df/d5d/namespacecv_1_1bgsegm.html

[6] A. B. Godbehere and K. Goldberg, “Algorithms for visual tracking of visitors under variable-lighting conditions for a responsive audio art installation,” *Control. Art Inq. Intersect. Subj. Object.*, pp. 181–204, 2014, doi: 10.1007/978-3-319-03904-6_8.

[7] A. C. Cob-Parro, C. Losada-Gutiérrez, M. Marrón-Romera, A. Gardel-Vicente, and I. Bravo-Muñoz, “Smart video surveillance system based on edge computing,” *Sensors*, vol. 21, no. 9, 2021, doi: 10.3390/s21092958.

[8] D. G. Lema, R. Usamentiaga, and D. F. García, “Quantitative comparison and performance evaluation of deep learning-based object detection models on edge computing devices,” *Integration*, vol. 95, no. May 2023, p. 102127, 2024, doi: 10.1016/j.vlsi.2023.102127.

[9] G. K. Erabati, N. Gonçalves, and H. Araújo, “Object Detection in Traffic Scenarios - A Comparison of Traditional and Deep Learning Approaches,” pp. 225–237, 2020, doi: 10.5121/csit.2020.100918.

[10] OpenCV Zoo, “GitHub - opencv/opencv_zoo: Model Zoo For OpenCV DNN and Benchmarks.” Accessed: Nov. 16, 2024. [Online]. Available: https://github.com/opencv/opencv_zoo

[11] L. Deligiannidis, “Remote Video Surveillance,” *Proc. - 2021 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2021*, pp. 771–776, 2021, doi: 10.1109/CSCI54926.2021.00064.

[12] Federal Trade Commission, “FTC Says Ring Employees Illegally Surveilled Customers, Failed to Stop Hackers from Taking Control of Users’ Cameras | Federal Trade Commission.” Accessed: Nov. 23, 2024. [Online]. Available: https://www.ftc.gov/news-events/news/press-releases/2023/05/ftc-says-ring-employees-illegally-surveilled-customers-failed-stop-hackers-taking-control-users

[13] David Varghese, “Network Address Translation (NAT): The What, Why and How Explained.” Accessed: Nov. 24, 2024. [Online]. Available: https://blog.davidvarghese.net/posts/nat-explained/

[14] AnyConnect, “STUN, TURN, and ICE NAT Traversal Protocols.” Accessed: Nov. 24, 2024. [Online]. Available: https://anyconnect.com/stun-turn-ice/

[15] A. Baretto, N. Pudussery, V. Subramaniam, and A. Siddiqui, “Real-Time WebRTC based Mobile Surveillance System,” *Int. J. Eng. Manag. Res.*, vol. 11, no. 3, pp. 30–35, 2021, doi: 10.31033/ijemr.11.3.4.

[16] Mozilla Developer Network, “Using WebRTC data channels - Web APIs.” Accessed: Nov. 20, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Using_data_channels#security