CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Abandoned Object Detection in Vehicle Cabin Environment

Master's thesis in Computer science and engineering

Hao Hu

Yuhua Cheng

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Abandoned Object Detection in Vehicle Cabin Environment

Hao Hu
Yuhua Cheng

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Abandoned Object Detection in Vehicle Cabin Environment
Hao Hu
Yuhua Cheng

Supervisor: Pedro Petersen Moura Trancoso, Computer Science and Engineering
Advisor: Anshuman Dubey, SmartEye AB
Examiner: Pedro Petersen Moura Trancoso, Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2024

Abandoned Object Detection in Vehicle Cabin Environment

Hao Hu, Yuhua Cheng
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Abandoned object detection in the vehicle cabin environment is pivotal for the convenience of passengers. Existing methods for abandoned object detection are either aimed at outdoor surveillance or rely too much on training data. This study aims to develop an efficient algorithm for detecting abandoned objects in a vehicle cabin environment that relies less on training data and generalizes to untrained objects. The main approach involves utilizing a simple background model to extract candidate abandoned objects, followed by feature extraction using MobileNetV2 pre-trained on ImageNet-k for accurate detection. By comparing features before and after human presence, false positive proposals are effectively filtered out. To make the proposed method suitable for embedded execution, optimizations are performed to improve efficiency. We evaluate the proposed method on common left-behind objects videos with a Jetson Nano device. The results prove the efficacy and efficiency of the proposed method. However, the study is still limited by the lack of research on dynamic cameras, low-contrast infrared images, and persistent shadows.

Keywords: Abandoned object detection, Vehicle cabin monitoring, Background model, Feature extraction, Computer Vision, Embedded ML.

Acknowledgements

We want to thank Smart Eye for giving us the opportunity to do this thesis in their company by supporting us with all the data and devices and for all the guidance they have given us (including the tasty fruit and Fika). We would greatly like to thank our company supervisor Anshuman Dubey for all the discussion sessions and daily meetings. We would also like to thank our Chalmers supervisor and examiner Pedro Petersen Moura Trancoso for his patient guidance and advice.

Hao Hu, Yuhua Cheng, Gothenburg, 2024-06-26

Contents

List of Figures

List of Tables

1

Introduction

For decades, taxis have been crucial for urban transportation. Still, the rise of ride-sharing platforms like Uber has introduced new challenges such as passengers frequently leaving items behind [1]. Taxi drivers may not always check seats promptly to inform passengers of abandoned items. What's more, automated features and the emergence of driverless cars diminish this human element. As we move towards futuristic transport modes, finding efficient lost and found solutions becomes increasingly crucial.

Existing methods for abandoned object detection primarily concentrate on detecting abandoned objects in public settings such as train stations and intersections. Nevertheless, current research lacks extensive studies specifically focused on abandoned object detection within vehicle cabin environments. While some researches exist in this area, they typically rely heavily on supervised learning and extensive datasets. What's more, the trained model tends to perform badly for unseen objects and cannot recognize untrained categories. Hence, it is imperative to devise a methodology that is less learning-based but rather rule-based for detecting abandoned objects within vehicle cabins.

Smart Eye aims to solve this issue with its focus on camera-based solutions to monitor drivers, enhancing safety in mobility [2]. Recognizing the growing importance of improving both its products and passenger convenience, Smart Eye is developing a solution capable of detecting abandoned objects in vehicle cabins without requiring extra training data. With Smart Eye, our research concentrates on this functionality, leveraging advanced algorithms specifically designed for abandoned object detection in vehicle cabin environments.

1.1 Problem Statement

In addition to the contextual aspects of the study, there exists a requirement for new methodologies in the field of abandoned object detection in vehicle cabin setting research. The majority of existing abandoned object detection algorithms rely on either foreground information derived from background models or large amounts of training data for object detection. Short-term and long-term background models are considered one of the effective ways to extract abandoned objects. One problem is that other objects, like slightly moved safety belts, can also be mistaken for abandoned items, causing false positives. The false positive areas are highlighted as

new foregrounds, leading to a significant increase in false alarms and poor surveillance system performance. Although algorithms relying on traditional computer vision techniques are limited in accuracy or performance, the latest advancements in artificial intelligence have paved practical avenues for enhancing various applications such as object detection, segmentation, and tracking. However, machine learning, particularly deep learning-based methods, typically requires extensive resources for collecting and annotating training data. The deep learning models tend to behave better than traditional methods on objects that are trained on. But for objects that are not trained on, the model is more likely to perform worse leading to a low recall rate. Hence, the need for a system that solves the above-mentioned issues that is capable of swiftly and precisely identifying left behind objects in the cabin becomes imperative.

1.2 Purpose

This study aims to develop an accurate and efficient algorithm for detecting abandoned objects in vehicle cabin environments that rely less on training data.

Three key research questions have been defined to accomplish the goal of this study.

RQ1: *How can we detect abandoned objects of all categories in the vehicle cabin?*

In cars, a diverse array of items is frequently forgotten, including mobile phones, wallets, keys, and headphones, as observed in [1]. Detecting all these items poses a significant challenge due to their diverse nature, spanning across various categories. Due to the impracticality of acquiring an extensive dataset encompassing all potential abandoned items, utilizing object detection methods such as YOLOv3 in [3] becomes highly improbable. Meanwhile, existing traditional methods like [4], are mostly developed for outdoor surveillance, not for vehicle cabins. How to adapt the previous research for our vehicle setting and detect a wide range of categories of objects proposes a challenge.

RQ2: *How can we remove false positives detected for abandoned objects?*

False positives are likely to occur due to the fluctuating presence of objects within cars, such as safety belts and seat cover adjustments resulting from human activity. How to differentiate between true positive and false positive abandoned object cases in vehicle cabin settings is important for the success of the solution.

RQ3: *How can we optimize the designed algorithm for abandoned object detection so that it can execute efficiently on embedded devices with limited computation?* The solution for abandoned object detection in the vehicle cabin environment, like all other services provided by Smart Eye, needs to run on the embedded device in the vehicle. Besides, the algorithm should respond to left-behind objects in a short time. This makes it necessary to optimize our solution for hardware with limited hardware resources like computation power and memory, especially when you have multiple models running at the same time.

1.3 Scope

In our research, our primary focus was on abandoned objects captured by fixed infrared image cameras. We didn't deal with situations where the camera moved while detecting objects. We also did not consider live subjects such as pets or infants. Objects already in the car that had been taken away were also excluded from our study. These allowed us to streamline our research focus and avoid duplicating existing studies in this area. Excluding these factors also helped us simplify the case and have a more targeted analysis.

1.4 Ethical Consideration

Implementing abandoned object detection in vehicle cabins raises several ethical concerns. Firstly, the privacy of individuals recorded by cabin cameras is a major issue. Ensuring all individuals are aware of and consent to the surveillance is crucial. Secondly, the security of the collected data is critical, as video recordings and related data must be stored securely to prevent unauthorized access and data breaches. Additionally, this technology could alter passenger behavior and cause discomfort, necessitating a careful balance to avoid unnecessary alerts. Finally, there is a risk of misuse for unauthorized surveillance, thus strict guidelines and ethical usage policies are needed.

1.5 Thesis outline

The introduction sets the stage by outlining the problem statement, purpose, scope, and thesis outline, establishing the context for the research endeavor. Following this, the background section delves into existing literature, examining non-neural network and neural network-based methods. The theory section includes introductions to infrared imaging, Gaussian mixture models, and various image processing techniques. The method section provides a detailed overview of the proposed approach, covering aspects such as background modeling, filtering out false positives, and contiguous frame tracking. Subsequently, the results section presents how we collect the video data, optimization of the proposed method targeting Jetson Nano, and the detection results on test videos obtained through the implementation of the proposed method. Then, the discussion section offers a thorough analysis and interpretation of these results and outlines avenues for limitations and future research. Finally, the conclusion encapsulates the key insights derived from the study, highlighting its contributions to the field.

2

Related Work

This chapter provides an overview of the methodologies employed in literature on abandoned object detection. Section 2.1 summarises the literature on non-neural network based methods, while Section 2.2 gives an outline of neural network based methods.

2.1 Non-Neural Network Based Method

This section will discuss conventional computer vision techniques for moving object detection, focusing on their computational complexity and adaptability. We will explore methods like frame difference [5], optical flow [6], statistical learning[7], and introduce advanced approaches like camouflage modeling [8], dual Gaussian mixture models with the SVM-based classification methods [9] and dual background models [4].

The frame difference method [5] is straightforward and easy to implement. However, its accuracy is compromised due to background brightness changes, which can lead to misjudgments.

Optical flow algorithms [6] aim to compute the velocity field of moving objects in a sequence of images. It is complex due to its high computational demands and the substantial amount of data required for effective training and operation.

Statistical learning [7] uses algorithms to identify patterns and relationships in large datasets for predictive modeling and decision-making. Like optical flow, statistical learning is also one of the most complex algorithms among conventional computer vision techniques, requiring more time than other methods. Additionally, the statistical learning method demands numerous training samples and has high computational complexity.

Zhan *et al.* [10] proposed a method for moving object detection based on frame difference and edge detection which keeps a small computation budget while overcoming the sensitivity to light with the frame difference method. The proposed method achieved the highest recognition rate and time efficiency among the data they gathered compared to all other conventional methods investigated. However, this method based on frame difference and edge detection can not adapt to slow, gradual changes in the background as it only compares differences between consecutive frames.

The issues of slow, gradual changes were solved in background models In a related study, Zhang *et al.* [8]. This research introduced a method called camouflage modeling (CM) to detect camouflaged foreground pixels by combining global and local models, using Bayesian principles, initially for visible light images but potentially useful for low-contrast infrared imagery. However, this approach can be computationally complex due to maintaining and updating multiple models and performing Bayesian inference, and it may struggle with rapid scene changes or highly dynamic environments.

Based on camouflage modeling, Wahyono *et al.* [9] introduced a method based on dual Gaussian mixture models for cumulative dual foreground difference in stationary object detection. They integrated an SVM-based classifier to verify region candidates and classify them as vehicles, humans, or other objects. However, this method has limitations when handling full illumination changes such as from cloudy to sunny conditions. Thus more powerful background modeling should be considered. The long short-term background model has the same performance as the dual Gaussian mixture models with the SVM-based classification method and was less computationally expensive.

In their work, Lin *et al.* [4] introduced the current state-of-the-art background model-based method for detecting abandoned objects using temporal consistency modeling and back-tracking verification. This method employs a Gaussian Mixture Model (GMM) for background subtraction. This method maintained two models, one for long-term and one for short-term, to track foreground objects. By applying a state machine to these models, it identified static object candidates. A back-tracing algorithm was then employed to confirm whether these static object candidates were indeed abandoned by someone. However, it could not deal with moving objects at a very slow speed.

These studies relying on multiple background models are unable to effectively address the issue of false positives in vehicle cabin settings, as they cannot differentiate between objects that have been relocated due to passengers' dynamic interaction with the background, shadows because of light changes, or abandoned objects. Besides, these background modeling-based methods cannot deal with the varying time range of passengers in the car.

2.2 Neural Network Based Method

Neural network-based methods for abandoned object detection can be classified into classification, detection, and segmentation categories while detection-based methods account for most of them.

For classification-based methods, Contractor *et al.* [11] used InceptionV3 [12] to distinguish between scenes with and without abandoned luggage. Although the method achieved high accuracy on the dataset, it cannot give out the abandoned object location information.

For segmentation-based methods, according to Park *et al.* [13], they employed a

hybrid approach combining a dual background model with Mask R-CNN. Initially, candidate stationary objects are extracted from the dual background model. Subsequently, the determination of these candidate stationary objects is refined by considering various scenarios using Mask R-CNN.

For detection-based methods, Smeureanu *et al.* [14] proposed a two-stage model composed of static object detection and abandoned object recognition. It combined the traditional method with the CNN method which yields better performance than a strong CNN baseline method. However, these approaches are susceptible to overfitting issues since they train sub-areas of the experimental video background as negative classes. This would result in challenges of generalizing to new environments without extensive training data. Lwin and Tun [15] introduced a method utilizing YOLOv4 for identifying abandoned objects in video surveillance. Their study devised a framework that enhanced YOLOv4's capabilities specifically for detecting abandoned items. They employed a self-assembled dataset for training purposes, where the neural network was trained to recognize six specific objects: people, backpacks, handbags, books, hats, and backpacks (a duplicate was mentioned). Conversely, we wanted to avoid the need for extensive labeled datasets. Thus this leveraged difficulties in gathering data. In their paper referenced as [16], Shyam *et al.* proposed a dual background model employing the SILTP-ViBe modeling method [17], in conjunction with PFSM to track foreground pixel states. They also utilized the Single Shot Multibox Detector (SSD) [18] to identify stationary objects as potential candidates. However, this method automatically labeled all objects within the candidate area as abandoned, disregarding the possibility of foreground regions formed by disappearing objects. As a result, it may erroneously classify stolen items as abandoned. According to their work, Liu *et al.* [19] introduced an abandoned object detection approach that did not rely on background modeling. This method combined a pedestrian detector trained with the YOLO deep learning model and a frame difference technique for scene analysis. By focusing solely on frames captured before a pedestrian entered and after they exited the scene, this method remained resilient to lighting variations, thanks to its exclusion of background modeling. Perrett *et al.* [13] proposed a Faster-RCNN based [20] left-behind object detection method in the vehicle cabin environment. It can effectively detect abandoned objects under a single Near Infra-Red camera mounted in the ceiling. However, it failed to detect objects on a small scale or have a similar color to the background seat. Notably, this is the first research paper we have found using the object detection method targeting the vehicle cabin environment.

These neural network-based research studies share valuable insights relevant to our goal of detecting abandoned objects in vehicle cabins using infrared imaging. All the neural network-based methods suffer from the same problem: relying on training data and generalizing badly to untrained objects. However, due to limitations in our dataset for training purposes, we are compelled to explore alternative methods that require less extensive training. It is also worth mentioning that only one of the studies reviewed utilized infrared images, which makes our research with infrared images as input more valuable.

3

Theory

This chapter explains the theories and terminologies used in our thesis, providing a foundation for understanding the methods and analyses presented in subsequent chapters.

3.1 Infrared Imaging

Infrared imaging is the foundation of our research as it allows us to capture images based on infrared radiation rather than visible light, thus ensuring well-function during low light conditions. An infrared image recorded by an infrared camera will appear as a grayscale image where the intensity of each pixel corresponds to the infrared radiation of that pixel. The higher the infrared radiation the higher the the pixel intensity. Since infrared images depend on infrared radiation rather than light, they provide useful information about the properties of the objects and are widely used in several applications such as night vision. Several known issues with infrared images are low contrast and poor texture information [21]. For example, objects that have similar infrared radiation parameters will be recorded as similar colors even if their appearance is completely different color under visible light.

3.2 Image Processing

In this section, we will discuss some image processing techniques used in our method.

3.2.1 Gaussian Blur

To address the issue of noise and detail in images, we use Gaussian blur, a popular image processing technique. Gaussian blur is an image processing technique used to reduce noise and detail in an image while preserving its overall structure. The "Gaussian Blur" technique involves convolving an image with a kernel defined by a Gaussian function. The Gaussian function is defined as [22]:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

3.2.2 Morphological Image Processing

Morphological image processing techniques, such as erosion and dilation, are pivotal for the preprocessing tasks for our method due to their ability to reduce noises within mask images. Erosion and dilation form the fundamental operations in morphological image processing. Erosion reduces the size of foreground structures, while dilation increases their size [23]. They are often used together to reduce noise in the mask images where noisy small white pixels are removed after morphological processing.

3.3 Background Modeling Methods

In this section, we will discuss theories related two background modeling-based methods we explored.

3.3.1 Gaussian Mixture Model

The Gaussian Mixture Model (GMM) was selected for detecting moving objects. A Gaussian Mixture Model is a commonly employed method for detecting moving objects within video sequences captured by stationary cameras. Gaussian distribution, also called the Normal distribution, is a continuous probability distribution, given by

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma|^{1/2}}exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\} \tag{3.1}$$

where μ is a D-dimensional mean vector, Σ is a D $\times$ D covariance matrix, which describes the shape of the Gaussian and $|\Sigma|$ denotes the determinant of Σ.

The Gaussian distribution displays symmetry around its mean and is defined by both the mean and the standard deviation. However, the unimodal nature of a single Gaussian distribution fails to adequately represent the multiple-density regions often encountered in practical multimodal datasets. As a result, it cannot effectively capture the heterogeneous resource usage behaviors observed in cloud workloads. To address this limitation, more complex multimodal distributions can be accurately modeled using a mixture of Gaussian distributions.

A Gaussian Mixture Model (GMM) is an unsupervised clustering technique that generates ellipsoidal-shaped clusters by estimating probability densities through the Expectation-Maximization algorithm. Each cluster is represented as a Gaussian distribution. Unlike K-Means, which consider only the mean, GMMs utilize both the mean and the covariance. This enables GMMs to offer a more precise quantitative assessment of fitness for a given number of clusters.

A Gaussian Mixture Model (GMM) is formulated as a linear combination of basic Gaussian probability distributions, and is expressed as

$$p(x) = \Sigma_{k=1}^{K}\pi_k\mathcal{N}(x|\mu_k,\Sigma_k) \tag{3.2}$$

where K denotes the number of components, while π_k is referred to as the mixing coefficient, providing an estimation of the density associated with each Gaussian

component. The Gaussian density given by $N(x|\mu_k, \Sigma_k)$, is called a component of the mixture model. Each component k is delineated by a Gaussian distribution with mean μ_k, covariance Σ_k and the mixing coefficient π_k.

3.3.2 K Nearest Neighbour(KNN)

KNN was chosen as another method for detecting moving objects within stationary camera footage. A general version of the nearest neighbor technique bases the classification of an unknown sample on the "votes" of K of its nearest neighbor rather than on only it's on a single nearest neighbor. If the costs of error are represented in the collection of its K nearest neighbors [24].

Let $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ be the k nearest neighbors of $\mathbf{x}$. Then, for classification, the predicted output $\hat{y}$ is often determined by a majority vote among the labels of the k nearest neighbors:

$$\hat{y} = \text{argmax}_y \sum_{i=1}^{k} I(y_i = y)$$

where $I(\cdot)$ is the indicator function that indicates the membership of an element in the set. In regression tasks, $\hat{y}$ is commonly calculated as the average of the output values of the k nearest neighbors:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i$$

3.4 Image Similarity Measurement

A neural network is used to extract the feature vector which is then used to calculate similarity between image pairs in our method. Similarity is then used to filter out false positive cases caused by background movement. We will discuss theories and techniques related to it in this section.

3.4.1 ImageNet-1k

ImageNet-1k has been utilized for pre-trained MobileNetV2 in this study. ImageNet-1k is a benchmark dataset widely used in computer vision research and machine learning. It consists of 1.46 million images across 1,000 categories, serving as a comprehensive resource for training and evaluating image classification models [25]. Developed by the ImageNet project, this dataset has played a pivotal role in advancing the field of deep learning, particularly through competitions like the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [26].

3.4.2 MobileNetV2

In our study, MobileNetV2 served as an effective feature extractor. MobileNetV2 [27] is a pre-trained on ImageNet-1k (also referred to as ILSVRC 2012, a collection of 1.3 million images and 1,000 classes) as a feature extractor. MobileNetV2 is a type of architecture in which the basic building block is a bottleneck depth-separable convolution with residuals [27]. It features two types of blocks, depicted in Fig. 1. Each block consists of three layers. The first layer involves 1x1 convolutions with "ReLU6". The second layer employs depth-wise convolution, and the third layer applies 1x1 convolution without non-linearity. The initial layer acts as a one-stride residual block. Table 1 indicates that the second layer also functions as a residual block with a stride of 2, used for compression.

The approach to object detection involves classification to identify the input class and regression to adjust the bounding box. Most backbone networks for detection, except for the final fully connected layers, are primarily designed for classification tasks. The backbone network acts as a straightforward feature extractor for object detection, taking input images and generating feature maps for each [28].

3.4.3 Cosine Similarity

Cosine similarity was utilized to compute the similarity between potential abandoned objects based on features extracted by MobileNetV2. Cosine similarity(CS) was used to calculate the similarity score between two proposed images. Cosine similarity (CS) between two vectors x and y is defined as:

$$Cosine(x, y) = \frac{x \cdot y}{|x||y|} \tag{3.3}$$

The cosine similarity metric possesses a unique property that renders it well-suited for metric learning tasks: its resultant similarity measure always falls within the range of -1 and +1. This characteristic, as illustrated in Equation 3.3, facilitates the design of a simple yet effective objective function for various learning tasks.

3.5 Libraries

Libraries are integral components in computational algorithms. In the context of our algorithm for abandoned object detection, these libraries are introduced to underscore their critical roles in optimizing and implementing key computational tasks.

3.5.1 NCNN

NCNN [29] is a high-performance neural network inference computing framework meticulously optimized for mobile platforms. From its inception, NCNN has been meticulously engineered for deployment and usage on mobile phones. Notably, NCNN stands out for its lack of third-party dependencies, ensuring seamless integration and efficient performance. Moreover, NCNN boasts cross-platform compatibility

and outperforms all known open-source frameworks when executed on mobile phone CPUs.

Developers benefit from NCNN's efficiency by effortlessly deploying deep learning algorithm models to mobile platforms, thereby empowering the creation of intelligent applications that bring artificial intelligence to users' fingertips. NCNN's robust implementation has garnered widespread adoption, with numerous Tencent applications, including QQ, Qzone, WeChat, Pitu, leveraging its capabilities to enhance user experiences and drive innovation in the mobile landscape.

In this project, we use ncnn as a neural network inference engine, ncnn can run both on CPU and GPU devices.

3.5.2 OpenCV

OpenCV [30] is an open-source computer vision library. It provides libraries and tools in the fields of computer vision, image processing, and artificial intelligence and is widely used by the computer vision industry.

With its extensive collection of pre-built functions and modules, it simplifies the development of applications that require computer vision capabilities.

In this project, we use OpenCV for project setting up, image processing, visualization, and deep learning inference.

3.5.3 TensorRT

TensorRT [31] is a high-performance deep learning inference library developed by NVIDIA. It is designed to optimize and deploy trained neural networks for inference on various NVIDIA GPU architectures, delivering low-latency and high-throughput performance.

With TensorRT, developers can efficiently deploy deep learning models in production environments, including applications such as image classification, object detection, natural language processing, and speech recognition. By leveraging the power of NVIDIA GPUs, TensorRT accelerates inference tasks by optimizing the computation graph, reducing memory footprint, and exploiting parallelism inherent in GPU architectures.

3.5.4 CUDA

CUDA [32] (Compute Unified Device Architecture) is a parallel computing platform and programming model developed by NVIDIA. It enables developers to harness the computational power of NVIDIA GPUs to accelerate a wide range of compute-intensive tasks.

At its core, CUDA provides a scalable and efficient approach to parallel computing by offloading compute-intensive portions of an application to the GPU. This allows developers to leverage the massively parallel architecture of GPUs, which consist of thousands of cores capable of executing multiple threads simultaneously.

CUDA programming involves writing parallel code in the CUDA C/C++ programming language, which extends the standard C/C++ language with GPU-specific features. Developers can write CUDA kernels, which are specialized functions that execute in parallel on the GPU. These kernels are then launched from the CPU and run concurrently across multiple threads on the GPU.

3.5.5 Vulkan

Vulkan [33] is a low-overhead, cross-platform graphics and compute API developed by the Khronos Group. It provides developers with fine-grained control over graphics and computing tasks, allowing for high-performance rendering and computation on modern GPUs (Graphics Processing Units).

Introduced as the successor to OpenGL, Vulkan was designed from the ground up to address the limitations of traditional graphics APIs and to take full advantage of modern GPU architectures. Its key design principles include efficiency, scalability, and flexibility, making it suitable for a wide range of applications, from gaming and virtual reality to scientific visualization and machine learning.

3.6 Hardware

Compared to the cloud, neural network inference on the edge provides privacy, low bandwidth requirements, and real-time processing advantages. In this section, we will introduce Jetson Nano - a common low-end edge device equipped with a GPU for neural network inference.

The Jetson Nano is a compact, powerful computing platform developed by NVIDIA, designed to bring the power of artificial intelligence to the edge. Targeted at developers, makers, and students, the Jetson Nano enables the creation of high-performance, low-power AI systems. It features a 128-core Maxwell GPU, a quad-core ARM Cortex-A57 CPU, and 4GB of LPDDR4 memory, making it well-suited for tasks such as image recognition, object detection, and other machine learning applications. With extensive support for popular high-performance libraries optimized for GPU such as CUDA, cuDNN and TensorRT, the Jetson Nano provides a robust foundation for innovative projects in robotics, IoT, and embedded systems.

4

Methods

The main goal of this study is to design a system that can detect abandoned objects in vehicle cabins and give out an alarm signal to passengers timely. In this chapter, we will discuss existing state-of-the-art methods for abandoned object detection and explain our proposed method step by step.

4.1 Input and Output

The input to the abandoned object detection system is infrared video frames recorded with the vehicle cabin settings. The motivation behind using infrared cameras while not visible light cameras is the company Smart Eye wants to detect objects reliably during the daytime and nighttime. All the existing solutions for cabin monitoring are based on infrared image input as well.
The expected output should be an alarm signal if there is an abandoned object from passengers in the vehicle cabin.

4.2 Long-Term and Short-Term Integration Background Modeling

After gathering the necessary data, we adopted the current state-of-the-art classical method using temporal consistency modeling and back-tracing verification developed by Kevin et al. in[4] for abandoned object detection. This approach as shown in Algorithm 1 employed both short- and long-term background models to isolate foreground objects from input images. Each pixel undergoes classification using a 2-bit code, followed by a framework that discerns static foreground regions by scrutinizing the temporal evolution of code patterns. Foreground objects were then detected as abandoned objects.

Initially, we outline the long- and short-term models constructed within our methodology for static foreground detection. The algorithm we proposed as Algorithm 1 commenced with a generic background modeling technique implemented at two distinct learning rates. We selected Gaussian Mixture Model (GMM) and K-Nearest Neighbors (KNN) as our background modeling method.

To implement the long- and short- term based background model, we tried to adapt GMM. Our approach involved iteratively processing each pixel within the input video

Algorithm 1 Background Modeling Algorithm

```
1: procedure BackgroundModeling(Image I_t, Background Model B)
2:     for (x, y) in I_t do
3:         if I_t(x, y) ∈ B(x, y) then
4:             Classify (x, y) as background pixel
5:         else
6:             Classify (x, y) as foreground pixel
7:         end if
8:     end for
9:     for Newly identified background pixel (x, y) do
10:        Update B(x, y) using I_t(x, y)
11:    end for
12: end procedure
```

frames and updating the background model parameters using GMM. Parameters such as mean, covariance, and weight, were continuously refined as new frames were processed. We also used Gaussian Blur to reduce noise in our data, which is a technique that helps to smooth out sudden changes and diminish high-frequency components related to noise.

Additionally, we integrated dynamic adjustments into the learning rate parameter. This parameter controlled the speed at which the background model adjusted to scene changes. A higher learning rate facilitated swift adaptation to variations in the scene, while a lower rate ensured a more gradual update, potentially improving the stability of background segmentation over time. Consequently, we utilized a higher learning rate of 0.01 for the short-term model and a higher learning rate of 0.001 for the long-term model.

After exploring the Gaussian Mixture Model (GMM) for the background model, we also experimented with the K-Nearest Neighbors (KNN) approach as an alternative method. We aimed to evaluate the performance of different methods in detecting abandoned objects to determine the most effective approach.

As a result, the KNN-based background model was selected as it had a better performance for precision as shown in Figure 4.1. By integrating the KNN-based background model, our approach adeptly discerned between static background elements and moving foreground objects during the Background Segmentation phase.

In our system, an abandoned object is defined as an item that the long-term model consistently identifies as a foreground object over time. The short-term model, with its faster updating rate, classifies the abandoned luggage as a background object. We define objects within bounding boxes detected solely by the short-term model, but not by the long-term model, as abandoned objects. Figure 4.2 illustrates an example of our detection results.

We faced several challenges when implementing this method and struggled to find effective solutions for them:

(a) Frame Example (b) Short-term Model of GMM (c) Long-term Model of GMM

(d) Frame Example (e) Short-term Model of KNN (f) Long-term Model of KNN

Figure 4.1: Performance of GMM and KNN

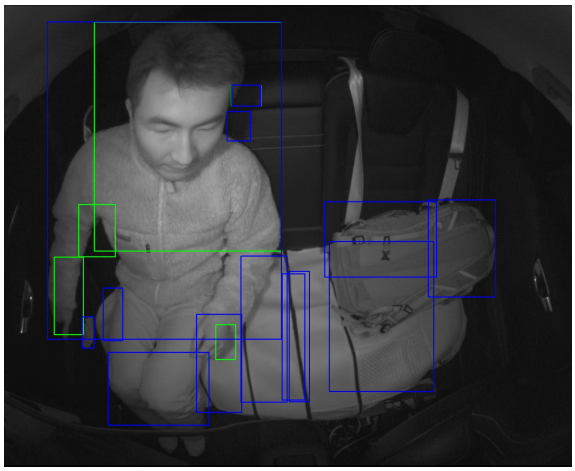


Figure 4.2: Detection Result

1. Accurately defining human remains a challenge, leading to instances where stationary body parts, such as legs and certain sections of the head, were mistakenly classified as abandoned objects. Thus an algorithm that can detect humans accurately is imperative.
2. The inability to filter out slightly moved objects posed another hurdle. For instance, in Figure 4.2, the detection of seams and seat belts as abandoned objects highlights this issue. Finding methods to effectively eliminate false positives like these is essential.
3. The use of learning rates presented challenges due to varying passenger durations. Since a passenger may stay in a car for different time ranges, like 10 minutes, 1 hour, or 2 hours, it is impossible to adjust the learning rate to make the abandoned object stay in the long-term model and not in the short-term model in every time range. Thus we needed to find a way to avoid needing to adjust to time-range uncertainty.

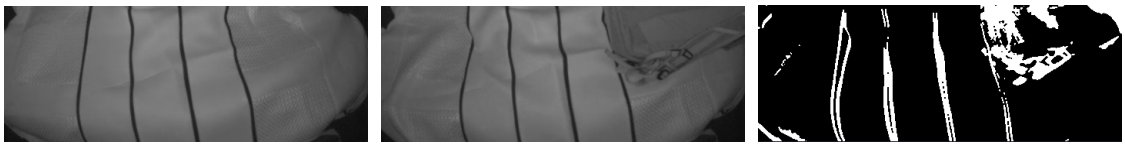
To address these challenges, we require a method capable of accurately detecting humans, filtering out lightly moved objects already present in the car, and avoiding the need for adjustments to time-range uncertainty. These issues cannot be effectively resolved using traditional background models. Therefore, we opted to develop our proposed method.

4.3 Proposed Method

To precisely detect humans, remove false positives, and circumvent the necessity for adjustments to time-range uncertainty, we came up with the following ideas:

1. First, we decided to utilize the human detection algorithm developed by Smart Eye. It is a highly effective algorithm for detecting human presence and counting the number of individuals within a scene. Upon this, we capture segments of the test video before a human enters and after he/she leaves. Then we we can apply a simplified background model on the two segments of video to differentiate between background and foreground. Based on this we were able to detect humans and avoid uncertain time ranges that needed to be adjusted.
2. After implementing the human detection algorithm and a simplified human detection method, we encountered the persistent issue of false positives, such as misidentifying seat covers and seat belts, as depicted in Figure 4.3. To tackle this challenge, we utilized a pre-trained neural network as a feature extractor to effectively filter out these false positives.

Based on these ideas, our proposed methodology involved the following steps as shown in Figure 4.4. Firstly, we leveraged a method provided by Smart Eye to ascertain the presence of humans within the video stream. This allowed us to segment the input video into two distinct phases: pre-human entry and post-human departure. Subsequently, we employed a simple background model for detecting candidate objects through foreground analysis. In the background model, information was acquired regarding the scene prior to human entry and subsequent to human



(a) Before entering (b) After leaving (c) True and false positives

Figure 4.3: False positives introduced by interaction with background

departure. Candidate proposals for abandoned objects were then generated by comparing the disparities between the scenes before human entry and after human departure. To be able to compare the characteristics of candidate objects before and after human interaction, we utilized a pre-trained MobileNetV2 model. This ensured that moved objects were not erroneously identified as abandoned items, thereby minimizing false positives. If the candidate proposals were identified as false positives, they were disregarded, and the detection process continued with subsequent frames. Alternatively, if a candidate proposal was deemed valid, an alarm was triggered to indicate that an object had been abandoned.

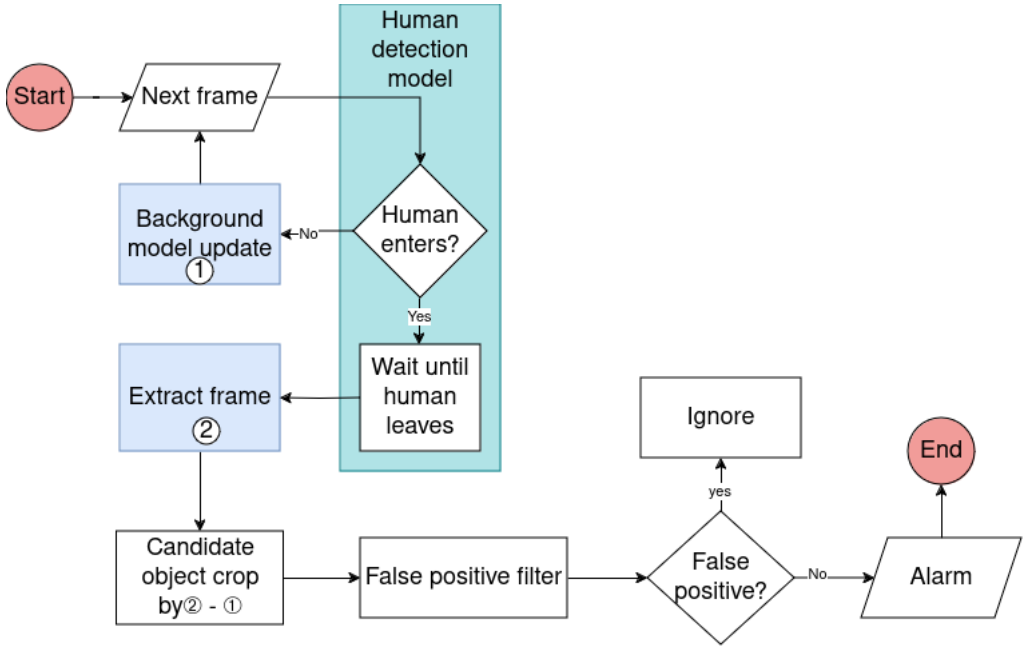
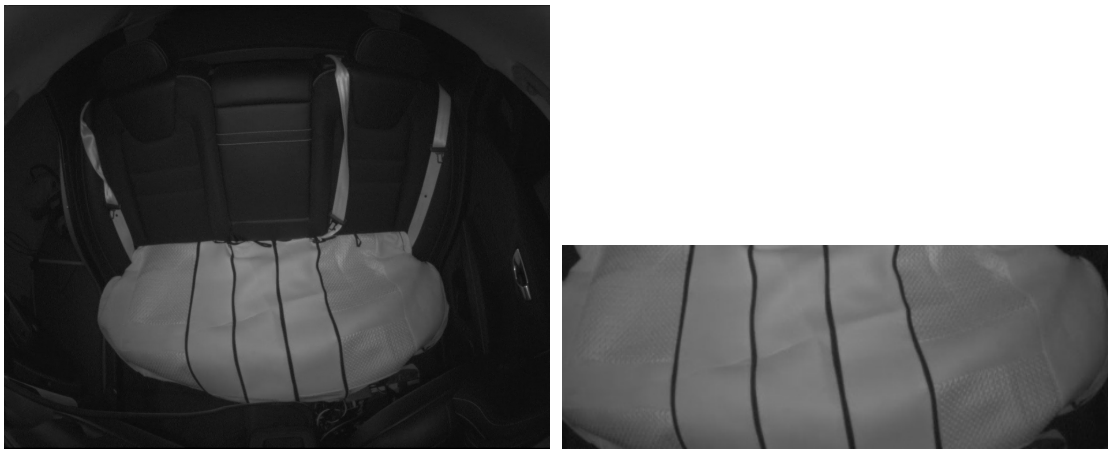


Figure 4.4: Flowchart of our algorithm

4.3.1 Preprocessing

Smart Eye possessed a highly effective algorithm for detecting human presence and counting the number of individuals within a car. Upon this module, we developed a method to partition the given video into two segments: before a human enters and after a human leaves. These segments were then merged to create a whole entity. Additionally, we selected the specific area in the backseat as the Region of Interest (ROI) for detecting abandoned objects, assuming a fixed camera position. This left out unrelated areas to the algorithm and also reduced the computation complexity.



(a) Whole image captured by camera (b) Selected region of interest

Figure 4.5: Whole image vs Selected ROI

(a) Foreground mask from background model (b) Foreground mask after dilation

Figure 4.6: Dilation morphological processing

4.3.2 Background Model

After the necessary preprocessing steps, a simplified background model was used to extract candidate abandoned objects. First, both the input and background frames were converted to floating-point representation to enable subsequent calculations. The background model first learned from the frames before the person entered as background. After the learning period, for each new incoming input, the algorithm compared it with the background model. If the difference of the pixel was larger than a certain value, it was treated as foreground; otherwise, it was considered background. Meanwhile, for the areas treated as background, the algorithm updated the background model to adjust to changes in the scene, such as slowly changing light. The result was a preliminary foreground mask computation, as shown in Figure 4.6a. It outlined potential areas of interest like moving objects or scene changes.

4.3.3 Noise Removal

From the foreground mask image, we captured the candidate abandoned objects. However, numerous small, disconnected regions and noise were present. To enhance the image quality, morphological dilation techniques were utilized to merge these small regions, resulting in fewer noisy proposals. The dilated foreground mask image is depicted in Figure 4.6b.

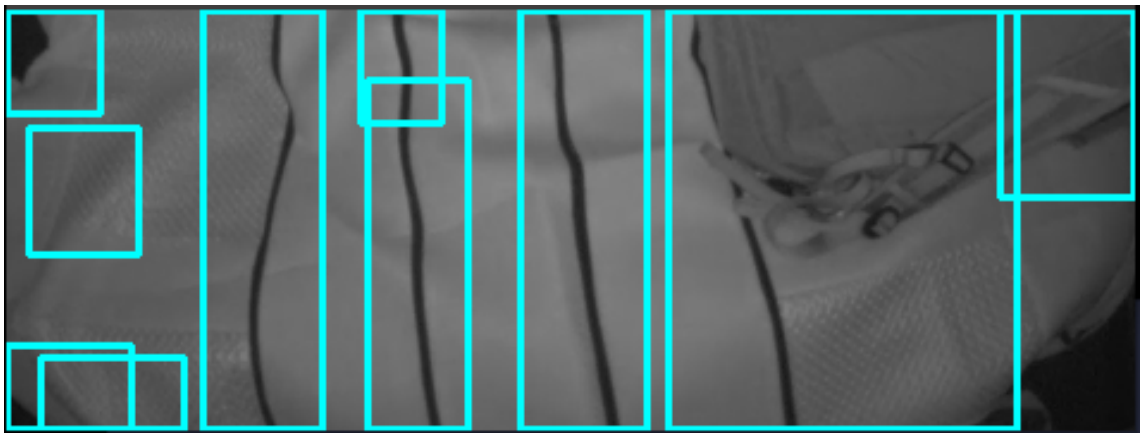


Figure 4.7: Candidates generated from foreground mask

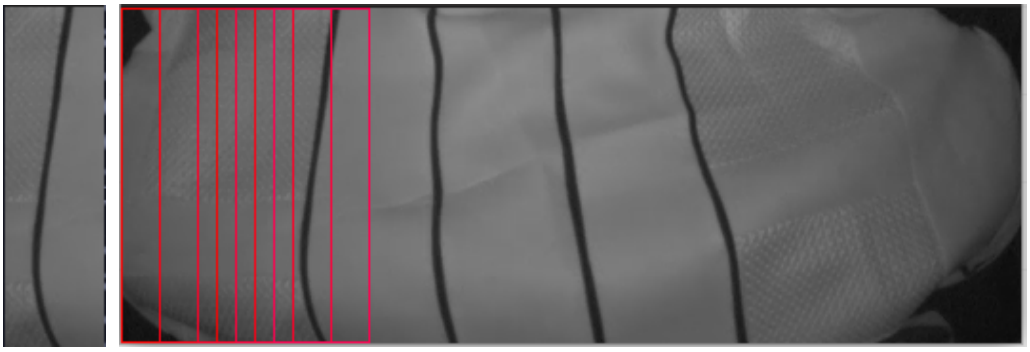
4.3.4 Filter Out False Positive

From the processed foreground mask, obtained after removing noise, candidate proposal boxes were extracted. These boxes represented areas with the potential of containing left-behind objects, identified from the connected contour areas in the mask, as illustrated in Figure 4.7.

Although morphological dilation effectively reduces the number of proposals and assists in noise reduction, the problem of false positives remained unsolved. They arose when pre-existing objects were displaced after human departure rather than indicating newly abandoned items. Furthermore, background movements, such as a shifting seat cover, pose difficulties for the current algorithm in accurately detecting false positives. To address these issues, a novel approach has been developed, leveraging enhanced object features from algorithms including background subtraction, foreground subtraction, and noise removal. This algorithm extracted candidate object proposals from the pre-processed mask based on identified contours and bounding boxes. Subsequently, false positive proposals were filtered out by comparing them with the background region using a sliding window approach.

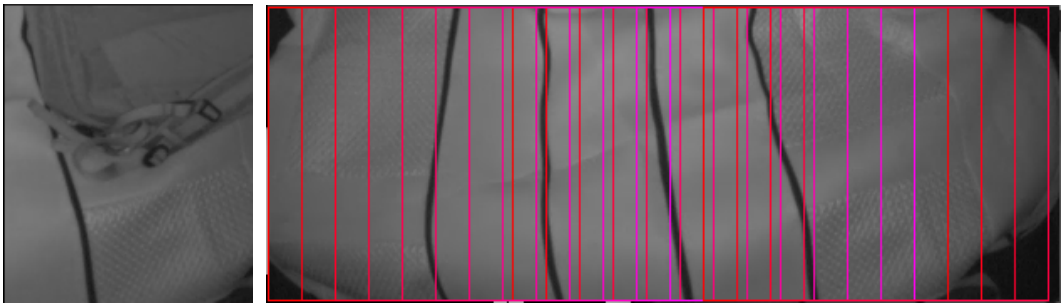
To break this down into details, first, we used the contours and bounding boxes to find potential abandoned objects in the mask we made earlier in subsection 4.3.2. Then, for each proposal, we got object patches from the frame before a person entered using sliding windows. These windows, which were a set size, moved across the image at a fixed interval. At each position, the window grabbed a patch the same size as the proposal from the foreground mask. We repeated this process for the whole image, creating a set of background patches covering the entire area of interest.

After extracting the proposal and patch pairs, we determined their similarity by extracting features from each using a neural network and calculating their cosine similarity. This allowed us to effectively assess their resemblance. To calculate the similarity between proposal-patch pairs, we utilized the pre-trained MobileNetV2 model, which was trained on ImageNet. The second last layer was utilized as the feature layer to extract features from the images. Once the features were extracted from the proposal and patch images, we calculated similarity using the cosine



(a) FPP (b) Sliding window

Figure 4.8: Sliding window for false positive proposal case, FPP stands for false positive proposal



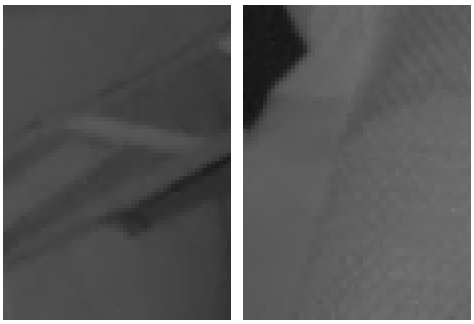
(a) TPP (b) Sliding window

Figure 4.9: Sliding window for true positive proposal case, TPP stands for true positive proposal

similarity score between the proposal feature and background patch feature. This similarity score indicated how closely their feature vectors aligned, suggesting their resemblance. A higher cosine similarity score suggests a greater likeness between the two objects, implying a higher probability that they are the same objects within the scene.

Based on the similarity scores, we determined whether each proposal constituted a false positive based on a minimum similarity score threshold value, which was set at 0.85 according to our empirical experience. For each proposal, if there was a patch with a similarity score exceeding this threshold, then the proposal was marked as a false positive because it strongly resembled a background object, suggesting it existed before the passenger entered. In such cases, the sliding window exited early, as depicted in Figure 4.8. Otherwise, proposals with similarity scores below the threshold were considered true positives, as they did not closely resemble any background component. Thus, they were regarded as abandoned objects once all the window searches were finished, as illustrated in Figure 4.9.

We have also experimented with the segmentation model based on MobileNet created by Smart Eye. It was trained on a dataset comprising 47 common category labels



(a) Proposal (b) Patch

Figure 4.10: A proposal-patch pair example in Video No.3

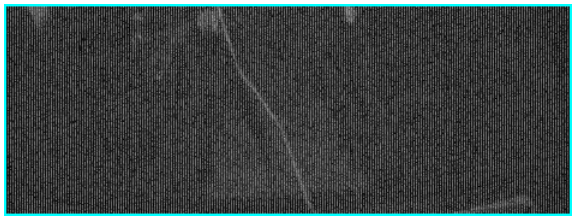


Figure 4.11: False positive caused by flashing light

commonly found in car cabin environments. The segmentation model performed worse than MobileNetV2 due to its less stable distribution of similarity scores, which made setting a threshold challenging. Moreover, its similarity scoring proved to be less reliable, yielding less consistent scores compared to MobileNetV2. For one instance in video No.3 Figure 4.10, the segmentation model assigned a similarity score of 0.899 for this proposal-patch pair, whereas MobileNetV2 scored it at 0.763. This discrepancy is noteworthy, especially considering it is a part of an abandoned object, where a lower similarity score would be more appropriate.

4.3.5 Contiguous Frames Tracking

With the above steps, we filtered out false positives such as moved seat covers. However, another issue persisted: the occurrence of false positives due to light flashes or sudden noise such as in Figure 4.11. To address this, we implemented contiguous frame tracking. Alarm signals for abandoned objects were issued only when detected consistently over a series of frames. Specifically, we would only trigger an alarm for abandoned objects after consistently detecting them over five consecutive frames.

5

Results

In this chapter, we discuss how we setup the development environment, collected our data, the optimizations we performed to enable embedded execution for the proposed solution, and the final detection results on the collected videos using the proposed method.

5.1 Environment Setup

The whole project was carried on Jetson-Nano 4GB with NVIDIA Jetson Nano Developer kit Jetpack 4.6.4. It is equipped with Ubuntu 18.04. The CUDA version is 10.2.3, the cuDNN version is 8.2.1.32, the TensorRT version used is 8.2.1.9. The OpenCV version used is 4.8.0 compiled with CUDA support. This information can be acquired using the jtop tool[1]. The ncnn version used is 20240102 release[2].

The project code was developed in C++11 and compiled with the '-O3' optimization flag.

5.2 Data Collection

In a spatio-temporal context, an object is considered abandoned if its owner has exited a predefined detection area for a duration exceeding a specified time period t [34]. Based on the definition of an abandoned object, we consider it to possess the following factors: 1. It was not present in the background before a human entered. 2. It remained in the frame for several frames after the human departed.

With this principle, we visited Smart Eye's test car facility to collect data on abandoned objects in vehicle cabins. We captured 14 videos of various abandoned objects and drove around outdoors to simulate different lighting conditions and light changes. To ensure clarity for future solution implementation, we conducted data collection under specific conditions, including the use of an infrared camera, maintaining a fixed camera position, and assuming reliable performance of Smart Eye's human detection function.

[1] https://github.com/rbonghi/jetson_stats
[2] https://github.com/Tencent/ncnn/releases/tag/20240102

The incorporation of infrared (IR) imagery in our system was chosen because it is essential for its reliable performance, especially in low-light conditions such as during night time operations. IR technology enables us to capture clear and detailed images even when ambient light is scarce, ensuring accurate detection capabilities in challenging environments. Moreover, IR images are less sensitive to changes in ambient lighting, allowing our system to maintain consistent performance across varying light conditions. This resilience to light fluctuations enhances the overall robustness and effectiveness of our system.

The camera used to record the test video remained fixed throughout the process to prevent any changes in the position of objects, which could lead to false alarms in the system.

As shown in Table 5.1, for the videos, we experimented with various common items commonly abandoned, such as baby seats, headphones, bags, hats, and more. These objects are chosen based on the statistics of common left behind objects within the company and cover different scales, contrasting with the background. Additionally, we tested objects abandoned in different locations within the back seats, including both the right and left sides. We also assessed the impact of objects abandoned with seat covers and tested scenarios where the seat cover was intentionally moved to determine if such movement could potentially trigger false positives. Furthermore, we conducted experiments with special cases, including moving objects within the camera scene.

After gathering the test videos, we developed methods for abandoned object detection.

Table 5.1: Summary of Test Videos of Abandoned Objects

Test Video List			
Index	Abandoned Object	Seat Cover	Note
1	Airpods	Yes	
2	Baby seat	No	
3	Bag	Yes	
4	Bag	No	
5	Bag	No	
6	Coat	Yes	
7	Headphone	No	
8	Headphone	No	
9	Hat	No	Moved
10	Key	Yes	
11	Phone	Yes	
12	Phone	No	
13	Wallet	Yes	
14	Wallet	No	

5.3 Optimizations

Since our system needed to be run on embedded devices, efficiency was as critical as the well-functioning of the system. Therefore, after the algorithm was set up, we undertook many optimization techniques which will be discussed in the following subsections. The target device we chose is the Jetson Nano 4GB. Since our algorithm involves neural network inference part, it's more practical to run it on a device with an AI accelerator.

5.3.1 Profiling

Before optimization began, the entire algorithm was profiled to identify its bottlenecks. We selected two sample videos to simplify matters and consider time constraints for testing purposes. The first video depicted a scenario with a bag left behind and a moved seat cover, resulting in numerous false positive proposals (index 3 in Table 5.1). The second video featured a baby seat left behind without a seat cover (index 2 in Table 5.1). The neural network employed for filtering out false positives ran on the NCNN library on the CPU.

Table 5.2: Time profiling result with NCNN CPU engine

video index	**background model**	**filter out false positive**	**total time**
3	0.512	48.709	49.221
2	1.077	29.592	30.669

5.3.2 Neural Network Inference Engine Benchmark

As we can see from the Table 5.2, the most time-consuming part of the algorithm is filtering out false positives. This is reasonable since it ran neural network inference in a sliding window. In order to optimize the neural network inference part, we tested three types of inference engines on the target device - OpenCV, NCNN, and TensorRT. OpenCV and NCNN can run both on CPU and GPU while TensorRT can only run on NVIDIA GPU, leading us to five neural network inference settings. The NCNN CPU result is given out in Table 5.2 already. Below we list the profiling results of other options. All the time results are measured in the units of seconds.

Table 5.3: Time and GPU memory profiling result with NCNN GPU engine

video index	**GPU MEM(MB)**	**filter out false positive**
3	64	14.589
2	64	2.355

Given the results above, considering both time efficiency and memory footprint perspectives, we chose NCNN as the default neural network inference engine to further our optimizations.

Table 5.4: Time profiling result with OpenCV CPU engine

video index	GPU MEM(MB)	filter out false positive
3	-	53.031
2	-	30.294

Table 5.5: Time and GPU memory profiling result with OpenCV CUDA engine

video index	GPU MEM(MB)	filter out false positive
3	204	54.287
2	204	11.541

5.3.3 Reducing Input Size for Feature Extractor optimization

Initially, we used the MobileNetV2 pre-trained on the ImageNet-1k dataset with image size 224 as a feature extractor. But actually, lots of the proposal patches were not that large, we can use smaller input image sizes to reduce the total computation. Using the pre-trained models in repository [3], we successfully reduced the model size from 224 to 96. The profiling results of using input size 96 in the NCNN inference engine are listed in Table 5.7 below.

We can find that using a smaller input size for the pre-trained model significantly improves its efficiency. Usually, using a smaller input size will lead to a lower accuracy performance model. However, this did not affect our case here since we only used the output feature map to calculate cosine similarity.

5.3.4 Background Model Optimization

After implementing all the optimizations mentioned above, the time latency of the background model was of the same magnitude as the filter-out false positive part. In this part, we begin to optimize the simple background model.

Previously, the background model utilized floating point for data storage and computation. Since the incoming data was of uint8 type, we had to handle data type conversion from uint8 to floating point when processing an image, as well as conversion from floating point to uint8 when returning results to the caller. To mitigate the overhead of data conversion and floating point calculation, we transitioned the background model to utilize the uint8 data type for computation. This change eliminated the necessity for data conversion and enhanced computational efficiency.

Table 5.6: Time and GPU memory profiling result with TensorRT engine

video index	GPU MEM(MB)	filter out false positive
3	252	54.271
2	252	2.355

[3] https://github.com/d-li14/mobilenetv2.pytorch

Table 5.7: Time profiling result with NCNN GPU engine with input size 96

video index	GPU MEM(MB)	filter out false positive
3	63.3	2.766
2	63.3	0.718

Table 5.8: Time profiling result with comparison of float32 and uint8 background model

video index	bg model float32	bg model uint8	speedup
3	0.512	0.213	2.4
2	1.077	0.513	2.1

Furthermore, we observed that within the nested-for loop responsible for background per-pixel updates, an if condition was used to check if each pixel value equaled 0. This condition checking rendered the nested loop unparallelizable. To address this, we replaced the nested for loop with the cv::addWeighted function and utilized a mask update function for updating the background model. Inside cv::addWeighted, all pixel-wise operations were parallelized. However, the mask update function still involved condition checking. The original core code is presented in Listing A.1, while the modified version is shown in Listing A.2. This optimization yielded a slight improvement, as demonstrated in Table 5.9.

Table 5.9: Time profiling result with nested loop parallelization

video index	nested for loop	parallelized for loop	speedup
3	0.213	0.181	1.18
2	0.513	0.367	1.40

The above optimizations combined reduce the background model latency almost 2.8 times speedup for the given two test samples as shown in Table 5.10.

5.3.5 Optimizations Combined

The optimizations for the filtering false positive part and background model part were combined. The result has been recorded in Table 5.11.

We compared the total runtime of the initial version where no optimizations were applied, and the neural network ran on CPU, with the final version where all mentioned optimizations had been applied, and the neural network ran on GPU. As observed in Table 5.12, the remarkable speedup achieved makes it much more practical to run our algorithm on embedded devices.

5.3.6 Extra Heuristic

Another optimization we devised was to incorporate more heuristics into the sliding window search process. One heuristic involved ignoring cases where objects were displaced within the vehicle, focusing solely on objects introduced by passengers. In

Table 5.10: Time profiling result with original and optimized background model

video index	original	optimized	speedup
3	0.512	0.181	2.82
2	1.077	0.367	2.86

Table 5.11: Time profiling result with all optimization combined

video index	background model	filter out false positive	total time
3	0.181	2.766	2.947
2	0.367	0.718	1.085

such instances, the search region should be proximate to the proposal. Since, in the background, the nearby region of the proposal does not encompass the proposed object if it is a true positive (i.e., a left-behind object), and if it is a false positive (such as a seat cover or seat belt), the displaced region is assumed to be relatively small. Leveraging this assumption, a heuristic was applied to the sliding window search process. We expanded the proposal box area to include the surrounding region in the background image and compared the proposal image with the background image using a similarity threshold. If the similarity score exceeded a certain value, we inferred the proposal to be a false positive; otherwise, it was considered a true positive—a genuine left-behind object.

This optimization helped us avoid the complexity of the sliding window search. The result with the optimization applied was recorded in Table 5.13. However, because the heuristic altered the algorithm's logic, making it unable to detect objects moved significantly in space, it is presented only as an option for further speedup, and the result is not considered as part of the final speedup achieved in the optimization process.

5.4 Detection Results

In this section, the result of the background model and false positive elimination are summarized in Table 5.14. The first column indicates the index of each video. The second column of the table indicates whether abandoned objects were successfully detected as candidate proposals for each video, as shown in Figure A.1. The third column shows whether all false positives were successfully removed for each video. The final column assesses whether the alarm accurately notified that the objects were abandoned, which means if the alarm notified about the correct object and no false positives were shown in the notification. The comprehensive results can be

Table 5.12: Total speedup after optimizations applied

video index	original total time	final total time	speedup
3	49.221	2.947	16.7
2	30.669	1.085	28.3

Table 5.13: Time profiling result with heuristic for sliding window

video index	GPU MEM(MB)	filter out false positive
3	63.3	0.835
2	63.3	0.289

found in Figure A.2.

Table 5.14: Background Model and False Positive Removal Results

Index	Background Model	False Positive	Detection Successful
1	No	Yes	No
2	Yes	Yes	Yes
3	Yes	Yes	Yes
4	Yes	Yes	Yes
5	Yes	Yes	Yes
6	Yes	Yes	Yes
7	Yes	Yes	Yes
8	No	-	No
9	Yes	Yes	Yes
10	Yes	Yes	Yes
11	Yes	Yes	Yes
12	Yes	Yes	Yes
13	Yes	Yes	Yes
14	No	-	No

In our experimentation across all 14 videos, the algorithm successfully extracted candidate objects from 11 videos, while in 3 videos, abandoned objects were not detected as candidates. In the first input video, although the background model did not succeed in extracting abandoned objects, all the false positives were filtered out. For the 8th and 14th input videos, since no candidates were extracted from the background model, no objects could proceed to the next step for false positive filtering. In all other test videos, the false positives were successfully filtered out. As a result, we achieved successful detection in 11 instances, while the accuracy reached 78.57%.

6

Discussion

In summarizing our research endeavors, our key accomplishments lie in the successful development of an algorithm tailored to our specific objective: efficiently detecting abandoned objects within vehicle cabins. Additionally, our exploration yielded intriguing findings regarding both the results obtained and potential avenues for future work.

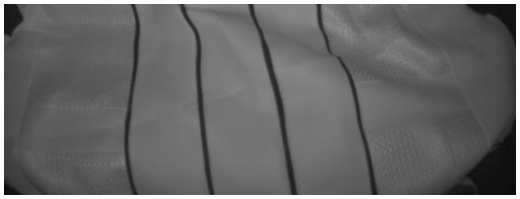
6.1 Result Analysis

In our results, there were 3 cases of detection failure, all we regard to be attributed to low contrast. These cases are delineated in Figure 6.1. For instance, as depicted in Figure 6.1(a) and (b), an AirPod was left in the upper left corner in Video No.1. Its detection eluded us due to its color blending seamlessly with the background. Similarly, in Figure 6.1(c) and (d), a black headphone was abandoned against a black background, further complicating detection. Finally, in Figure 6.1(e) and (f), a black wallet was situated in the upper left portion of the scene, rendering it indistinguishable from the background due to matching colors.

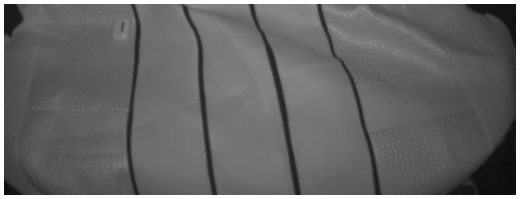
The instances where these objects weren't successfully detected can be considered reasonable because of the inherent difficulties posed by low-contrast scenarios introduced by infrared image input. In such situations, where abandoned objects blend closely with the background in terms of color and texture, their detection was inherently challenging. Additionally, even the human eye may struggle to distinguish these objects under similar conditions. Hence, encountering instances where detection algorithms falter in such demanding visual environments is understandable. Neither the object detection method nor the background modeling method can overcome the low contrast issue better than our method.

6.2 Model Selection

In this section, the process of model selection is discussed, where two primary methodologies were considered: Gaussian Mixture Model (GMM) and K-Nearest Neighbors (KNN) for background model in long-term and short-term integration background model, alongside the evaluation of two distinct neural network architectures for feature extraction: MobileNetV2 and a segmentation model.

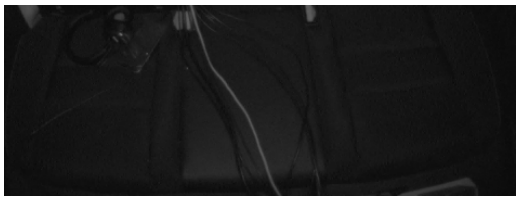


(a) Before human entry video No.1

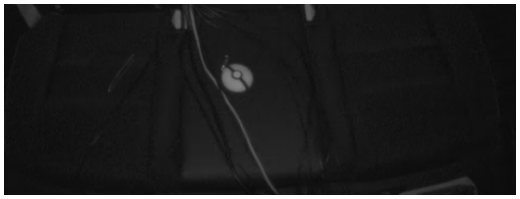


(b) After human exit video No.1

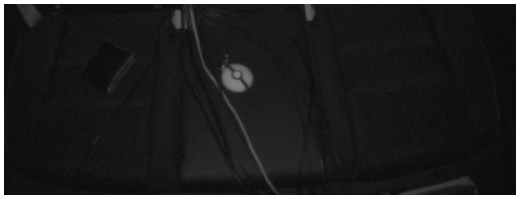
(c) Before human entry video No.8



(d) After human exit video No.8



(e) Before human entry eideo No.14

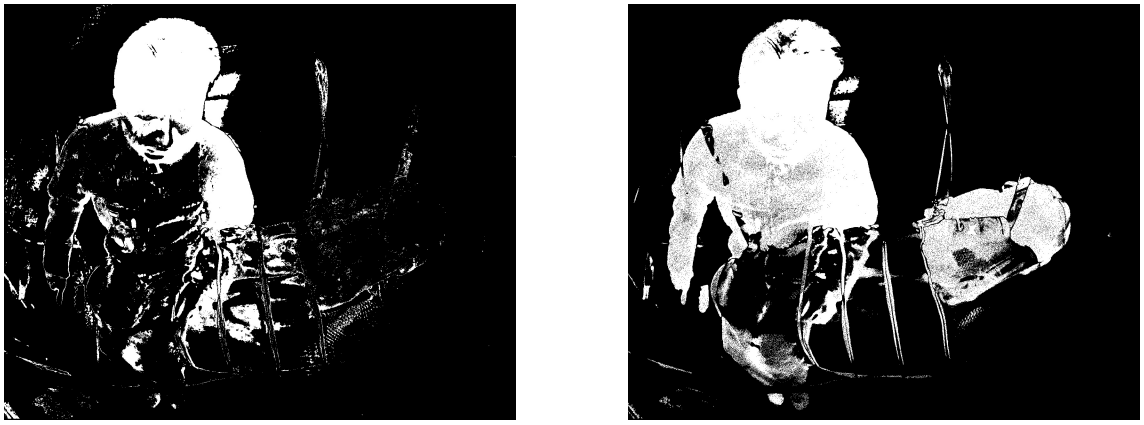


(f) After human exit video No.14

Figure 6.1: Comparison of scene before and after human entry in video No.1, 8, 14

6.2.1 Background Model Selection

For the background modeling task in Section 4.2, our initial approach involved experimenting with Gaussian Mixture Model (GMM) and K-Nearest Neighbors (KNN) to evaluate their performance. Upon comparison, KNN demonstrated superior results as shown in Figure 6.2.



(a) GMM (b) KNN

Figure 6.2: Performance of GMM and KNN for long-term model at frame 800 of video No.3

This outcome can be attributed to KNN's sensitivity to data. KNN, being a non-parametric algorithm, does not make any assumptions about the underlying distribution of the data. This allows it to adapt more flexibly to the data's inherent patterns and local structures, leading to more accurate predictions. On the other hand, GMM assumes that the data follows a Gaussian distribution, which might not capture the complexity of the actual data distribution as effectively.

However, it's noteworthy that both KNN and GMM entail significant computational costs. Therefore, we created an alternative background model that selectively compares the difference between frames captured before a human enters and frames captured after the human leaves. This method proved to be less time-consuming and delivered excellent results for several reasons:

1. Reduced Computational Load. KNN and GMM require significant processing power, especially for large datasets. KNN involves distance calculations for each query point, and GMM involves iterative processes to fit the data to multiple Gaussian distributions. The proposed method, however, performs basic statistical operations, making it considerably faster and less resource-intensive.

2. Sufficient Accuracy. In scenarios where extreme precision is not necessary, the proposed method provides adequate accuracy. In our case, detecting changes before and after human presence did not require the complex modeling of GMM or the precise neighbor comparisons in KNN. The proposed approach is

sufficiently accurate for identifying changes in the scene and provides enough proposals for the later stage of processing.

6.2.2 Feature Extraction Model Selection

During our evaluation of MobileNetV2 and the segmentation model developed by Smart Eye, we uncovered compelling insights. MobileNetV2 demonstrated superior speed and accuracy compared to the segmentation model for feature extraction. Several factors may contribute to this observation:

1. MobileNetV2 has fewer parameters, totaling 3.5 million, compared to the segmentation model, which has 5.3 million parameters. A lower number of parameters typically indicates a less complex model architecture, leading to reduced computational requirements. Consequently, a model with fewer parameters generally demands fewer computational resources during both training and inference phases. This makes MobileNetV2 more efficient and suitable for applications where computational power is limited.
2. MobileNetV2 was trained on 1000 classes which is a larger and more diverse dataset than the segmentation model which was trained on 47 classes. Being trained on more classes can help a model with fewer weights generalize better to unseen data,

6.3 Methods Comparison

In this section, we compare our proposed method with both the deep learning-based approach and the background modeling-based approach. Our proposed method and the deep learning-based method effectively address the issue of passengers' occupancy timescales by focusing specifically on periods when passengers are not seated in the cabin. But for the background modeling method, it's hard to adjust the background model updating rate according to passengers' in-cabin time. For the background movement problem, by utilizing a neural network as a feature extractor and calculating similarity scores, we are able to distinguish between false positive and true positive proposals. Furthermore, our method's capability to detect objects beyond those trained by a feature extractor allows for generalization to untrained objects. However, all three methods suffer from low contrast input problems. The comparison of these three methods is summarized in Table 6.1.

Table 6.1: Comparison of Approaches in Abandoned Object Detection

	Deep learning based method	Long- and short-term background modeling	Our approach
Solve the passengers' timescale of occupancy	Yes	No	Yes
Solve the background movement (seat cover/safety belt)	Yes	No	Yes
Solve generalization & reliance on training data	No	Yes	Yes
Solve low contrast input	No	No	No

7

Conclusion

In conclusion, this study has successfully develop an efficient algorithm for detecting abandoned objects in a vehicle cabin environment that does not rely on training and can generalize to a wide range of objects. We have identified utilizing the simple background model detailed in Section 4.2 to extract candidate abandoned objects. Then by comparing the features of proposed candidate abandoned objects before and after human presence using similarity score calculated from feature vector extracted using MobileNetv2 pretrained on ImageNet, we can effectively filter out false positives exceeding a specified similarity threshold. Finally, by profiling and optimizing the proposed solution on Jetson Nano, the speed has been improved by orders of magnitude to guarantee efficient execution on embedded devices. The results met our general expectations of efficiently detecting abandoned object detection methods without the need for extensive data training. The study addressed and solved three key research questions proposed at Section 1.1:

RQ1: *How can we detect abandoned objects of all categories in the vehicle cabin?*

To solve the challenge of detecting a diverse array of items that can be forgotten in cars, the study employed a simple background model to extract extract candidate objects by comparing frames before and after human presence. This is based on the human detection model developed by Smart Eye. With this method, the algorithm can generalize to various object categories without the need for an extensive dataset of all potential abandoned items.

RQ2: *How can we remove false positives detected for abandoned objects?*

The problem of false positives is solved by comparing the features of candidate abandoned objects before and after human presence. Using a similarity score calculated from feature vectors extracted with MobileNetV2, false positives were effectively filtered out by exceeding a specified similarity threshold.

RQ3: *How can we optimize the designed algorithm for abandoned object detection so that it can execute efficiently on embedded devices with limited computation?*

By profiling and optimizing the proposed solution on the Jetson Nano, the speed of the algorithm was improved. Specific methods to enhance performance include converting the background model from float32 to uint8, parallelizing nested for loops, moving the neural network from CPU to GPU, and reducing the MobileNetV2 input size from 224 to 96.

The results analysis demonstrated that the proposed method meets the general expectations for efficient and accurate abandoned object detection. Some limitations were left for future work:

7.1 Limitations

As mentioned in the introduction, our research has certain limitations as we specifically focus on abandoned objects detected by stationary cameras. Consequently, our research did not consider methods for overcoming the following factors:

1. Low contrast issues pose significant challenges, such as the difficulty in detecting abandoned objects that blend seamlessly with the background due to similar color and texture.
2. Camera Movement During Object Detection. The proposed algorithm is not equipped to handle scenarios where the camera is moved. If the camera position changes during object detection, the method cannot successfully process and analyze the frames.
3. Persistent shadows, like a tree shadow consistently cast in the same spot, can be mistakenly detected as abandoned objects.

While our current research did not encompass these factors, they undoubtedly still hold significance for future investigation.

7.2 Future Work

In considering future directions for research, several challenges remain to be addressed to enhance the performance of abandoned object detection in dynamic environments:

1. Infrared image input leads to low contrast between object and background. One of the issues brought about by using infrared image input in our system was the low contrast problem. For example, the system was unable to recognize black phones and certain material seats because they had similar radiation distribution under infrared cameras. This can be mitigated by using some infrared image enhancement techniques or designing better image processing pipelines so that the input to the designed system has higher contrast which will result in a high recall rate.
2. Movement of the camera during object detection. Using object tracking algorithms alongside motion analysis is an efficient method for detecting abandoned objects with dynamic cameras. By monitoring object movement in the camera's field of view and analyzing motion characteristics like sudden disappearance or reduced intensity over time, potential abandoned objects can be identified.
3. Presence of persistent shadows in the background, potentially leading to false positives. Addressing the challenge of persistent shadows requires the development of shadow-resistant object detection algorithms. This could involve

exploring innovative approaches, such as adaptive thresholding techniques or the incorporation of shadow detection modules within object detection pipelines. Additionally, leveraging deep learning architectures trained on diverse datasets containing shadowed scenarios may enhance the model's ability to distinguish between shadows and actual objects.

Focusing on these areas in future research, efforts can lead to advancements in object detection systems, enhancing their reliability and effectiveness in real-world applications to further advance this study.

Bibliography

[1] *Uber lost & found index*, https://www.uber.com/newsroom/2022-uber-lost-found-index/, Accessed on: Insert Date, 2022.

[2] Smart Eye, *Automotive solutions*, https://www.smarteye.se/solutions/automotive/, Accessed: 2024-05-19, 2024.

[3] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. arXiv: 1804.02767 [cs.CV].

[4] K. Lin, S.-C. Chen, C.-S. Chen, D.-T. Lin, and Y.-P. Hung, "Abandoned object detection via temporal consistency modeling and back-tracing verification for visual surveillance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1359–1370, 2015. DOI: 10.1109/TIFS.2015.2408263.

[5] M. Xiao, C. Han, and X. Kang, "A background reconstruction for dynamic scenes," Aug. 2006, pp. 1–7. DOI: 10.1109/ICIF.2006.301727.

[6] J.-Q. Wang, Z.-L. Shi, and S.-B. Huang, "Detection of moving targets in video sequences," vol. 32, pp. 5–8, Dec. 2005.

[7] Y. Li, S. Wang, Q. Tian, and X. Ding, "Feature representation for statistical-learning-based object detection: A review," *Pattern Recognition*, vol. 48, no. 11, pp. 3542–3559, 2015, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2015.04.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320315001570.

[8] X. Zhang, C. Zhu, S. Wang, Y. Liu, and M. Ye, "A bayesian approach to camouflaged moving object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 9, pp. 2001–2013, 2017. DOI: 10.1109/TCSVT.2016.2555719.

[9] Wahyono, R. Pulungan, and K.-H. Jo, "Stationary object detection for vision-based smart monitoring system," in *Intelligent Information and Database Systems*, N. T. Nguyen, D. H. Hoang, T.-P. Hong, H. Pham, and B. Trawiński, Eds., Cham: Springer International Publishing, 2018, pp. 583–593.

[10] C. Zhan, X. Duan, S. Xu, Z. Song, and M. Luo, "An improved moving object detection algorithm based on frame difference and edge detection," in *Fourth International Conference on Image and Graphics (ICIG 2007)*, 2007, pp. 519–523. DOI: 10.1109/ICIG.2007.153.

[11] "Cnns for surveillance footage scene classification." (2018), [Online]. Available: https://arxiv.org/pdf/1809.02766.pdf.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. arXiv: 1512.00567. [Online]. Available: http://arxiv.org/abs/1512.00567.

[13] H. Park, S. Park, and Y. Joo, "Detection of abandoned and stolen objects based on dual background model and mask r-cnn," *IEEE Access*, vol. 8, pp. 80 010–80 019, 2020. DOI: 10.1109/ACCESS.2020.2990618.

[14] S. Smeureanu and R. T. Ionescu, "Real-time deep learning method for abandoned luggage detection in video," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 1775–1779. DOI: 10.23919/EUSIPCO.2018.8553156.

[15] S. P. Lwin and M. T. Tun, "Deep convonlutional neural network for abandoned object detection," *Int Res J Mod Eng Technol Sci*, vol. 4, no. 01, pp. 1549–1553, 2022.

[16] D. Shyam, A. Kot, and C. Athalye, "Abandoned object detection using pixel-based finite state machine and single shot multibox detector," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6. DOI: 10.1109/ICME.2018.8486464.

[17] H. Xu and F. Yu, "Improved compressive tracking in surveillance scenes," Jul. 2013, pp. 869–873. DOI: 10.1109/ICIG.2013.176.

[18] W. Liu, D. Anguelov, D. Erhan, *et al.*, "Ssd: Single shot multibox detector," in *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 21–37, ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0_2. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[19] W. Liu, P. Liu, C. Xiao, and R. Hu, "General-purpose abandoned object detection method without background modeling," in *2021 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2021, pp. 1–5. DOI: 10.1109/IST50367.2021.9651400.

[20] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. arXiv: 1506.01497. [Online]. Available: http://arxiv.org/abs/1506.01497.

[21] C. Zhao, J. Wang, N. Su, Y. Yan, and X. Xing, "Low contrast infrared target detection method based on residual thermal backbone network and weighting loss function," *Remote Sensing*, vol. 14, no. 1, 2022, ISSN: 2072-4292. DOI: 10.3390/rs14010177. [Online]. Available: https://www.mdpi.com/2072-4292/14/1/177.

[22] S. Misra and Y. Wu, "Chapter 10 - machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales with feature extraction and feature ranking," in *Machine Learning for Subsurface Characterization*, S. Misra, H. Li, and J. He, Eds., Gulf Professional Publishing, 2020, pp. 289–314, ISBN: 978-0-12-817736-5. DOI: https://doi.org/10.1016/B978-0-12-817736-5.00010-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128177365000107.

[23] K. A. M. Said and A. B. Jambek, "Analysis of image processing using morphological erosion and dilation," *Journal of Physics: Conference Series*, vol. 2071, no. 1, p. 012 033, Oct. 2021. DOI: 10.1088/1742-6596/2071/1/012033. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2071/1/012033.

[24] T.-L. Pao, W.-Y. Liao, and Y.-T. Chen, "A weighted discrete knn method for mandarin speech and emotion recognition," in Nov. 2008, ISBN: 978-953-7619-29-9. DOI: 10.5772/6370.

[25] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[26] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, *Imagenet training in minutes*, 2018. arXiv: 1709.05011 [cs.CV].

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," Jun. 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.

[28] M. Almghraby and A. O. Elnady, "Face mask detection in real-time using mobilenetv2," *International Journal of Engineering and Advanced Technology*, vol. 10, no. 6, pp. 104–108, 2021.

[29] Tencent, *Ncnn: Neural Compute Network Neural Network Framework*, https://github.com/Tencent/ncnn, Year of Last Update.

[30] OpenCV, *OpenCV: Open source computer vision library*, https://github.com/opencv/opencv, Year of Last Update.

[31] NVIDIA. "TensorRT Documentation," NVIDIA Developer. (Year of Access), [Online]. Available: https://developer.nvidia.com/tensorrt.

[32] NVIDIA. "CUDA Toolkit," NVIDIA Developer. (Year of Access), [Online]. Available: https://developer.nvidia.com/cuda-toolkit.

[33] The Khronos Group. "Vulkan," The Khronos Group. (Year of Access), [Online]. Available: https://www.vulkan.org/.

[34] M. Bhargava, C.-C. Chen, M. S. Ryoo, and J. K. Aggarwal, "Detection of abandoned objects in crowded environments," in *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, 2007, pp. 271–276. DOI: 10.1109/AVSS.2007.4425322.

A

Appendix

A.1 Test Video Index and File References

This table lists the test videos used in our experiment on abandoned object detection. Each video is indexed and named according to the content and scenario it captures, providing a clear reference for further analysis.

Table A.1: Summary of Test Videos of Abandoned Objects

Test Video List	
Index	Abandoned Object
1	yuhua_airpods_with_bg_right.mp4
2	hao_baby_right_to_left.mp4
3	yuhua_bag_with_bg_left.mp4
4	yuhua_bag_left.mp4
5	yuhua_bag_right.mp4
6	yuhua_coat_right.mp4
7	hao_headphone_right.mp4
8	yuhua _headphone_right.mp4
9	yuhua_hat_moved_left.mp4
10	yuhua_key_with_bg_left.mp4
11	yuhua_phone_with_bg_right.mp4
12	hao_phone_right.mp4
13	yuhua_wallet_left.mp4
14	yuhua_wallet_right.mp4

A.2 Results of All Input Videos

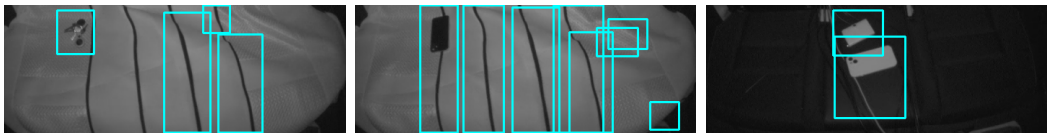
This section presents the results of the background model and false positives filtering from all 14 input videos.

Figure A.1 showcases the results of our background modeling approach across different test videos. Each subfigure (a) to (n) presents the output of the background model, illustrating its effectiveness in distinguishing foreground objects from the background under various conditions.

(a) BG model result 1 (b) BG model result 2 (c) BG model result 3

(d) BG model result 4 (e) BG model result 5 (f) BG model result 6

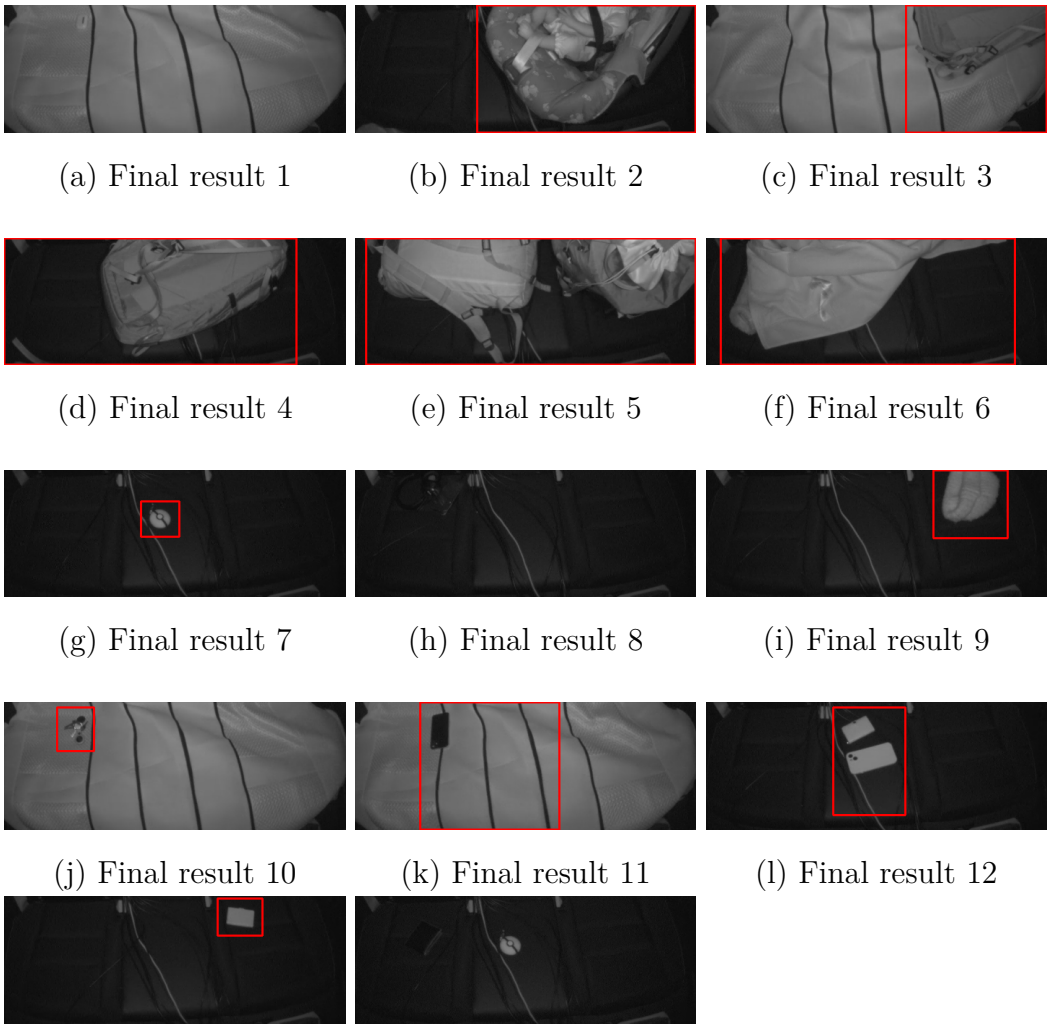
(g) BG model result 7 (h) BG model result 8 (i) BG model result 9



(j) BG model result 10 (k) BG model result 11 (l) BG model result 12

(m) BG model result 13 (n) BG model result 14

Figure A.1: Background Model Results



(a) Final result 1 (b) Final result 2 (c) Final result 3

(d) Final result 4 (e) Final result 5 (f) Final result 6

(g) Final result 7 (h) Final result 8 (i) Final result 9

(j) Final result 10 (k) Final result 11 (l) Final result 12

(m) Final result 13 (n) Final result 14

Figure A.2: Final model results

Figure A.2 presents the final results of our model after applying our algorithm including the background model and false positives filtering. Each subfigure (a) to (n) depicts the detected abandoned objects in the test videos, demonstrating the model's accuracy and robustness in identifying and isolating these objects from their surroundings.

A.3 Optimization of Background Model

Listing A.1: Code before parallelizing

```
for (int i = 0; i < img_input.rows; i++) {
  for (int j = 0; j < img_input.cols; j++) {
    if (img_foreground.at<uchar>(i, j) == 0) {
      img_background.at<uchar>(i, j) =
        alpha_detection * img_input.at<uchar>(i, j) +
```

```
            (1 - alpha_detection) * img_background.at<uchar>(i, j);
        }
    }
}
```

Listing A.2: Code after removing if statement and parallelizing

```
cv::Mat update_mask = (img_foreground == 0);
cv::Mat blended;
cv::addWeighted(img_input, alpha_detection, img_background,
    1 - alpha_detection, 0, blended);
blended.copyTo(img_background, update_mask);
```