Methods and algorithms for detecting and classifying moving objects based on the evaluation of an object's geometrical parameters in an image sequence for smart lighting systems

Von dem Promotionszentrum
Ingenieurwissenschaften und Informationstechnologien
an der Hochschule Anhalt
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

Doktor
(Dr.-Ing.)

Vorgelegt von
M.Eng. Ivan Matveev

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr.-Ing. habil. Christian Hentschel, Brandenburgische Technische Universität
2. Prof. Dr.-Ing. Stephan Schmidt, Hochschule Merseburg
3. Prof. Dr.-Ing. Eduard Siemens, Hochschule Anhalt

Die Verleihung des akademischen Grades erfolgt mit Bestehen
der Verteidigung am 25.01.2024 mit dem Gesamturteil *magna cum laude*.

Acknowledgment

First of all, I express my gratitude to my supervisor Ingo Chmielewski for his wise advice and support. I tried to take over his experience with technical systems. He showed me how theoretical and implementation problems could be solved. I am grateful to my co-supervisor Dmitry Kachan for the structural and valuable comments that helped to improve the work. I want to thank my advisor Eduard Siemens, who has tirelessly supported me all these years. I take my example from him in enthusiasm and love for work. My thanks go to the entire FILA Lab staff and, in particular, to Kirill Karpov, a co-author of many publications.

I would like to acknowledge the financial support in the form of a Ph.D. scholarship from the Saxony-Anhalt state, which has kept me alive during my dissertation work. The university's technical base, ES-LAB and FILA laboratories' equipment played an essential role in my work.

In addition, I cannot fail to mention my sister, parents, and grandparents, who have always supported me in my endeavors and instilled in me a thirst for experimentation, curiosity, and open-mindedness. Last but not least, I am grateful to my wife Nadia, who has been with me all these years, tolerated all the difficulties associated with the dissertation, and did not let me give up in difficult moments.

Abstract

Modern machine learning based object detection methods have high accuracy; however, at the same time, they require the increased computing power of processing units. Thus, widespread low-performance IoT devices generating a substantial amount of data cannot apply corresponding machine learning algorithms for real-time data processing due to a lack of local computational resources. The Dimensional Based Object Detection (DBOD) algorithm for low-performance single-board computers is developed and evaluated in the course of this dissertation. The proposed algorithm exploits geometrical features of objects in an image and real-world scene parameters (e.g. camera's focal length, height and angle of installation) as classification features. Extraction and classification of these features are computationally simple procedures that single-board computers can execute in real-time. The algorithm is focused on detecting and classifying objects that are the most expected in urban environments: pedestrians, bicyclists, and vehicles. The algorithm can be applied for processing video sequences captured by a CCTV camera. A method for fast generating synthetic training features for the DBOD has been proposed. The algorithm DBOD has been tested on real and synthetically generated datasets. The results have shown that low-performance systems, such as popular Raspberry Pi, are capable of object classification with the required frame rate and accuracy for smart city applications.

Zusammenfassung

Moderne, maschinenlernen-basierte Objekterkennungsmethoden weisen eine hohe Genauigkeit auf, erfordern jedoch gleichzeitig eine hohe Rechenleistung der Verarbeitungseinheiten. Daher können weit verbreitete IoT-Geräte mit geringer Leistung, die eine beträchtliche Menge an Daten erzeugen, aufgrund fehlender lokaler Rechenressourcen keine entsprechenden Algorithmen für maschinelles Lernen zur Datenverarbeitung in Echtzeit anwenden. Der Algorithmus Dimensional Based Object Detection (DBOD) für leistungsschwache Einplatinencomputer wird im Rahmen dieser Dissertation entwickelt und evaluiert. Der vorgeschlagene Algorithmus nutzt geometrische Merkmale von Objekten in einem Bild und reale Szenenparameter (z.B. Brennweite, Höhe und Installationswinkel der Kamera) als Klassifikationsmerkmale. Die Extraktion und Klassifizierung dieser Merkmale sind rechnerisch einfache Verfahren, die von Einplatinencomputern in Echtzeit ausgeführt werden können. Der Algorithmus konzentriert sich auf die Erkennung und Klassifizierung von Objekten, die in städtischen Umgebungen am häufigsten erwartet werden: Fußgänger, Radfahrer und Fahrzeuge. Der Algorithmus kann für die Verarbeitung von video sequences eingesetzt werden, die von einer CCTV-Kamera aufgenommen wurden. Darüber hinaus wurde eine Methode zur schnellen Erzeugung synthetischer Trainingsmerkmalen für den DBOD vorgeschlagen. Der Algorithmus DBOD wurde an realen und synthetisch erzeugten Datensätzen getestet. Die Ergebnisse zeigen, dass Systeme mit geringer Leistung, wie der beliebte Raspberry Pi, in der Lage sind, Objekte mit der erforderlichen Bildrate und Genauigkeit für Smart-City-Anwendungen zu erkennen und zu klassifizieren.

Contents

List of Figures

List of Tables

Acronyms

AP Average Precision . . . . . 19, 36 f., 69

BS Background Subtraction . . . . . 21, 37, 39 f., 43, 48 f., 69

CDF Cumulative distribution function 49

CLAHE Contrast Limited Adaptive Histogram Equalization 49

CNN Convolutional Neural Network . . . . . 19, 28–31, 45, 76 f.

CV Computer Vision . . . . . 17 f., 21, 25, 39, 43, 45 f.

DBOD Dimensional Based Object Detection 59, 65, 69, 73, 76–79

DL Deep Learning 28, 37, 45

F1 F1-measure . . . . . 19, 35 f., 39, 43 f., 46, 70, 78

FN False Negative . . . . . 19, 23 f., 35 f., 69 f., 72, 78

FP False Positive . . . . . 19, 24, 27 f., 30, 35 ff., 39, 69 f., 72, 76, 78

FPS Frames per Second . . . . . 19, 27, 30 f., 37, 39, 43 ff., 75, 77 f.

GPU Graphical Processing Unit . . . . . 17, 28, 30 f., 45

HAOV Horizontal Angle of View 32

HOG Histograms of Oriented Gradients . . . . . 27 f., 30, 37, 45

ICF Integral Channel Features 27

IoT Internet of Things 17, 19, 46, 73

IOU Intersection over Union 34, 74

IR Infrared . . . . . 17, 24 f., 39, 43 f., 46, 49, 59, 64 ff., 69

mAP mean Average Precision . . . . . 19, 31, 36 f., 70, 74, 76, 78

ML Machine Learning 25, 28

MOG Mixture of Gaussians 42, 44, 46, 48, 78

R-CNN Region-based Convolutional Neural Network . . . . . 30 f., 73, 75 ff.

ROI Regions of Interest . . . . . 30 f., 39, 47 f., 58

RW Radio-wave 25, 44

SL Smart Lighting . . . . . 17, 19, 22, 32 ff., 37, 44, 46, 58

SoC System on Chip 45

SSD Single Stage Detector . . . . . 31, 37, 45, 73 f.

SVM Support Vector Machine . . . . . 28, 30 f.

TN True Negative 19, 35

TP True Positive 19, 34–37, 69

US Ultrasonic . . . . . 23 f., 44

VAOV Vertical Angle of View 32

YOLO You Only Look Once . . . . . 31 f., 37, 45, 73

Introduction

Relevance of the topic. Conventional lighting systems provide illumination permanently during the nighttime; however, pedestrian and vehicle traffic intensity in peripheral urban areas at this time of the day is not significant. The proposed method has been developed in the context of Smart Lighting (SL) systems, particularly the SmartLighting project [1], [2]. The project aims at energy saving by applying an intelligent management approach for street lighting systems. The SL system turns on illumination depending on the presence of pedestrians, bicyclists, and cars in the lighting area. Lamps of such a system are connected via a decentralized mesh wireless network that predicts the most probable object's movement vector based on information from the multiple lights. Street lighting optimization via such an intelligent approach improves the environmental situation and reduces expenses on street illumination. The detection subsystem is a vital component affecting the workability and efficiency of the lighting system.

A variety of detection methods considers the task of outdoor object detection. Conventional methods exploit objects' physical phenomena and properties, such as Infrared (IR) radiation, the reflection of waves, the Doppler effect, etc., to detect moving objects. However, conventional moving object detection methods have a number of disadvantages, in particular, short detection range, high error rate, and high cost. Moreover, such methods have limitations regarding the type and speed of the detecting object. The detection accuracy of such methods often depends on operating conditions.

Recently, Computer Vision (CV) methods have achieved high accuracy in the field of detection and object classification. At the same time, such methods require the significant computing power of devices, making it impossible to process video data in real-time autonomously on low-performance single-board computers [3], even with the external hardware acceleration units [4]. Thus, developing a new, efficient algorithm of low computational complexity for object detection is an essential task for intelligent city systems.

The smart city concept is based on data collection by the Internet of Things (IoT) sensors. The typical sensors are cameras used in intelligent city services, such as video surveillance and traffic monitoring. Cameras produce a significant amount of video data, usually centrally processed by high-performance computers equipped with Graphical Processing Units (GPUs) [5]. Some services that use a camera as a sensor may require decentralized data processing performed directly on IoT devices, primarily due to the lack of a stable Internet connection. Transferring video data from multiple IoT devices over wireless networks for processing in a cloud can be inappropriate due to latency, loss, and poor connection stability. These factors affect the performance of services that

require immediate reaction to events in the real world. At the same time, single-board computers are becoming more powerful and less expensive; however, video processing in the real-time mode remains challenging due to the limited computational resources of single-board computers.

Existing detection methods do not meet all the SmartLighting requirements simultaneously, the main of which are:

- the ability to detect and classify moving objects which are the most expected on a pedestrian or roadway and in need of lighting: pedestrians, cyclists, and vehicles;
- low computational requirements of the detection algorithm due to the use of budgetary and low-performance computers (Raspberry Pi, Orange Pi, etc.) as data processing devices;
- the ability to detect objects moving at speeds up to 60 km/h, at distances up to 25 m from the base of a street light.

The goal of this work is aimed at solving the problem of efficient moving object detection and classification in an urban environment, using low-performance computers as data processing devices. The following tasks have to be answered to achieve the goal:

1. To analyze the existing detection approaches and methods in order to identify the limitations and disadvantages considering the established requirements.
2. To develop a detection algorithm that provides the required accuracy and computational complexity, running on devices with low computing power.
3. To develop an algorithm for feature extraction, which provides the extraction of unique characteristics of an object in the real world based on its geometric parameters in the image and the camera parameters used for shooting.
4. To develop a methodology for collecting synthetic data that will generate plausible features of objects for training the classifier based on logistic regression.
5. To evaluate the effectiveness of the developed method on data sets collected in scenarios close to exploitation.
6. To compare the effectiveness of the developed method with existing alternatives in terms of average accuracy and performance.

Novelty:

1. For the first time, an original analytical study summarizes existing conventional sensor-based and Computer Vision methods in the context of the assigned detection problem.
2. A proposed algorithm allows objects to be detected in single-channel low-resolution frames (from 320×240 px) at 128 frames per second on a low-performance computer. The algorithm provides the required accuracy (no more than 30 % false positive and 15 % false negative error rates), which is unattainable by existing CV algorithms. The above-mentioned accuracy and performance are achieved by a proposed method for computationally inexpensive

extraction of object features. The feature extraction method uses information from images and camera installation parameters, while existing methods extract information from pictures only. In particular, formulas for calculating an object's geometric characteristics and coordinates in the real world are proposed.

3. A proposed method for generating synthetic scenarios of object movement replaces gathering the real data; simplifies and speeds up the process of collecting and preparing data for training a classifier. In contrast to existing simulation methods, the proposed one does not require manual 3D reconstruction of the expected environments in detail and generates training features out of given numerical ranges.

Practical relevance. The developed detection method can be used to detect moving objects by IoT devices in traffic counting or alarm systems. The detection method has been integrated into a street lighting system as part of the SmartLighting project, replacing conventional passive infrared sensors, which do not provide the required detection range. For the first time, the detection method allows producing different actions depending on the class of moving objects.

Research methodology and methods. Experiments have been carried out on real and synthetic datasets. Measurements and statistical analysis were used to obtain accuracy and performance metrics for the developed detection algorithm on the target computing platform: True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) error rates, Average Precision (AP), mean Average Precision (mAP), F1-measure (F1) and average Frames per Second (FPS) rate. Based on the above metrics, compliance with the requirements of the SL systems was assessed. Also, a comparison with modern detection algorithms based on Convolutional Neural Networks (CNNs) is performed.

Chapter 1

The problem statement. Existing detection methods

The chapter introduces the concept of an intelligent lighting system and its requirements for the object detection method. Topical research related to the problem of object detection has been completed that includes an analytical study of conventional methods and CV algorithms. The detection methods are analyzed, taking into account the specifics of the detection task. As a result of the analysis, a selected theory is justified, considering the system's requirements. The required vertical ($VAOV_{min}$), horizontal ($HAOV_{min}$) camera angles of view, and the required output frame rate of the algorithm (FPS_{req}) have been calculated. Performance comparison of existing CV algorithms on devices with limited computational resources is performed in the frame of the analytical study. Background Subtraction (BS) studies are performed to find the most appropriate approach for fast segmenting nighttime datasets. The materials presented in the first chapter were published in [6]–[10].

1.1 A smart lighting concept

One of the developing areas in an intelligent city concept is smart lighting systems. Such systems aim to solve the problem of efficient energy consumption, light pollution reduction, and remote monitoring and control via integrating information and communication technologies.

Innovative lighting systems have several advantages over conventional ones. While conventional systems illuminate an area with a constant brightness level regardless of traffic or weather conditions, smart lighting allows dynamic configuration of the illuminating area size and related parameters depending on external events.

The proposed detection method has been developed in the context of the SmartLighting project [1], [2], Koethen, Germany. Each lamp of such a system is a node equipped with a single-board computer, a detection module, and an interface for wireless data transmission. After detecting an object, a group of neighboring lamps is switched on in the direction of the object's movement. The speed and vector of the object are determined due to the data exchange between nodes, and the necessary dynamic lighting zone is provided (Figure 1.1). This approach saves up

to 80 to 90 % of electricity on street lighting.

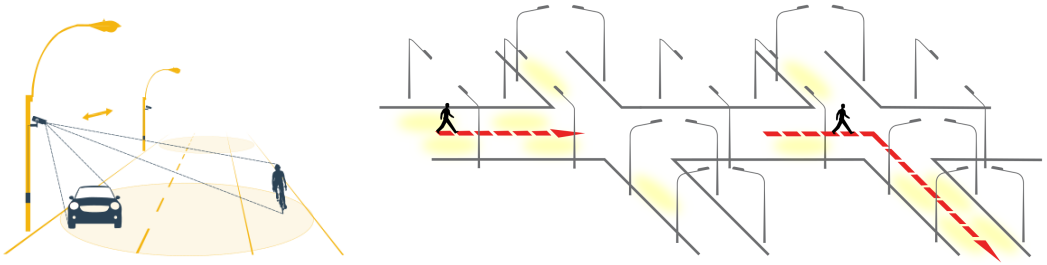


Figure 1.1: The scheme of the SmartLighting system

Several companies are currently developing and producing intelligent lighting systems, including Twilight, Trilux, Leipziger Leuchten, and eSave. Available on the market solutions for event-based lighting have limitations regarding possible operating scenarios, mainly due to their object detection method (typically, passive infrared). The constraints to effective detection range make it problematic to detect an object in advance that is relevant for streets and crossroads, where the size of a control area is 15 to 25 m. Moreover, restrictions concern the type (only pedestrians and cyclists) and the speed of the object being detected.

1.2 The smart lighting system requirements for the detection method

The detection method used in the SL system must meet several requirements to ensure that the lighting system performs at its stated efficiency:

1. Detection of objects most expected on a pedestrian or roadway and in need of illumination. Target objects are **pedestrians**, **cyclists**, and **vehicles**. The SL can use the information about the object type to calculate the required light area, for example, provide a larger light area or specific light intensity for vehicles.

2. Detection of objects moving at **speeds** up to 60 km/h, at distances up to 25 m. Speed objects are limited based on the maximum allowed speed within the city limits [11], while the distance is determined by the expected size of road infrastructure elements in the suburban area: 2 to 4 vehicle traffic lines and 2 pedestrian sidewalks [12], [13].

3. **Detection accuracy** is comparable to state-of-the-art detection methods. The SL is more tolerant of false negative errors because they can be compensated by true positive detection in subsequent frames. The false positive errors lead to excessive energy spending on lighting, while reducing false negative errors improves the timeliness of illumination.

4. **Low computational requirements** of the detection algorithm due to using low-cost and low-power computers (Raspberry Pi, Orange Pi, etc.) as processing devices.

5. **Spatial object coordinate determination** is required for zoning lighting areas, particularly for lighting objects moving at a given range of distances.

6. **Sensor position** on the lighting pole in the height range of 2.5 to 12 m. The value of the lower limit of the range is due to anti-vandal measures in urban environments, while the upper limit is limited to the height of streetlight poles.

7. **Modifiability** in case of additional functionality. Possible modifications include subsystems for tracking an object and estimating its speed, as well as counting the number of objects to collect information about the intensity of traffic.

1.3 Conventional methods

In this paper, conventional methods refer to methods used to detect objects long before the computer vision era. In conventional approaches, the detection decision is made by an electronic circuit or microprocessor logic based on signals from a primary transducer. These methods found wide application due to the reliability and ease of operation of such devices. Conventional technologies for detecting moving objects are typically based on infrared, ultrasonic, and radio wave methods [14].

Ultrasonic (US) methods can be divided into *active* and *passive* detection approaches. *Active ultrasonic* methods use two detection principles. US sensors based on the application of the Doppler effect evaluate the frequency shift between the sent and reflected signals that occurs when an object moves, allowing inferring the size and speed of an object. Another type of US sensors is based on recording the amplitude and propagation time of the signal reflected from the object. Amplitude is used to identify the received signal by comparing the original and received values, while the propagation time allows calculating the distance to the object. Sensors of this principle have found wide application in robotics [15], where the robot's orientation in space can be done by ranging.

Active US sensors are non-contact, have a short response time (1 to 100 µs), and are very sensitive and reliable [16]. The detection logic is implemented by an electronic circuit, for example, based on a comparator, and therefore, object detection does not require high computational costs. Using an array of networked US rangefinders can detect the presence and track the trajectory of objects in the room [17].

The US method is applicable for object classification. The most apparent approach is estimating a passing object's height by US ranging. In the work of R. Stiawan et al.[18], vehicle detection and classification are performed by measuring the unoccupied height of the vehicle in the room. In a study by T. Damarla et al. [19], the classification between humans and horses is done by analyzing the micro-Doppler effect [20] observed in the reflected signal. The different radial velocities of human and animal limbs allow the classification of these objects using machine learning. Classification can also be done by a neural network using the shape of the reflected signal in the time domain as features for classification [21].

A disadvantage of US sensors is the influence of environmental conditions, such as temperature change and air turbulence, on the detection accuracy [22], [23]. Distance error correction mechanisms have been proposed to minimize this factor; for example, in the work of M. Paulet et al. [24],

the error is corrected depending on the current temperature. The detection range of commercial and widely available US sensors remains limited [17], [24], resulting in their indoor usage, where the control area does not exceed 10 m [25]. That includes security and alarm systems, where ultrasonic sensors can register even small movements of objects [26]. Another disadvantage is the influence of rotation/tilting angles of reflecting object surface on the detection accuracy; in particular, an undesirable signal reflection angle can lead to FN detection errors. Additionally, some materials can disturb detection accuracy [27].

In contrast to active US detection methods, *passive methods* involve analyzing the pattern of vibrations in the US range, which occur when the object moves. Periodic vibrations corresponding to a particular gait allow us to identify and classify a moving pedestrian. A study by A. Ekimov et al. [28], [29] proposed recognizing a pedestrian's gait by analyzing the ultrasonic signature in the frequency domain, providing a detection range of up to 17 m. However, passive US methods are poorly applicable outdoors, where the chaotic movement of multiple objects at different distances is assumed, leading to a low signal-to-noise ratio.

Infrared (IR) detection methods include *passive* and *active* detection approaches. Devices based on the *passive-infrared* method use the pyroelectric effect to detect the IR radiation of an object in the far spectral range (8 to 14 µm). In the most common implementation of such sensors, the IR radiation of a moving object is alternately (due to the motion of the object) focused by the Fresnel lens segments, which creates voltage changes at the output of the pyroelectric cell. The use of Fresnel lenses is also determined by the necessity of pulse focusing of infrared radiation on the pyroelectric element because the output level of the element decreases when the emission is constant. Passive IR sensors are widely used for indoor and outdoor object detection due to their cheapness, unpretentiousness, ease of use, and low power consumption [30].

The main limitations of such sensors are the short detection range up to 10 to 12 m [31]–[35], high FP error rate due to changing environmental conditions (due to temperature drift on the pyroelectric cell), and high FN error rate when objects move too slowly/rapidly. Slowly moving objects cause minor voltage disturbances that are difficult to distinguish from ambient temperature variations [26]. Besides, applying the passive IR sensors with the aim of object classification is problematic due to their operation principles.

To solve the problems of existing sensors, M. Kastek et al. [36] propose a particular structure of a passive IR sensor equipped with two pyroelectric elements. Appropriate sensors [37] compensate for errors arising from varying illumination and temperature. The proposed signal analysis method [38] allows the signal of interest to be distinguished at low signal-to-noise ratio values, making it possible to detect pedestrians indoors at distances up to 140 m.

In addition to pyroelectric elements, an array of thermobatteries, each consisting of series-connected thermocouples (for example, Panasonic's GRID-EYE implementation [39]), can be used to convert thermal radiation. The averaged thermal image of the room background obtained by such a sensor allows segmentation of a moving person by subtracting the following thermal images from the background [40]. Bicubic interpolation is used to locate and track objects more accurately on low-resolution frames (8×8 cells in the case of GRID-EYE). Thermal imaging, combined with computer vision techniques, also offers several ways to detect pedestrians. Object

areas can be extracted from a thermal image, depending on the object's temperature, or by applying a fusion of camera and thermal imager data. Such methods have relatively high accuracy, but they are sensitive to changes in ambient light and require an expensive thermal imager (compared to a camera) [41]–[44].

A typical implementation of the *active* IR method is based on interrupting the IR beam between emitter and receiver by an object. This method requires the source and receiver to be installed at opposite points of the control zone. In this configuration, the detection range of commercial active infrared sensors can be up to 80 to 100 m.

In a study by D. Noyce et al. [45], the active IR method is used for rangefinding to classify pedestrians, bicyclists, and vehicles. Several IR emitters and receivers are mounted perpendicularly above the control zone plane. When an object moves in the control zone, the reflection distance of the infrared radiation changes. Measuring the distance to the reflection surface allows for building a map of the object's height, length, and width. The object's speed is calculated given the time difference between the successive intersection of the first and i-th beams. This method's true positive detection probability was 92 % for pedestrians and cyclists [45]. It is worth noting that this method requires the device's calibration depending on its installation's angle and height.

Radio-wave (RW) detection methods use the same signal properties and physical effects as ultrasonic methods. However, the signals of the RW methods are in the ultra-wideband (UWB) range and represented by short pulses. Due to the relative technical complexity of generating and processing radio signals, such sensors are mainly used in high-performance systems such as drones, game industry devices, autonomous vehicles, and alarm systems [30], [46].

The automotive applications gather spatial frequency and amplitude maps to detect moving objects. Each cell of the map corresponds to a segment of the control area, assigned to a value of frequency and intensity of the reflected UWB pulses. These maps are used to extract the characteristic features of a pedestrian in the work of A. Bartsch et al. [47], in particular, the object's size, shape, and frequency histogram (Doppler spectrum), which allow the classification of objects based on the weight function without machine learning appliance. As a result, the detection accuracy was 95.3 % for pedestrians at distances up to 35 m [47]. However, this detection system has a high cost because the radar must have a high spatial and frequency resolution.

1.4 Computer vision methods

Computer Vision (CV) methods aim at extracting information from an image and interpreting it. The image contains real-world data converted into some form: an n-dimensional image from one or multiple cameras, a video sequence, a thermal image, etc. Data interpretation may consist of the detection, classification, and tracking of objects in order to make an automated decision. Today, computer vision systems are widely used practically, for example, in automotive, robotics, or video surveillance.

1.4.1 The early era of computer vision

At the beginning of the CV era (from 2000 to 2010), the model's computational complexity significantly impacted detection speed due to the low performance of computers. Classical object detection methods include two-stage Machine Learning (ML)-based classifiers. In the first stage, different features are extracted from the image. After that, the features are classified by a predefined classifier. The classification quality depends on the type of extracted features and the used classification approach.

Viola-Jones method is one of the earliest face detectors based on extracting features from an image area using Haar features [48]. Haar features are adjacent rectangular areas (Haar primitives, Figure 1.2) superimposed on an image segment at multiple positions and scales. Where an image segment — is the area of the image bounded by the scanning window. The feature is a difference between pixels' intensities covered by the white areas and the intensities under the black areas. Each primitive reflects possible characteristics of the object: the presence of corners and lines between dark and light areas of the image. For example, the areas around the eyes are the darkest for a human face, while the cheeks and forehead are relatively light.

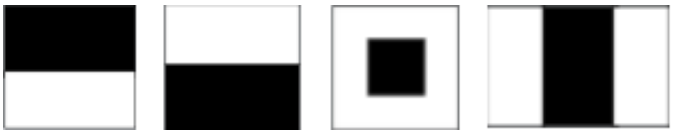


Figure 1.2: Haar features used in the Viola-Jones method [49]

The image is converted into an intermediate integral form to speed up the calculation of features in the Viola-Jones method. Each point of the integral image is calculated as the sum of pixels to the left and above the considered point [50]:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y'), \tag{1.1}$$

where $ii(x,y)$ — integral image;

$i(x',y')$ — the original image;

The selection of the most informative features and training of classifiers is performed using the AdaBoost adaptive boosting algorithm. AdaBoost builds a strong classifier from a linear combination of weak weighted classifiers. Individually, each weak classifier has a low classification accuracy, slightly higher (>0.5) than the random decision. The weak classifier is a function comparing the Haar feature value to some threshold value that separates the training samples [51]:

$$h(x,f,p,\theta) = \begin{cases} 1, & \text{if } pf(x) < p\theta \\ 0, & \text{otherwise} \end{cases} \tag{1.2}$$

where f is– the value of the Haar feature in the considered image region x;

θ — threshold value;

p — polarity indicating the direction of the inequality sign.

During the iterations of the boosting training, the best threshold and polarity minimizing the error are chosen for the weak classifier. The best feature-threshold pair is then selected, leading to a significant reduction in the number of features considered. At the end of each iteration, the weights of the weak classifiers are updated.

Classification of the image region is performed using a cascade of strong classifiers, where each strong classifier handles a group of features. Thus, the area classified as a *not targeted* object is not considered by the subsequent strong classifiers of the cascade, which increases the detection speed. Each subsequent classifier is more complex than the previous because it processes more features, while increasing detection accuracy by minimizing FP errors.

A distinctive feature of this method is the ability to operate on low-performance devices in real-time, with a processing rate equal to 15 FPS [48]. The typical problems for this method are the sensitivity to the illumination of the image scene and the noise component. Also, a complex background texture significantly reduces the method's accuracy [52].

This method showed high accuracy for frontal face detection of 77.8 %, with 5 FP errors (out of 149 faces) on the MIT [51] dataset. However, the algorithm's accuracy decreases significantly when the faces are rotated relative to the camera. There is a problem of feature dispersion when objects of the same class appear under different object angles and illumination. In the case of applying this method for multi-class classification, the problem of feature similarity between objects of different classes, aggravated by varying scene parameters and textures of object material, may appear. Attempts to use this type of pedestrian detection result in a significant error rate, making its use for such tasks impractical. For example, in pedestrian detection on the CALTECH dataset, the *error rate* was 94.7 % [53].

Integral Channel Features (ICF) detector proposed by P. Dollár et al. in 2009 [54] with the improvement in [55], [56] presents an extension of a Viola–Jones detector. In addition to the Haar features, ICF can use gradients, filtered channel features [57], and convolutional channel features [58]. J. Sochman proposed an ICF-like solution using Wald's sequential probability ratio test and AdaBoost algorithm for decision-making [59]. R. Juránek et al. adapted an ICF detector for video steaming tasks [60], [61]. ICF-based detectors are still improving nowadays [62].

Histograms of Oriented Gradients (HOG) was considered by N. Dalal and B. Triggs, who in their work [63] studied the parameters of HOG features for pedestrian detection. In this method, gradient directional histograms of an image segment act as object features. The gradient vector for each image pixel shows the change in pixel intensity along the horizontal and vertical axes.

The gradient of the image function is a vector of its partial derivatives [64]:

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \\ \frac{\delta f}{\delta y} \end{bmatrix} = \begin{bmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{bmatrix}, \tag{1.3}$$

where $f(x,y)$ is a pixel intensity function of the image;
$\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}$ — derivatives of x and y.

The sliding window bounding the image segment is divided into n cells. For each cell, g values and θ gradient directions are calculated (expression 1.4). For the pedestrian detection problem, the proposed sliding window size is 64×128 px [63], which corresponds to an elongated geometric shape of the pedestrian along the vertical axis.

$$\begin{aligned} g &= \sqrt{g_x^2 + g_y^2} \\ \theta &= \arctan(\frac{g_y}{g_x}) \end{aligned} \tag{1.4}$$

In discrete form, the gradient is calculated by convolving the image segment and kernel of the form $[-1\ 0\ 1]$ — when convolving the kernel and image along the x-axis and $[-1\ 0\ 1]^T$ — along the y-axis. When plotting the n_i cell histogram, each pixel contributes a weighted gradient value to the histogram of directions from 0 to 180°. Normalization of the gradient value on a sliding window region of n cells is performed to minimize the effect of image contrast and brightness on the feature. The HOG feature is the concatenated histograms of all n cells in the sliding window region. Machine learning methods classify the resulting vector of histograms. The Support Vector Machine (SVM) was used to classify the feature vector in the original work [63].

N. Dalal and B. Triggs showed that HOG features could reduce the *FP error rate* by more than one order of magnitude compared to existing Haar features [63]. Although HOG can be used to detect many classes of objects, the method was developed to solve the problem of pedestrian detection, being one of the most accurate approaches for this purpose for a long time [65].

Using a sliding window in this method requires high computational cost, making it poorly applicable for low-performance real-time computers. The predefined shape of the sliding window corresponding to the expected object complicates the multi-class classification task, as it requires multiple scanning of the image with windows of different shapes, significantly affecting the system's performance. In addition, HOG features are sensitive to the object's rotation in the image [66].

1.4.2 Methods based on artificial neural networks

The development of GPU-based computing contributed to the widespread usage of Deep Learning (DL) algorithms for object detection. In practice, they outperform conventional ML-based solutions in terms of detection accuracy; however, they require substantial resources for learning and are usually slower in recognition.

Convolutional Neural Networks (CNNs). CNNs are the most suitable solution for object detection among all existing neural network structures. Various CNN modifications have been developed during the last two decades [67]. In contrast to conventional methods of computer vision, a distinctive feature of the methods based on artificial neural networks is the principle of feature formation. Considered Haar and HOG features are determined in advance by a human at the design stage of the detection method, which limits the amount of information extracted from the image. The CNNs build features during training independently by selecting the neuron weights (convolutional kernel values) by minimizing the loss function. CNN detection, in general, involves two steps: extraction of object features by CNN and subsequent classification of features

by machine learning (e.g., SVM or neural network).

The CNN architecture typically includes the following layers:

- input layer — source image;
- convolutional layer — a set of feature maps (filters) obtained by convolution of the input layer and kernels with different weight coefficients. The shape of the kernel is determined depending on the structure of the input layer. For a single-channel image, the shape is a 2D array, whereas, for an RGB image, the kernel has a cubic shape. The convolutional layer includes an activation layer that transforms the result of each convolution through a non-linear activation function, such as ReLU [68];
- subsampling layer — feature maps compacted by finding the maximum, average, or L2-norm in a cell combining neighboring pixels;
- fully-connected layer is a classifier associating the previous convolutional layer (feature set) with the output classes. Thus, the filter weighting factor of the convolutional layer determines the class of the object.

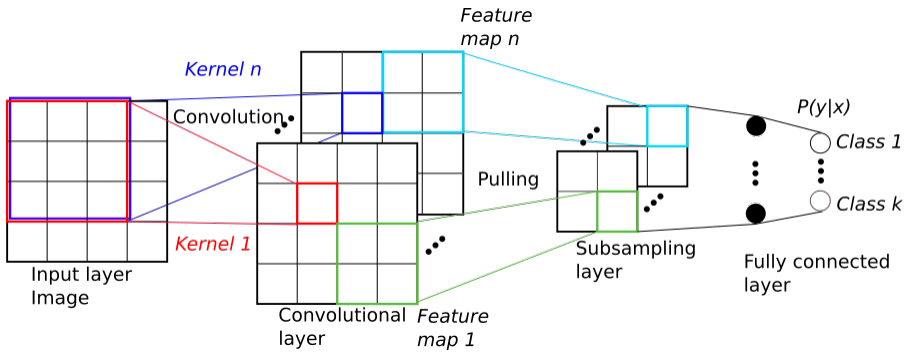


Figure 1.3: Example of an elementary convolutional neural network architecture

The first convolutional layer of the network contains primitive image features, such as angles, lines, circles, etc. As moving deeper into the network, more generalized characteristics of the image are formed by convolving the feature maps and kernels. In a real CNN, the convolutional and subsampling layers are repeatedly alternated to decrease the dimensionality and increase the level of abstraction of feature maps. For example, the AlexNet [69] network consists of 3 convolutional layers, 2 subsampling layers, and 3 fully-connected layers. The fully-connected layer determines which high-level feature maps correlate with a specific type of object.

The training algorithm for the CNN can be the gradient descent algorithm [70]. When training the network, the convolutional kernel weights are set randomly, leading to high loss function values. The loss function can be expressed as the root mean square error:

$$J = \sum \frac{1}{2}(y - x)^2, \tag{1.5}$$

where x — the predicted network value;

y — the reference value.

Backpropagation is performed using the derivative of the loss function. The weighting coefficients most affecting the value of the loss function are determined. The weight coefficients are then updated in the direction of the gradient.

In contrast to classical neural networks, where a neuron processes each pixel with an individual weighting coefficient, the convolution method allows using uniform kernel weights for the whole image (regardless of position). Feature extraction and classification by CNNs allow for achieving high accuracy compared to conventional approaches. AlexNet CNN achieved an *error rate* of 0.37 (with 0.1 FP detection per image) on the Caltech Pedestrian Dataset [71], whereas the conventional Viola-Jones and HOG methods have this rate on the same dataset equal to 0.95 and 0.68 [72].

The disadvantage of CNN is the need for empirical configuration of the network architecture and its parameters (number of layers, kernels, subsampling function, etc.), which significantly affects the performance and efficiency of the network. In addition, a classical CNN with AlexNet-like architecture requires a fixed resolution of the input image since the final classifier takes a fixed-length feature vector [73] as input. The result of CNN detection is the degree of confidence (probability) of object presence in the image; thus, CNN solves the problem of classifying images, but not the localization of the object in the image. The sliding window technique can detect multiple objects of different classes in the same image, allowing for localizing the object [74]; however, this approach is characterized by redundant calculations (extraction of features in all possible sliding window positions), which reduce the detection speed.

The main disadvantage of CNN is its high computational complexity. There are implementations of detection systems where a CNN has been combined with the background subtraction techniques to speed up segmentation [5]. However such approaches require a GPU to operate in real-time. An increase of the CNN detection frame rate is possible by applying a hardware acceleration unit (e.g., Neural Compute Stick [75]): for a single-board computer Raspberry Pi 3 B+, the AlexNet video processing speed was 4 FPS [76].

Currently, networks such as GoogleNet [77], VGG [78], ResNet [79], and MobileNet [80] are widely used, which provide a lower error rate [81] than AlexNet; however, AlexNet outperforms them in the detection speed [82].

Region-based Convolutional Neural Network (R-CNN). Unlike conventional convolutional neural networks, where a sliding window defines the Regions of Interests (ROIs), the main difference of this network type is the algorithm for finding regions of interest. The region-search algorithm pre-segments the image into regions containing the object of interest with the highest probability, reducing the number of regions compared to the sliding window technique. R. Girshick et al.[83] used Selective Search [84] as a search algorithm for R-CNN, which performs segmentation by finding similar features of regions in different color spaces (color, texture, size, and degree of homogeneity). This search algorithm has a high segmentation *recall* of (99 % on the Pascal 2007 TEST dataset [84]).

In the next stage, the segmented regions are scaled to the same size, regardless of the initial size and aspect ratio. The result of CNN processing is a feature vector with the same length for all regions of interest. The resulting vectors are independently processed by the linear regression

blocks of the bounding rectangle and SVM classification. The linear regression model produces the numerical coordinates of the bounding rectangle in the image, whereas the classification is performed by n binary SVMs, where n is the number of classes.

The R-CNN algorithm showed a 30 % increase in mAP on the PASCAL VOC dataset in 2012, compared to the best previous algorithm [83], [85]. One disadvantage of the method is the complexity of training because it requires training the algorithm components independently: convolutional network for feature extraction, linear regression for refining the coordinates of the bounding rectangle, and SVM classifiers. The Selective Search algorithm segments 2000 potential regions of interest per image, which is redundant. Thus, processing one image using GPU requires about 14 s [65], making it impossible to apply the method in real-time.

Subsequent modifications of the method — **Fast R-CNN** [86] and **Faster R-CNN** [87] aim to speed up processing by optimizing the method of ROIs finding. Unlike R-CNN, where features are extracted from each ROI independently, Fast R-CNN builds a feature map for the entire image once, into which potential ROI is projected. The network layer (*ROI pooling*) converts the regions of interest into fixed-size regions fed into the fully-connected network layer. The ROIs are searched by the Selective Search algorithm, similar to the R-CNN. Fast R-CNN performs feature extraction, classification, and refinement of the bounding box coordinates by a single network: the SVM classifier is replaced by a Softmax layer (multidimensional logistic function), and the linear regression block is located parallel to the classification block. With a slight improvement in detection accuracy (66 % vs. 62 % mAP compared to the original R-CNN), Fast R-CNN enabled GPU-based image processing at 3 FPS [86].

In Faster R-CNN, in order to increase the performance, the Selective Search algorithm is replaced by a separate neural network (Region Proposal Network) following the last convolutional layer of CNN. The Region Proposal Network "slides" the window over the feature map, suggesting rectangular regions for each window position, where the rectangular regions contain objects of interest with the highest probability. The selected regions of interest are scaled by the ROI pooling network layer, which then performs classification and refinement of the coordinates of the bounding rectangle. While maintaining the detection accuracy of Fast R-CNN, Faster R-CNN has increased the processing speed to 5 FPS using GPU [87].

Single Stage Detectors (SSDs) are represented by algorithms such as You Only Look Once (YOLO), SSD, RetinaNet, etc. A distinctive feature of such algorithms is that the extraction of YOLO features is performed once per image [88]. The algorithm divides the image into a grid of rectangular cells. The probabilities and coordinates of the bounding rectangle are computed for each cell simultaneously. The cell is assigned to the object and contains its parameters when the center of the object matches that cell. Independent logistic classifiers perform multi-label classification. This approach enables the algorithm to make predictions faster and computationally lighter than the two-stage detectors.

A distinctive feature of algorithms based on single-stage detectors is high speed: the YOLO algorithm achieved a processing speed equal to 45 FPS, using GPU, with an accuracy of 63 % mAP on the PASCAL VOC data set 2007 [88]. As a consequence of analyzing the entire image rather than independent regions as in R-CNN, single-stage detectors take contextual information such

as background objects into account.

However, a grid structure limits the density and size of objects in the frame since each cell can only correlate with a single object. Subsequent improvements to the YOLOv2 [89] and YOLOv3 [90] algorithms partially solve this problem by improving accuracy and maintaining high detection rates.

Despite the wide variety of proposed neural network configurations, achieving the highest accuracy and detection speed is still challenging. Two-stage detectors usually have higher accuracy, while one-stage are typically faster. The trade-off can be found using the optimal strategy proposed in [91].

1.5 Requirements for camera parameters and algorithm performance

Requirements for camera parameters such as Vertical Angle of View (VAOV) and Horizontal Angle of View (HAOV) are imposed to ensure that the sensor covers the entire control area. Also, the required performance of the algorithm depends on these camera parameters. The vertical camera angle of view limits the size of the monitoring area along the Z axis and the maximum detection range (Figure 1.4a).

In Figure 1.4a, the two objects, o and o', are located at the minimum and maximum distances of the monitoring area from the AC lighting pole. Calculating the required $VAOV_{min}$ considers the outer vertical points of the object o' in the camera's field of view at the maximum distance AD_2 from the camera. The value of AD_2 depends on the blind spot size AD_1, which is determined by the angle α between the pole AC and the lower boundary of the camera field of view CD_1. Thus, the value of minimal vertical angle of view $VAOV_{min}$ is defined as:

$$\begin{aligned} VAOV_{min} &= \arctan\left(\frac{AD_2}{AC - D_2H'_o}\right) \\ AD_2 &= AD_1 + D_1D_2 \\ AD_1 &= AC \cdot \frac{\sin\alpha}{\sin\left(90^\circ - \alpha\right)} \end{aligned}, \tag{1.6}$$

where D_1D_2 is the length of the control area defined by SL system requirements;

AC — the minimal possible height of camera installation;

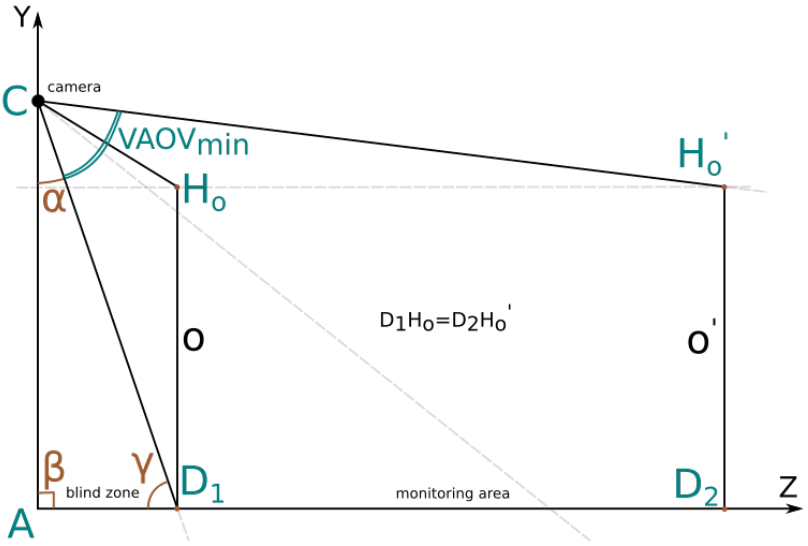
$D_2H'_o$ — maximal height of the object o'.

The minimal horizontal camera angle of view $HAOV_{min}$ is estimated from $VAOV_{min}$ and the aspect ratio:

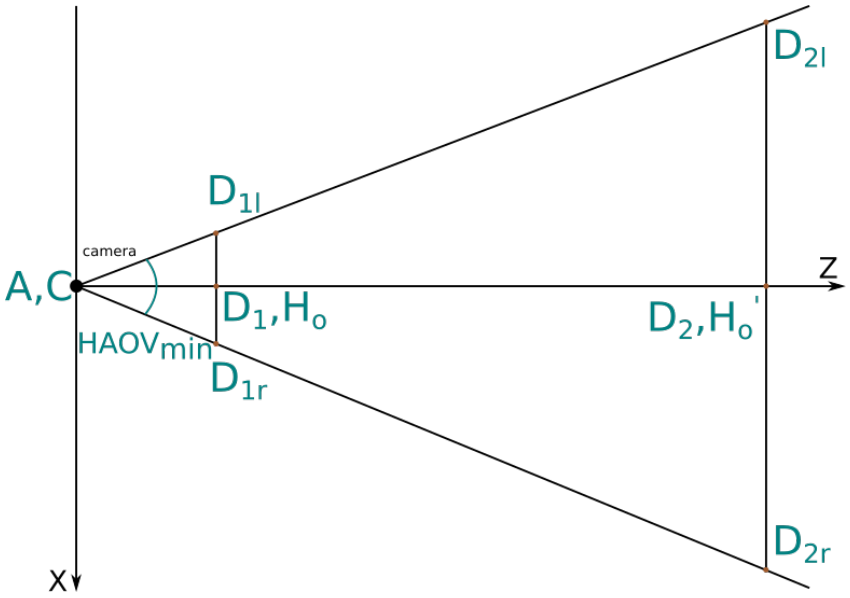
$$HAOV_{min} = VAOV_{min}\frac{w_{img}}{h_{img}}, \tag{1.7}$$

where w_{img}, h_{img} — frame width and height.

The minimum acceptable frame rate FPS_{req} is determined from the time for a point moving at constant speed v_o to pass the distance $D_{1l}D_{1r}$ (Figure 1.4b) in the camera field of view, at the minimum distance from the camera CH_o (Figure 1.4a). Thus, FPS_{req} is expressed as:



(a) $VAOV_{min}$, side view



(b) $HAOV_{min}$ top view

Figure 1.4: 3D scene parameter projections

$$FPS_{req} = \frac{v_o}{2\tan(\frac{HAOV_{min}}{2})\sqrt{(AC - D_1H_o)^2 + AD_1^2}} \tag{1.8}$$

The following scene parameters are used to calculate FPS_{req} in the worst-case scenario:

- $D_1D_2 = 25\,\text{m}$, based on SL system requirements for maximum detection range;
- $AC = 2.5\,\text{m}$, based on SL system requirements for the minimum height of the detection device;
- $D_1H_o = D_2H_o' = 2\,\text{m}$, selected as the maximum height of the detection object. The geometric parameters of detecting objects are discussed in Section 2.2;
- $\alpha = 0°$, due to the possible adjacency of the control area to a street lighting pole: in some configurations, the pedestrian/traffic zone starts directly from the lighting pole;

- $\frac{w_{img}}{h_{img}} = \frac{4}{3}$, chosen from a range of aspect ratios used by modern cameras ($\frac{4}{3}$, $\frac{16}{9}$, $\frac{16}{10}$, etc.) as the worst case scenario where $HAOV$ has the lowest value;

- $v_o = 16.6\,\text{m/s}$, based on the SL system requirements for maximum object speed;

Given the scene parameters in the above-mentioned extreme case, the following requirements are imposed on a camera and processing performance:

- The required frame rate should be not less than $FPS_{req} \approx 10$;

- The vertical camera angle of view should be not less than $VAOV_{min} \approx 90\,\text{deg}$;

- The horizontal camera angle of view should be not less than $HAOV_{min} \approx 120\,\text{deg}$.

Applying wide-angle camera lenses allows for achieving the calculated $VAOV_{min}$ and $HAOV_{min}$. An appropriate correcting algorithm may be required to compensate for distortions caused by the wide-angle lenses.

1.6 Metrics for accuracy and performance evaluation

Determination of **Basic Characteristics** is based on a comparison of the reference data and results of model classification. The preparation of the data for classification involves the selection of regions of interest by a bounding rectangle.

The Jaccard index, better known in computer vision as Intersection over Union (IOU), is used to determine the similarity between the detection result and the reference. This coefficient is defined as the ratio of overlap between the reference and detected areas to their total area [92]:

$$\text{IOU}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{1.9}$$

where A, B — reference and detected rectangular areas.

The **Confidence Score** is a probabilistic prediction of the classifier, used to determine the basic characteristics of a classifier. The confidence score is the probability that a detected region characterized by a set of features corresponds to a particular object class.

The basic characteristics of the model are defined as:

- True Positive (TP) decisions — objects classified as target objects while being target objects. A solution is marked as TP when the following set of conditions are satisfied:

$$\text{TP}(A, B) = \begin{cases} 1, & \text{if } (IoU(A, B) > T_{IoU}) \wedge (P_c(B) > T_{Pc}) \\ 0, & \text{otherwise} \end{cases}, \tag{1.10}$$

  where T_{IoU} — the threshold value of area similarity, chosen depending on the detection algorithm used and optimized a posteriori based on the obtained *precision* and *recall* of the model. In general, the value of T_{IoU} varies in the range from 0.25 to 0.8 [93];

  P_c — confidence score predicted by a classifier;

Table 1.1: Binary confusion matrix

		Predicted value	
		Class 0	Class 1
True value	Class 0	TN	FP
	Class 1	FN	TP

T_{Pc} — confidence threshold, generally equal to 0.5, optimized depending on the obtained model accuracy metrics and datasets peculiarities. The standard threshold value, taken as 0.5, may not be optimal in case of a class imbalance (significant quantitative difference between instances of different classes).

- True Negative (TN) decisions — objects assigned to any other class, at that, these objects are actually objects of another class;
- False Positive (FP) decisions — type 1 errors. Objects of a non-target class classified as a target class;
- False Negative (FN) decisions — type 2 errors. Ignored objects, which are the target objects.

A **Confusion Matrix** can be used for a visual representation, combining the model's basic characteristics. An example of the confusion matrix structure for the case of binary classification is presented in Table 1.1.

A confusion matrix can be constructed for the multi-class classification problem, where the diagonal elements correspond to the TP decisions for each class. *Precision*, *Recall*, and *F1-measure* can be calculated based on the basic characteristics.

Precision of a model is the proportion of correct predictions out of the total number of positive predictions [94]:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1.11}$$

Recall or sensitivity, in general, is the fraction of true predictions out of real positive decisions [94]:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1.12}$$

F1-measure (F1) is used for the weighted evaluation of precision and recall. The F1 is calculated as a harmonic mean between precision and recall [95]:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1.13}$$

Accuracy of a binary classifier is calculated using the basic characteristics of the model [95]:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1.14}$$

The **Precision-Recall curves** are used to explore the model given a set of possible classifier thresholds and evaluate the trade-off between precision and recall. The confidence score is compared with some threshold value to convert the value of the confidence score into a binary form. If the resulting confidence score for a class exceeds this threshold value, the object is classified as that class. Thus, iterating the threshold value from 0 to 1, the resulting precision and recall pairs are plotted. The precision-recall curves show adequate classifier performance on an imbalanced dataset compared to ROC curves. An example of the precision-recall curves for the perfect and real classifiers is shown in Figure 1.5.

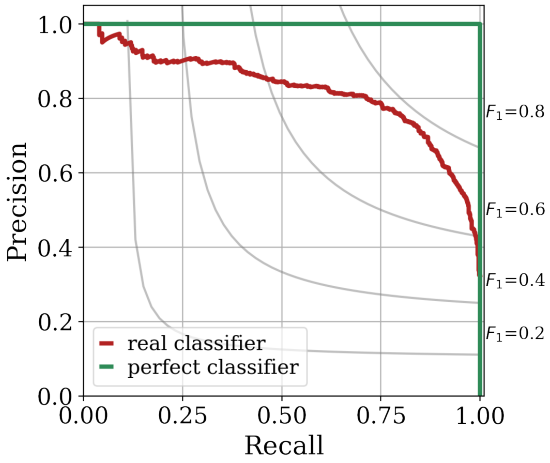


Figure 1.5: Example of precision-recall curves

Average Precision (AP) is a metric that numerically characterizes the precision-recall curve. Average precision summarizes the precision-recall curve as the average of the precision values obtained at each threshold, using the difference between recalls calculated on previous and current thresholds as a weighting factor. The value of this metric is equivalent to the area under the precision-recall curve and, unlike F1-measure, the average precision considers the characteristics of the classifier at different thresholds. The average precision of the classifier is calculated through the following expression [96]:

$$\mathrm{AP} = \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{r+1}$$

$$p(r) = \max_{r' \geqslant r} p(r')$$

where r — the calculated values of recall;

p — interpolated precision over recall values. Interpolation allows smoothing the curve in the presence of variations of precision.

Mean Average Precision (mAP) allows for estimating average precision when multiple classes are present. The following methods of averaging are distinguished for calculating the mAP:

- *micro mAP* — AP is calculated based on the total number of characteristics (TP, FP, FN) for all classes;

- *macro mAP* — AP is determined for each class individually; after that, the average value is found;

- *weighted mAP* — AP is determined for each class individually; after that, a weighted average is found, where the weight of the class is determined by the number of TP for it, so the dominant class has the most significant influence.

The choice between *micro mAP*, *macro mAP*, or *weighted mAP* depends on the specific task and the properties of the data sets. For example, weighted mAP or micro mAP can hide the accuracy of classes presented with low frequency [97], which is often the subject of consideration. In recent years, mAP has become a standard accuracy evaluation metric for classifiers on open data sets PASCAL VOC [98], ImageNet [99], COCO [100], etc.

1.7 Performance comparison of the computer vision methods

Based on the SL system (Section 1.5) requirements for using low-cost single-board computers as processing devices, the Raspberry Pi platform was chosen to represent low-performance embedded microcomputers to compare FPS rates (Tables 1.2 and 1.3). *Precision metrics* allow estimating the relative accuracy between methods only within a particular source because different authors use unequal metrics to estimate algorithm accuracy and datasets of varying nature and complexity. Section 1.6 presents a detailed discussion of the accuracy metrics of the detection algorithms (FP coefficient, Accuracy, and mAP).

According to Tables 1.2 and 1.3, the highest FPS is achieved on Raspberry Pi 3B+ by a lightweight version of the YOLO algorithm — YOLOv3-tiny, using hardware acceleration of neural network calculations by NCS device (YOLOv3-tiny and NCS). This frame rate does not meet the SL system requirements for algorithm speed (required frame rate — 10 FPS). In addition, the cost of the accelerator device is several times higher than the Raspberry Pi 3B+, which makes this approach uneconomical for the SL system.

A possible way to increase the speed of detection algorithms is to reduce the number of regions of interest and their size through pre-segmentation. Pre-segmentation can be performed through signal measurement and comparison in the time domain in order to distinguish moving objects from the background signal. Methods for such segmentation based on BS are discussed in Section 1.8. This acceleration approach has been applied to Deep Learning methods (SSD-MobileNet and BS) and conventional computer vision methods (HOG and BS). The achieved frame rate of the combined approaches is significantly higher than that of the original methods (HOG and SSD-MobileNet); however, the obtained values also do not meet the SL requirements.

Also, it is worth noting that the methods based on DL have high accuracy. In work [3], algorithms based on YOLO (SSD-GoogleNet and L-CNN) have several times lower FP error rates compared to the conventional methods (Viola–Jones and HOG).

Table 1.2: Characteristics of the detection algorithms running on a Raspberry Pi 3B+

	Method	FPS	Accuracy metrics, %		Dataset	Resolution, px
[3]	Viola–Jones	1.82	FP	26.3	ImageNet, VOC07	224 × 224
	HOG	0.30		14.5		
	SSD-GoogleNet	0.39		5.3		
	L-CNN	1.79		6.6		
[101]	Viola–Jones	1.88	Acc.	99.0	Faces	320 × 240
[4]	SSD-MobileNet	0.50	mAP	72.3	People	300 × 300
	YOLOv3-tiny	1.17		52.7		416 × 416
	YOLOv3-tiny and NCS	5.55		52.7		416 × 416
[102]	SSD-MobileNet	0.50	—		—	—
	SSD-MobileNet and NCS	3.50				
[103]	SSD-MobileNet and BS	2.90	Acc.	91.2	People	128 × 128
[104]	SSD-MobileNet	1.55	Acc.	93.0	People	224 × 224
	Tiny YOLO	1.05		67.0		
[105]	SSD-MobileNet	0.55	Acc.	85.0	—	300 × 300

FP — False Positive coefficient;
mAP — mean Average Precision;
Acc. — Accuracy

Table 1.3: Characteristics of the detection algorithms running on a Raspberry Pi 1B

	Method	FPS	Accuracy metrics, %		Dataset	Resolution, px
[106]	HOG	0.39	Acc.	65.0	CAVIAR	384 × 288
	HOG and BS	1.94		49.1		
[107]	HOG and BS	0.40	Acc.	83.5	Pedestrians	—
[108]	YOLO	1.89	Acc.	99.5	Faces	46 × 46
	Viola–Jones	2.36	—		—	—

Acc. — Accuracy

1.8 Image segmentation methods

The choice of the most suitable image/video segmentation method should be based on analytical and experimental studies of existing segmentation methods used to solve the problem of locating moving objects in the frame with the highest accuracy and speed.

Image segmentation groups pixels related to the object of interest by a set of features. Segmentation errors are expressed in excessive or insufficient clustering of the object pixels. The result of segmenting a single object (not partially obscured by obstacles in the original image) in the form of several areas is considered insufficient segmentation. A case of excessive segmentation is merging several different areas into a single ROI.

The segmentation task is an essential pre-processing step in many computer vision methods for object detection and classification [109] or used separately to group pixels by color, intensity, texture, etc. [110]. The choice of segmentation technique depends on the type, properties, quality, or nature of the input data in the time domain (static image or video stream), so the specific technique is chosen based on the requirements of the final application [111].

Static image segmentation techniques cluster image regions based on pixel properties without considering the information about the image changes in the time domain, while video stream segmentation techniques allow efficient detection of moving objects through a technique called Background Subtraction (BS). Considering that the lighting system requires the detection of moving objects, the object detection applying CV methods can be accelerated using BS algorithms. BS methods are more suitable for detecting moving objects because they do not require the high computing power of devices to work in real-time. Besides, applying BS methods minimizes the probability of segmentation of undesirable static objects of a scene that can lead to FP errors.

According to the given detection task, the selected BS method must satisfy the following requirements:

- detection speed of the method and low computational performance requirements for the processing device — given the requirements for the algorithm to operate in real-time on low-performance devices, the selected BS method must provide minimum computational costs. The performance can be evaluated by the FPS metric;

- high coefficient of algorithm accuracy and segmentation quality — can be evaluated with the metrics *F1-measure*, *precision*, and *recall*;

- stability of the algorithm to oscillatory changes of scene areas — the influence of cyclic motions, such as leaf fluctuations, water fluctuations, etc., should be taken into account by the background model and have the most negligible impact on the segmentation quality,

- the ability of the method to handle night scenes captured by a camera with a near-IR emitter.

1.8.1 Analysis of background subtraction methods

Based on the analysis of existing analytical papers [112]–[122], four BS methods were selected according to the criterion of the highest performance: *Frame Difference*, *Adaptive Background Learning*, *Single Gaussian*, and *Mixture of Gaussians*-based methods.

BS methods based on neural networks, such as simplified SOBS [123] or DSTEI [124], can achieve high-quality segmentation but, at the same time, are computationally expensive [121]. Therefore, the chosen method must compromise the accuracy requirements and the mathematical complexity affecting the processing speed.

BS methods generally include *initializing the background*, *detecting areas belonging to the foreground*, and *updating the background model* [112]. *Background initialization* — forming the background model based on a limited number of input frames. Initialization can be performed by numerous approaches, such as statistical or neural network-based approaches. *Detecting areas belonging to the foreground* — new frames are compared to the background model. Finding the difference by subtraction allows for selecting foreground areas. *Updating the background model* — the shooting scene in the real world changes over time. Thus, the background model must adapt to changing environmental conditions given the updating rate of the background model. Appeared objects that do not subsequently move must be gradually integrated into the averaged background model.

The mathematical description of the methods presented below is considered for single-channel images, as the images obtained from the camera at night are taken in grayscale. The considered BS methods use the same assumption that the observed video sequence is taken against a stationary background with a moving object in front of it. The assumption that the pixel intensity distribution of the moving object is different from the background is described by the following expression:

$$F_t(u,v) = \begin{cases} 1, & \text{if } d(K_t(u,v), B_t(u,v)) > T \\ 0, & \text{otherwise} \end{cases}, \tag{1.15}$$

where d is a function of the distance (difference) between the pixel intensity of the current frame and the background model;

u,v — coordinates of the pixel in the image;

T — threshold value is more important for dynamic scenes than static ones. It is chosen empirically, depending on the method used;

$F_t(u,v)$ — a function of the pixel intensity of the binary mask containing the moving object. At time t the values of this function are 255 and 0 otherwise;

$K_t(u,v)$ — the pixel intensity function of the frame from the video sequence at a particular point in time t;

$B_t(u,v)$ — a function of the intensity of the background pixels at a particular time t.

Frame difference is the simplest background subtraction method. To construct a binary mask, the distance function d_f is defined as:

$$d_f = |K_t(u,v) - B_t(u,v)| \tag{1.16}$$

In the case of *frame difference*, the background model is formed from the previous image as $B_t(u,v) = K_{t-1}(u,v)$.

This method is robust to rapid changes in the background scene, while remaining simple to implement. The ease of implementation makes this method particularly applicable to real-time systems [125].

Adaptive Background Learning involves using the arithmetic mean of the pixels of the preceding m frames to form the initial B background model (although a median can also be used) [126]:

$$B_t(u,v) = \frac{1}{m}\sum_{n=1}^{m} K_{t-n}(u,v) \tag{1.17}$$

The foreground objects are defined using this background model according to the expression 1.16. In order to adapt to the changed scene conditions, the background model is updated as follows:

$$B_t(u,v) = (1-\alpha)B_{t-1}(u,v) + \alpha K_t(u,v), \tag{1.18}$$

where α — is a constant update coefficient of the background model, which takes values from 0 to 1.

The relatively low computational complexity of the method allows using it on low-performance devices with a high frame rate. The ability of the method to modify the background model according to the current frame allows it to be used in scenes with dynamic backgrounds. The main disadvantage of this method is the need to select the coefficient of updating the background model according to the moving object's speed, size, and distance [121]. In addition, with large values of the updating coefficient, a "tail" behind the moving object may occur [127].

The method using **one Gaussian** (Gaussian probability distribution) involves building a background model for each pixel using a probability density function that is updated with the arrival of new frames [128]. In this case, the pixel with a low probability of being in the background model is treated as belonging to a moving foreground object.

The distribution density function of each pixel is described by the mean μ and the variance σ^2 for the pixel p over m previous frames. At the first moment, μ and σ^2 are equal to some standard values: the value of μ is equated to the pixel value of the first frame, while σ^2 is calculated using nearby pixels.

The following expressions are used to update the background model:

$$\begin{gathered} \mu_{t,p} = \rho k_{t,p} + (1-\rho)\mu_{t-1,p} \\ \sigma^2_{t,p} = l_p^2\rho + (1-\rho)\sigma^2_{t-1,p} \\ l_p = |k_{t,p} - \mu_{t,p}| \end{gathered}, \tag{1.19}$$

where $k_{t,p}$ is the pixel intensity value p at a particular time t;

ρ — a constant determining the window size for filling the probability density function;

l — the distance between the average pixel value and its actual value.

Thus, the distance function between the background model and the new frame checks whether the pixel belongs to the confidence interval of its distribution:

$$d_g = \frac{(|K_t(u,v) - \mu_{t-1}(u,v))|}{\sigma_t(u,v)} \tag{1.20}$$

The value of the deviation threshold T, when using this distance function in the expression 1.16, is usually chosen as 2.5 [121].

The method has a high computational speed and low memory consumption. Using a single Gaussian often leads to faulty segmentation in scenes with complex background textures. In addition, this method is characterized by the experimental selection of threshold and learning coefficient values.

The **Mixture of Gaussians (MOG)** is a statistical recursive background subtraction method that handles fluctuations of background elements (e.g., leaves of trees) using a mixture of several probability density functions. Independent distribution models simulate local pixel intensity variations, allowing us to account for the oscillatory nature of changes in background texture. The pixel intensity is compared to the background probability models. If the probability of a pixel being in one of the distributions is low, the object is treated as foreground. The most common algorithms of this type are GMG, KNN, MOG, and MOG2 [129]–[133]. These methods are suitable for motion detection, including near-infrared images, that satisfy the requirements of an intelligent lighting system.

The probability of a pixel appearing with a certain intensity is expressed as:

$$P(k_{t,p}) = \sum_{i=1}^{M} \omega_{i,t,p} N(k_{t,p}, \mu_{i,t,p}, \sigma_{i,t,p}), \tag{1.21}$$

where M — number of Gaussians to simulate intensity variations;

$k_{t,p}$ — intensity value of a pixel p at a specific time t;

$\omega_{i,t,p}$ — the weight of the Gaussian i in the total mixture at time t for pixel p;

$N(k_{t,p}, \mu_{i,t,p}, \sigma_{i,t,p})$ — a Gaussian distribution model given by the following probability density function:

$$N(k_{t,p}, \mu_{i,t,p}, \sigma_{i,t,p}) = \frac{1}{\sigma_{i,t,p}\sqrt{2\pi}} \exp^{-\frac{1}{2}(\frac{k_{t,p} - \mu_{i,t,p}}{\sigma_{i,t,p}})^2} \tag{1.22}$$

As in the single Gaussian case, checking a pixel k_p for falling into existing Gaussians i is done through the expression 1.20. The result is compared to the rejection threshold. When a pixel hits an existing Gaussian, it is treated as background. After that, the parameters $\mu_{i,t,p}$, $\sigma_{i,t,p}$ of the background pixel model are updated according to the expression 1.19 and the weighting factor $\omega_{i,t,p}$ as:

$$\omega_{i,t,p} = (1 - \alpha)\omega_{i,t-1,p} + \alpha \tag{1.23}$$

where α — the update coefficient.

When a pixel is not in the distribution, the values $\mu_{i,t,p}$, $\sigma_{i,t,p}$ are not updated, while the weight coefficient of the Gaussians decreases $\omega_{i,t,p} = (1-\alpha)\omega_{i,t-1,p}$ and all the weight coefficients are normalized. A new Gaussian replaces the Gaussian with the lowest pixel weight $k_{t,p}$. All the Gaussians are sorted by the criterion $\frac{\omega_{i,t,p}}{\sigma_{i,t,p}}$ and only the most likely ones are considered as background.

$$H = \underset{h}{\operatorname{argmin}}(\sum_{i=1}^{h} \omega_i > J), \tag{1.24}$$

where J — threshold value.

The pixels with intensities distanced by more than 2.5 standard deviations from the most probable background distributions are segmented as foreground objects.

This method can handle complex background textures consisting of several intensity components and is robust to gradual changes in scene illumination. The disadvantage of this method is its high computational complexity and the need to select thresholds and model update coefficients. In addition, when the noise component of the image is high, pre and post-filtering are required using this method.

1.8.2 Experimental studies of background subtraction methods

Experimental studies of BS methods were conducted to identify the most appropriate method on datasets collected at night, which are close to the expected operational scenarios. The datasets contain low-resolution image sequences (up to 320×240 px) captured at night by a camera equipped with an IR light. The F1-measure (F1) was used as an accuracy metric, while the performance was evaluated by the Frames per Second (FPS) metric. A comparison of the obtained metrics with the works of other authors is presented in Table 1.4.

A comparison of the characteristics of the methods is made based on four works:

1. Experiments are conducted on custom datasets described in Section 3.1.1 using a Raspberry Pi 3 B+ single board computer. The data were partitioned using OpenCV CVAT [134] software. The methods used to perform the experiments are implemented in the open C++ library of background subtraction methods bgslibrary [135]. The BS method parameters used are presented in Table 1.5.

2. A. Sobral et al. [112] tested the algorithms on the BCM [136] dataset, which contains real and synthetically generated videos of urban environments from a static camera. The data includes challenging environmental scenarios for segmentation: fog, cloud cover, wind (affecting the fluctuations of background objects), and scenes with sudden changes in lightness.

3. K. Sehairi et al. [121] measured algorithm accuracy using the CDnet 2014 [137] dataset, which includes video sequences during nighttime, inclemency, dynamic backgrounds, etc.

4. S. Tommesani [138] compared the performance of the presented algorithms on an Intel I3 2370 processor using images with a resolution of 320×190 px.

Table 1.4: Comparison of BS methods

Metric		Frame difference	Adapt. back. learning	Single Gaussian	MOG
F1	Exp.*	0.56	0.63	0.66	0.73
	[112]	0.80	0.84	0.85	0.85
	[121]	0.19	0.28	0.37	0.25
Precision	Exp.*	0.65	0.73	0.73	0.81
	[112]	0.93	0.88	0.92	0.91
	[121]	0.47	0.44	0.49	0.61
Recall	Exp.*	0.50	0.56	0.61	0.68
	[112]	0.70	0.81	0.80	0.79
	[121]	0.28	0.36	0.30	0.21
FPS	Exp.*	267.00	87.00	77.00	60.00
	[121]	36.00	33.35	0.11	0.02
	[138]	2468.00	336.00	379.00	290.00

*Experiments on owned datasets

Table 1.5: Used BS parameters

Method	Parameters
Frame difference	$T = 10$
Adapt. back. learning	$T = 10$
Single Gaussian	$T = 2.5$; $\alpha = 0.05$
Mixture of Gaussians	$M = 5$; $\alpha = 0.01$; $J = 0.03$; $T = 2.5$

Despite the different nature of the data sets, all the studies have similar gradations of F1 and FPS metrics. The complexity of the dataset explains the low F1 values from [121] relative to the other sources.

According to the experimental results, the highest F1 value (0.73) was achieved by the MOG method providing 60 FPS. The frame difference method provides the highest performance (267 FPS) due to computational simplicity; however, its application is problematic in scenes with fluctuating background elements, reflected in the lowest F1 values (0.56). It should be noted that none of the considered methods are stable to sudden changes in illumination, which can be partially compensated by filtering in the subsequent stages of the algorithm [117].

1.9 Section summary

Devices based on IR, US, or RW methods can operate in limited visibility conditions. They are characterized by high energy efficiency, low cost, and ease of use. Applying the passive IR sensors with the aim of object classification is problematic due to their operation principles. A disadvantage of active US sensors is the influence of the environmental conditions, materials, and installation parameters on detection accuracy. RW sensors have high sensitivity [14], which often causes false positive responses due to vibrations or minor movements in a monitoring area [7]. Existing passive US and hybrid systems based on a combination of RW-IR [139] or US-IR [140] are designed to detect objects indoors, and their application in the urban street environment is problematic. One of the critical limitations of passive IR, US, or RW is the relatively short detection range, which does not exceed 12 m [7], [8], [25], [31]–[35], [141]–[143], that does not meet the requirements of the SL system for the detection range. Solving the classification problem by active IR or active US rangefinding requires a specific location of sensors relative to the monitoring zone (for example, perpendicularly over the area [18], [45]), which is challenging to implement in a city street environment and contradicts the SL system requirements for the sensor's location.

Computer vision methods can detect objects in images or videos. The classical Viola-Jones and HOG algorithms require training the classifier with data closest to the expected operating conditions. These methods are not universal with regard to shooting parameters and scene configuration: the position and rotation of objects in space, lighting, and frame resolution significantly affect the detection accuracy. Appeared at the beginning of the CV era, these methods are based on the exhaustive search technique, which requires high computational costs and, therefore, cannot be applied to modern low-performance System on Chips (SoCs) [3], [101], [106]–[108].

In recent years, DL models have gained popularity due to the increase in GPUs' processing power. In the field of object classification and detection, the most famous methods are based on CNNs and their modifications, such as R-CNN, YOLO, SSD, etc. DL models have high accuracy, however, they remain too slow for real-time operation on single-board computers [3], [104], [105], [108].

The necessary camera parameters and the algorithm performance were calculated. The size of the observing area in the image depends on the camera installation height and distance to the lighting area. In the worst scenario of physical camera installation, the required frame rate should be $FPS_{req} \approx 10$, while $VAOV_{min} \approx 90\,\text{deg}$ and $HAOV_{min} \approx 120\,\text{deg}$. The calculated $VAOV_{min}$ and $HAOV_{min}$ angles can be achieved by wide-angle lenses. Such lenses introduce significant distortions in the image, which may require correcting them using an appropriate method.

Based on the given requirements, the Raspberry Pi 3 platform was chosen as a representative of low-performance embedded microcomputers for comparing the FPS performance of the CV algorithms in a frame of the analytical study. The study showed that the conventional CV algorithms ($FPS_{max} = 1.82$) and algorithms based on CNNs ($FPS_{max} = 1.55$) could not achieve the required detection speed on the target platform ($FPS_{max} < FPS_{req}$) and, therefore, do not meet the requirements. The considered speeding-up techniques of CNNs by hardware acceleration of calculations ($FPS_{max} = 5.55$) or preliminary segmentation by background subtraction

($FPS_{max} = 2.9$) cannot provide the required frame rate [4], [102], [103], [106], [107].

Comparing conventional detection approaches to CV methods, the later are preferable for solving the problem of object detection in the frame of the SL system:

- CV methods allow for detecting objects at a greater range than existing commercial sensors can provide. However, the detection range is limited by the camera's viewing angle, image resolution, and IR illumination power;
- CV methods are more versatile since the detection parameters can be changed without modifying the hardware;
- wide variety of camera installation locations relative to the monitoring area is possible;
- the cost of CMOS matrices, especially for low-resolution cameras, is comparable to most budget representatives of conventional methods.

The main disadvantage of the existing CV methods is the high requirements for the computing power of devices. Thus, the developed method should fill the gap between computationally complex CV algorithms and low-performance computers. Nevertheless, the task of object detection applying CV methods in this scenario has a number of challenges:

- the images are grayscale and have poor contrast since they are captured in the near-infrared spectrum (during nighttime).
- making a switching decision in real-time is necessary because objects can move with high velocity. The detection speed of single-board IoT devices is slow because their processing units do not include additional instructions existing in "big" non-embedded processors.
- the image resolution is lowered to 320×240 px in order to reduce the amount of processing data.
- synthetic data can compensate for the deficiency of labeled training data from real street environments [144].

Several background subtraction methods have been analyzed. Applying such methods can speed up detection by computationally inexpensive image segmentation and, as a result, improve the performance of a computer vision algorithm. All the considered background subtraction methods can operate on low-performance computers in real-time and provide a sufficient frame rate. The method based on a Mixture of Gaussians has shown the highest F1-measure on the nighttime dataset.

Chapter 2

Proposed detection method

The chapter describes the structure and main stages of the developed algorithm. The proposed feature extraction method is presented. Characteristics of the expected urban objects and their features are considered. The method for generating synthetic data for training the classifier is described. The results presented in the second chapter were published in [8], [145]–[147].

2.1 Structure of the detection method

A scheme of the detection method proposed in the course of this Ph.D. includes six stages and is shown in Figure 2.1.

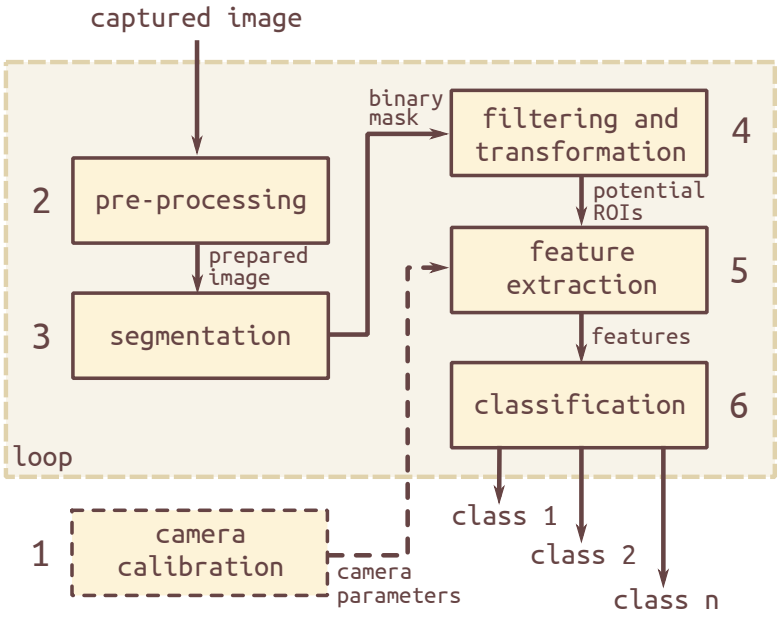


Figure 2.1: Proposed detection algorithm scheme

1. As a result of camera calibration, the intrinsic and extrinsic parameters of the camera, as well as the lens distortion coefficients, are determined. Calibration can be performed by well-known methods, for example, by algorithms Tsai R. [148], Zhang Z. [149], etc.

2. The pre-processing unit receives images taken by a camera, fixed on a static object, and directed to the monitoring area. Adaptive histogram equalization is applied to a sequence of input images to improve the results of subsequent background subtraction in night scenes. The contrast stabilization technique and its parameters can be changed depending on the specific operational case. Camera lens distortions can be corrected at the current stage by applying the correction to all pixels of the image, or at the feature extraction stage, only to the ROI to reduce the number of redundant calculations. The image correction is performed using the lens distortion coefficients obtained in the calibration step.

3. The resulting image is processed by a Background Subtraction (BS) algorithm to segment the image and obtain a binary foreground mask containing moving objects. The choice of the BS method for video stream segmentation is justified by low computational costs relative to segmentation methods based on statistical clustering or neural networks [150]. BS methods generate an averaged background model, adapting to gradual changes in illumination and fluctuating scene parameters [151]. These methods have some limitations, including the inability to distinguish static objects, which does not contradict the requirements of the smart lighting system. The BS method based on MOG was used in the proposed detection algorithm. The rationale for choosing the segmentation method is discussed in Section 1.8.

4. At the stage of filtering and transformation, a morphological opening is applied to a binary image, which includes subsequent erosion and dilation operations performed with the same kernel. The erosion removes noise caused by minor fluctuations and changes in lighting, while the dilation increases the ROI that has been cut off by erosion [152]. An additional dilation is performed to merge the subareas of interest broken up incorrectly during the segmentation process, primarily due to insufficient scene illumination. The primary parameters of objects, such as the contour area, the bounding rectangle, and the object's coordinates in an image, are determined using the OpenCV computer vision library, which, in turn, uses Green's theorem to calculate. Small objects are removed by comparing the contour area to a statistically defined threshold. The filtering and transformation techniques used are described in more detail in Section 2.1.4.

5. The camera intrinsic parameters, obtained during the calibration process, allow converting the primary parameters of objects into several features (*distance*, *estimated width*, *height*, and *contour area*) that describe the geometric shape and coordinates of the object in the real world. A detailed description of this stage is given in Section 2.1.5.

6. A trained classifier, based on logistic regression, decides whether the candidate's attributes belong to a particular class (for example, a pedestrian, a cyclist, or a car). The classification stage is described in Section 2.1.6.

2.1.1 Camera calibration

A matrix of internal camera parameters and lens distortion coefficients are determined during the calibration. The resulting camera calibration parameters are used in the feature extraction stage

of the developed detection method.

A widely used calibration approach is the Zhang [149] algorithm, which uses a checkerboard as the calibration object. The use of the checkerboard is motivated by the ease of detecting the coordinates of the rectangles' corners. On a number of chessboard images, projective transformations between chessboard points in the real world and the image are determined from different angles. The resulting projective transformations are used to estimate the internal camera parameters by applying a closed-form linear solution [153]. The camera calibration also determines lens distortion coefficients used in lens distortion modeling. The distortion components are radial and tangential. Radial distortion is axisymmetric and can be described by a radial function, as proposed in the classical Brown-Conrady [154] lens distortion model. Tangential distortion occurs when the camera matrix and lens are not parallel. The radial components of distortion are more than an order of magnitude greater than the tangential components, so in practice, tangential distortion is often ignored [155].

2.1.2 Pre-processing

The problem of insufficient image contrast getting especially relevant in low-light conditions when using a camera with an IR emitter. The pixel intensities are distributed in a narrow range in such cases, which makes their differentiation problematic, in particular, in the next segmentation stage. An adaptive contrast stabilization technique is applied to improve the contrast component of the input image. This technique redistributes the pixel intensities through a transformation function determined by neighboring pixels. The adaptive contrast stabilization algorithm for a single-channel image includes the following steps:

- An image is split into regions. For each region, the transform function is calculated — Cumulative distribution function (CDF) (cumulative histogram containing pixel intensities and their number in the image);
- new values of pixel intensities are calculated for each region according to the following histogram stabilization expression:

$$h(v) = round(\frac{\mathrm{CDF}(v) - \mathrm{CDF}_{min}}{(M \cdot N)} - \mathrm{CDF}_{min}) \cdot (L - 1), \tag{2.1}$$

where $round$ — is a rounding function to the nearest integer;

CDF_{min} — a minimum nonzero value of the cumulative probability function;

M, N — width and height of the region, px;

L — number of pixel intensity levels.

The Contrast Limited Adaptive Histogram Equalization (CLAHE) [156] is used to minimize the noise component of the area that can be amplified during stabilization. The pixel histograms exceeding a threshold value are clipped by evenly redistributing them across the rest of the histogram cells.

2.1.3 Background subtraction method

According to the comparative analysis (Section 1.8.2), the BS method based on a Mixture of Gaussians has shown acceptable performance and the highest accuracy during trial on the near-IR nighttime dataset. In the following work, the Mixture of Gaussians is used as a method of video stream segmentation. The proposed detection algorithm, however, is not focused on a particular segmentation method, which can be chosen depending on a particular task and expected operating conditions.

2.1.4 Filtering and transformation

The primary filtering is performed at the *filtering and transformation* stage, and the binary mask and segmented objects of interest are prepared for the next feature extraction stage. The basic geometric parameters are extracted from the objects of interest.

Morphological filtering

The binary mask obtained as a result of segmentation, containing the objects of interest, is subjected to preliminary filtering from noises to exclude them from the subsequent processing. Noises on the binary mask often arise due to reflections of a light beam (produced by headlights or flashlights) from the surfaces.

Morphological opening (opening) [64] is used to filter the binary image. The morphological opening involves sequential erosion and dilation operations performed with the same structural element:

$$A \circ B = (A \ominus B) \oplus B, \tag{2.2}$$

where A — binary image;

B — a structural element, which is a binary geometric shape (e.g., a rectangle);

$\ominus$, $\oplus$ — erosion and dilation operations, respectively.

During the opening operation, noises (points, lines) with a smaller diameter than the structural element are removed. While this operation introduces distortions to the geometric shape of the object of interest, smoothing the contours, these distortions can be neglected when the size of the structural element is much smaller than the size of the expected objects of interest.

Finding basic geometric parameters of an object

The filtered binary foreground mask is used as input data to find the basic geometric parameters of the object. The following geometric parameters of the object in an image with resolution $u_{max} \times v_{max}$ are determined if the object is present in the image:

- area of the object in pixels (ca_{px}). The area is found through standard methods of the *OpenCV* computer vision library, which subsequently uses Green's theorem for computing;
- bounding rectangle of the object, represented by the coordinates of one of the corners and lengths of the sides (u_o, v_o, w, h), and its area in pixels (ca_{Rpx});

Filtering by minimum contour area

The large noises are not eliminated during the morphological opening. The obtained contour areas are filtered by another method to reduce the number of candidates processed in the subsequent stages of the algorithm. Undesirable candidates are filtered out by comparing the contour area ratio of the object to an experimentally defined threshold:

$$s_{ca} = \begin{cases} 1, & \text{if } f_{ca} < t_{ca} \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

$$f_{ca} = 100 \cdot \frac{ca_{px}}{u_{max} \cdot v_{max}} \tag{2.4}$$

where f_{ca} — contour area ratio defined as the ratio of an object area to the total area of the image;

t_{ca} — threshold value.

Finding the threshold value is based on a statistical analysis of experimental data. Three datasets have been formed using self-gathered and publicly available data:

- Non-informative objects (normal) are noises and artifacts remaining after the morphological opening in a binary mask in normal weather conditions.
- Informative objects (normal) - objects of interest (pedestrians, bicyclists, and cars) captured in normal weather conditions.
- Non-informative objects (rain) are noises and artifacts remaining after the morphological opening in a binary mask. The noises are mainly caused by rainfalls of different intensities and oscillation of grass or bushes due to gusts. The dataset is built using self-collected and publicly available data, which includes scenarios during day and night. The dataset details are shown in Table 2.1.
- Non-informative objects (snow) are noises and artifacts remaining after the morphological opening in a binary mask. The noises are mainly caused by snowfalls of different intensities. Swaying branches of the trees are present in the dataset. The dataset is formed from public data and includes only daytime scenes. The dataset details are given in Table 2.2.

The datasets with *non-informative objects (normal)* and *informative objects (normal)* include day and night scenes that have been gathered in normal weather conditions without precipitations. An experimental setup, scenarios, and description of the dataset are presented in Sections 3.1 and 3.2. The contour area ratio (f_{ca}) is calculated for each dataset according to Formula 2.4. Dispersion of the contour area ratio for four datasets is shown in Figure 2.2.

The interquartile range (IQR) is used to find the boundaries for outliers. An upper whisker defines the contour area threshold for the non-informative data:

$$t_{ca} = Q3 + 1.5 \cdot IQR \tag{2.5}$$

Table 2.1: Properties of the dataset containing noises in rainy weather

Scenario	**Source**	**Time of day**	**Moving objects**	**Precipitations**	**Wind speed**
Light rain	*self	day	grass, raindrops	0.2 mm	24 km/h
Moderate rain	[157]	night	raindrops	-	-
	[158]	night	raindrops	-	-
	*self	night	trees, raindrops	0.8 mm	15 km/h
	*self	day	trees, raindrops	1.2 mm	15 km/h
Heavy rain	[159]	night	reflections, raindrops	-	-
	[160]	day	bushes, raindrops	-	-
*self — data gathered by the author					

Table 2.2: Properties of the dataset containing noises during snowfall

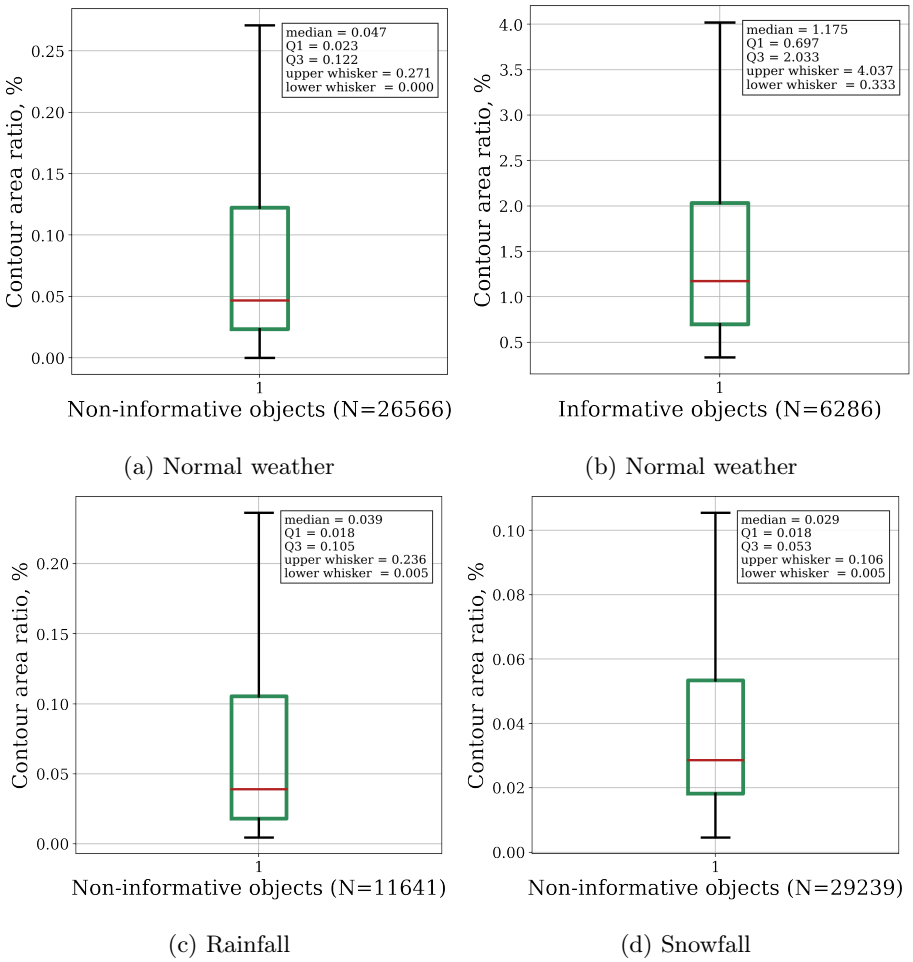
Scenario	**Source**	**Time of day**	**Moving objects**
Light snowfall	[161]	day	snowflakes
Moderate snowfall	[162]	day	snowflakes, branches
Heavy snowfall	[163]	day	snowflakes, branches

Among the datasets containing non-informative data in different weather conditions (*non-informative objects (normal/rain/snow)*), the maximal value at the upper whisker (0.271 %) is obtained for the dataset gathered under normal weather conditions - *non-informative objects (normal)*. The more considerable variation of the contour area ratio for non-informative data in *normal* weather scenario compared to scenarios with precipitations is explained by the presence of moving lighting sources in the *normal* dataset. The moving lighting sources produce more significant artifacts in the foreground mask than rain or snowfall. Thus, the chosen threshold value for the *normal* dataset filters out noises appearing during rain or snowfall.

The chosen threshold filters most non-informative data and minimizes interference with informative objects (lower whisker is 0.333 %) since the outlier boundaries for both distributions are not overlapping. The proposed filtering approach reduces the number of undesired objects on the order of magnitude.

2.1.5 Proposed feature extraction method

A projection of a 3D object in the image is determined by a camera's extrinsic and intrinsic parameters. Extrinsic parameters of the camera are coordinates in space relative to the object and rotation angles around the axes. Intrinsic parameters are focal length, the sensor's physical dimensions, and the image's optical center. The feature extraction method uses these parameters to roughly estimate the geometric dimensions of an object in the real world. It is important to



(a) Normal weather (b) Normal weather

(c) Rainfall (d) Snowfall

Figure 2.2: Box-and-whisker plots for contour area ratio for different weather conditions

note that the proposed method is not aimed at accurately measuring the geometric dimensions of an object; however, it allows identifying unique features that make it possible to differentiate the types of objects at the classification stage. The obtained geometric parameters are referred to as *estimated* since the calculated values may differ from the actual dimensions of the object.

Calculation of the distance to the object.

The distance between the camera and the object is determined to estimate an object's size in the real world. The method transforms the coordinates of the image into real-world coordinates. For that, several pre-requirements must be met:

- an object intersects a ground surface in the real world at least at one point;
- the lowest object point in the image lies on a ground surface in the real world;
- known intrinsic camera parameters such as equivalent focal length, physical dimensions of the sensor, and image resolution;
- known extrinsic parameters, which include tilt angle and real-world coordinates of the camera relative to the horizontal surface;
- known distortion coefficients of the camera lenses.

Camera lens distortion is corrected for radial and tangential factors using the Brown-Conrady optical distortion model [154]:

$$\begin{aligned} r &= \sqrt{(u'-u_c)^2+(v'-v_c)^2} \\ u &= u'(1+k_1r^2+k_2r^4+k_3r^6)+2p_1u'v'+p_2(r^2+2u'^2), \\ v &= v'(1+k_1r^2+k_2r^4)+p_1(r^2+2v'^2)+2p_2u'v' \end{aligned} \tag{2.6}$$

where u, v — coordinates of the corrected pixel along the abscissa and ordinate axes, px;

u', v' — coordinates of the distorted pixel, px;

r — a function of the coordinates u', v', depending on the distance to the optical center of the lens (the central pixel of the image with coordinates u_c, v_c);

k_1, k_2, k_3 — radial distortion coefficients;

p_1, p_2 — tangential distortion coefficients.

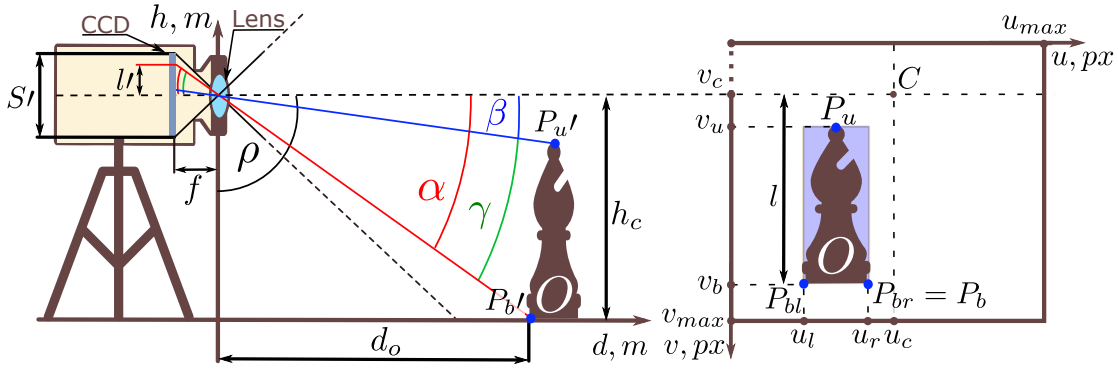


Figure 2.3: Object distance estimation scheme: real-world (left) and image plane (right) scenes

According to Figure 2.3 (right), one of the bottom pixels of object O is the point $P_b(u_r, v_b)$ in the image with a resolution of $u_{max} \times v_{max}$. The image principal point $C(u_c, v_c)$ corresponds to the camera's optical center, which for an ideal camera is equivalent to the center pixel of the image.

Given the distance $l = v_b - v_c$, as the distance along the v axis between the lowest point of the object and the principal point of the image in pixels, it is possible to find the equivalent distance $l\prime$ in units of the 3D scene (mm):

$$\frac{S\prime}{l\prime} = \frac{v_{max}}{l} \Rightarrow l\prime = \frac{S\prime l}{v_{max}} = \frac{S\prime}{v_{max}}(v_b - v_c) \tag{2.7}$$

where v_b is the lower coordinate of the object in the image, px;

$S\prime$ - camera sensor height, mm;

v_{max} - image resolution along the v axis (image height), px.

The angle α is calculated using the segment $l\prime$ from Expression 2.7 as the first cathetus and the focal length f as the second one:

$$\alpha = \arctan\left(\frac{l\prime}{f}\right) = \arctan\left(\frac{S\prime}{f v_{max}}(v_b - v_c)\right), \tag{2.8}$$

where f is the focal length of the camera, mm.

The distance to the object is calculated based on the height h_c and the camera angle ρ, in the same way, as the angle α between the bottom point of the object O and the camera's optical center had been found. Distance is found using the following expression:

$$d_o = \frac{h_c}{\tan\left(\alpha + (90° - \rho)\right)}, \tag{2.9}$$

where $(90° - \rho)$ is a tilt of the camera with respect to the ground, deg.

Object height estimation

The estimated object height is found using the calculated distance and known intrinsic camera parameters. According to Figure 2.3, pixels P_b and P_u correspond to the image plane's lowest and highest points of the object O. The angle between these points is found using Expression 2.8. The angle $\eta \sim \gamma$ (shown in Figure 2.4) is calculated using Formula 2.10.

$$\gamma = \alpha - \beta = \arctan\left(\frac{Sl}{fv_{max}}(v_b - v_c)\right) - \arctan\left(\frac{Sl}{fv_{max}}(v_u - v_c)\right), \tag{2.10}$$

where v_b and v_u are the lowest and highest coordinates of the object along the v axis, px.

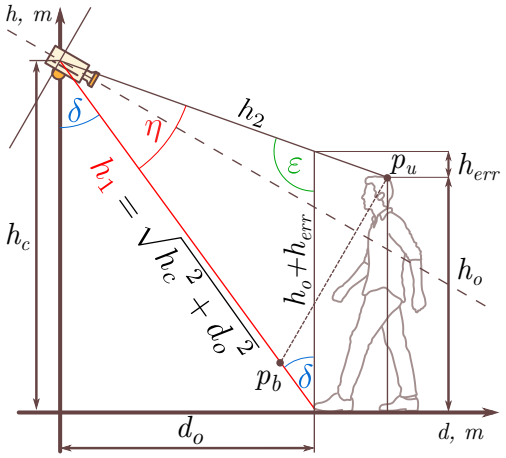


Figure 2.4: Object height estimation scheme

The angle $\delta = \arctan\left(\frac{d_o}{h_c}\right)$, therefore, $\varepsilon = 180° - \eta - \delta$. Given all the angles and one side of the triangle, the *estimated height* of the object is calculated as follows:

$$h_e = h_o + h_{err} = h_1 \cdot \frac{\sin\eta}{\sin\varepsilon} \tag{2.11}$$

The result of Expression 2.11 is the height of the actual object and the non-negative estimation error due to the geometric shape of the object and the camera setup parameters.

Object width estimation

The estimated width of an object is calculated via an inverse projective transformation to reconstruct real-world coordinates from image points. The inverse projective transformation can be expressed in the Pinhole Camera Model [164]. The following expression describes the Pinhole Camera Model:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot [R|t] \cdot W = \begin{bmatrix} f_x & 0 & u_c \\ 0 & f_y & v_c \\ 0 & 0 & 1 \end{bmatrix} \cdot \left[\begin{array}{ccc|c} 1 & 0 & 0 & t_1 \\ 0 & \cos\theta & -\sin\theta & t_2 \\ 0 & \sin\theta & \cos\theta & t_3 \end{array}\right] \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.12}$$

where u, v — coordinates of the projected point on the image plane, px;

K is a matrix of intrinsic camera parameters which describes the relationship between a point in the camera coordinate system and a point in the image coordinate system; the matrix includes the focal lengths f_x, f_y in pixels and the central coordinates of the image c_x, c_y, px;

$[R|t]$ is a matrix of extrinsic camera parameters that describes the position and direction of the camera in the real world. It includes the rotation matrix R, in which rotation is expected only around the X axis by θ degrees; and the matrix of extrinsic parameters also contains a transition vector t_1—t_3, whose values can be set to zero if the shift is included in real world coordinates X, Y, Z;

W — the matrix of coordinates of a real-world point, expressed through the representation of the coordinates of the point (X, Y, Z) in the homogeneous system.

The inverse perspective transformation can be found based on the forward transformation 2.12 as:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = [R|t]^{-1} \cdot K^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \cdot z_{cam}, \tag{2.13}$$

where X, Y, Z - coordinates of a point in the real world, m;

$[R|t]^{-1}$ - inverse matrix of extrinsic camera parameters;

K^{-1} is the inverse matrix of the camera's intrinsic parameters;

u, v - coordinates of the projected point on the image plane, px;

z_{cam} - scaling factor equal to the distance to the object in the camera coordinate system, m.

The image point with coordinates u v is pre-corrected to eliminate lens distortion. Being interested only in the z_{cam} parameter and considering rotation around the X axis, z_{cam} is derived through the simplification of $[R|t] \cdot W$ and has the following form:

$$z_{cam} = Y \cdot \sin\theta + Z \cdot \cos\theta \tag{2.14}$$

According to Figure 2.3, the terms of the expression are the height of the camera $Y = h_c$, the distance to the object $Z = d_o$, and camera rotation angle $\theta = 90° - \rho$. Thus, the inverse transformation (Expression 2.13) allows estimating the X coordinate of an object point in 3D space, which corresponds to the projection of this point into the image with coordinates u, v. The estimation is based on the assumption that the lowest point of the object lies on the ground surface.

To estimate the width of an object in the real world, the left-most $P_{bl}(u_l, v_b)$ and right-most $P_{br}(u_r, v_b)$ bottom points of the object's bounding box in the image are reconstructed in the 3D

scene. The reconstructed points P_{bl} and P_{br} have X_l, Y_l, Z_l and X_r, Y_r, Z_r coordinates in the real world. The estimated width of the object is found as the distance between these points. When calculating this distance, only the X coordinates are taken into account since the values of the Y and Z coordinates for all points on the lower side of the bounding box are the same:

$$w_e = w_o + w_{err} = \sqrt{(X_r - X_l)^2} \tag{2.15}$$

Contour area estimation

The estimated contour area of an object (ca_e), expressed in real-world units, is calculated by the ratio of the bounding box area to the object's contour area in the image and the real world.

$$\frac{h_{px} \cdot w_{px}}{ca_{px}} = \frac{l_e \cdot w_e}{ca_e} \Rightarrow ca_e = \frac{ca_{px} \cdot l_e \cdot w_e}{h_{px} \cdot w_{px}}, \tag{2.16}$$

where h_{px}, w_{px} - height and width of object's bounding rectangle in image, px;

ca_{px} - a contour area of the object, px.

2.1.6 Object classification

Reaction to a motion is made only for specific objects, depending on their geometric parameters. The object type is determined using a trained logistic regression classifier. The classifier is trained using real and synthesized features of objects. The type of object is described by a number of features that characterize its geometric parameters, in particular, the *estimated width*, *height*, *area*, *distance* to the object, and the parameters of the 3D scene - the *angle* of the camera relative to the ground surface and the *height* of the camera. The parameters of the 3D scene compensate for estimation errors (l_{err}, w_{err}) appearing at the feature extraction stage. The logistic regression method was used as a statistical model for multi-class classification since this model does not require significant computational resources, providing the ability to adjust the parameters of the target function and easy regularization [165].

Logistic regression uses a logistic function (sigmoid) as a hypothesis to model a probabilistic output in which a range of values lies in the interval [0, 1]. The logistic function is represented as [166]:

$$\begin{gathered} h_\beta(x_i) = \frac{1}{1 + e^{-z}}, \\ z = \beta^T x_i \end{gathered} \tag{2.17}$$

z — a regression equation in vector form, the polynomial order of which is chosen depending on the specific dataset;

β — vector of model parameters;

x_i — vector of input features of point i.

The choice of parameters β determines the probability that the input features x belong to a particular class y:

$$h_\beta(x_i) = P(y = 1|x_i; \beta) = 1 - P(y = 0|x_i; \beta) \tag{2.18}$$

The β adjustment is performed by minimizing the loss function on the training dataset, for example, through the gradient descent algorithm. The training sample is marked so that the feature set for each point i corresponds to a binary label of y belonging to a specific class. The loss function for logistic regression can be expressed as [167]:

$$J(\beta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(h_\beta(x_i)) + (1-y_i)\log(1-h_\beta(x_i)), \tag{2.19}$$

where n — the number of elements in the training sample;

y_i — a binary label of class affiliation.

The updating of model parameters via gradient descent is done simultaneously:

$$\beta_j = \beta_j - \alpha\frac{\partial}{\partial\beta_j}J(\beta) = \beta_j - \frac{\alpha}{n}\sum_{i=1}^{n}(h_\beta(x_i) - y_i)x_{ij}, \tag{2.20}$$

where α is the learning rate coefficient of the model.

The multi-class classification problem is minimized to binary if several classes $y \in \{0, 1...k\}$ are present.. For each class, hypotheses $h_{\beta,k}(x_i)$ are constructed, transforming the binary labels of the data set so that for the class in question $y = 1$, while for all others $y = 0$. As a result, the hypothesis of the class with the highest probability for the input feature set x_i is chosen.

2.2 Classes and characteristics of detecting objects.

The following object classes are expected in the SL street environment: *pedestrian*, *cyclist*, and *vehicle*. These classes can also be combined into a common metaclass - the *target group*. Any other object that does not belong to the target group's classes is considered a noise class. The noises are the ROIs appearing due to rapid changes in scene illumination or oscillation of background objects (e.g., trees). The class noise is not the target class but the supporting one. The *target group* class is introduced for a binary classification between the *target group* and *noise*. The misclassification problem between classes within the *target group* is considered less significant than the *target group / noise* classification since all classes require lighting at night. As a result of the analytical study, the most probable parameters of objects in the real world were distinguished (Table 2.3).

Table 2.3: Selected boundaries of geometrical parameters for the *target group* [168]–[175]

Class	Target group					
	Pedestrian		Bicyclist		Vehicle	
	min	max	min	max	min	max
Width, m	0.30	0.64	0.30	0.64	1.40	1.80
Height, m	1.15	2.00	1.50	1.90	1.40	2.00
Depth, m*	0.30	0.79	1.50	1.70	3.70	6.50
Speed, m/s	1.10	4.50	1.50	9.00	4.00	20.00

*Maximal step size for a pedestrian

Environmental conditions, properties of 3D scenes, and properties of objects, such as lighting and object material, can distort the geometric parameters of objects. These distortions lead to insufficient or incorrect segmentation of objects. Moreover, each class introduces its specific distortions. The pedestrian class is subject to non-expected moving behavior. The pedestrian changes its geometric parameters while walking because of the non-rigid nature of the human body. Several distortions are common to the cyclist and vehicle classes. For example, in some night scenes, the object and the light produced by its headlights (reflections from asphalt, etc.) can merge into one object. Motion Blur can occur due to slow shutter speeds, especially when capturing fast-moving objects. Large-sized vehicles moving at certain angles can cause significant estimation errors during feature extraction.

2.3 Synthetic data generation methods

The purpose of the synthetic data generation is to collect plausible geometric features of the classes for training the classifier. Using synthetic data speeds up the process of collecting and preparing data for training, unlike empirical data collection via a real camera. Given the lack of real data for training the classifier in the public domain (nighttime IR datasets with known intrinsic and extrinsic camera parameters), the method for generating synthetic data can be especially important (Table 2.4). The synthetic generators provide detailed parameters about the scene configuration, which are required by the Dimensional Based Object Detection (DBOD) algorithm, e.g., cameras' focal length, height, incline, and sensor dimensions.

Table 2.4: Properties comparison of synthetic datasets to publicly available ones [176]

	Real (open access)	**Synthetic**
Nighttime scenes	very limited	any
Camera parameters	usually not provided	any
Illumination	daytime/street lights/IR	daytime/street lights/IR
Camera position	arbitrary	smart-lighting specific

The synthetic data has been generated by two methods to find the most accurate solution for the DBOD algorithm: *perspective projection* and *full scene reconstruction.*

2.3.1 Perspective projection method (method 1 - proposed in the course of the Ph.D.)

The essence of the technique is to project the prepared 3D models of objects into the image plane to generate a foreground binary mask. During the projective transformation, the object is proportionally scaled within the specified range of geometric dimensions in the real world. The coordinates are shifted within the boundaries of the control area ($35 \times 35 \times 12$ m). Rotation of the object around the Y axis in the range of $0 - 360$ deg is performed to simulate the object's movement direction. The object's rotation around the X axis in the range of $0 - 90$ deg relative to

the ground is performed to simulate the camera tilt angle. The convolution of the original image and the point spread function introduces the linear motion blur effect. The function arguments (the pixel movement length during exposure and the angle with respect to the horizontal axis) are determined through the projective transformation of an object point moving at a constant speed in the real world into the image plane. Morphological erosion is applied to the image to emulate insufficient scene illumination, which is typical for the real dataset. The distorted image is filtered and transformed as described in Section 2.1.4. The feature extraction method uses the distorted binary image with a segmented object to calculate the features of the object. The *feature extraction* step is identical to the one described in Section 2.1.5. The resulting object features are associated with the parameters used to generate the scene and used for training the classifier. *The noises* are generated numerically as vectors of features out of known ranges. A schematic of the synthetic data generation method is shown in Figure 2.5.

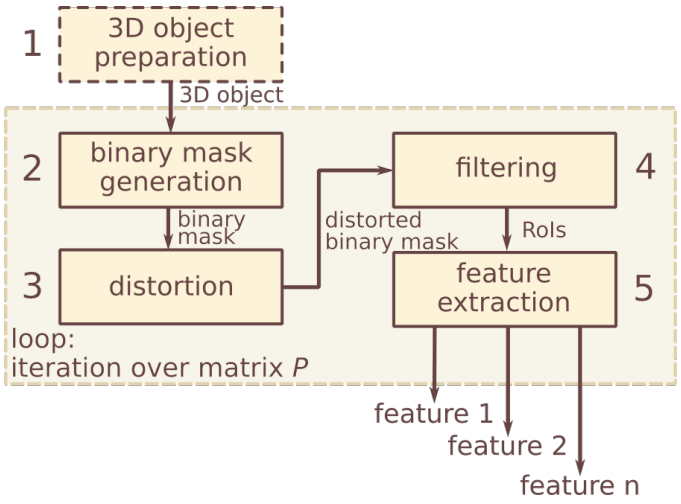


Figure 2.5: Scheme of synthetic feature generation

The parameters for generating the scene and the synthetic object in it are defined in the P matrix. Each row of the matrix contains a set of parameters for the current iteration of the algorithm. The matrix P is an array corresponding to specific object models.

3D object preparation. The 3D models are prepared for each class (described in Section 2.2). Preparation of 3D models consists of modifying the motion poses of the original 3D model templates available publicly [177]. 3D models of the *pedestrian* class are represented by the most likely poses of its movement — walking, running, and standing still poses. The models of the *cyclist* class are expressed in different positions of a person on a bicycle and types of bicycles. The models of the *transport* class are represented by different types of vehicles. In addition, in order to speed up the subsequent data collection process, the number of vertices and polygons of the models is reduced to no more than 500 triangular polygon vertices per object. Examples of prepared 3D objects are shown in Figure 2.6. Manipulations for 3D model preparation were performed using *Blender 2.81a* software. Prepared 3D models are saved in files of format *"Wavefront.obj"* containing coordinates of vertices and corresponding polygons to simplify the file reading by the data collection algorithm. Each object is an array of coordinates of object vertices, where each object vertex is a vector of coordinates (p_x, p_y, p_z) in homogeneous form.

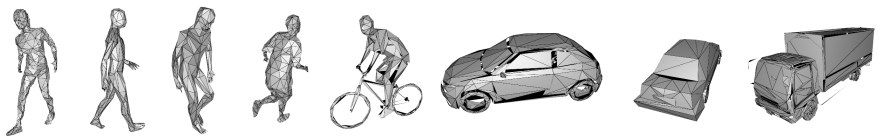


Figure 2.6: Examples of 3D objects used for *method 1*

Binary mask generation involves a series of linear transformations with the specified scene and 3D model parameters.

The object's scaling is done proportionally, according to the expected dimensions in Table 2.3. The total scaling factor v_s is defined as $v_s = \frac{v_t}{v_o}$, where v_t is– the table size along the desired axis; v_o is the initial size of the object, defined as the distance between the maximum and minimum coordinate along the desired axis. Thus, minimum (v_{smin}) and maximum (v_{smax}) scaling factors considering the height, width, and depth of the object (along axes x, y, and z in the 3D scene) are defined as:

$$v_{smin} = min(\frac{v_{tx}}{v_{ox}}, \frac{v_{ty}}{v_{oy}}, \frac{v_{tz}}{v_{oz}}), v_{smax} = max(\frac{v_{tx}}{v_{ox}}, \frac{v_{ty}}{v_{oy}}, \frac{v_{tz}}{v_{oz}}) \tag{2.21}$$

Scaling of object points is performed using the following linear transformation in the homogeneous coordinate system:

$$S_v p = \begin{bmatrix} v_x & 0 & 0 & 0 \\ 0 & v_y & 0 & 0 \\ 0 & 0 & v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} v_x p_x \\ v_y p_y \\ v_z p_z \\ 1 \end{bmatrix}, \tag{2.22}$$

where S_v is the scaling matrix;

p — object point coordinate matrix;

v_x, v_y, v_z — scaling vector, where $v_x = v_y = v_z = v_s$ in case of proportional scaling of the object;

p_x, p_y, p_z — coordinates of the 3D vertex.

The object is rotated around the Y-axis from 0 to 360° to simulate the movement direction. The rotation is the product of the rotation matrix and the coordinates of the object vertices:

$$R_y p = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \cos\theta & \sin\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}, \tag{2.23}$$

where R_y is the rotation matrix;

θ — rotation angle.

The object coordinates are translated into a range of 3D coordinates of the modeled control area to generate scenarios with different objects' locations. In the 3D scene, the object has coordinates t_x, t_y, and t_z in the ranges from t_{xmin} to t_{xmax}, which corresponds to the lateral displacement; from t_{ymin} to t_{ymax} — corresponding to the camera location height; from t_{zmin}

to t_{zmax} — the object distance from the camera. The dimensions of the control area used for modeling were $35 \times 35 \times 12$ m, where 12 m is the maximum expected height of the camera location.

The coordinates of the object in 3D space are iterated through the coordinate array C, where $C_i = (t_x, t_y, t_z)$, which includes all possible combinations of object coordinates in the control area. The coordinates translation is performed through the following transformation:

$$T_v p = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} t_x + p_x \\ t_y + p_y \\ t_z + p_z \\ 1 \end{bmatrix}, \tag{2.24}$$

where T_v — translation matrix;

t_x, t_y, t_z — translation vector.

The object is rotated around the X-axis from 0 to 90° to simulate the camera tilt angle. The rotation matrix is presented earlier in the expression 2.12.

The *projective transformation* projects the object points in 3D space onto the 2D plane, given the camera parameters (expression 2.12).

Thus, all the transformation steps of the object vertex into an image point can be represented by the following matrix product:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot R_x \cdot T_v \cdot R_y \cdot S_v \cdot \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}, \tag{2.25}$$

where u, v — the desired projection of a 3D object vertex in an image;

K — a matrix of intrinsic camera parameters;

R_x, R_y — rotation matrices around X and Y axes;

T_v — a translation matrix;

S_v — a scaling matrix;

p_x, p_y, p_z — coordinates of an object vertex in the 3D space.

Thus, the complete transformation matrix has the following form:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u_c v_x \cos\theta_x \cos\theta_y - f_x v_x \sin\theta_y & f_x v_y \cos\theta_y + u_c v_y \cos\theta_x \sin\theta_y & u_c v_z \sin\theta_x & f_x t_x + u_c(t_y \sin\theta_x + t_z \cos\theta_x) \\ v_c v_x \cos\theta_x \cos\theta_y - f_y v_x \sin\theta_x \cos\theta_y & v_c v_y \cos\theta_x \sin\theta_y - f_y v_y \sin\theta_x \sin\theta_y & f_y v_z \cos\theta_x + v_c v_z \sin\theta_x & f_y(t_y \cos\theta_x - t_z \sin\theta_x) + v_c(t_y \sin\theta_x + t_z \cos\theta_x) \\ v_x \cos\theta_x \cos\theta_y & v_y \cos\theta_x \sin\theta_y & v_z \sin\theta_x & t_y \sin\theta_x + t_z \cos\theta_x \end{bmatrix} \cdot \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{2.26}$$

Distortion of the binary mask is performed to approximate synthetic data to the real datasets. One of the most influential distortion components is the motion blur effect, which occurs when

relative motion between a camera and subject is present during capturing [178]. Simulation of the motion blur effect allows taking into account the imperfection of a capturing device. The real-world scene parameters affecting the motion blur are exposure time, object velocity, object coordinates, and motion angle. The proposed distortion method iterates the exposure time and the object's velocity in the expected ranges, introducing the blurring effect depending on the coordinates and angle of the object. A distorted binary mask is formed at each iteration, which is subsequently processed by the feature extraction algorithm.

The distorted image with the blurring effect $i(u, v)$ is obtained as a result of the convolution between the original image $f(u, v)$ and the point spread function $d(u, v)$ [179]:

$$i(u, v) = f(u, v) * d(u, v) \tag{2.27}$$

The point spread function for the homogeneous linear motion blur is expressed as [180]:

$$d(u, v; L_{px}, \phi) = \begin{cases} \dfrac{1}{L_{px}}, & \text{if } \sqrt{u^2 + v^2} \leqslant \dfrac{L_{px}}{2} \wedge \dfrac{u}{v} = -\tan\phi \\ 0, & \text{otherwise} \end{cases}, \tag{2.28}$$

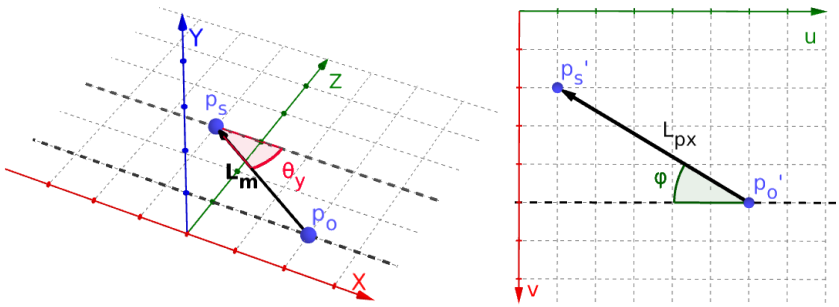
where u, v — coordinates of the pixel;

L_{px} — length of pixel motion during exposure, px;

ϕ — the angle of motion relative to the horizontal axis, rad.

The arguments of function 2.28 are the angle ϕ and the length of pixel motion L_{px}. The determination of these arguments is based on point motion modeling with the given parameters of the 3D scene and the subsequent projective transformation.

As a base point, the central-frontal point of the object p_o, lying at its base on the ground plane (the XZ plane), is chosen. The base point p_o has coordinates X_o, Y_o, and Z_o, which are set at the stage of coordinates *translation*. The motion length L_m of point p_o during the exposure time is calculated in the 3D scene as $L_m = v \cdot t_e$, where v is the constant velocity of a point, which is selected from the range of velocities $[v_{min}, v_{vmax}]$ corresponding to the object class, according to Table 2.3; t_e — exposure time, chosen from the range $[t_{emin}, t_{emax}]$ determined experimentally, where t_{emin} corresponds to the minimum exposure time at which the blur effect is not noticeable.



(a) 3D scene (b) Image plane

Figure 2.7: Moving point parameters

The projective transformation of points p_o and p_s allows estimating the arguments of the

blurring function for the moving object in the image plane. These points are defined in the image plane as $p_{o\prime}$ and $p_{s\prime}$ with coordinates $(u_{po\prime}, v_{po\prime})$ and $(u_{ps\prime}, v_{ps\prime})$. The angle θ_y is set at *the rotation stage around the Y-axis* to simulate object movement direction. Coordinates of a new point $p_{s\prime}$ in the image plane are calculated given the angle θ_y and the length of motion L_m:

$$\begin{bmatrix} u_{ps\prime} \\ v_{ps\prime} \\ 1 \end{bmatrix} = K \cdot R_x \cdot T_{ps} = K \cdot R_x \cdot \begin{bmatrix} X_{po} + L_m \cos(\theta_y) \\ Y_{ps} \\ Z_{po} + L_m \sin(\theta_y) \\ 1 \end{bmatrix}, \tag{2.29}$$

where the coordinate Y_{ps} is assumed to be equal to the coordinate of the source point since the motion of objects is expected only in the XZ plane.

Then, the length of motion L_{px} and the angle ϕ with respect to the horizontal axis u in the image plane are defined as:

$$\begin{aligned} L_{px} &= \sqrt{(u_{po\prime} + u_{ps\prime})^2 + (v_{po\prime} + v_{ps\prime})^2} \\ \phi &= \frac{\arcsin(\sqrt{(v_{po\prime} - v_{ps\prime})^2})}{L_{px}} \end{aligned} \tag{2.30}$$

The resulting object contours are processed by the proposed feature extraction method to collect data for training the classifier. The features of the class *noise* are generated numerically as a uniform distribution, with the feature values lying outside the intervals of the *target group*.

2.3.2 Full scene reconstruction method (method 2 - proposed by Karpov et al.)

The method for synthetic data generation proposed by Karpov et al. [144] was used as a second synthesizing method for accuracy comparison. The 3D scene is modeled using the Unity 3D engine to emulate a dark urban environment. During the modeling, 3D objects of the *target group* are moving at typical speeds along predefined trajectories. Compared to the perspective projection method, the current approach allows synthesizing realistic illumination conditions (camera's IR light, lamps of vehicles and bicyclists, streetlights). In addition, the distribution of *noise* features is more realistic (corresponds to a real dataset). Examples of generated scenes by this method are shown in Figure 2.8.

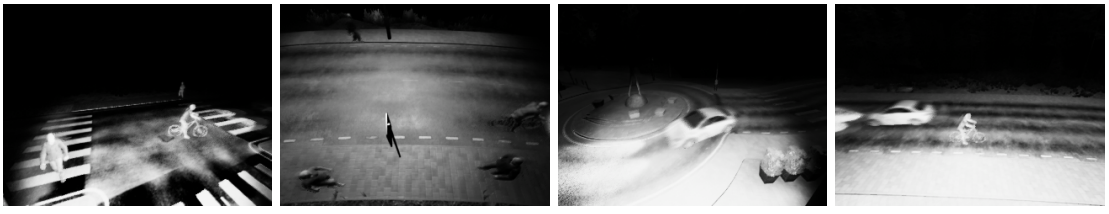


Figure 2.8: Synthetically generated scenes by *method 2*

Chapter 3

Evaluation of the detection method

The third chapter presents the testing results of the proposed detection system. The chapter describes the experimental setup, an evaluation of the detection algorithm accuracy, and a comparison of the accuracy to existing algorithms. The performance of the algorithm is estimated. The results presented in the third chapter were published in [146], [181].

3.1 Experimental setup

3.1.1 Real installation

Real data collection was carried out using several installations with various extrinsic and intrinsic parameters of the cameras. Four different cameras are used to confirm that the proposed feature extraction method can take the camera's intrinsic parameters into account. Real data was collected in six scenarios during the dark time of the day: two parking zones and three city streets (Figure 3.1). The daytime scenarios are arranged additionally to test the workability of the algorithm in daylight scenes. All the objects move at distances of up to 50 m. The camera installation height varies in the range of 3 to 5 m. All the images are grayscale and scaled to a resolution of 320×240 px. The low image resolution positively impacts the processing speed of low-performance computers. Examples of images are shown in Figure 3.1. Static objects are excluded from the dataset since the DBOD is an algorithm based on background subtraction that makes it possible to segment only objects which change location between successive frames.

A **The nighttime parking area** includes moving objects at distances up to 25 m. The scene contains pedestrians and cyclists with and without a flashlight. The web camera's IR-cut filter has been removed. The camera is equipped with an external IR emitter.

B Another **nighttime parking area** with pedestrians, cyclists with and without a flashlight, and vehicles with enabled car lights are present at distances up to 25 m. The camera with the removed IR-cut filter is equipped with an external IR emitter.

C **Nighttime city street (1)** includes pedestrians and cyclists with flashlights and vehicles with enabled car lights on distances up to 50 m. The camera has an embedded IR illumination.

D **Daytime city street (1)** includes moving objects at distances up to 50 m, particularly pedestrians, cyclists, and cars.

E **Nighttime city street (2)** contains moving objects at distances up to 30 m. Images are captured using a camera with a near-IR emitter. Pedestrians, bicyclists, and vehicles are present in this scene.

F **Daytime city street (2)** contains pedestrians, bicyclists, and vehicles moving at distances up to 30 m.

Table 3.1: Datasets parameters

Scene	**A**	**B**	**C**	**D**	**E**	**F**
Times of day	Night	Night	Night	Day	Night	Day
Description	Parking 1	Parking 2	City street 1		City street 2	
Camera	Logitech C920 HD	Microsoft HD-3000	Gadinan CCTV		RPi Camera v2	
Height	3 m	3.1 m	5 m		4 m	
Angle	13 deg	22 deg	17 deg		25 deg	
Focal length	3.67 mm	0.73 mm	3.6 mm		2.8 mm	
Sensor dim	4.8×3.6 mm	0.7×0.52 mm	3.45×1.94 mm		3.68×2.76 mm	
Objects distance	up to 25 m	up to 25 m	up to 50 m		up to 30 m	

(a) (b) (c) (d)

(e) (f)

Figure 3.1: Images from the real dataset

3.1.2 Synthetic installation (method 1)

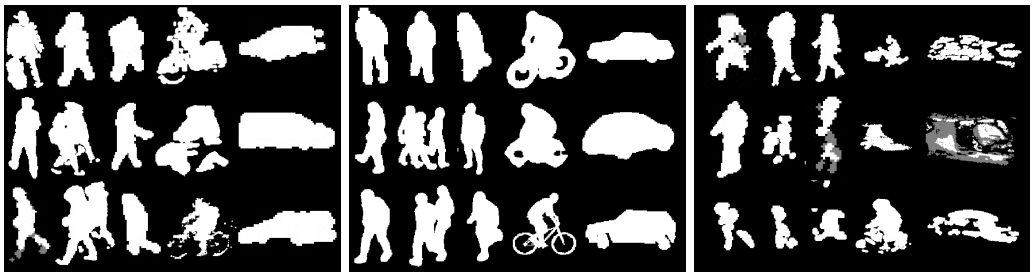
The synthetic setup contains scenes generated by the proposed perspective projection method. A camera is installed on heights from 2 to 12 m. The focal length of the camera varies from 2.8 to 3.6 mm. The target objects move at distances up to 35 m from the camera. The urban environment in scenes is not modeled.

3.1.3 Synthetic installation (method 2)

The synthetic setup contains eleven scenes in a city environment (buildings, streets, crossroads, street lights, sidewalks) obtained using Unity3D based smart-light urban simulator [144]. The simulated light sources are an infra-red camera spotlight, street lights, and vehicle/bicyclist lamps. A camera is set on street lights (height 3 to 5 m above the ground level) and directed to a modeled carriageway with sidewalks. Moving objects are located at distances up to 30 m from the camera. The camera's focal length varies from 2.8 to 3.6 mm. Such scene configuration is expected in the scenario of future smart lighting and smart city systems.

3.2 Datasets

Real foreground masks (Figure 3.2a) were used to validate synthetically generated images. The resulting contours of objects (Figure 3.2) are processed by the proposed feature extraction method to collect data for training the classifier.



(a) Foreground masks from real dataset (b) Result of masks generation by *method 1* (c) Result of masks generation by *method 2*

Figure 3.2: Binary masks of objects of different classes

Four datasets have been built for algorithm performance evaluation:

- Real nighttime dataset
- Real daytime dataset
- Synthetic nighttime dataset (*method 1*)
- Synthetic nighttime dataset (*method 2*)

Two real datasets have been gathered using the experimental setup described in Section 3.1. Two synthetic datasets have been generated by the described in Section 2.3 methods (*method 1* and *method 2*). The datasets do not include images captured during rainy, snowy, or windy weather. A proportion of a test split relative to a training split is 0.3 for all the scenarios. Characteristics of the datasets used for training and testing are shown in Table 3.2.

The distribution of the real features is shown in Figures 3.3 and 3.4, while synthetic features generated by different methods are shown in Figures 3.5 and 3.6.

During the day scenario, the amount of *noise* is significantly less than at night. Such difference is explained by the absence of moving light sources causing *noises* during the day. Due to the

Table 3.2: Characteristics of the dataset used for evaluating the DBOD algorithm

Dataset	Noise	Pedestrian	Bicyclist	Vehicle
Real night	3713	1929	1129	550
Real day	232	1709	682	287
Synth 1 night	1358	1169	1189	1249
Synth 2 night	5939	1783	1783	1793

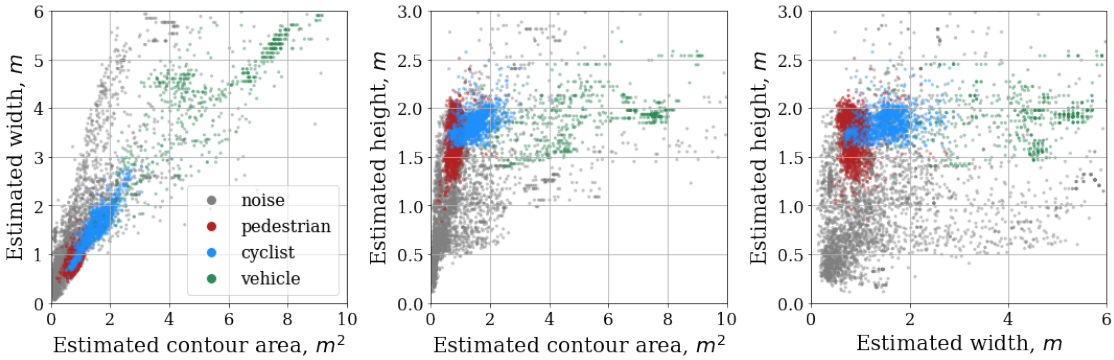


Figure 3.3: An example of real features distribution (night)

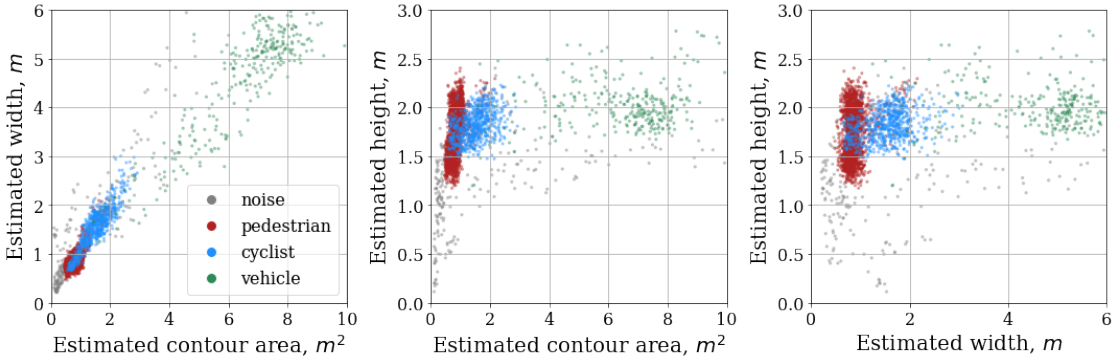


Figure 3.4: An example of real features distribution (day)

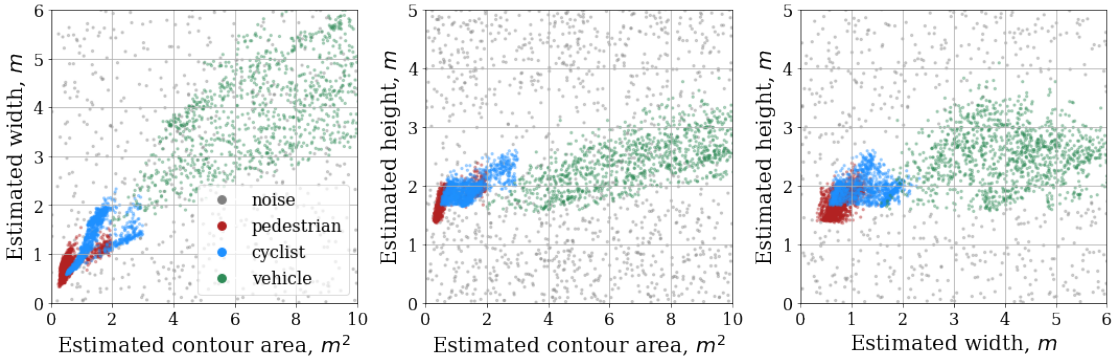


Figure 3.5: An example of synthetic features distribution generated by *method 1* (night)

chosen generation technique, the noises are distributed uniformly in the dataset generated by *method 1* (Figure 3.5). *Pedestrian* and *cyclist* class features are located close to each other, complicating their differentiation at certain angles of movement that can lead to classification

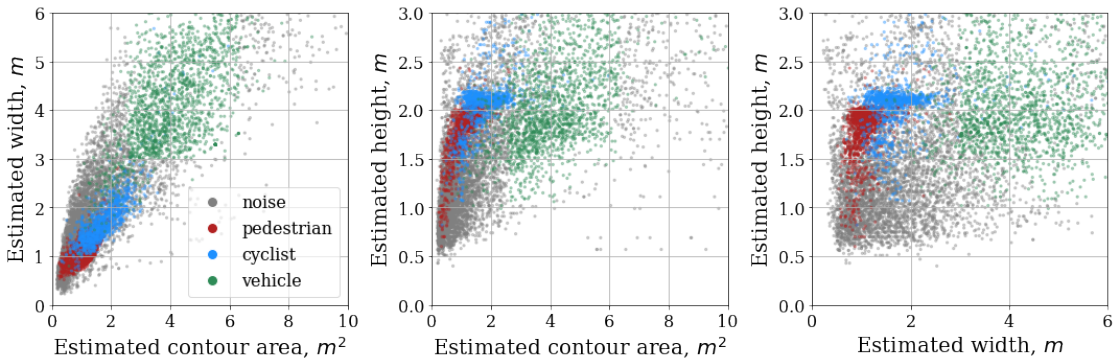


Figure 3.6: An example of synthetic features distribution generated by *method 2* (night)

errors. The geometrical features of the *vehicle* class make it possible to draw the decision-making boundary. The *vehicle's* features are distanced from the other classes in all the scenarios, as shown in the graphs. The spread of features for all the scenarios is in Appendix A. The number of features was increased by the polynomial combinations of the second degree to separate the *target class* from the surrounding *noise*.

3.3 Accuracy evaluation

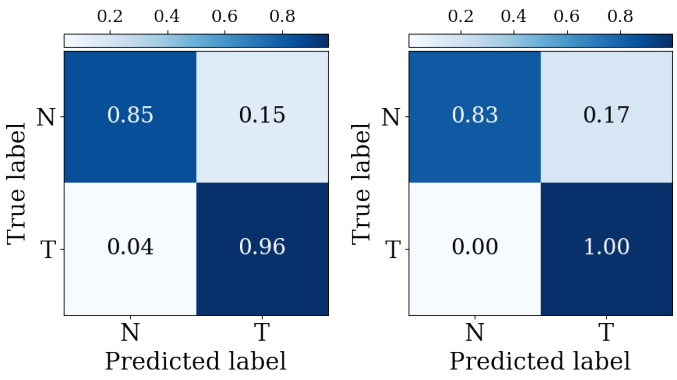
The accuracy of the proposed DBOD algorithm has been experimentally evaluated on the real day- and nighttime datasets. The accuracy of the algorithm trained on real and synthetic data has been compared. A comparison of accuracy between the DBOD and several state-of-the-art algorithms is also presented in this section.

3.3.1 Real scenarios during day and night

Annotated scenes have been divided into nighttime and daytime and, afterward, were classified by the trained logistic regression model. The result of classification can be summarized in corresponding confusion matrices.

Binary classification

According to Figures 3.7a and 3.7b (the darker color, the greater probability), the FN rate for the *target group* is higher at nighttime than at daytime (**0.04** vs. **0**). Such error rates can be explained by poorer segmentation of the *target group* on long distances due to insufficient IR illumination. The FP errors for the *target group* at night (**0.15**) are mainly caused by the interpretation of the object's lights as a target object, especially by reflections of the vehicle's lights from the ground and some vertical surfaces (i.e., windows). The same parameter for daytime scenes is **0.17**. These errors are frequently caused by wrong BS segmentation and object shadows interpreted as a target object. The amount of noises occurring in the nighttime scenario is noticeably larger than in the daytime scenario; however, their geometrical features allow distinguishing the *noise* class with a higher probability. The daylight scene illumination provides a better BS segmentation, which resulted in a high TP rate of **1** for the *target group*.



(a) Real nighttime (b) Real daytime

Figure 3.7: Binary confusion matrices (N - *noises*, T - *target group*)

Evaluation of the classifiers utilizing Average Precision has shown values of 0.93 for nighttime and 0.98 for daytime classifiers.

Multi-class classification

The problem of misclassification inside the *target group*, particularly for *pedestrians* and *cyclists*, remains relevant for both nighttime and daytime scenes, as shown in Figures 3.8a and 3.8b. Obtained FP rates for *pedestrians* toward *cyclists* are **0.24** for a night and **0.27** for day scenes. In these scenarios, the FP decisions appear when objects move under certain angles, which do not allow obtaining distinguishable features. A high FP rate is also related to intersecting parameters of classes, which are shown in Table 2.3.

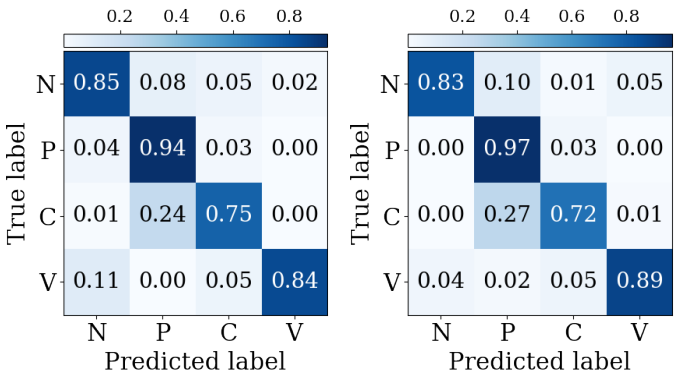
The *vehicle* has the lowest FP rates towards any other class due to the infrequency of 2D projection, wherein features of the *vehicle* and any other class of the *target group* are overlapped. The *vehicle* is interpreted as *noise* in some projections, resulting in a FN rate of **0.11** for night and **0.04** for day scenes. This rate can be reduced by extending the geometrical parameters of the *vehicle* from Table 2.3. However, this leads to precision reduction of the system in general.

Considering the precision-recall curves (Figure 3.8), the cyclist class shows the smallest area under the curve in both scenarios. The resulting macro average curve is similar in night and day datasets showing that the accuracy of the classifier is comparable in both scenarios.

Accuracy metrics are also summarized in Table 3.3. Obtained macro F1 and mAP scores show that the classifiers detect objects with high accuracy. The similar general accuracy of daytime and nighttime classifiers allows us to conclude that the proposed algorithm can operate in day and night scenarios with comparable performance.

3.3.2 Real vs. synthetic scenarios during night

Intelligent lighting systems are applied in streets with various environmental conditions like illumination level, traffic distribution, camera position, etc. It is often difficult to train a classification model on the data obtained from several real scenes. Instead, the model has to be trained using a substantial amount of generic urban scenes. However, obtaining the data containing various



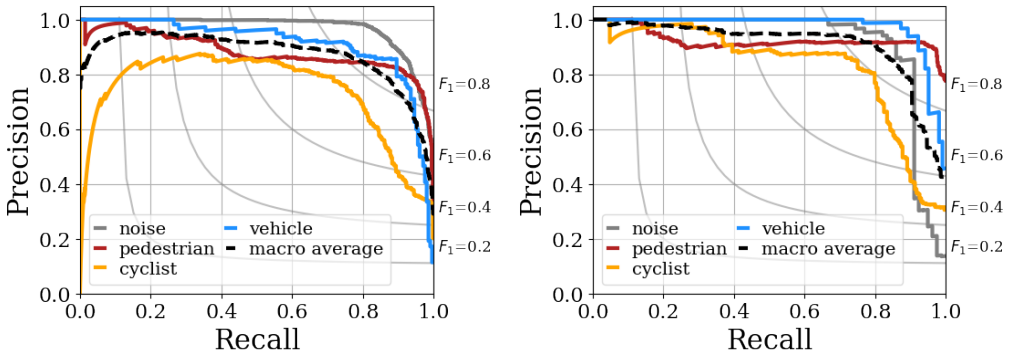
(a) Real nighttime (b) Real daytime

Figure 3.8: Multi-class confusion matrices (N - *noises*, P - *pedestrian*, C - *cyclist*, V - *vehicle*)

Table 3.3: Accuracy metrics for the DBOD algorithm in night and day scenarios

Class	**Metric**	**Real night**	**Real day**
Noise	F1	0.90	0.88
	Precision	0.85	0.83
	Recall	0.96	0.94
	AP	0.96	0.91
Pedestrian	F1	0.84	0.93
	Precision	0.94	0.97
	Recall	0.77	0.89
	AP	0.88	0.92
Bicyclist	F1	0.75	0.79
	Precision	0.75	0.72
	Recall	0.75	0.87
	AP	0.73	0.82
Vehicle	F1	0.90	0.91
	Precision	0.84	0.89
	Recall	0.87	0.94
	AP	0.91	0.97
All	**macro mAP**	0.87	0.91
	macro F1	0.84	0.88

scenes of an actual city is challenging. Complete reconstruction of several scenes identically, reflecting the real ones in a virtual environment, is ineffective since it restricts the training dataset. Therefore, the synthetic dataset represents a generic urban environment under various conditions. To evaluate the accuracy performance of the algorithm being trained on synthetic datasets, the following scenarios have been used:

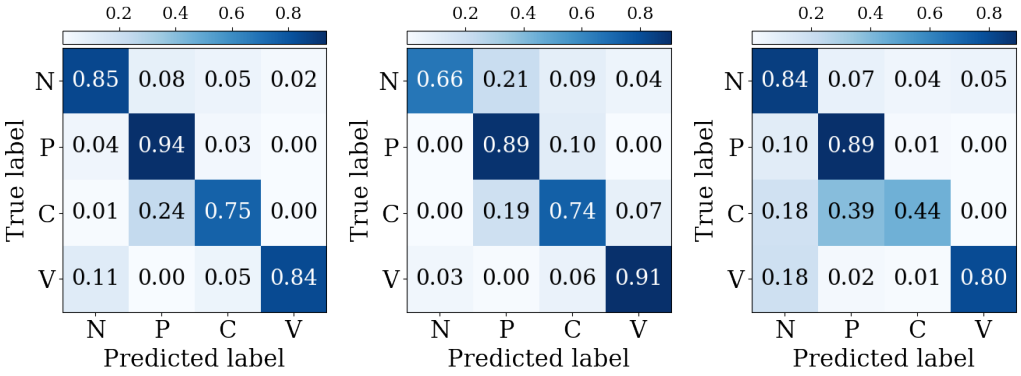


(a) Real nighttime (b) Real daytime

Figure 3.9: Precision-recall curves for real scenarios

- *Real - real* - the model is trained and tested using a real nighttime dataset only (Section 3.3.1).
- *Synth 1 - real* - training is performed on a synthetic nighttime dataset generated by *method 1*, while real images are used for testing.
- *Synth 2 - real* - the classifier is trained using synthetic nighttime data generated by *method 2* and tested on real images.

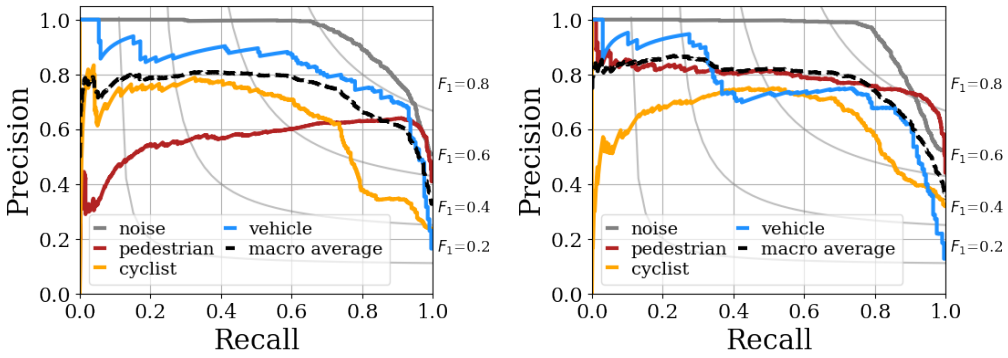
The *Synth 1 - real* and *Synth 2 - real* scenarios are used to prove the validity of the synthetic dataset relative to the real dataset. The classification result of the logistic regression is summarized in the corresponding confusion matrices (Figure 3.10).



(a) Real - real (b) Synth 1 - real (c) Synth 2 - real

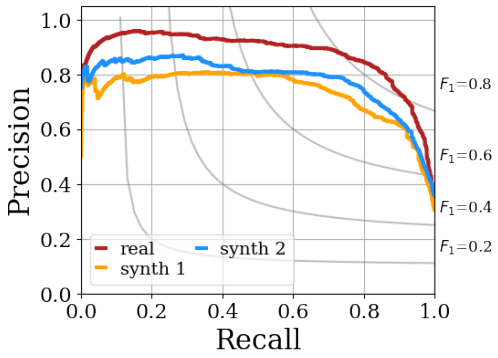
Figure 3.10: Multi-class confusion matrices

In all the test scenarios, there is a problem of misclassification between the *pedestrian* and *cyclist* classes. This problem is explained by the similar geometric parameters of these objects in the image plane at some angles of movement and can be seen in Figures 3.3, 3.5, and 3.6. The class *vehicle* has the lowest FP error rate relative to any other class of the *target group* due to the rare occurrence of such a 2D projection in which features of the *vehicle* and any other class intersect. However, in some projections, the *vehicle* can be interpreted as *noise*, leading to high values of the FN coefficient (Figure 3.10a and 3.10c).



(a) Synth 1 - real

(b) Synth 2 - real



(c) Macro average comparison

Figure 3.11: Real vs. synthetic precision-recall curves

From the obtained macro accuracy metrics (Table 3.4), it can be seen that the accuracy of the classifier trained using synthetic data is lower than in the direct scenario (Real - real). Comparing the methods for generating synthetic data, *method 2* showed the highest accuracy values, which is explained by the richer scene detailing during modeling.

3.3.3 Comparison to the existing algorithms

The state-of-the-art DL models have been compared to the proposed DBOD algorithm [176]. The selected models include one-stage (*YOLOv3 tiny*, *ssdlite_ mobilenet_ v2*, *ssd_ mobilenet_ v1*, *ssd_ mobilenet_ v2_ quantized*) and two-stage (*faster_ rcnn_ inception_ v2*) detectors.

YOLO has a special, "tiny" implementation for constrained environments such as single-board computers. Since the task requires real-time image processing on the IoT devices, the tiny version has been tested. The used *YOLOv3 tiny* model is implemented within a Darknet framework.

Sdlite_ mobilenet_ v2 is a light modification of the SSD model, providing slightly better detection performance than the original SSD model. It is based on MobileNetV2 neural network architecture.

Ssd_ mobilenet_ v1 is a classic SSD model based on MobileNetV1 neural network architecture. It provides better accuracy compared to SSDLite; however, the object detection speed might be less efficient.

Faster_ rcnn_ inception_ v2 implements the Faster R-CNN model, providing even better accuracy than SSD models. However, a particular disadvantage of this model can be the considerably

Table 3.4: Accuracy of the DBOD algorithm in different test scenarios

Class	**Metric**	**Real-real**	**Synth 1-real**	**Synth 2-Real**
Noise	F1	0.90	0.80	0.85
	Precision	0.85	0.99	0.86
	Recall	0.96	0.66	0.84
	AP	0.96	0.94	0.94
Pedestrian	F1	0.84	0.74	0.77
	Precision	0.94	0.63	0.70
	Recall	0.77	0.91	0.87
	AP	0.88	0.56	0.78
Bicyclist	F1	0.75	0.64	0.53
	Precision	0.75	0.60	0.74
	Recall	0.75	0.70	0.41
	AP	0.73	0.63	0.63
Vehicle	F1	0.90	0.78	0.75
	Precision	0.84	0.67	0.69
	Recall	0.87	0.91	0.83
	AP	0.91	0.80	0.76
All	**macro mAP**	0.87	0.73	0.78
	macro F1	0.84	0.74	0.73

higher amount of computational resources required during the detection process compared to the rest of the models. This drawback might negatively impact its deployment on the embedded devices.

Ssd_ mobilenet_ v2_ quantized is an 8-bit quantized SSD model, which provides worse accuracy but much higher detection speeds, which is essential on the embedded platforms. The model was trained using the TensorFlow framework and then was converted to a TFLite (TensorFlow Lite) format, primarily used on low-performance machines.

The above 4 DL models are part of a TensorFlow-v1 framework. The TensorFlow framework presents an open-source library for machine learning and AI, presenting a flexible set of tools to train and evaluate deep-learning models on given pre-labeled datasets. These machine learning models for object classification were selected from the official TensorFlow 1 Detection Model Zoo for further training and evaluation.

The models have been tested on real and synthesized (by *method 2*) test datasets. The obtained accuracy metrics for all the testing scenarios are represented in Table 3.5. The mAP is calculated at an IOU of 0.5. To evaluate the accuracy and performance of the algorithms being trained on real and synthetic datasets, the following four scenarios have been used:

- *Real-real* - the model is trained and tested using a real dataset.

- *Synth-synth* - the model is trained and tested using a synthetic dataset only.
- *Real-synth* - the classifier is trained using real data and tested on synthetic images.
- *Synth-real* - training is performed on a synthetic dataset, while real images are used for testing.

Table 3.5: Experimental quality metrics

	mAP; IOU = 0.5			
	Real-real	**Synth-synth**	**Real-synth**	**Synth-real**
DBOD	0.87	0.85	0.79	0.78
YOLOv3 tiny	0.97	0.96	0.03	0.07
ssdlite_mobilenet_v2	0.93	0.89	0.10	0.02
ssd_mobilenet_v1	0.88	0.85	0.17	0.01
faster_rcnn_inception_v2	0.98	0.95	0.10	0.36
ssd_mobilenet_v2_quantized	0.95	0.52	0.03	0.10

As shown in Table 3.5, the CNN-based models were trained well enough to recognize the objects of three classes on real and synthesized data (scenarios *Real-real* and *Synth-synth*) with high precision. However, these models struggle to be tested in cross scenarios (*Real-synth* and *Synth-real*), demonstrating very low precision. This result is caused by the difference between real and synthesized datasets in terms of scene parameters: illumination, image contrast, camera parameters, etc.

The R-CNN-based detector shows slightly better results in the *Synth-real* scenario; however, the achieved prediction accuracy is still not appropriate. In addition, this is the slowest and the most complicated model, as shown in Table 3.6.

Table 3.6: Average performance metrics of the models deployed on Raspberry Pi 4

Model Name	**FPS**	**CPU, %**	**Memory, MB**
DBOD	128.00	247	112
YOLOv3 tiny	0.46	373	122
ssdlite_mobilenet_v2	2.34	247	380
ssd_mobilenet_v1	2.21	257	395
faster_rcnn_inception_v2	0.19	377	767
ssd_mobilenet_v2_quantized	3.47	287	197
CPU and memory usage have been measured using top utility			

All the models were deployed on a Raspberry Pi 4 platform to collect real-time performance metrics for each tested model. The performance metrics are associated with how fast a given model handles the object detection, measured in an average number of processed frames per

second (FPS). Additional metrics included the consumed CPU and memory resources during the operation. CPU usage is related to the computational complexity of the model, memory usage depends on the framework optimization, and average FPS shows the algorithm's performance on the platform.

Tables 3.5 and 3.6 show that the DBOD algorithm is the fastest solution that can operate adequately in all the testing scenarios. In contrast to the DL models, which consider object appearance in an image, the algorithm shows appropriate accuracy in cross-testing (*Real-synth*, *Synth-real*) since the detection is performed based on rough geometrical measures of an object. Nevertheless, the accuracy in cross-testing is lower than in direct scenarios due to imperfections of the proposed simulation. Similar average accuracy metrics were obtained when testing the algorithm in direct scenarios (*Real-real* or *Synth-synth*).

3.3.4 Chapter conclusion

The experiments were carried out using urban environment datasets containing objects of the target group moving at speeds up to 60 km/h and distances at least 25 m. According to the results of the experiments on real daytime and nighttime datasets, the FP coefficient for the target group class did not exceed 0.17 in the case of binary noise/target group classification. The value of the Average Precision for the daytime classifier is 0.98, while the same metric for the nighttime classifier is 0.93.

Considering the multi-class classification, the daytime classifier showed a slightly higher accuracy ($mAP = 0.87$ vs. $mAP = 0.91$). The similar values of obtained mean Average Precision make it possible to detect objects by the proposed algorithm during day and night scenarios with comparable accuracy. The values of FP coefficients ($0.24 - 0.27$) in the case of the multi-class classification show that the algorithm frequently misclassifies cyclists as pedestrians.

It is preferable to use real data for training; however, in case of a lack of real data, the dataset can be substituted by the synthetic one, accompanied by detection accuracy reduction. In the case of training the model using synthetic data, the best results are achieved by the full scene reconstruction method (*method 2*), providing $mAP = 0.78$. The classifier trained on the data generated by the proposed *method 1* showed a lower accuracy of $mAP = 0.73$. The main advantage of the perspective projection method (*method 1*) for data synthesizing is the possibility to generate features from numerically given ranges of parameters (only initial object model formation is required to be done manually), avoiding detailed and time-consuming urban 3D scene design.

Comparison to modern DL models is performed using real and synthetic data (generated by *method 2*). In the testing *Synth-synth* and *Real-real* scenarios, the DL solutions mostly provide a more accurate detection than DBOD. The highest accuracy has been achieved by the Faster R-CNN model providing $mAP = 0.98$ in the *Real-real* scenario. The DBOD algorithm has the lowest accuracy in these scenarios ($mAP = 0.87$ in the *Real-real* scenario) compared to the CNNs.

In contrast, the DL models show extremely low accuracy in the cross-testing scenarios (*Real-synth* and *Synth-real*). Such results are explained by the peculiarities of synthetic and real

datasets (scenes have different physical parameters), described in Section 2.3. In most cases, it turned out that the training and test datasets have no common features, which do not allow training neural networks on the proposed synthetic data. The DBOD algorithm is more robust in these scenarios that is confirmed by the detection accuracy of $mAP = 78$ (*Synth-real*) and $mAP = 79$ (*Real-synth*).

Some studies show [182] that neural networks can be trained on synthetic data and have higher accuracy than DBOD, when the real environment is reconstructed identically. Nevertheless, with the absence of a sufficient amount of real data and the lack of accurate information about the environment characteristics, training the DBOD algorithm on generic synthetic data provides relatively high precision. At the same time, neural networks are ineffective in these conditions. It makes the DBOD algorithm the most suitable approach for intelligent lighting systems.

The one-stage detectors show the highest detection speed among the CNNs ($0.46 <= FPS_{SSD} <= 2.34$) on Raspberry Pi 4. Being a two-stage detector, Faster R-CNN is slower (FPS = 0.2). Thus, CNN-based methods can hardly detect fast-moving objects such as cars on the considered embedded computing platforms. The DBOD algorithm has the highest detection speed (FPS = 128). The achieved FPS indicates the high applicability of the proposed algorithm structure and the feature extraction method on devices with low computing power.

Conclusion

The main task of this dissertation was to develop a method for detecting objects in an urban environment that meets the requirements of intelligent lighting systems. The analysis of existing methods and algorithms, taking into account their compliance with the requirements for lighting systems, has shown that computer vision methods and techniques are the most appropriate approach for solving the problem.

The background subtraction method used for segmentation allows limiting the regions of interest to moving objects, reducing the redundant computations critical for real-time operation on low-performance computing devices. In addition, foreground segmentation by background subtraction reduces the likelihood of False Positive errors in the system due to the reduced number of regions of interest to be processed. As a result of empirical studies on nighttime datasets, the most suitable background subtraction method for use in the detection method, according to F1-measure and Frames per Second metrics, is the method based on the Mixture of Gaussians.

The proposed algorithm includes an original feature extraction method that uses intrinsic and extrinsic camera parameters for the calculation. The feature extraction method allows extracting unique geometric characteristics of objects from the image for classification by a statistical model based on logistic regression. Also, the method estimates the spatial coordinates of the object in the real world, which makes it possible to configure the size of the illumination area. A synthetic data generation technique was proposed to generate the required scenarios and extract training features from them in order to train the classifier. This technique minimizes the time spent collecting real reference data and the subsequent process of manual data annotation.

The algorithm's efficiency was confirmed by the accuracy and performance metrics obtained in experimental studies on real nighttime and daytime datasets, which include low-resolution single-channel images. The obtained accuracy on the nighttime dataset is slightly lower than on the daytime dataset. The obtained False Negative and False Positive error rates show a problem of misclassification inside the target group, particularly between a cyclist and pedestrian classes. In the case of binary classification, there is a noticeable False Positive error rate (0.17) for the target group toward the noise class.

The possibility of training the proposed Dimensional Based Object Detection algorithm by synthesized data has been studied. For that, two approaches for synthetic data generation were compared. The proposed approach provides more straightforward and faster feature generation based on numerically given scene parameters. In contrast, the classifier trained using the alternative approach provides higher accuracy. The accuracy of the classifier trained on synthetic data is lower than the accuracy of the classifier trained on real data. The synthetic features can still be used for training, when the amount of the real data is not sufficient.

The effectiveness of the developed method is also confirmed by the mean Average Precision metric, which has a value comparable to the modern detection methods based on convolutional neural networks. The developed method significantly outperforms existing CNN-based approaches in terms of Frames per Second on low-performance devices, making the detection of fast-moving

vehicles possible.

Despite the achieved efficiency, the Dimensional Based Object Detection algorithm has several peculiarities, requiring preliminary calibration of the camera, a specific position of the object in the frame relative to the ground surface, and known extrinsic camera parameters. In addition, the method cannot be applied to detect objects in scenes with a highly dynamic background or moving camera. The classification is based on the geometric characteristics of the object, which can lead to classification errors when the geometric shapes of the objects are similar.

Author's publications

[6] I. G. Matveev, A. S. Goponenko, A. V. Yurchenko, and M. V. Kovalev, "Development of the detection subsytem for smart lighting systems based on BeagleBone microcompiter," *Polzunovskij vestnik*, vol. 196, no. 3, pp. 126–130, 2015, ISSN: 2072-8921.

[7] I. Matveev, E. Siemens, D. A. Dugaev, and A. Yurchenko, "Development of the detection module for a SmartLighting system," in *Proc. of the 5th International Conference on Applied Innovations in IT, (ICAIIT)*, vol. 5, Koethen, Germany: Anhalt University of Applied Sciences, Mar. 16, 2017, pp. 87–94, ISBN: 978-3-96057-024-0.

[8] I. Matveev, E. Siemens, A. Yurchenko, and D. Kuznetsov, "Development and experimental investigations of motion detection module for smart lighting system," in *IOP Conf. Series: Materials Science and Engineering*, vol. 132, Tomsk, Russia, 2016, pp. 1–6. DOI: 10.1088/1757-899X/132/1/012010.

[9] I. G. Matveev and A. S. Goponenko, "Development of the detection module," in *Proceedings of Information and Measuring Techniques and Technologies Conference*, Tomsk, Russia: Tomsk Polytechnic Univ., 2015, pp. 247–250.

[10] A. S. Goponenko and I. G. Matveev, "Overview of motion and presence detection systems used in smart lighting systems," in *Proceedings of Information and Measuring Techniques and Technologies Conference*, Tomsk, Russia: Tomsk Polytechnic Univ., 2015, pp. 241–246.

[145] I. Chmielewski, I. Matveev, and E. Siemens, "Object detection based on analysis of a sequence of images," European pat. 3 611 655 A1, Feb. 19, 2020.

[146] I. Matveev, K. Karpov, I. Chmielewski, E. Siemens, and A. Yurchenko, "Fast object detection using dimensional based features for public street environments," *Smart Cities*, vol. 3, no. 1, pp. 93–111, Mar. 2020, Number: 1 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/smartcities3010006.

[147] I. Matveev, I. Chmielewsky, E. Siemens, and A. Yurchenko, "Method for object detection using analysis of near-infrared images processed by background subtraction techniques," in *Proceedings of Abstracts of XIV International conference ETAI 2018*, Struga, Macedonia, Sep. 20, 2018, pp. 28–29, ISBN: 978-9989-630-89-87.

[176] I. Matveev, K. Karpov, M. Iushchenko, *et al.*, "Comparative analysis of object detection methods in computer vision for low-performance computers towards smart lighting systems," *Progress in Advanced Information and Communication Technology and Systems*, vol. 2, p. 13, Oct. 2022, (Pending, accepted for publication), ISSN: 2367-3370.

[181] I. Matveev, K. Karpov, E. Siemens, and A. Yurchenko, "The object tracking algorithm using dimensional based detection for public street environment," *Eurasian Physical Technical Journal*, vol. 17, no. 2, pp. 123–127, Dec. 2020, ISSN: 1811-1165. DOI: 10.31489/2020No2/123-127.

References

[1] E. Siemens, "Method for lighting e.g. road, involves switching on the lamp on detection of movement of person, and sending message to neighboring lamps through communication unit," German pat. 102 010 049 121, International Classification H05B37/02; Cooperative Classification Y02B20/72, H05B37/0227, H05B37/0245, Y02B20/44, H05B37/0263; European Classification H05B37/02B4, H05B37/02B6, H05B37/02B6P, Apr. 26, 2012.

[2] S. Zinov and E. Siemens, "The smart lighting concept," in *Proceeding of the first Workshop on Problems of Autonomous Power Systems in the Siberian Region.*, Köthen, 2013.

[3] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, and T. R. Faughnan, "Real-time human detection as an edge service enabled by a lightweight CNN," in *2018 IEEE International Conference on Edge Computing (EDGE)*, Jul. 2018, pp. 125–129. DOI: 10.1109/EDGE.2018.00025.

[4] L. Wei Yang and C. Yen Su, "Low-cost CNN design for intelligent surveillance system," in *2018 International Conference on System Science and Engineering (ICSSE)*, ISSN: 2325-0909, Jun. 2018, pp. 1–4. DOI: 10.1109/ICSSE.2018.8520133.

[5] C. Kim, J. Lee, T. Han, and Y.-M. Kim, "A hybrid framework combining background subtraction and deep neural networks for rapid person detection," *Journal of Big Data*, vol. 5, no. 1, p. 22, Jul. 10, 2018, ISSN: 2196-1115. DOI: 10.1186/s40537-018-0131-x.

[6] I. G. Matveev, A. S. Goponenko, A. V. Yurchenko, and M. V. Kovalev, "Development of the detection subsytem for smart lighting systems based on BeagleBone microcompiter," *Polzunovskij vestnik*, vol. 196, no. 3, pp. 126–130, 2015, ISSN: 2072-8921.

[7] I. Matveev, E. Siemens, D. A. Dugaev, and A. Yurchenko, "Development of the detection module for a SmartLighting system," in *Proc. of the 5th International Conference on Applied Innovations in IT, (ICAIIT)*, vol. 5, Koethen, Germany: Anhalt University of Applied Sciences, Mar. 16, 2017, pp. 87–94, ISBN: 978-3-96057-024-0.

[8] I. Matveev, E. Siemens, A. Yurchenko, and D. Kuznetsov, "Development and experimental investigations of motion detection module for smart lighting system," in *IOP Conf. Series: Materials Science and Engineering*, vol. 132, Tomsk, Russia, 2016, pp. 1–6. DOI: 10.1088/1757-899X/132/1/012010.

[9] I. G. Matveev and A. S. Goponenko, "Development of the detection module," in *Proceedings of Information and Measuring Techniques and Technologies Conference*, Tomsk, Russia: Tomsk Polytechnic Univ., 2015, pp. 247–250.

[10] A. S. Goponenko and I. G. Matveev, "Overview of motion and presence detection systems used in smart lighting systems," in *Proceedings of Information and Measuring Techniques and Technologies Conference*, Tomsk, Russia: Tomsk Polytechnic Univ., 2015, pp. 241–246.

[11] "Speed and speed management," European Commission, Directorate General for Transport, Brussel, Belgium, Standard, Feb. 2018, 35 p.

[12] "La center road standards ordinance. chapter 12.10. public and private road standards," La Center, Washington, USA, Standard, 2009.

[13] "Code of practice (part-1)," Institute of Urban Transport, Delhi, Standard, 2012, 78 p.

[14] E. Yavari, H. Jou, V. Lubecke, and O. Boric-Lubecke, "Doppler radar sensor for occupancy monitoring," in *2013 IEEE Topical Conference on Power Amplifiers for Wireless and Radio Applications*, Jan. 2013, pp. 145–147. DOI: 10.1109/PAWR.2013.6490217.

[15] C. Canali, G. De Cicco, B. Morten, M. Prudenziati, and A. Taroni, "A temperature compensated ultrasonic sensor operating in air for distance and proximity measurements," *IEEE Transactions on Industrial Electronics*, vol. IE-29, no. 4, pp. 336–341, Nov. 1982, ISSN: 1557-9948. DOI: 10.1109/TIE.1982.356688.

[16] A. M. Zungeru, "Design and development of an ultrasonic motion detector," *International Journal of Security, Privacy and Trust Management*, vol. 2, no. 1, pp. 1–13, Feb. 28, 2013, ISSN: 23194103, 22775498. DOI: 10.5121/ijsptm.2013.2101.

[17] C. G. Raghavendra, S. Akshay, P. Bharath, M. Santosh, and D. Vishwas, "Object tracking and detection for short range surveillance using 2d ultrasonic sensor array," in *2016 International Conference on Circuits, Controls, Communications and Computing (I4C)*, Oct. 2016, pp. 1–4. DOI: 10.1109/CIMCA.2016.8053267.

[18] R. Stiawan, A. Kusumadjati, N. S. Aminah, M. Djamal, and S. Viridi, "An ultrasonic sensor system for vehicle detection application," *Journal of Physics: Conference Series*, vol. 1204, p. 012 017, Apr. 2019, ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1204/1/012017.

[19] T. Damarla, M. Bradley, A. Mehmood, and J. M. Sabatier, "Classification of animals and people ultrasonic signatures," *IEEE Sensors Journal*, vol. 13, no. 5, pp. 1464–1472, May 2013, ISSN: 2379-9153. DOI: 10.1109/JSEN.2012.2236550.

[20] V. Chen, Fayin Li, Shen-Shyang Ho, and H. Wechsler, "Micro-doppler effect in radar: Phenomenon, model, and simulation study," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 2–21, Jan. 2006, ISSN: 0018-9251. DOI: 10.1109/TAES.2006.1603402.

[21] M. Ecemis and P. Gaudiano, "Object recognition with ultrasonic sensors," in *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No.99EX375)*, Nov. 1999, pp. 250–255. DOI: 10.1109/CIRA.1999.810057.

[22] S. Jeon, E. Kwon, and I. Jung, "Traffic measurement on multiple drive lanes with wireless ultrasonic sensors," *Sensors*, vol. 14, no. 12, pp. 22 891–22 906, Dec. 2014, Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s141222891.

[23] A. Kianpisheh, N. Mustaffa, P. Limtrairut, and P. Keikhosrokiani, "Smart parking system (SPS) architecture using ultrasonic detector," *International Journal of Software Engineering and Its Application*, vol. 6, no. 3, pp. 51–58, Jul. 3, 2012.

[24] M. V. Paulet, A. Salceanu, and O. M. Neacsu, "Ultrasonic radar," in *2016 International Conference and Exposition on Electrical and Power Engineering (EPE)*, Oct. 2016, pp. 551–554. DOI: 10.1109/ICEPE.2016.7781400.

[25] J. S. Wilson, Ed., *Sensor technology handbook*, Amsterdam ; Boston: Elsevier, 2005, 691 pp., ISBN: 978-0-7506-7729-5.

[26] D. Caicedo and A. Pandharipande, "Ultrasonic array sensor for indoor presence detection," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, ISSN: 2219-5491, Aug. 2012, pp. 175–179.

[27] M. Pandey and G. Mishra, "Types of sensor and their applications, advantages, and disadvantages," in *Emerging Technologies in Data Mining and Information Security*, A. Abraham, P. Dutta, J. K. Mandal, A. Bhattacharya, and S. Dutta, Eds., ser. Advances in Intelligent Systems and Computing, Singapore: Springer, 2019, pp. 791–804, ISBN: 9789811315015. DOI: 10.1007/978-981-13-1501-5_69.

[28] A. Ekimov and J. M. Sabatier, "Passive ultrasonic method for human footstep detection," in *Unattended Ground, Sea, and Air Sensor Technologies and Applications IX*, vol. 6562, International Society for Optics and Photonics, May 4, 2007, p. 656 203. DOI: 10.1117/12.716899.

[29] ——, "Rhythm analysis of orthogonal signals from human walking," *The Journal of the Acoustical Society of America*, vol. 129, no. 3, pp. 1306–1314, Mar. 2011, ISSN: 1520-8524. DOI: 10.1121/1.3533694.

[30] T. Teixeira, G. Dublon, and A. Savvides, "A survey of human-sensing: Methods for detecting presence, count, location, track, and identity," ENALAB, Vol. 1, No. 1, Sep. 2010, p. 41.

[31] K. C. Liddiard, "PIR security sensors: Developing the next generation," in *Infrared Technology and Applications XXXIII*, vol. 6542, International Society for Optics and Photonics, May 14, 2007, pp. 1–9. DOI: 10.1117/12.719118.

[32] S. Akbas, M. A. Efe, and S. Ozdemir, "Performance evaluation of PIR sensor deployment in critical area surveillance networks," in *2014 IEEE International Conference on Distributed Computing in Sensor Systems*, May 2014, pp. 327–332. DOI: 10.1109/DCOSS.2014.56.

[33] S. Adarsh, S. M. Kaleemuddin, D. Bose, and K. I. Ramachandran, "Performance comparison of infrared and ultrasonic sensors for obstacles of different materials in vehicle/ robot navigation applications," *IOP Conference Series: Materials Science and Engineering*, vol. 149, p. 012 141, Sep. 2016, ISSN: 1757-8981, 1757-899X. DOI: 10.1088/1757-899X/149/1/012141.

[34] "PIR motion sensor. high density long distance detection type. EKMB/EKMC series.," Panasonic, Kadoma, Japan, Catalog-Datasheet, 2019.

[35] "Outdoor PIR sensors datasheet: Zc-pir-wall.," Atex Ltd, Dublin, Ireland, Catalog-Datasheet v1.3, May 31, 2017, p. 3.

[36] M. Kastek, H. Madura, and T. Sosnowski, "Passive infrared detector used for infrastructure protection," presented at the SAFE 2009, Rome, Italy, Jun. 19, 2009, pp. 61–70. DOI: 10.2495/SAFE090071.

[37] M. Kastek, T. Sosnowski, and T. Piątkowski, "Passive infrared detector used for detection of very slowly moving of crawling people," *Opto-Electronics Review*, vol. 16, no. 3, pp. 328–335, Sep. 1, 2008, ISSN: 1896-3757. DOI: 10.2478/s11772-008-0022-3.

[38] H. Madura, "Method of signal processing in passive infrared detectors for security systems," in *Computational Methods and Experimental Measurements XIII*, vol. I, Prague, Czech Republic: WIT Press, Jun. 11, 2007, pp. 757–768, ISBN: 978-1-84564-084-2. DOI: 10.2495/CMEM070741.

[39] "Grid-EYE infrared array sensor," Panasonic, Kadoma, Japan, Catalog-Datasheet 2021.4, Mar. 2021, p. 49.

[40] A. D. Shetty, Disha, S. B., and S. K., "Detection and tracking of a human using the infrared thermopile array sensor — "grid-EYE"," in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, Jul. 2017, pp. 1490–1495. DOI: 10.1109/ICICICT1.2017.8342790.

[41] M. Bertozzi, A. Broggi, A. Fascioli, T. Graf, and M. Meinecke, "Pedestrian detection for driver assistance using multiresolution infrared vision," *IEEE Transactions on Vehicular Technology*, vol. 53, no. 6, pp. 1666–1678, Nov. 2004, ISSN: 0018-9545. DOI: 10.1109/TVT.2004.834878.

[42] A. Fernandez-Caballero, M. Lopez, and J. Serrano-Cuerda, "Thermal-infrared pedestrian ROI extraction through thermal and motion information fusion," *Sensors*, vol. 14, no. 4, pp. 6666–6676, Apr. 10, 2014, ISSN: 1424-8220. DOI: 10.3390/s140406666.

[43] X. Zhao, Z. He, S. Zhang, and D. Liang, "Robust pedestrian detection in thermal infrared imagery using a shape distribution histogram feature and modified sparse representation classification," *Pattern Recognition*, vol. 48, no. 6, pp. 1947–1960, Jun. 1, 2015, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.12.013.

[44] E. Jeon, J.-S. Choi, J. Lee, *et al.*, "Human detection based on the generation of a background image by using a far-infrared light camera," *Sensors*, vol. 15, no. 3, pp. 6763–6788, Mar. 19, 2015, ISSN: 1424-8220. DOI: 10.3390/s150306763.

[45] D. A. Noyce, A. Gajendran, and R. Dharmaraju, "Development of bicycle and pedestrian detection and classification algorithm for active-infrared overhead vehicle imaging sensors," *Transportation Research Record*, vol. 1982, no. 1, pp. 202–209, Jan. 1, 2006, ISSN: 0361-1981. DOI: 10.1177/0361198106198200125.

[46] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1239–1258, Jul. 2010, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2009.122.

[47] A. Bartsch, F. Fitzek, and R. H. Rasshofer, "Pedestrian recognition using automotive radar sensors," in *Advances in Radio Science*, vol. 10, Copernicus GmbH, Sep. 18, 2012, pp. 45–55. DOI: https://doi.org/10.5194/ars-10-45-2012.

[48] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, Kauai, HI, USA: IEEE Comput. Soc, 2001, pp. I–511–I–518, ISBN: 978-0-7695-1272-3. DOI: 10.1109/CVPR.2001.990517.

[49] S. Guennouni, A. Ahaitouf, and A. Mansouri, "A comparative study of multiple object detection using haar-like feature selection and local binary patterns in several platforms," in *Modelling and Simulation in Engineering*, 2015. DOI: 0.1155/2015/948960.

[50] S. Ehsan, A. F. Clark, N. ur Rehman, and K. D. McDonald-Maier, "Integral images: Efficient algorithms for their computation and storage in resource-constrained embedded vision systems," *Sensors (Basel, Switzerland)*, vol. 15, no. 7, pp. 16 804–16 830, Jul. 10, 2015, ISSN: 1424-8220. DOI: 10.3390/s150716804.

[51] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 1, 2004, ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000013087.49260.fb.

[52] M. Chaudhari, S. sondur, and G. Vanjare, "A review on face detection and study of viola jones method," *International Journal of Computer Trends and Technology*, vol. 25, no. 1, pp. 54–61, Jul. 25, 2015, ISSN: 22312803. DOI: 10.14445/22312803/IJCTT-V25P110.

[53] M. H. Putra, Z. M. Yussof, S. I. M. Salim, and K.-C. Lim, "Convolutional neural network for person detection using YOLO framework," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 2, pp. 1–9, 2017, ISSN: 2289-8131.

[54] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *Proceedings of the British Machine Vision Conference*, Meeting Name: British Machine Visoin Conference (BMVC) 2009, London: BMVC Press, 2009, pp. 91.1–91.11, ISBN: 978-1-901725-39-1. DOI: 10.5244/C.23.91.

[55] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI: IEEE, Jun. 2012, pp. 2903–2910, ISBN: 978-1-4673-1228-8. DOI: 10.1109/CVPR.2012.6248017.

[56] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool, "Seeking the strongest rigid detector," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3666–3673.

[57] S. Zhang, R. Benenson, B. Schiele, *et al.*, "Filtered channel features for pedestrian detection.," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Issue: 2, vol. 1, Boston, MA, USA: IEEE, Jun. 7, 2015, p. 4. DOI: 10.1109/CVPR.2015.7298784.

[58] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Convolutional channel features," in *Proceedings of the IEEE international conference on computer vision*, Santiago, Chile: IEEE, Dec. 7, 2015, pp. 82–90. DOI: 10.1109/ICCV.2015.18.

[59] J. Sochman and J. Matas, "Waldboost-learning for time constrained sequential detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 2, San Diego, CA, USA: IEEE, 2005, pp. 150–156, ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.373.

[60] R. Juránek, "Detection of dogs in video using statistical classifiers," in *International Conference on Computer Vision and Graphics*, Springer, 2008, pp. 249–259.

[61] P. Zemcik, R. Juránek, P. Musil, M. Musil, and M. Hradis, "High performance architecture for object detection in streamed videos," in *2013 23rd International Conference on Field programmable Logic and Applications*, IEEE, 2013, pp. 1–4.

[62] A. Chesalin and S. Grodzenskiy, "Modification of the WaldBoost algorithm to improve the efficiency of solving problems of technical diagnostics of electrical systems," in *2019 International Conference on Electrotechnical Complexes and Systems (ICOECS)*, Oct. 2019, pp. 1–4. DOI: 10.1109/ICOECS46375.2019.8950002.

[63] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, Jun. 2005, pp. 886–893. DOI: 10.1109/CVPR.2005.177.

[64] R. C. Gonzalez, R. E. Woods, and B. R. Masters, *Digital Image Processing*, 3rd ed. United States: Prentice-Hall, Inc., 2006, 976 pp., ISBN: 978-0-13-168728-8.

[65] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv:1905.05055 [cs]*, May 13, 2019.

[66] Z. Luo, J. Chen, T. Takiguchi, and Y. Ariki, "Rotation-invariant histograms of oriented gradients for local patch robust representation," in *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, ISSN: null, Dec. 2015, pp. 196–199. DOI: 10.1109/APSIPA.2015.7415502.

[67] L. Jiao, F. Zhang, F. Liu, *et al.*, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2939201.

[68] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," *arXiv:1803.08375 [cs, stat]*, p. 7, Feb. 7, 2019. arXiv: 1803.08375.

[69] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 24, 2017, ISSN: 00010782. DOI: 10.1145/3065386.

[70] D. Mandic, "A generalized normalized gradient descent algorithm," *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 115–118, Feb. 2004, Conference Name: IEEE Signal Processing Letters, ISSN: 1558-2361. DOI: 10.1109/LSP.2003.821649.

[71] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, Apr. 2012, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2011.155.

[72] D. Tome, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi, and S. Tubaro, "Deep convolutional neural networks for pedestrian detection," *Signal Processing: Image Communication*, vol. 47, pp. 482–489, Sep. 1, 2016, ISSN: 0923-5965. DOI: 10.1016/j.image.2016.05.007.

[73] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool Publishers, Feb. 13, 2018, 209 pp., ISBN: 978-1-68173-022-6.

[74] J. P. Mueller and L. Massaron, *Deep Learning For Dummies*. John Wiley & Sons, Apr. 17, 2019, 425 pp., ISBN: 978-1-119-54303-9.

[75] "Intel neural compute stick 2," Intel, US, Datasheet 000055126, 2017.

[76] D. Pena, A. Forembski, X. Xu, and D. Moloney, "Benchmarking of CNNs for low-cost , low-power robotics applications," in *RSS 2017 Workshop: New Frontier for Deep Learning in Robotics*, 2017, p. 5.

[77] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jun. 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[78] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556 [cs]*, pp. 1–14, Apr. 10, 2015. arXiv: 1409.1556.

[79] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv:1512.03385 [cs]*, pp. 1–12, Dec. 10, 2015. arXiv: 1512.03385.

[80] A. G. Howard, M. Zhu, B. Chen, *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861 [cs]*, p. 9, Apr. 16, 2017.

[81] M. Z. Alom, T. M. Taha, C. Yakopcic, *et al.*, "The history began from AlexNet: A comprehensive survey on deep learning approaches," *arXiv:1803.01164 [cs]*, p. 39, Sep. 12, 2018.

[82] J. Johnson, *Benchmarks for popular CNN models*, original-date: 2016-07-13T06:46:20Z, Feb. 20, 2020.

[83] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv:1311.2524 [cs]*, p. 21, Oct. 22, 2014.

[84] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, Sep. 2013, ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-013-0620-5.

[85] K. Li, W. Ma, U. Sajid, Y. Wu, and G. Wang, "Object detection with convolutional neural networks," *arXiv:1912.01844 [cs]*, p. 28, Dec. 4, 2019.

[86] R. Girshick, "Fast r-CNN," presented at the Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

[87] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015, p. 14.

[88] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 779–788, ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.91.

[89] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[90] ——, "YOLOv3: An incremental improvement," *arXiv:1804.02767 [cs]*, Apr. 8, 2018.

[91] P. Soviany and R. T. Ionescu, "Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction," in *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, IEEE, 2018, pp. 209–214.

[92] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," *arXiv:1902.09630 [cs]*, Apr. 14, 2019. arXiv: 1902.09630.

[93] X. Ye and X. Ma, "Improved multi-object tracking algorithm for forward looking sonar based on rotation estimation," in *Intelligent Robotics and Applications*, H. Yu, J. Liu, L. Liu, Z. Ju, Y. Liu, and D. Zhou, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2019, pp. 171–183, ISBN: 978-3-030-27532-7. DOI: 10.1007/978-3-030-27532-7_15.

[94] M. Hossin and S. M.N, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, pp. 01–11, Mar. 31, 2015. DOI: 10.5121/ijdkp.2015.5201.

[95] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," *ArXiv*, 2020.

[96] N. Zeng. "An introduction to evaluation metrics for object detection," NickZeng. (Dec. 16, 2018), [Online]. Available: https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/ (visited on 01/22/2020).

[97] C. Yu, F. Li, G. Li, and N. Yang, "Multi-classes imbalanced dataset classification based on sample information," in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, ISSN: null, Aug. 2015, pp. 1768–1773. DOI: 10.1109/HPCC-CSS-ICESS.2015.327.

[98] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010, ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-009-0275-4.

[99] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[100] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft COCO: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_48.

[101] I. G. N. M. K. Raya, A. N. Jati, and R. E. Saputra, "Analysis realization of viola-jones method for face detection on CCTV camera based on embedded system," in *2017 International Conference on Robotics, Biomimetics, and Intelligent Computational Systems (Robionetics)*, Aug. 2017, pp. 1–5. DOI: 10.1109/ROBIONETICS.2017.8203427.

[102] N. A. OTHMAN and I. AYDIN, "A new deep learning application based on movidius NCS for embedded object detection and recognition," in *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Oct. 2018, pp. 1–5. DOI: 10.1109/ISMSIT.2018.8567306.

[103] T. G. Ostby, "Object detection and tracking on a raspberry pi using background subtraction and convolutional neural networks," Master Thesis, University College of Southeast Norway, Norway, 2018.

[104] A. Gunnarsson and M. Davidsson, "Real time object detection on a raspberry pi," Bachelor Degree Project, Linnaeus University, Faculty of Technology, Department of computer science and media technology (CM)., Växjö & Kalmar, Småland, Sweden, 2019, 20 pp.

[105] F. Mehmood, I. Ullah, S. Ahmad, and D. Kim, "Object detection mechanism based on deep learning algorithm using embedded IoT devices for smart home appliances control in CoT," *Journal of Ambient Intelligence and Humanized Computing*, p. 17, Mar. 22, 2019, ISSN: 1868-5145. DOI: 10.1007/s12652-019-01272-8.

[106] M. Noman, M. H. Yousaf, and S. A. Velastin, "An optimized and fast scheme for real-time human detection using raspberry pi," in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Nov. 2016, pp. 1–7. DOI: 10.1109/DICTA.2016.7797008.

[107] W. F. Abaya, J. Basa, M. Sy, A. C. Abad, and E. P. Dadios, "Low cost smart security camera with night vision capability using raspberry pi and OpenCV," in *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, Nov. 2014, pp. 1–6. DOI: 10.1109/HNICEM.2014.7016253.

[108] O. Durr, Y. Pauchard, D. Browarnik, R. Axthelm, and M. Loeser, "Deep learning on a raspberry pi for real time face recognition," p. 5, 2015. DOI: 10.2312/egp.20151036.

[109] M. Waseem Khan, "A survey: Image segmentation techniques," *International Journal of Future Computer and Communication*, pp. 89–93, 2014, ISSN: 20103751. DOI: 10.7763/IJFCC.2014.V3.274.

[110] A. Norouzi, M. S. M. Rahim, A. Altameem, *et al.*, "Medical image segmentation methods, algorithms, and applications," *IETE Technical Review*, vol. 31, no. 3, pp. 199–213, May 4, 2014, ISSN: 0256-4602. DOI: 10.1080/02564602.2014.906861.

[111] J. Acharya, S. Gadhiya, and K. Raviya, "Segmentation techniques for image analysis: A review," *Int. J. Comp. Sci. and Manage. Research*, vol. 2, pp. 1218–1221, Jan. 1, 2013.

[112] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4–21, May 1, 2014, ISSN: 1077-3142. DOI: 10.1016/j.cviu.2013.12.005.

[113] Y. Dhome, N. Tronson, A. Vacavant, *et al.*, "A benchmark for background subtraction algorithms in monocular vision: A comparative study," in *2010 2nd International Conference on Image Processing Theory, Tools and Applications*, ISSN: 2154-5111, Jul. 2010, pp. 66–71. DOI: 10.1109/IPTA.2010.5586792.

[114] C. Lallier, E. Reynaud, L. Robinault, and L. Tougne, "A testing framework for background subtraction algorithms comparison in intrusion detection context," in *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug. 2011, pp. 314–319. DOI: 10.1109/AVSS.2011.6027343.

[115] J. S. Lumentut, F. E. Gunawan, and Diana, "Evaluation of recursive background subtraction algorithms for real-time passenger counting at bus rapid transit system," in *International Conference on Computer Science and Computational Intelligence (ICCSCI 2015)*, vol. 59, Jan. 1, 2015, pp. 445–453. DOI: 10.1016/j.procs.2015.07.565.

[116] D. H. Parks and S. S. Fels, "Evaluation of background subtraction algorithms with post-processing," in *2008 IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance*, Sep. 2008, pp. 192–199. DOI: 10.1109/AVSS.2008.19.

[117] P.-M. Jodoin, "Comparative study of background subtraction algorithms," *Journal of Electronic Imaging*, vol. 19, no. 3, p. 033 003, Jul. 1, 2010, ISSN: 1017-9909. DOI: 10.1117/1.3456695.

[118] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, Sep. 1999, 255–261 vol.1. DOI: 10.1109/ICCV.1999.791228.

[119] S.-C. S. Cheung and C. Kamath, "Robust background subtraction with foreground validation for urban traffic video," *EURASIP Journal on Advances in Signal Processing*, vol. 2005, no. 14, p. 726 261, Aug. 25, 2005, ISSN: 1687-6180. DOI: 10.1155/ASP.2005.2330.

[120] T. Bouwmans, F. E. Baf, and B. Vachon, "Background modeling using mixture of gaussians for foreground detection - a survey," *Recent Patents on Computer Science*, vol. 1, no. 3, pp. 219–237, 2008. DOI: 10.2174/1874479610801030219.

[121] K. Sehairi, C. Fatima, and J. Meunier, "Comparative study of motion detection methods for video surveillance systems," *Journal of Electronic Imaging*, vol. 26, no. 2, pp. 1–30, Apr. 25, 2017, ISSN: 1017-9909. DOI: 10.1117/1.JEI.26.2.023025. arXiv: 1804.05459.

[122] T. Ko, S. Soatto, and D. Estrin, "Background subtraction on distributions," in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2008, pp. 276–289, ISBN: 978-3-540-88690-7. DOI: 10.1007/978-3-540-88690-7_21.

[123] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1168–1177, Jul. 2008, ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2008.924285.

[124] G. Jing, C. E. Siong, and D. Rajan, "Foreground motion detection by difference-based spatial temporal entropy image," in *2004 IEEE Region 10 Conference TENCON 2004.*, vol. A, Nov. 2004, 379–382 Vol. 1. DOI: 10.1109/TENCON.2004.1414436.

[125] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984, ISSN: 1939-3539. DOI: 10.1109/TPAMI.1984.4767596.

[126] X.-j. Tan, J. Li, and C. Liu, "A video-based real-time vehicle detection method by classified background learning," *World transactions on engineering and technology education*, vol. 6 (1), pp. 189–193, 2007.

[127] J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," in *Proceedings Second IEEE Workshop on Visual Surveillance (VS'99) (Cat. No.98-89223)*, Jun. 1999, pp. 74–81. DOI: 10.1109/VS.1999.780271.

[128] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, Jul. 1, 1997, ISSN: 0162-8828. DOI: 10.1109/34.598236.

[129] T. Trnovszký, P. Sýkora, and R. Hudec, "Comparison of background subtraction methods on near infra-red spectrum video sequences," *Procedia Engineering*, 12th international scientific conference of young scientists on sustainable, modern and safe transport, vol. 192, pp. 887–892, Jan. 1, 2017, ISSN: 1877-7058. DOI: 10.1016/j.proeng.2017.06.153.

[130] A. B. Godbehere, A. Matsukawa, and K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation," in *2012 American Control Conference (ACC)*, Montreal, QC: IEEE, Jun. 2012, pp. 4305–4312, ISBN: 978-1-4577-1096-4. DOI: 10.1109/ACC.2012.6315174.

[131] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, May 2006, ISSN: 01678655. DOI: 10.1016/j.patrec.2005.11.005.

[132] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems*, Springer, Boston, MA, 2002, pp. 135–144, ISBN: 978-1-4613-5301-0. DOI: 10.1007/978-1-4615-0913-4_11.

[133] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004*, vol. 2, Cambridge, UK: IEEE, Aug. 2004, pp. 28–31, ISBN: 978-0-7695-2128-2. DOI: 10.1109/ICPR.2004.1333992.

[134] *Opencv/cvat*, original-date: 2018-06-29T14:02:45Z, Jan. 21, 2020.

[135] A. Sobral, *BGSLibrary: An OpenCV c++ background subtraction library*, Rio de Janeiro, Brazil, 2013.

[136] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequièvre, "A benchmark dataset for outdoor foreground/background extraction," in *Computer Vision - ACCV 2012 Workshops*, J.-I. Park and J. Kim, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2013, pp. 291–300, ISBN: 978-3-642-37410-4. DOI: 10.1007/978-3-642-37410-4_25.

[137] "A video database for testing change detection algorithms," Change Detection. (), [Online]. Available: http://changedetection.net/ (visited on 01/20/2020).

[138] S. Tommesani. "Comparing background subtraction algorithms - stefano tommesani," Comparing background subtraction algorithms. (Aug. 16, 2013), [Online]. Available: http://tommesani.com/index.php/video/comparing-background-subtraction-algorithms.html (visited on 01/20/2020).

[139] T. Pfeifer and D. Elias, "Commercial hybrid IR/RF local positioning system," in *KiVS Kurzbeiträge*, University of Leipzig, Germany, 2003, pp. 119–127, ISBN: 978-3-8007-2753-7.

[140] Y. W. Bai, C. C. Cheng, and Z. L. Xie, "Use of ultrasonic signal coding and PIR sensors to enhance the sensing reliability of an embedded surveillance system," in *2013 IEEE International Systems Conference (SysCon)*, Apr. 2013, pp. 287–291. DOI: 10.1109/SysCon.2013.6549895.

[141] B. Mustapha, A. Zayegh, and R. K. Begg, "Ultrasonic and infrared sensors performance in a wireless obstacle detection system," in *Modelling and Simulation 2013 1st International Conference on Artificial Intelligence*, Dec. 2013, pp. 487–492. DOI: 10.1109/AIMS.2013.89.

[142] J. DuCarme, "Developing effective proximity detection systems for underground coal mines," in *Advances in Productive, Safe, and Responsible Coal Mining*, J. Hirschi, Ed., Woodhead Publishing, Jan. 1, 2019, pp. 101–119, ISBN: 978-0-08-101288-8. DOI: 10.1016/B978-0-08-101288-8.00003-1.

[143] T.-S. Chu, J. Roderick, S. Chang, T. Mercer, C. Du, and H. Hashemi, "A short-range UWB impulse-radio CMOS sensor for human feature detection," in *2011 IEEE International Solid-State Circuits Conference*, ISSN: 0193-6530, Feb. 2011, pp. 294–296. DOI: 10.1109/ISSCC.2011.5746325.

[144] K. Karpov, I. Luzianin, M. Iushchenko, and E. Siemens, "Urban environment simulator for train data generation toward CV object recognition," in *Proceedings of International Conference on Applied Innovations in IT (ICAIIT)*, Koethen, Germany, 2021. DOI: 10.25673/36585.

[145] I. Chmielewski, I. Matveev, and E. Siemens, "Object detection based on analysis of a sequence of images," European pat. 3 611 655 A1, Feb. 19, 2020.

[146] I. Matveev, K. Karpov, I. Chmielewski, E. Siemens, and A. Yurchenko, "Fast object detection using dimensional based features for public street environments," *Smart Cities*, vol. 3, no. 1, pp. 93–111, Mar. 2020, Number: 1 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/smartcities3010006.

[147] I. Matveev, I. Chmielewsky, E. Siemens, and A. Yurchenko, "Method for object detection using analysis of near-infrared images processed by background subtraction techniques," in *Proceedings of Abstracts of XIV International conference ETAI 2018*, Struga, Macedonia, Sep. 20, 2018, pp. 28–29, ISBN: 978-9989-630-89-87.

[148] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, Aug. 1987, ISSN: 2374-8710. DOI: 10.1109/JRA.1987.1087109.

[149] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000, ISSN: 01628828. DOI: 10.1109/34.888718.

[150] A. Hardas, D. Bade, and V. Wali, "Moving object detection using background subtraction, shadow removal and post processing," in *IJCA Proceedings on International Conference on Computer Technology*, Sep. 19, 2015, pp. 1–5.

[151] M. Piccardi, "Background subtraction techniques: A review," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, ISSN: 1062-922X, vol. 4, Oct. 2004, 3099–3104 vol.4. DOI: 10.1109/ICSMC.2004.1400815.

[152] R. G. Abbott and L. R. Williams, "Multiple target tracking with lazy background subtraction and connected components analysis," *Machine Vision and Applications*, vol. 20, no. 2, pp. 93–101, Feb. 1, 2009, ISSN: 1432-1769. DOI: 10.1007/s00138-007-0109-8.

[153] W. Burger, "Zhang's camera calibration algorithm: In-depth tutorial and implementation," Hagenberg, Austria, 2016, p. 55. DOI: 10.13140/RG.2.1.1166.1688.

[154] D. C. Brown, "Close-range camera calibration," *Photogrammetric Engineering*, vol. 37, no. 8, pp. 855–866, 1971.

[155] W. Hugemann, "Correcting lens distortions in digital photographs," Ingenieurbüro Morawski + Hugemann, Leverkusen, Germany, 2010, p. 12.

[156] G. Yadav, S. Maheshwari, and A. Agarwal, "Contrast limited adaptive histogram equalization based enhancement for real time video system," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2014, pp. 2392–2397. DOI: 10.1109/ICACCI.2014.6968381.

[157] CCTV Aware. "Hikvision 4k dome IR footage whilst raining." (Apr. 28, 2018), [Online]. Available: https://www.youtube.com/watch?v=6Gii-F-FpRo (visited on 07/30/2022).

[158] Mike Miller. "Blink security camera shows what rain looks like in night vision." (Dec. 30, 2018), [Online]. Available: https://www.youtube.com/watch?v=njfPo-JToKY (visited on 07/30/2022).

[159] Supercircuits. "Alibi ALI-IPU3030rv IP camera: Parking lot in rain." (Aug. 20, 2014), [Online]. Available: https://www.youtube.com/watch?v=kVbVQ_i1p3E (visited on 07/30/2022).

[160] gatorpics09. "Testing sharx security camera SCNC3905 during a rain storm." (Mar. 24, 2015), [Online]. Available: https://www.youtube.com/watch?v=NXeHOIR2ooI (visited on 07/30/2022).

[161] SONALI RUPALI GRAM. "Snowfall cctv footge." (Feb. 19, 2021), [Online]. Available: https://www.youtube.com/watch?v=dbVo99OEN94 (visited on 07/31/2022).

[162] Relaxing Sounds Of Nature. "Relaxing snowfall 2 hours - sound of light wind breeze and falling snow in forest (part 2)." (Mar. 14, 2018), [Online]. Available: https://www.youtube.com/watch?v=vz91QpgUjFc (visited on 07/31/2022).

[163] Cat Trumpet. "Relaxing snowfall ~ heavy falling snow & the best relax music." (Nov. 27, 2016), [Online]. Available: https://www.youtube.com/watch?v=56WjHWD7fAo (visited on 07/31/2022).

[164] P. Sturm, "Pinhole camera model," in *Computer Vision: A Reference Guide*, K. Ikeuchi, Ed., Boston, MA: Springer US, 2014, pp. 610–613, ISBN: 978-0-387-31439-6. DOI: 10.1007/978-0-387-31439-6_472.

[165] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine Learning*, vol. 40, no. 3, pp. 203–228, Sep. 1, 2000, ISSN: 1573-0565. DOI: 10.1023/A:1007608224229.

[166] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, 2nd ed., ser. Springer Series in Statistics. New York: Springer-Verlag, 2009, ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7.

[167] M. Müller, "Generalized linear models," in *XploRe — Learning Guide*, W. Härdle, S. Klinke, and M. Müller, Eds., Berlin, Heidelberg: Springer, 2000, pp. 205–228, ISBN: 978-3-642-60232-0. DOI: 10.1007/978-3-642-60232-0_7.

[168] N.-H. Ho, P. H. Truong, and G.-M. Jeong, "Step-detection and adaptive step-length estimation for pedestrian dead-reckoning at various walking speeds using a smartphone," *Sensors (Basel, Switzerland)*, vol. 16, no. 9, Sep. 2, 2016, ISSN: 1424-8220. DOI: 10.3390/s16091423.

[169] S. Buchmüller and U. Weidmann, "Parameters of pedestrians, pedestrian traffic and walking facilities," ETH Zurich, Report, 2006, p. 57. DOI: 10.3929/ethz-b-000047950.

[170] T. Lee, M. Lim, T. Prather, and C. Welch, "Collision mitigation system: Pedestrian test target final design report," *Mechanical Engineering*, p. 340, Jun. 1, 2017.

[171] B. Blocken, T. v. Druenen, Y. Toparlar, *et al.*, "Aerodynamic drag in cycling pelotons: New insights by CFD simulation and wind tunnel testing," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 179, pp. 319–337, Aug. 1, 2018, ISSN: 0167-6105. DOI: 10.1016/j.jweia.2018.06.011.

[172] D. Fintelman, H. Hemida, M. Sterling, and F.-X. Li, "CFD simulations of the flow around a cyclist subjected to crosswinds," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 144, pp. 31–41, Sep. 2015, ISSN: 01676105. DOI: 10.1016/j.jweia.2015.05.009.

[173] M. S. Tarawneh, "Evaluation of pedestrian speed in jordan with investigation of some contributing factors," *Journal of Safety Research*, vol. 32, no. 2, pp. 229–236, Jun. 1, 2001, ISSN: 0022-4375. DOI: 10.1016/S0022-4375(01)00046-9.

[174] M. G. J. Gazendam and A. L. Hof, "Averaged EMG profiles in jogging and running at different speeds," *Gait & Posture*, vol. 25, no. 4, pp. 604–614, Apr. 1, 2007, ISSN: 0966-6362. DOI: 10.1016/j.gaitpost.2006.06.013.

[175] A. Kassim, L. Pascoe, K. Ismail, and A. E. H. Abd El Halim, "Vision-based analysis of cyclists' speed," presented at the Transportation Research Board 91st Annual Meeting, Number: 12-3441, Washington, US: Transportation Research Board, 2012, p. 17.

[176] I. Matveev, K. Karpov, M. Iushchenko, *et al.*, "Comparative analysis of object detection methods in computer vision for low-performance computers towards smart lighting systems," *Progress in Advanced Information and Communication Technology and Systems*, vol. 2, p. 13, Oct. 2022, (Pending, accepted for publication), ISSN: 2367-3370.

[177] "3d models for professionals: TurboSquid." (), [Online]. Available: https://www.turbosquid.com/ (visited on 03/25/2020).

[178] J. Zhang, F. He, and W. Li, "Motion blurring direction identification based on second-order difference spectrum," in *Computer and Computing Technologies in Agriculture IV*, D. Li, Y. Liu, and Y. Chen, Eds., vol. 345, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 102–109, ISBN: 978-3-642-18335-5.

[179] F. Brusius, U. Schwanecke, and P. Barth, "Blind image deconvolution of linear motion blur," in *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, G. Csurka, M. Kraus, L. Mestetskiy, P. Richard, and J. Braz, Eds., vol. 274, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 105–119, ISBN: 978-3-642-32349-2.

[180] R. L. Lagendijk and J. Biemond, "Chapter 14 - basic methods for image restoration and identification," in *The Essential Guide to Image Processing*, A. Bovik, Ed., Boston: Academic Press, Jan. 1, 2009, pp. 323–348, ISBN: 978-0-12-374457-9.

[181] I. Matveev, K. Karpov, E. Siemens, and A. Yurchenko, "The object tracking algorithm using dimensional based detection for public street environment," *Eurasian Physical Technical Journal*, vol. 17, no. 2, pp. 123–127, Dec. 2020, ISSN: 1811-1165. DOI: 10.31489/2020No2/123-127.

[182] H. Hattori, V. Naresh Boddeti, K. M. Kitani, and T. Kanade, "Learning scene-specific pedestrian detectors without real data," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3819–3827.

Appendices

Appendix A

Distribution of features

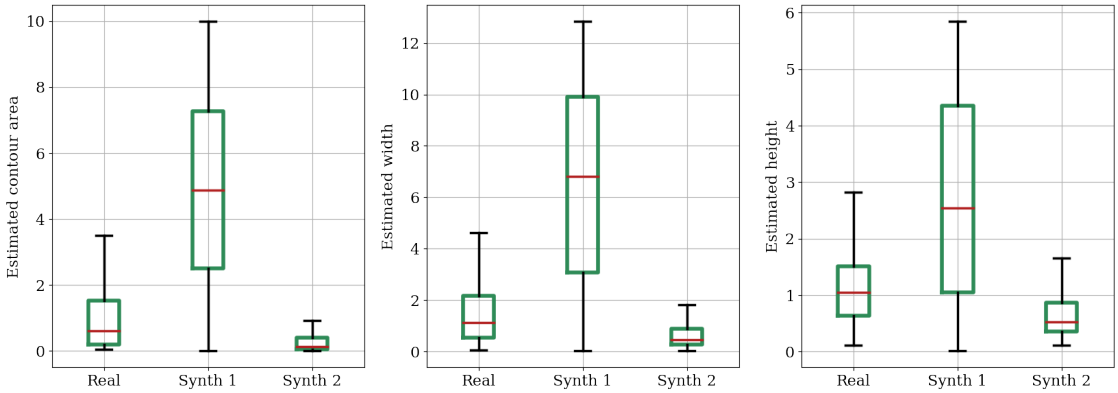


Figure A.1: Spread of features generated by different methods — noises

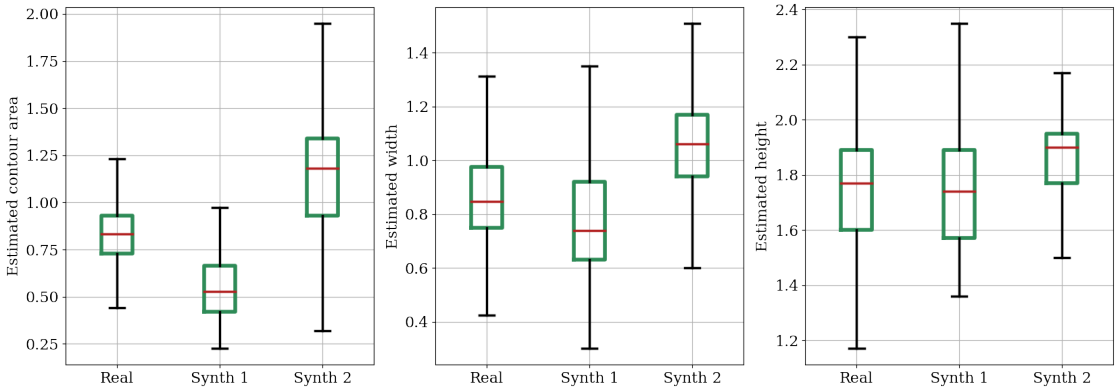


Figure A.2: Spread of features generated by different methods — pedestrian

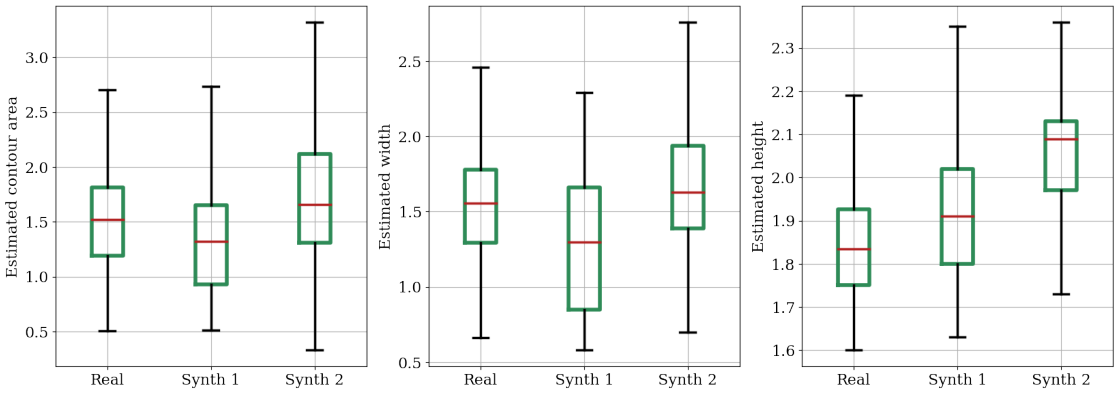


Figure A.3: Spread of features generated by different methods — cyclist

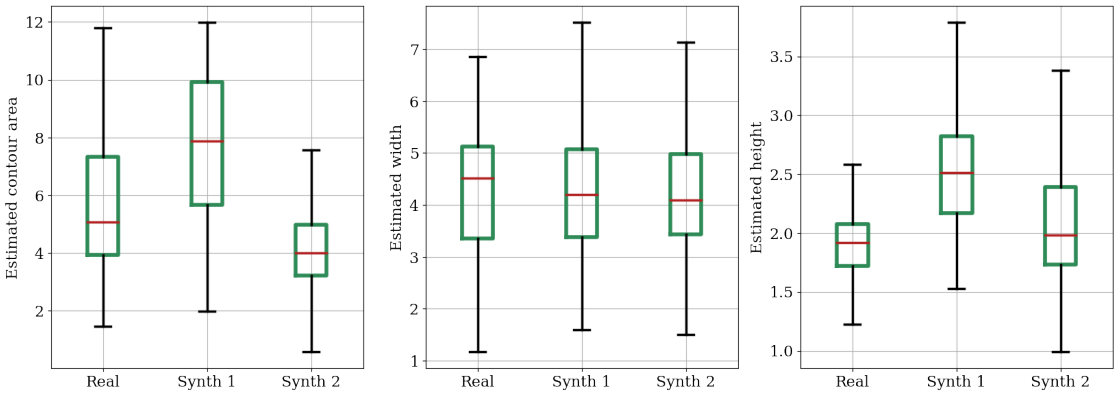


Figure A.4: Spread of features generated by different methods — vehicle

Appendix B

Detection examples

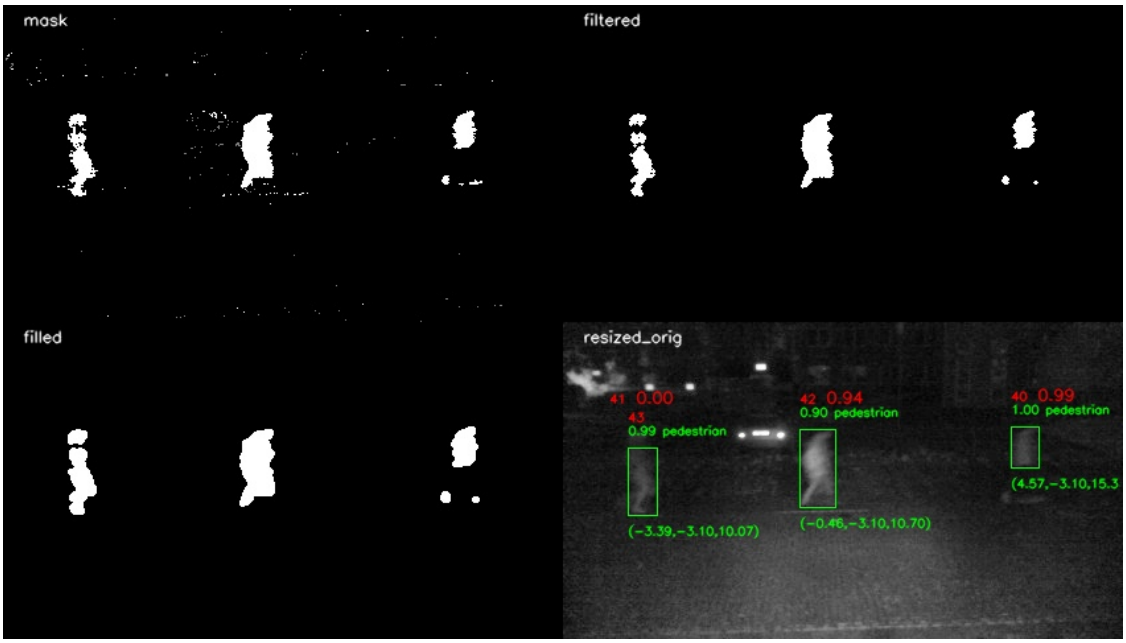


Figure B.1: Correct detection of three pedestrians

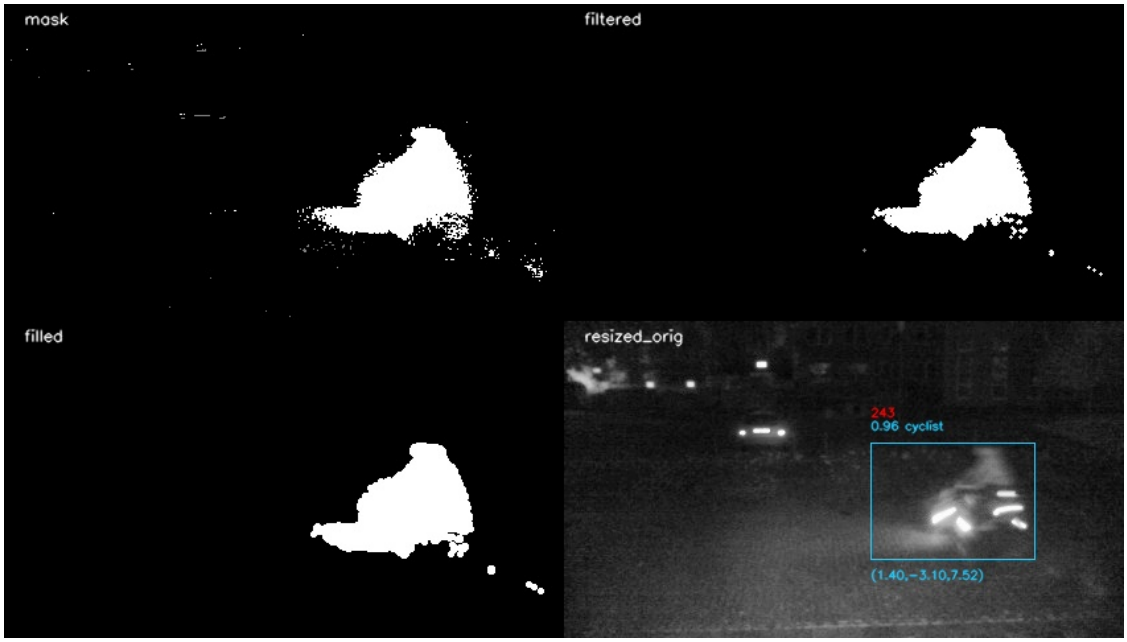


Figure B.2: Correct detection of a bicyclist

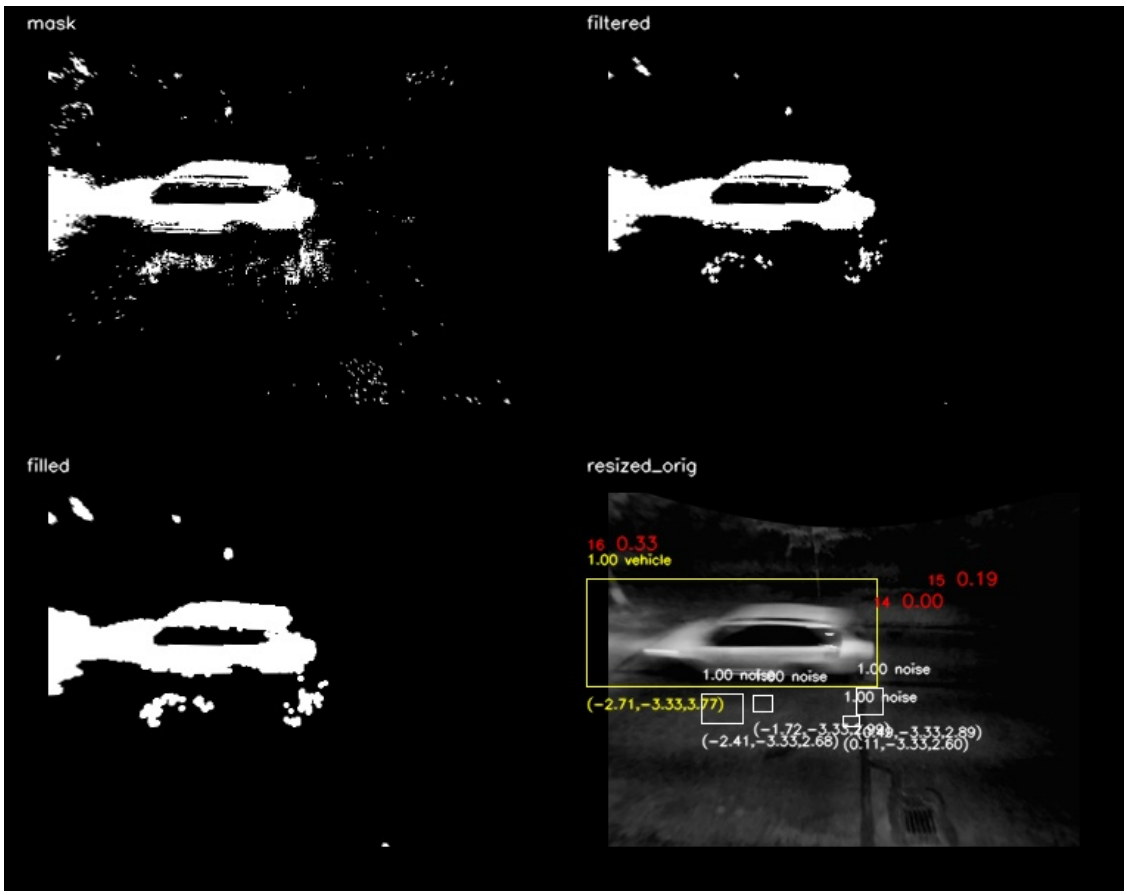


Figure B.3: Correct detection of a vehicle

Figure B.4: False negative pedestrian detection

Only one (object 6) out three pedestrians was detected due to insufficient segmentation at a distance of more than 30 m.

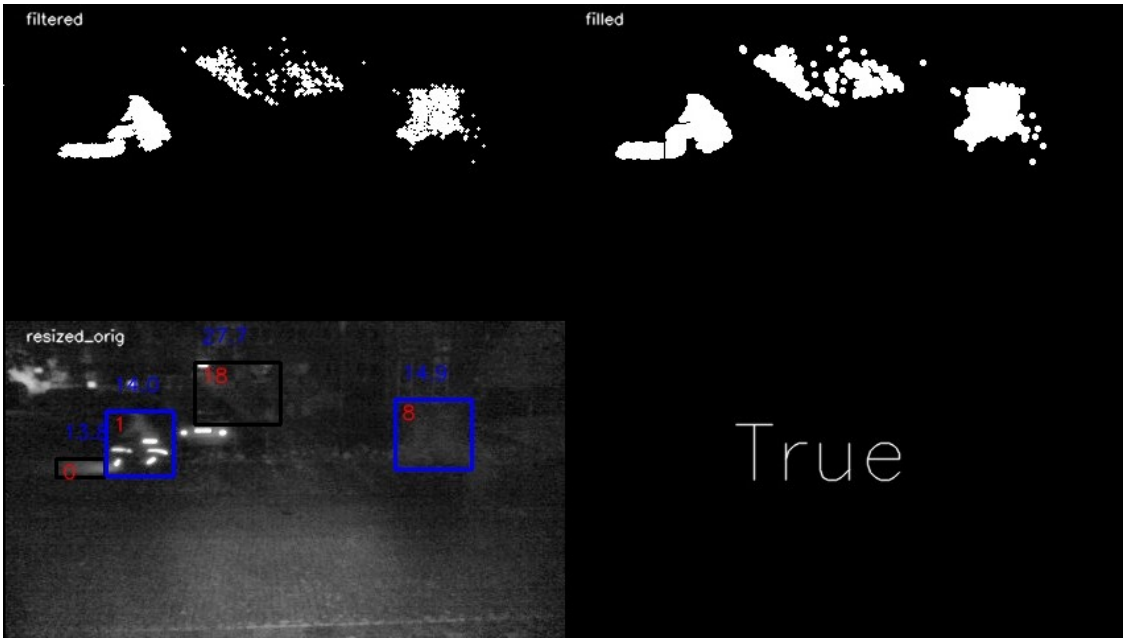


Figure B.5: False positive detection

Object 8 - FP detection caused by the movement of a cloud of smoke. Object 1 - true positive detection of a cyclist.

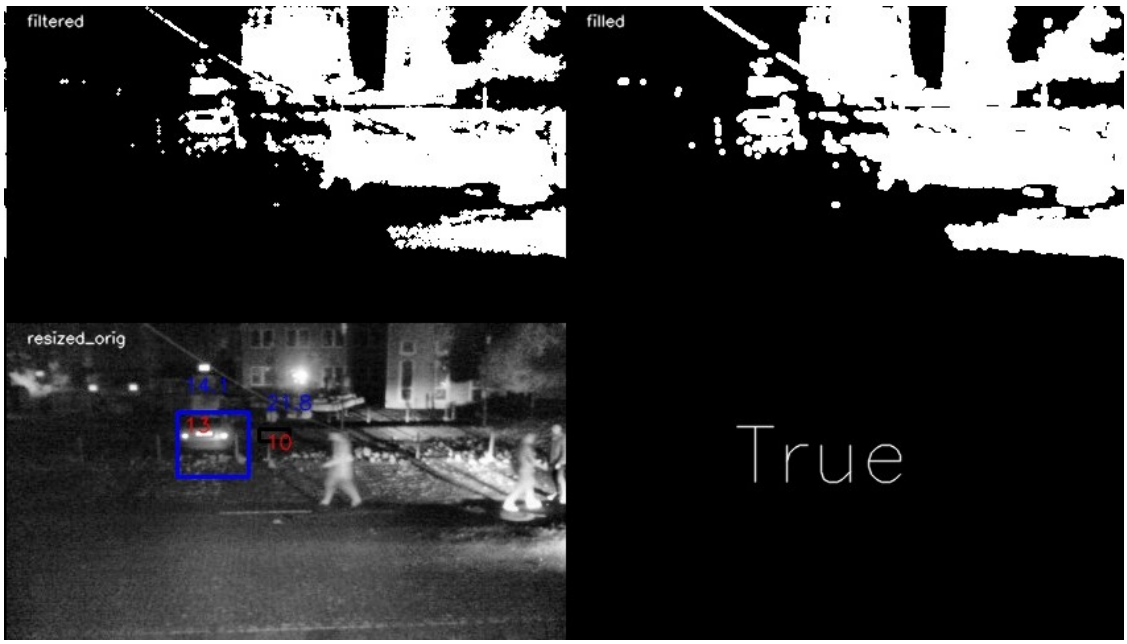


Figure B.6: False positive and false negative detection examples

The object 13 is a FP detection. Pedestrians present in the frame are not detected. Detection errors are caused by the reflection of light coming from a moving vehicle.

IoT Software Engineer

IVAN MATVEEV

Berlin

+49 17642010482

matveev.ivan.gr@gmail.com

linkedin.com/in/ivan-matveev-gr

WORK EXPERIENCE

Oct 2022 - Present

IoT Engineer

Aufzughelden by DigitalSpine

- **Databricks**
  - **Administration and Configuration** - Proficiently set up and managed Databricks configurations from inception.
  - **Medallion Architecture Implementation** - Established comprehensive architecture for all business data, integrating IoT data from the ground up.
  - **CICD Pipeline Design using Github Actions** - Designed robust Continuous Integration and Continuous Deployment (CICD) pipelines leveraging Github Actions.
- **Backend Microservices Development and Maintenance** - Engineered and sustained microservices on AWS, facilitating the implementation of new features.
- **AWS Infrastructure and Backend Migration** - Orchestrated seamless migration from AWS EC2 to AWS Fargate Spot, optimizing infrastructure efficiency and reducing costs.
- **Node.js App Enhancement on IoT Devices** - Implemented new features for applications on IoT devices, ensuring optimized functionality.
- Maintained clear and concise documentation of all configurations, architectures, and development processes, facilitating ease of understanding and knowledge transfer.

March 2021 - Sept 2022

Software Engineer

Dexor Technology GmbH (Extern in Safran Engineering Services)

Development of a vehicle charging software for VW Group:

- Model-based development (Matlab/Simulink + DSpace)
- Model and software in the loop testing (TPT)
- Requirements engineering
- Workflow automation with Python

Sept 2020 - March 2021

Research Engineer

Anhalt University of Applied Sciences

Development of an eco-friendly smart lighting system (distributed Linux-based IoT infrastructure containing 400+ nodes):

- Porting a lightweight object detection algorithm from Python to C++
- Optimization and Integration of the object detection algorithm
- Refactoring and optimization of Python projects
- Deployment of a Dockerized, Icinga-based monitoring system
- Automated configuration management using Ansible
- Cross compiling Linux kernel and modules for ARMv8 architecture
- Linux system administration including deployment and maintenance of JupyterHub

Oct 2016 - Sept 2020

Research Staff

Anhalt University of Applied Sciences

- Design and Python implementation of an object detection algorithm for low-performance single-board computers
- C implementation of OV7670 camera driver using Programmable Real-Time Units of Beaglebone Black SoC
- Automated unit testing of Python applications
- PhD scholarship of Saxony-Anhalt state
- Research in area of object detection and academic publishing (author of 1 patent and 6 publications)
- System administration of Linux-based IoT infrastructure
- Development of a custom API for an IoT application

SKILLS

Languages

English	Advanced
German	Intermediate
Russian	Native

Programming

Python	Advanced
Javascript	Advanced
Bash	Advanced
C/C++	Intermediate

OS

Arch-based	Debian-based

Libraries & frameworks

OpenCV	SciPy, Sklearn
NumPy, Pandas	Flask, Django
Matplotlib	RabbitMQ
Pytest	Selenium

Tools

Docker/ -compose	Git, SVN
Ansible	JupyterHub
Icinga2, Grafana	InfluxDB, Graphite
Jira, Confluence	CodeBeamer

EDUCATION

Nov 2021 - Present

PhD Candidate in Engineering. Thesis topic: Method for detecting and classifying moving objects based on evaluation of object geometrical parameters in an image sequence

Anhalt University of Applied Sciences

Koethen, Germany

Oct 2014 - Oct 2016

Master of Electrical and Computer Engineering. Thesis topic: Object detection module based on combination of radio-wave and passive infrared sensors

Anhalt University of Applied Sciences / Tomsk Polytechnic University

Koethen, Germany / Tomsk, Russia

Sept 2010 - Jun 2014

Bachelor of Instrumental Engineering

Tomsk Polytechnic University

Tomsk, Russia

Iptables	FFmpeg
Latex, Markdown	GitHub, GitLab
PIP, Conda	SQL
Jenkins	Make

AWS

IoT Core	Cloud Formation
Kinesis	DynamoDb
Lambda	ECS
ECR	S3