ABSTRACT

Real-time object detection is now a key capability for embedded computer vision platforms used in numerous applications, from surveillance to transportation and defense systems. However, the limited resource constraints and demands for low-latency accurate detection impose significant optimization challenges.

The key motivation is enabling real-time object detection using lightweight algorithms on embedded devices without powerful specialized hardware. The pipeline combines selective background subtraction using Zipfian Estimation techniques with an optimized HOG-SVM classifier to minimize computations. Additional techniques like multi-scale scanning via parallel classification threads are employed to enhance throughput.

This thesis proposes a real-time object detection system for embedded devices with stationary cameras that uses lightweight algorithms. It includes an overview of related work, presents the proposed system, and aims to evaluate its effectiveness through simulation tests on software written in C++ and videos from the MOT15 dataset. The system achieved frame rates between 15-24 FPS, demonstrating efficiency. However, the accuracy metric, mAP, showed that simpler non-neural methods resulted in less than 10% accuracy, highlighting the tradeoffs. While further development is required, this thesis provides a promising foundation for real-time detection that balances efficiency and accuracy and aims to be implemented into hardware.

Keywords: Object detection, Real-time, Background subtraction, Zipfian estimation, HOG features, SVM classification, Parallel processing, Multi-scale.

AUTHORSHIP

I have completed my graduation thesis, "**Developing a Real-Time Object Detection System on FPGA**" as a result of my labor under the guidance of my supervisors. Within the reference section of my thesis, I have correctly identified all sources used in its creation. I can attest that all data and results this thesis presents are accurate. I alone am responsible for any errors that may be found, subject to any disciplinary measures imposed by the University Paris-Scalay and the University of Engineering and Technology, Vietnam National University, Hanoi.

Hanoi, January 22nd, 2024

Author

ACKNOWLEDGEMENT

I want to express my sincere gratitude to the professors, students, staff, and institutions that played an essential role in my success throughout my Paris Saclay M2 Master's program in Communication and Data Engineering that cooperated with the VNU University of Engineering and Technology and Université Paris-Saclay.

I especially want to thank my supervisors, Dr. BUI Duy Hieu from VNU Information Technology Institute, Dr. TRAN Thi Thuy Quynh from VNU University of Engineering and Technology, and Dr. Erwan LIBESSART from CentraleSupélec for supervising my master's thesis. Especially Prof. TRAN Xuan Tu – Director of VNU Information Technology Institute, for providing valuable feedback. Completing this thesis was only possible with their guidance and support.

I also want to thank esteemed professors: Prof. Pierre DUHAMEL at Centrale Supélec, Prof. NGUYEN Linh Trung and Dr. NGUYEN Hong Thinh at VNU University of Engineering and Technology, and Prof. Arnauld BOURNEL at Université Paris-Saclay. Their guidance was crucial for my success during my Paris Saclay M2 Master's program, and I am confident that it will help me greatly in my future career.

Furthermore, I would like to thank my manager, MSc. DANG Van Huan at Viettel Aerospace Institute, thank you for providing me with the necessary support to complete my master's program.

Lastly, I would like to express my heartfelt appreciation to my family, girlfriend, friends, and colleagues for their unwavering support throughout this journey. I sincerely appreciate your help and am always grateful for your support.

Contents

List of Abbreviations

Abbreviation	Definition
2D	Two Dimensional
ADAS	Advanced Driver Assistance Systems
AI	Artificial Intelligence
AMD	Advanced Micro Devices
mAP	Mean Average Precision
ASIC	Application Specific Integrated Circuit
AVC	Advanced Video Coding
CBAM	Convolutional Block Attention Module
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
DMA	Direct Memory Access
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GB	Gigabyte
GE	Gate equivalents
GPU	Graphics Processing Unit
HD	High Definition
HOG	Histogram of Oriented Gradients
LTS	Long Term Support
MJPEG	Motion JPEG
MOT15	Multiple Object Tracking Benchmark 2015
MOTC	Multiple Object Tracking Challenge
MPSoC	MultiProcessor System On Chip
NMS	Non-Maximum Suppression
OS	Operating System
PASCAL	Pattern Analysis Statistical Modelling and Computational Learning
PETS09	Performance Evaluation of Tracking and Surveillance 2009
PTZ	Pan-Tilt-Zoom
RAM	Random Access Memory
RCNN	Regional Convolutional Neural Network
RGB	Red Green Blue
SIFT	Scale-Invariant Feature Transform
SPP	Spatial Pyramid Pooling
SRAM	Static Random Access Memory
SSD	Single Shot Detector
SVM	Support Vector Machine

TSMC	Taiwan Semiconductor Manufacturing Company
UHD	Ultra High Definition
VNU	Vietnam National University
VOC	Visual Object Classes
YOLO	You Only Look Once
ZCU106	Zynq UltraScale+ ZCU106 Evaluation Kit

List of Figures

List of Tables

Introduction

Real-time object detection is an essential computer vision application used in various fields. It enables automatic recognition of people in public areas and specialized systems used in military applications.

Real-time object detection is the capability of computer vision systems to locate and classify objects in images or videos quickly enough for use in practical applications. The crucial challenges for real-time object detection are accuracy, throughput, and computation complexity. Real-time applications are frequently utilized in mobile embedded systems with limited memory, computation power, and battery life.

There are two main types of object detection algorithms: the traditional method, which uses feature extraction algorithms, and deep learning, which uses neural networks. Deep learning methods can be further categorized into two types. The first type consists of region proposal object detection algorithms, which generate region proposal networks and classify these regions afterward. Examples of these algorithms include RCNN, SPP-net, Fast-RCNN, and Faster-RCNN. The second type is the regression object detection algorithm, which provides for algorithms like SSD and YOLO. Traditional methods, also known as non-neural methods, use hand-crafted feature extraction algorithms such as HOG, SIFT, Haar wavelets, and custom-designed filters based on expert domain knowledge.

While deep learning methods are often praised for their success in real-time object detection, traditional feature extraction algorithms still possess unique advantages in solving these problems. For instance, hand-crafted feature algorithms can run efficiently on essential CPU platforms, whereas deep neural networks often require powerful GPUs or dedicated hardware acceleration. Furthermore, hand-crafted feature algorithms rely heavily on expert domain knowledge or features of the object in question, making them less dependent on large labeled training datasets that can be costly to scale up.

This thesis proposes a system for real-time object detection in embedded devices that use stationary cameras and lightweight algorithms. The system utilizes the stationary camera's background characteristics, which remain almost constant. To detect objects, background extraction techniques are employed, and Zifian Estimation is used to detect moving objects to push into the object detection module. The HOG-SVM algorithm is also used for calculations, which are parallel at various scales to avoid missing objects. The algorithm employs quantized values to reduce the number of complex computations in the system. Overall, these suggestions help to improve the efficiency of the system.

The proposed system architecture involves using a sequence of images as input, which will be pre-processed using Zipfian Estimation Techniques. This technique is useful when dealing with dynamic objects like humans, as it calculates only those within the frame. The areas where motion is detected are then directed to the Object Detection block, which comprises a Scale Generator module, six HOG-SVM computation modules running in parallel in a multithread architecture, and an NMS algorithm module. The Scale Generator and multithread HOG-SVM computation modules are essential to reducing processing time when scaling the input image to detect all humans in multiscale. The results are then merged with the original image for viewing, storage, or transmission to a server for further processing.

The system being proposed is intended for deployment on FPGA. However, due to time constraints in this thesis, the proposed system will be halted on software at the simulation stage. The proposed system underwent testing on a C++ software platform on a computer with an Intel® Core™ i5-1035G4 processor. Input video sources were obtained from the MOT15 dataset. The evaluation of the system was based on speed and accuracy. The speed of the system was relatively good, but the accuracy was low. This could be attributed to various factors such as the complexity of the input images, limitations of the SVM training set, and not selecting suitable thresholds for each step like Zipfian Estimation, HOG-SVM, or NMS. Additionally, quantized values to optimize speed in steps contributed to the low accuracy results. However, the bright spot is that the low complexity system has been appropriately optimized for future

hardware platforms such as Xilinx's FPGA platform. Additionally, the above problems can be entirely improved by pushing our system into a specific situation, giving it an appropriate training set of SVM weights, and performing many corresponding experiments in many datasets with similar conditions with proposal situations to choose thresholds suitable for each step, from these things to complete the system.

The thesis is organized as follows:

Chapter 1 overviews real-time object detection and its challenges, applications, methods, and related works. The focus is developing a real-time system using lightweight algorithms optimized for hardware deployment on an FPGA platform.

Chapter 2 presents a lightweight motion detection algorithm for stationary cameras and the HOG-SVM object detection algorithm. The proposed system architecture, including block diagram and data flow, is based on state of the art.

Chapter 3 describes the experimental setup, metrics, and dataset to verify the proposed system. The system analysis is provided from the experimental results.

Chapter 1. Real-time Object Detection System

This chapter discusses Object Detection and its applications. It will provide an overview of Real-time Object Detection and its challenges. The following section will cover Object Detection methods and related works. Finally, the main focus of the thesis, which is to develop a real-time Object Detection system using lightweight algorithms for hardware deployment on the FPGA platform, will be explored in detail. This chapter is divided into four sections. Section 1.1 defines object detection and provides examples of its use in various applications. Section 1.2 discusses how achieving high accuracy and real-time performance can be challenging, particularly on platforms with limited resources. Section 1.3 reviews the current state of object detection methods, including deep learning and traditional computer vision algorithms, along with their limitations. Finally, the Chapter Conclusions are presented.

1.1. Object Detection Overview

Object detection [1] is a computer vision technology that enables machines to locate and identify objects of specific classes, such as people, vehicles, or animals, in digital images and videos. This technology is based on advanced machine learning algorithms that detect different objects and identify their presence and location in visual data.

Object detection has become an integral technology across many industries and applications. Surveillance systems enable the automatic detection of people and vehicles, which is critical for security tasks. In agriculture, object detection allows for counting livestock, monitoring crop growth, and detecting disease outbreaks automatically. Autonomous vehicles employ object detection to classify other cars, pedestrians, traffic signals, and road signs in real time to navigate safely. Or they are used in specialized systems in high-technology weapons like Kamikaze drones [2].

Factors such as color, contrast, quality, shape, and object orientation influence object detection accuracy. However, the orientation of an object is a significant challenge for object detection models. This is because many objects, including humans and animals, can take different poses and orientations, which can significantly impact

the accuracy of the AI model. It is important to note that object detection depends on the specific image and object being detected due to factors such as lighting, contrast, and object size. Therefore, it is crucial to carefully select the appropriate object detection model or algorithm for each application to ensure optimal accuracy in image processing.

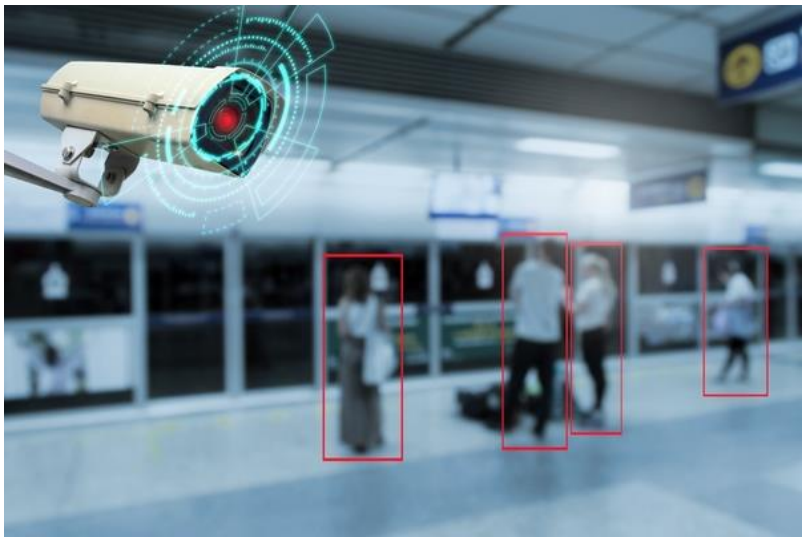


Figure 1 - Surveillance and Public Safety Camera [3].

Figure 2 - Autonomous surveillance along the border in the USA [4].

Object detection is a key technology in computer vision that enables automated systems to locate and identify objects in visual data. However, several variables can affect the accuracy of object detection. With advancements in deep learning and the use of traditional feature extraction algorithms with machine learning, there have been

many possibilities for the development of object detection. In object detection applications, real-time factors are always crucial. Therefore, this thesis aims to research real-time object detection, focusing on embedded edge devices.

1.2. Real-time Object Detection Challenges

Real-time object detection is a computer vision system that quickly locates and classifies objects in images or videos. It is a challenge to balance accuracy and speed.

Real-time processing is essential for many applications to operate successfully. With real-time processing, these applications are valuable in terms of use. Therefore, it is important to implement real-time processing to ensure these applications operate efficiently and effectively. For instance, even small edge devices such as car dash cameras Figure 3 are equipped with real-time vehicle detection functions to serve ADAS [5] features, while in tactical missiles using an Imaging Infrared seeker in Figure 4, it is necessary to detect the object quickly and decide to destroy the target. However, real-time object detection has many challenges.

Real-time systems need to process videos at high frame rates without any delay. These systems must balance speed and accuracy to achieve optimal performance. However, optimizing system latency across various components, including input data, processors, software, and output devices, can be complex.

As mentioned above, real-time object detection applications are often deployed in edge devices, in large numbers, and used as satellite devices in large systems. These size, power supply, processing performance, and cost are the fundamental problems. Complex methods using deep learning networks are very effective in accuracy and speed, but they require device hardware that uses high-performance GPUs or other specialized hardware. Balancing performance and device configuration is a big challenge in real-time object detection.

Figure 3 - Dash Camera [6].

Figure 4 - Imaging Infrared Seeker on the Missile [7].

Overall, real-time object detection is a crucial application across many fields. However, creating a real-time object detection system requires balancing speed, accuracy, performance, and hardware configuration. Depending on the specific problem requirements, a combination of traditional methods and deep learning can be used to meet the system's needs. This thesis aims to develop a low-complexity real-time object detection system that can be easily deployed to edge devices. The use case for this system is a stationary camera.

1.3. Related Works

In object detection, two methods exist: traditional and deep learning. Deep learning methods can be further categorized into two kinds. The first category is region proposal object detection algorithms, which generate region proposal networks and classify these region proposals afterward. Examples of these algorithms include RCNN [8] in 2014, Fast-RCNN [9], and Faster-RCNN [10] in 2015. The second

category is the regression object detection algorithm, which provides for algorithms like the YOLO [11] or SSD [12] in 2016. Traditional methods, or can say non-neural methods, use hand-crafted feature extraction algorithms; examples are HOG [13] in 2005, SIFT [14] in 1999, Haar [15] in 2004, etc.

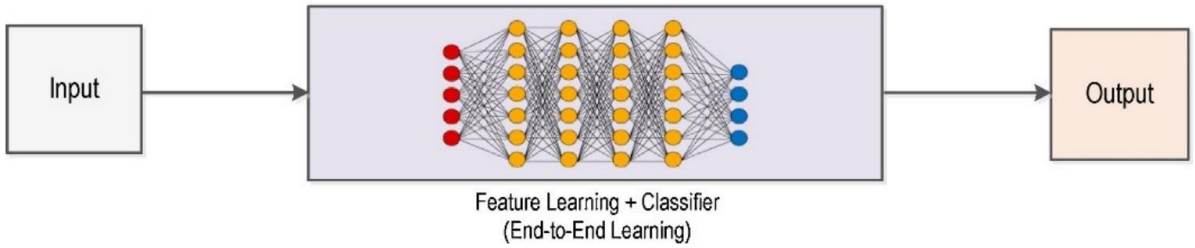


Figure 5 – Deep learning method [16].

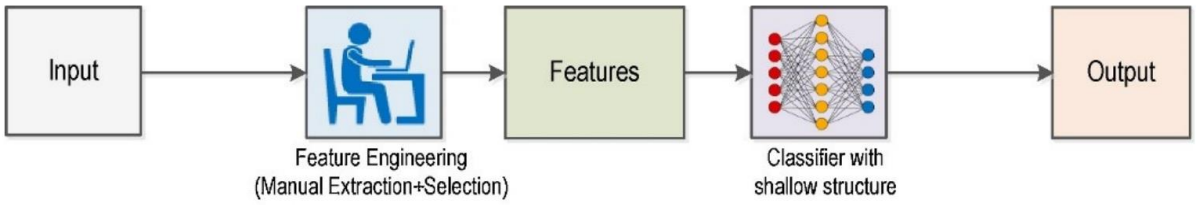


Figure 6 – Hand-crafted feature method [16].

The challenges of real-time object detection can be solved using deep learning, which has become a popular trend. In 2020, Tinier-YOLO [17] was proposed to yield a model size of 8.9MB while achieving 25 FPS real-time performance on Jetson TX1 and an mAP of 65.7% on the PASCAL VOC dataset and 34.0% on the COCO dataset. In 2022, [18] proposed an improved YOLOv5 by integrating CBAM and SENet to represent objects better and utilized multiscale detection for improved accuracy; the approach achieved up to 90.2% mAP. In 2023, YOLOv7 [19] surpassed all known object detectors in speed and accuracy range from 5 FPS to 120 FPS and had the highest accuracy, 56.8% AP, among all known real-time object detectors with 30 FPS.

While deep learning methods are often lauded for their success in real-time object detection, traditional algorithms still possess unique advantages in solving these problems. For instance, hand-crafted feature algorithms can run efficiently on essential CPU platforms. In contrast, deep neural networks often require powerful GPUs or dedicated hardware acceleration, as [20] in 2017 mentioned that the hand-crafted feature extraction algorithm used in this paper, HOG, has 311 to 13,486 times lower

power consumption and 35 times higher throughput than the deep learning method used in this paper, which is CNN. Furthermore, traditional algorithms rely heavily on expert domain knowledge or features of the object in question, making them less dependent on large labeled training datasets that can be costly to scale up; in 2020, [21] showed that limited training data could cause overfitting in machine learning models. Deep neural networks have millions of parameters that are difficult to tweak manually. On the other hand, engineers can transfer their insights into algorithms and tweak parameters for a broader range of images in the traditional way.

Object detection using images requires processing a vast amount of data. Therefore, executing these algorithms in real-time on edge devices requires special optimization techniques. Most works that support high-resolution and real-time computing employ the HOG-SVM algorithm. Various methods are proposed for improving the speed of the HOG-SVM algorithm, as mentioned in several works [22] in 2013. One approach is to increase the number of parallel computation blocks, which can speed up the calculations. For instance, in a particular work [23] in 2016, multiple computational blocks were used to calculate several image resolutions simultaneously. Another method suggested in the job [24] in 2019 is to optimize the HOG SVM concurrently to promote faster calculations. Some works have also proposed high-speed multi-object detection systems, such as the ones presented in [25] in 2019 and [26] in 2012. However, it is worth noting that these systems only work with images of a resolution of 512x512 pixels. In 2022, [27] present a real-time pedestrian detector using HOG feature extraction and SVM classification, capable of processing 4K video at 60 fps using an AMD Xilinx Zynq UltraScale+ MPSoC device. This is a huge step in real-time object detection because it can perform 60fps on a large UHD resolution.

Detecting objects in real-time is a difficult task that requires processing large amounts of image data efficiently. Deep learning methods can provide high accuracy but need powerful GPUs or specialized hardware. On the other hand, traditional computer vision algorithms, such as HOG-SVM, are lighter and more hardware-efficient, but their accuracy is lower. This thesis proposes developing a real-time object detection system that uses lightweight algorithms. The core of this system will

be the HOG-SVM object detection module. One technique that can detect motion in stationary camera situations is the Zipfian Estimation Technique [28]. This technique will help reduce the number of computations required. The project aims to implement this system on an FPGA hardware platform.

Feature extraction lightweight algorithms are highly suitable for implementation on FPGA hardware due to their efficiency, low power consumption, and the ability of FPGAs to be customized for parallel processing. FPGAs are known for their high efficiency and deterministic execution times, making them ideal for real-time applications. When implemented correctly, lightweight algorithms can be optimized to capitalize on these strengths. However, implementing complex algorithms or neural networks on FPGA hardware can be challenging due to the limited on-chip resources of FPGAs, which can be quickly outstripped by the demands of large neural networks. Designing and optimizing such algorithms on FPGAs requires a deep understanding of both the algorithmic intricacies and the FPGA architecture, adding complexity to the design process. There is also a trade-off between the flexibility of the FPGA and the efficiency gains from hardware-specific optimizations. Additionally, the development process for FPGA-based solutions is typically longer and more costly than for software implementations on general-purpose processors due to these complexities and the specialized skills required. Despite the potential benefits, the implementation of complex algorithms on FPGAs remains a challenging endeavor.

1.4. Conclusions

Real-time object detection is an essential technology with applications across many areas. However, achieving real-time speeds, high accuracy, and hardware constraints presents significant challenges.

Both deep learning and traditional computer vision methods have been used to solve these problems. However, there are tradeoffs between accuracy, speed, performance, and hardware configuration. Deep learning methods such as YOLO, R-CNN, or SSD can provide state-of-the-art accuracy but require powerful GPUs or specialized hardware. On the other hand, traditional algorithms like HOG-SVM are

more lightweight and hardware-efficient. However, they rely on feature engineering more heavily and may not match the accuracy of deep learning methods.

This thesis proposes a system that can detect objects in real time by utilizing the HOG-SVM algorithm as its core. Various techniques will be employed to enhance its speed and efficiency, including using background motion using the Zipfian Estimation method. The objective is to implement the complete system on an FPGA platform to leverage the hardware design optimization flexibility and parallelism.

Chapter 2. Proposed Real-time Object Detection System

The first chapter of this thesis provides an introduction to object detection and real-time object detection, and it also outlines the specific goals of the thesis. To create a system that detects objects in real-time on embedded edge devices using a stationary camera, it's recommended to use motion detection based on background subtraction using Zipfian Estimation Techniques. This technique helps identify the moving components in the frame, which can then be passed on to HOG-SVM calculation blocks to identify the objects accurately.

In the second chapter of the thesis, a system architecture will be proposed based on the idea above. The chapter will go into the different components of the system, including the motion detection block. The motion detection block works as the pre-processing part of the system. The object detection block uses HOG-SVM as its core and other components, such as the Bilinear Interpolation Scale Generator, which detects objects in multiscale, and the NMS, responsible for collecting the final results. Finally, the post-processing component is responsible for bringing the location of the recognized object back into the frame.

The chapter is divided into four sections. Section 2.1 outlines a system architecture that utilizes lightweight algorithms to detect humans. Section 2.2 describes a background subtraction algorithm, which helps detect motion in image frames from a stationary camera, thereby reducing computation compared to running object detection on every frame. Section 2.3 explains the use of Histogram of Oriented Gradients features with a Support Vector Machine classifier to detect humans in video frames after motion detection. Lastly, Chapter Conclusions are discussed.

2.1. Proposed System using Zifian Estimation Technique and HOG SVM Algorithm

This section presents a system architecture designed for detecting humans in video images. The system comprises several components, including motion detection, human detection through parallel processing, non-maximum suppression, and merging of detection results. The proposed approach offers optimized algorithms and a parallel

architecture to improve throughput. The following sections will discuss the Dataflow. and System Architecture in detail

2.1.1. Dataflow

(1) Pre-processing

Input: RGB Image data.

Output: Detected motion areas.

To process two image data, it is necessary first to convert them into grayscale images. This is achieved by computing the average pixel value of the Red, Green, and Blue channels, with each pixel being represented by an 8-bit number ranging from 0 to 255. Next, the Zipfian Estimation Techniques [28] are employed to obtain a list of contours that detect motion areas. This list of contours is then utilized to crop the input grayscale image and put them in human detection processing.

As shown in Figure 7, in the image with a resolution of *1300x982* (used 1.7 scales up) using *77202* sliding windows, we only push three images with resolutions *176x180*, *158x161*, and *182x190* into the object detection block, a total using *556* sliding windows, equivalent to *0.7%*.

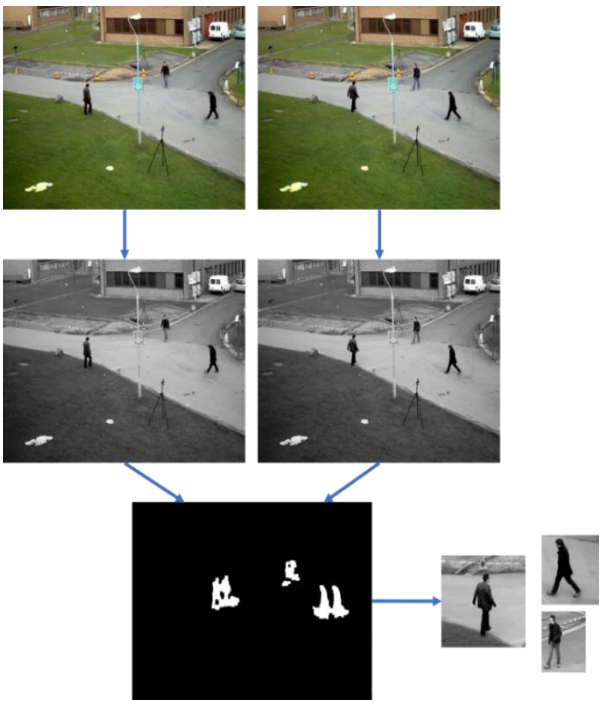


Figure 7 - Pre-processing data flow.

(2) Processing

Input: Detected motion areas

Output: Detected human regions.

For each motion detected area, scale it using the Bilinear Interpolation Scale Generator algorithm, each step of 1.05, decreasing until the height is less than *130* and the width is less than *66* (this is the minimum size of the *64x128* sliding window plus a 1-pixel outer border to implement the HOG-SVM algorithm). Divide these scaled images into six parallel HOG-SVM modules. The result will be pushed into the combined module and finally pushed through the NMS module to filter out the sliding window. The final result will be the area where the system determines to identify the object, precisely, in this case, a person.

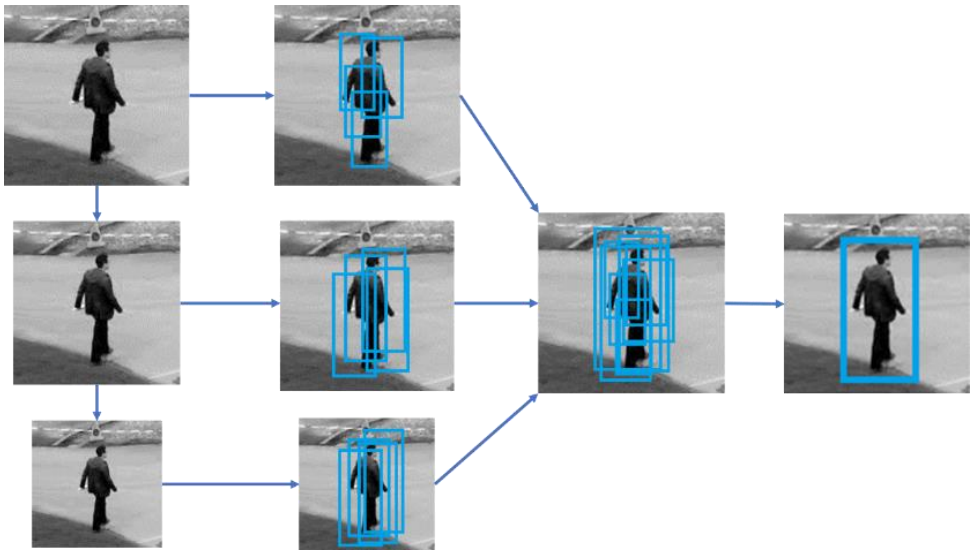


Figure 8 – Processing data flow.

(3) Post-processing

Input: Detected human areas and raw input image

Output: Detected human regions.

The contour result list is a crucial component that merges the output of the human detection process with the saved raw RGB image. Combining these two pieces of information, the contour result list generates the final results, accurately identifying

and outlining the detected objects or individuals. The result can be shown on a monitor, saved to a file, or sent to the database for future applications.

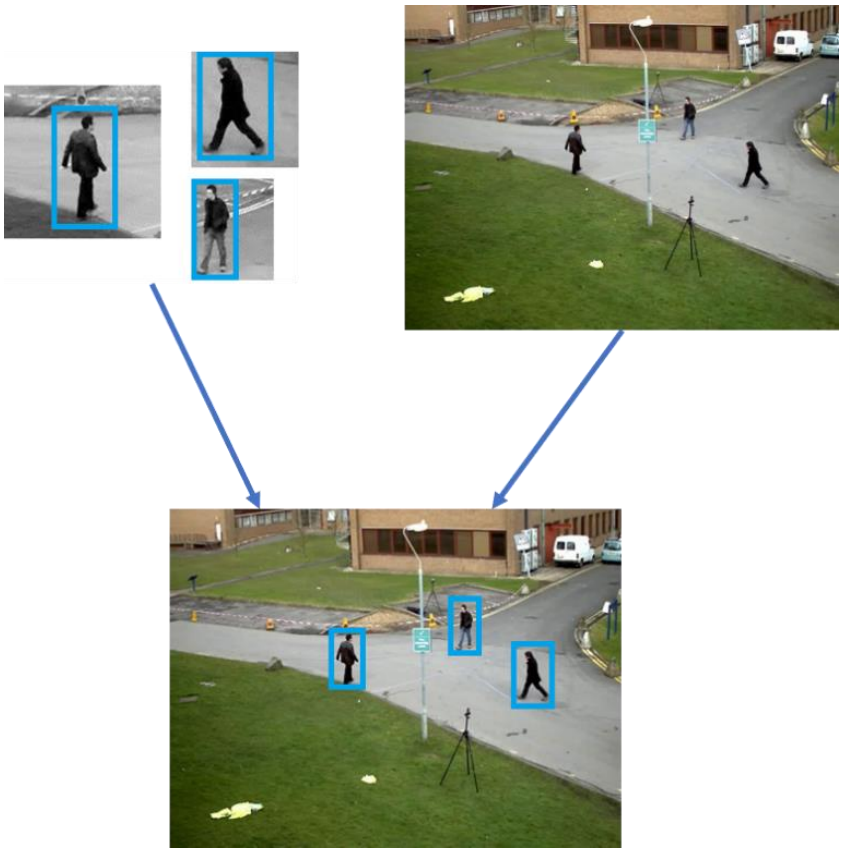


Figure 9 – Post-processing data flow.

2.1.2. Proposed System

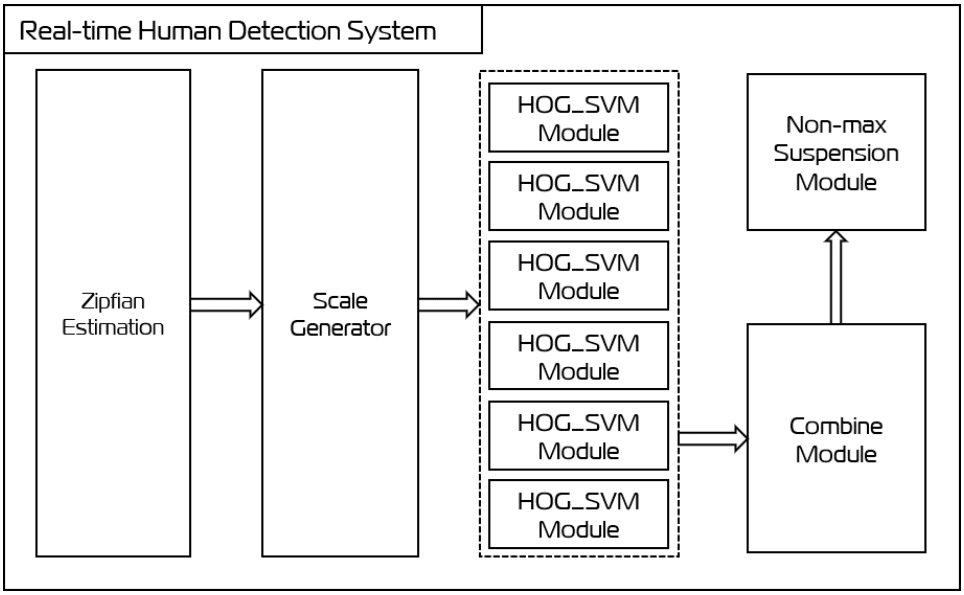


Figure 10 – Proposed system architecture.

The proposed system architecture involves using a sequence of images as input, which will be pre-processed with the help of Zipfian Estimation Techniques. This technique is proper when dealing with dynamic objects such as humans because it only calculates those in the frame's hold. The motion-detected areas are then directed to the Object Detection block, which comprises a Scale Generator module, six HOG-SVM computation modules that run in parallel in a multithread architecture, and an NMS algorithm module. The Scale Generator and multithread HOG-SVM computation modules are required to reduce processing time when scaling the input image to detect all humans in multiscale. The results are merged with the original image for viewing, storage, or transmission to a server for further processing.

Overall, the proposed system uses lightweight algorithms executed in parallel to detect humans in real-time in both video and image sequences. Pre-processing is done using motion detection before applying the HOG-SVM algorithm to reduce the search space. The system then outputs bounding boxes that outline the detected humans, which can be used in various applications. An efficient and accurate human detector is crucial in many areas, such as automated surveillance, advanced driver assistance systems, and human-computer interaction.

2.2. Lightweight Motion Detection Algorithm used on a stationary camera

The stationary camera is widely used for surveillance and monitoring in public areas. It is highly flexible, cost-effective, and easily deployed in large numbers due to its compact size and affordability. Its multifunctional nature makes it an ideal choice for deployment in public spaces. Stationary cameras are non-PTZ cameras installed in a fixed position with a predetermined field of view. These cameras may have adjustable zoom capabilities. As they make the background almost constant during operation, and people or objects appearing in the frame are dynamic, they only calculate those within the frame. This reduces the system's number of calculations, making it faster and enabling it to solve the input image with a higher resolution.

In 2014, The Zipfian Estimation [28] used MJPEG to extract moving and stationary blocks separately, resulting in a compression ratio twice as high as

conventional MJPEG. This method encodes only the residuals of the moving blocks, producing a similar quality and bit rate to H.264/AVC. It requires fewer operations than conventional MJPEG if the static scene is equal to or greater than 60%. Zipfian estimation was one of the fastest motion detection algorithms [29] in 2007, [30] in 2009, and the basic Σ-Δ background subtraction algorithm [31] in 2004. Overall, the Zipfian Estimation demonstrates how selective encoding of only dynamic regions can realize substantial compression gains without sacrificing visual quality or requiring extensive computations.

As mentioned above, the Zifian Estimation Module is a lightweight computing technique based on the Sigma-Delta algorithm to detect motion in the frame, as shown in Algorithm 1.

Algorithm 1 - Zipfian estimation [31].

find the greatest 2^{p} that divides (t mod 2^{m})
set $\sigma = \frac{2^m}{2^p}$
foreach pixel x **do**
  if Vt−1(x) > σ then
    if Mt−1(x) < It(x) **then** Mt(x) ← Mt−1 + 1
    if Mt−1(x) > It(x) **then** Mt(x) ← Mt−1 − 1
foreach pixel x **do**
  Ot(x) = |Mt(x) − It(x)|
foreach pixel x **do**
  if t mod TV = 0 **then**
    if Vt-1(x) < N × Ot(x) **then** Vt(x) ← Vt−1(x) + 1
    if Vt-1(x) > N × Ot(x) **then** Vt(x) ← Vt−1(x) − 1
foreach pixel x **do**
  if Ot(x) < Vt(x) **then** Et(x) ← 0

To begin with, the algorithm calculates a threshold based on the frame index *t* and frame index *t-1*. The value of the index is divided by 2^m (where *m* is the number of bits representing a pixel; in our case, $m = 8$ for grayscale image), and the remainder is denoted as *p*. Then, the threshold σ is equal to 2^m divided by 2^p. The background M_t is updated whenever variance V_{t-1} is more significant than σ. Next, O_t refers to the absolute difference between I_t and M_t. To avoid self-referencing, the variance V_t is

updated using a fixed period T_v. Usually, *a* T_v has a power of 2 within the range of 1 to 64. In our case, we have set T_v to 1. N is typically an amplification factor for the variance V_t, ranging from 1 to 4. In our case, $N = 2$. Finally, the movement or stillness of a pixel is determined by comparing its absolute difference with the variance.

Overall, the Zipfian Estimation shows how particular encoding of only dynamic regions can realize significant compression gains without sacrificing visual quality or requiring extensive computations. Additionally, these are lightweight techniques suitable for this thesis's purpose.

2.3. Object Detection using the HOG SVM Algorithm

As mentioned above, many works have been used and developed on the HOG-SVM algorithm. This algorithm was introduced in 2005 for human detection [13]. Histogram of Oriented Gradients (HOG) is a widely used feature for object detection that analyzes edge distribution to balance detection accuracy and complexity [32]. It is the standard baseline for object detection [33]. A Support vector machine (SVM) [34] is also used for classification based on extracted features.

2.3.1. State of the Art

(1) Gradient calculation optimizes

The HOG algorithm generates histograms with complex cells, including inverse tangent, square, square root, and floating point multiplication. Optimizing this step is crucial to improving the HOG-SVM algorithm module.

Pixel derivatives concerning x,y:

$$dx = I(x, y+1) - I(x, y-1);\ dy = I(x+1, y) - I(x-1, y)$$

Gradient magnitude and angle calculation:

$$\left|\left|\overrightarrow{M(x,y)}\right|\right| = \sqrt{d_x^2 + d_y^2};\ \alpha = \arctan(\frac{dx}{dy})$$

In 2017, a new technique [35] was proposed for generating cell histograms that simplify the process while producing accurate results. This approach involves bypassing the calculation of pixel gradients. The outcome of this research

demonstrates that the reconstruction error is less than 2% with an 8-bit fractional part length. Manipulating the precision of the gradient magnitude is simple using pre-defined sine and cosine values of quantized angles.

The work mentioned above presented a method where the primary gradient vector was split into two vectors, as shown in Figure 11. The values of these two vectors for pixel I(x,y) were calculated using, which takes into account the values of the four surrounding pixels I(x+1,y), I(x-1,y), I(x,y+1), and I(x,y-1).

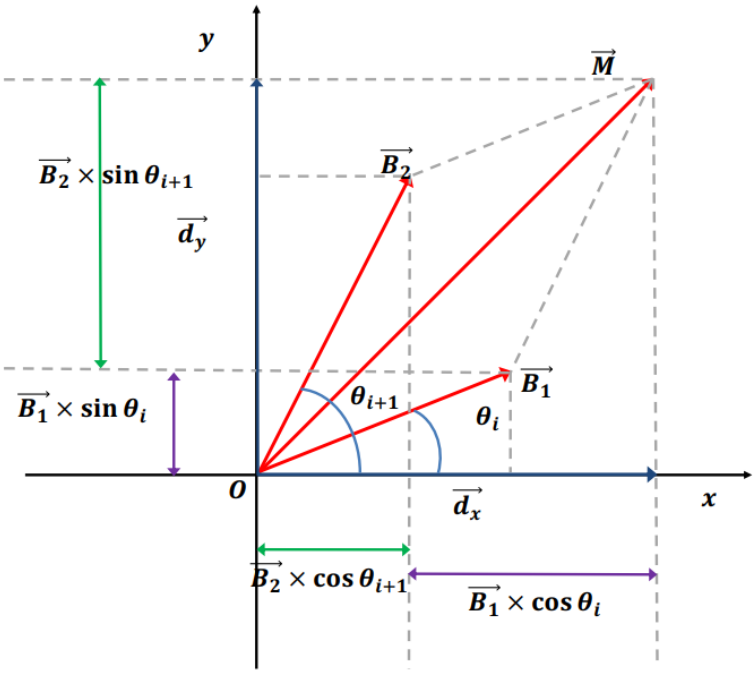


Figure 11 - Decomposition of a vector into the form of two vectors [35].

Two magnitudes are the solutions of the two equations:

$$\left|\left|\overrightarrow{B_{\theta_i}}\right|\right| = \frac{\sin(\theta_{i+1})\, d_x - \cos(\theta_{i+1})\, d_y}{\sin(20)};\ ||\overrightarrow{B_{\theta_{i+1}}}|| = \frac{\cos(\theta_i)\, d_y - \sin(\theta_i)\, d_x}{\sin(20)}$$

This thesis uses this technique to improve the HOG-SVM algorithm and decrease the number of calculations needed to process each sliding window.

(2) Implement HOG-SVM into hardware

In 2015, a real-time and energy-efficient multi-scale object detector hardware implementation was presented [23]. Parallel detectors with a balanced workload increase the throughput, enabling voltage scaling and energy consumption reduction.

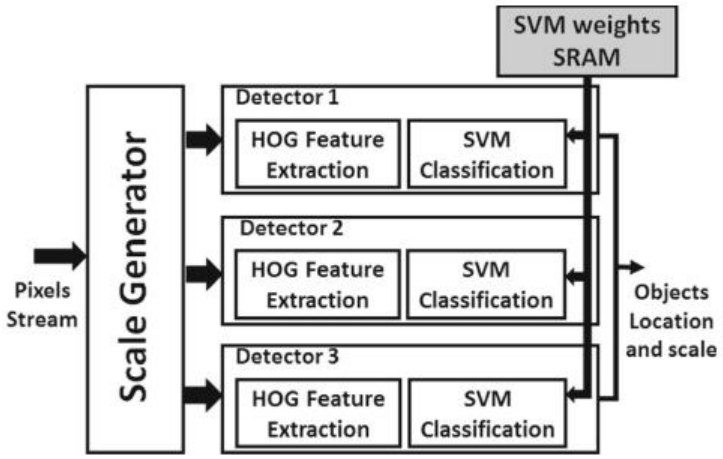


Figure 12 - Object detection system architecture [23].

In 2019, [24] a novel high-throughput hardware architecture was proposed for human detection by co-optimizing HOG feature generation and SVM classification. The architecture aimed at improving throughput by using a fast, highly parallel, and low-cost HOG feature generation in combination with a modified datapath for parallel computation of SVM and HOG feature normalization. The proposed architecture was implemented in TSMC 65nm technology with a maximum operating frequency of 500MHz, achieving a throughput of 139fps for full-HD resolution. The hardware area cost was about 145kGEs, along with 242kb SRAMs.

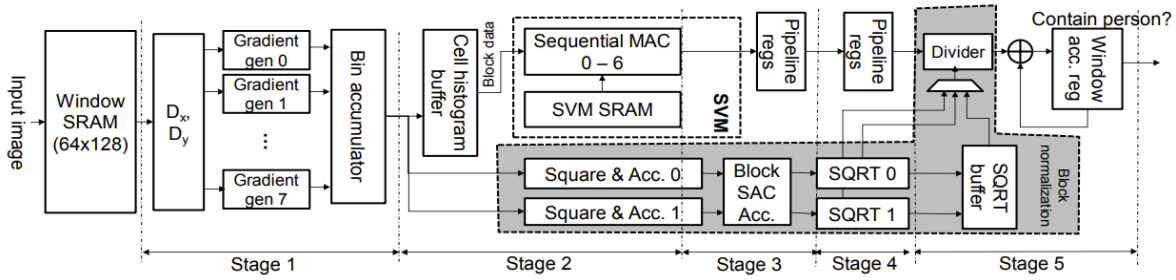


Figure 13 - Proposed block diagram of object detection [24].

In 2023, [36] a new system for detecting humans that use a HOG-SVM module in combination with Direct Memory Access (DMA) on Xilinx FPGA has been proposed. The design was implemented on the Xilinx FPGA Development Kit ZCU106 with direct memory access, which reduces the CPU load and improves the overall system performance.

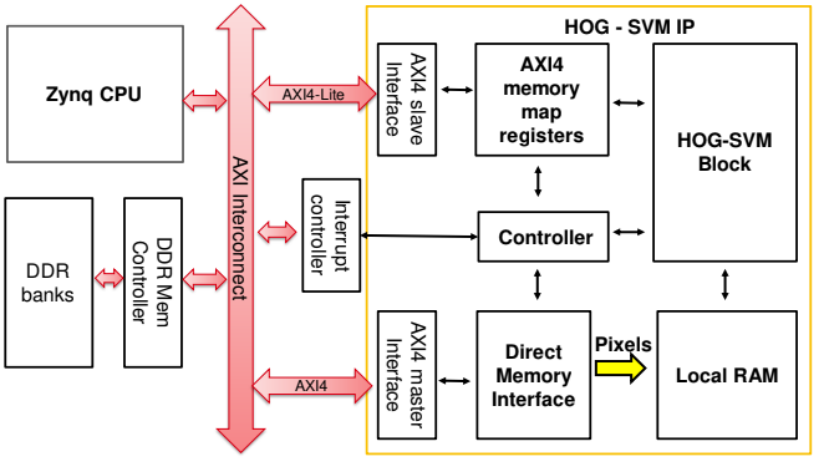


Figure 14 - Block diagram of the proposed human detection system [36].

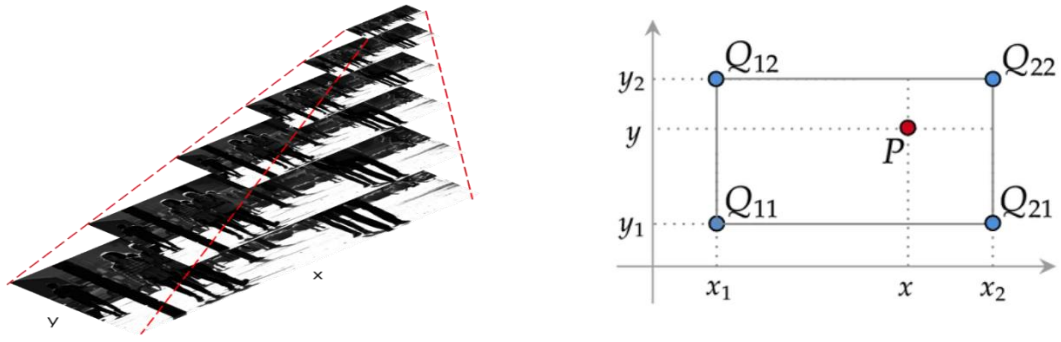
In conclusion, this section reviewed related research on optimizing and hardware implementing the HOG-SVM algorithm to improve efficiency and throughput for real-time object detection systems. One suggests bypassing complex gradient calculations by pre-quantizing sine and cosine values to reduce computations while preserving accuracy. Parallel architectures, deep pipelining, and direct memory access in FPGA/ASIC can achieve high frame rates and low latency. HOG-SVM is widely used for object detection but needs optimizations for real-time applications with power constraints.

2.3.2. Object Detection Block

As mentioned above, the Object Detection Block will include Bilinear Interpolation Scale Generator Module, HOG-SVM module, combine module, and NMS module.

(1) Bilinear Interpolation Scale Generator Module

Bilinear interpolation is a widely used technique in image scaling and 2D finite element analysis that involves linear interpolation in two dimensions. This method consists of averaging the data at each rectangle corner and determining the weights based on the distance between the point and those corners. For a given (x,y) position inside the rectangle, the weight of each corner increases as it approaches the tip of the rectangle. This method was implemented through Algorithm 2.



(a) (b)

Figure 15 - Bilinear Interpolation Scale Generator (a) and illustration (b).

Algorithm 2 – Bilinear Interpolation Scale Generator

```
get original height and width of input image
set new width is original_width / scale
set new height is original_height / scale
create output image of size (new height, new width)
   foreach pixel (x, y) in output_image:
      set src_x is x / scale and src_y is y / scale
      set x1 is floor(src_x) and y1 is floor(src_y)
      set x2 is min(x1 + 1, original width - 1)
      set y2 is min(y1 + 1, original height - 1)
      set w_x is src_x - x1 and w_y is src_y - y1
      set output image[y, x] is (1 - w_x) × (1 - w_y) ×
                        input image[y1, x1] + w_x × (1 - w_y) ×
                        input image[y1, x2] + (1 - w_x) × w_y ×
                        input image[y2, x1] + w_x × w_y ×
                        input image[y2, x2]
```

To resize an image, we first compute a new dimension for the output image. Then, for each output pixel, we find the corresponding pixel location in the input image and its four nearest neighbor pixels. We add distance-based interpolation weights to these pixels and assign a weighted average of them as the value of the output pixel. This process is repeated for every output pixel.

(2) HOG-SVM Module

As mentioned in Section 0, the HOG-SVM algorithm was chosen to process object detection tasks, in this case, focused human detection.

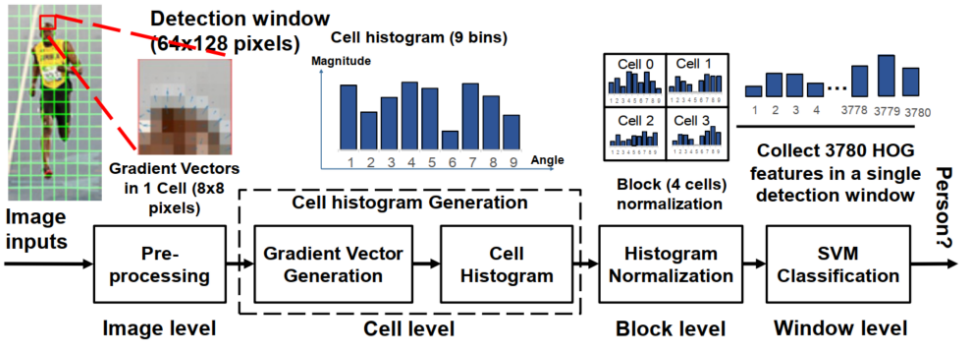


Figure 16 - HOG-SVM algorithm for human detection [23].

The algorithm processes a sliding window measuring *64 × 128* pixels extracted from the input image, then divided into several cells, each measuring *8 × 8* pixels. These cells generate a histogram that captures the gradient angle and magnitude. This involves accumulating gradient magnitudes into nine bins based on their respective angles. Furthermore, *2 x 2* cells are grouped to form a block, and the cell histograms are normalized using block data.

(a) Gradient calculation and Histogram generation

This is the first step of this algorithm, using the results of published work mentioned in section 2.3. It bypasses complex cells, including inverse tangent, square, square root, and floating point multiplication, to decrease the number of calculations needed to process each sliding window.

First, the values d_x and d_y are computed. In the second step, we have absolute values of d_x and d_y, combined with sign of dx and dy, to determine the two quantized angles θ_i; θ_{i+1} via a comparison between $A \times d_x$ and $B \times d_y$, where A is the dividend, and B is the divisor mention in [35]. To calculate two magnitudes, B_i and B_{i+1}, follow the two equations below. However, to be optimized using the values in [35], notice that the sine and cosine values have been rounded and multiplied by 256 in Table 1. Finally, these values will be multiplied by the round number $\frac{1}{\sin(20)} \approx 2 + \frac{15}{16}$ and divided by 256 (this is the secret of the number $\frac{47}{4096}$ in Algorithm 3).

Algorithm 3 – Gradient calculation.

set dx **is** I(x, y+1) - I(x, y-1)
set dy **is** I(x+1, y) - I(x-1, y)
if 17 × abs(dy) >= 3 × abs(dx) **then**
  angle1, angle2 are (170, 150) if dx × dy < 0 else (10, 30)
else if 168 × abs(dy) >= 97 × abs(dx) **then**
  angle1, angle2 are (150, 130) if dx × dy < 0 else (30, 50)
else if 73 × abs(dy) >= 87 × abs(dx) **then**
  angle1, angle2 are (130, 110) if dx × dy < 0 else (50, 70)
else if 4 × abs(dy) >= 11 × abs(dx) **then**
  angle1, angle2 are (110, 90) if dx × dy < 0 else (70, 90)
else
  angle1, angle2= (170, 10)
set beta1 **is** max(0, (dx × sin(angle1)_256 - dy × cos(angle1)_256) × 47/4096)
set beta2 **is** max(0, (dy × cos(angle2)_256 - dx × sin(angle2)_256) × 47/4096)

Table 1 - Sine and Cosine quantized value.

$\boldsymbol{\theta}$	$\boldsymbol{cos(\theta) * 256}$	$\boldsymbol{sin(\theta) * 256}$	$\boldsymbol{\theta}$	$\boldsymbol{cos(\theta) * 256}$	$\boldsymbol{sin(\theta) * 256}$
10	252	44	110	-88	241
30	222	128	130	-165	196
50	165	196	150	-222	128
70	88	241	170	-252	44
90	0	256			

To generate a histogram, the first step is to group 8x8 pixel gradients and divide them into nine bins. The details of this process are presented in Algorithm 4. Next, we apply the histogram generation on all *64x128* sliding windows of the input image and collect the list of histograms. This will generate all the histograms for the input image used for the next step.

Algorithm 4 – Histogram generation.

```
function getHistogram(beta1, beta2, angle1, angle2, numBins)
    create histogram with numBins bins
    foreach data point i:
        determine bin indices based on angle1[i] and angle2[i]
        increment bin1 by beta1[i]
        increment bin2 by beta2[i]
    end for
    return histogram
end function
```

(b) Descriptor generation and Result

After calculating histograms, the next step is to use l2 normalization to generate descriptors for each sliding window of the input image Algorithm 5. The descriptor will be saved as a vector with dimensions of *3780x1*. At the same time, the pre-trained SVM weights will be read from memory and saved as another vector of *3780x1*. Each descriptor vector will be transposed and multiplied with the SVM weight vector. The result of this multiplication will be a float number. The threshold is set to *0.4*. If the result of this multiplication is greater than or equal to *0.4*, then it can be considered that the area detected an object, in this case, a human, is detected.

Algorithm 5 – Description generation.

```
for row in range(numVertCells - 1) then
   for col in range(numHorizCells - 1) then
      for i in range(8):
         if col % 2 = 0 then
            set blockHist[i] is histograms[row][col][i]
            set blockHist[9+i] is histograms[row][col+1][i]
            set blockHist[18+i] is histograms[row+1][col][i]
            set blockHist[27+i] is histograms[row+1][col+1][i]
         else
            set blockHist[i] is histograms[row][col+1][i]
            set blockHist[9+i] is histograms[row][col][i]
            set blockHist[18+i] is histograms[row+1][col+1][i]
            set blockHist[27+i] is histograms[row+1][col][i]
      set l2_norm is np.linalg.norm(blockHist) + 0.01
      divide blockHist by l2_norm
      descriptor.append(blockHist)
```

(3) Combine Module and NMS Module

The combining module is a component that combines all the bounding boxes of varying scales into a single list. This list contains information such as the position, size, scale, and HOG-SVM calculation results of the sliding window. However, since this output may result in multiple bounding boxes for the same object, it can lead to information redundancy. Therefore, the NMS module is required to process the list and reduce the number of bounding boxes to generate the final bounding boxes list.

Figure 17 – Non-Maximum Suppression [37].

This section covers optimization techniques and hardware implementations of the HOG-SVM algorithm for real-time object detection. Gradient calculations are simplified to achieve high efficiency, and parallel architectures, deep pipelining, and

direct memory access is used. The proposed detection block includes the optimized HOG-SVM module and a bilinear scale generator, combined module, and NMS to enable multi-scale processing, consolidate outputs, and remove redundancies. This block is crucial to this thesis's proposed object detection system.

2.4. Conclusions

In this chapter, a real-time object detection system architecture, which focuses on detecting humans, has been proposed. The system employs lightweight algorithms such as Zipfian Estimation for motion detection and an optimized HOG-SVM pipeline for classification. The critical aspects of the system design have been highlighted, which include the use of parallel HOG-SVM computation threads and post-processing techniques using non-maximum suppression.

The entire process of data flow through each system stage has been analyzed. First, the motion detection stage identifies the regions of interest. These regions are then passed on to the classification stage, where the HOG-SVM computation technique is used to classify them. After classification, the areas are merged and filtered to produce the final human detection bounding boxes. These boxes are then placed onto the original input image for further use.

The system aims to use lightweight algorithms to perform real-time object recognition problems on high-resolution image input sources. This thesis is a stepping stone towards implementing the system into hardware in the future. In the next chapter, the thesis will present the process of running this system on different datasets using the C/C++ language. The system architecture offers a promising pipeline for efficient and accurate human detection for real-world applications.

Chapter 3. Implementation and Evaluations

In the previous chapter, we presented the proposed system architecture for our thesis. This chapter will demonstrate how we simulated the software system with C++ language. We utilized input images from video sources of the MOT15 dataset to calculate the system's mean average precision (mAP) and measured the processing frame rate per second. These results will help us analyze the effectiveness of the proposed system. This chapter is divided into four sections. Section 3.1 describes the software, hardware, datasets, and other components used to set up the experiment simulation environment. Section 3.2 explains the mean average precision (mAP) metric used to evaluate object tracking performance and the frames per second (fps) metric used to measure processing speed. Section 3.3 presents the experiment's results, including the measured mean average precision and processing frames per second of the system, and Chapter Conclusions.

3.1. Experiment setup environment

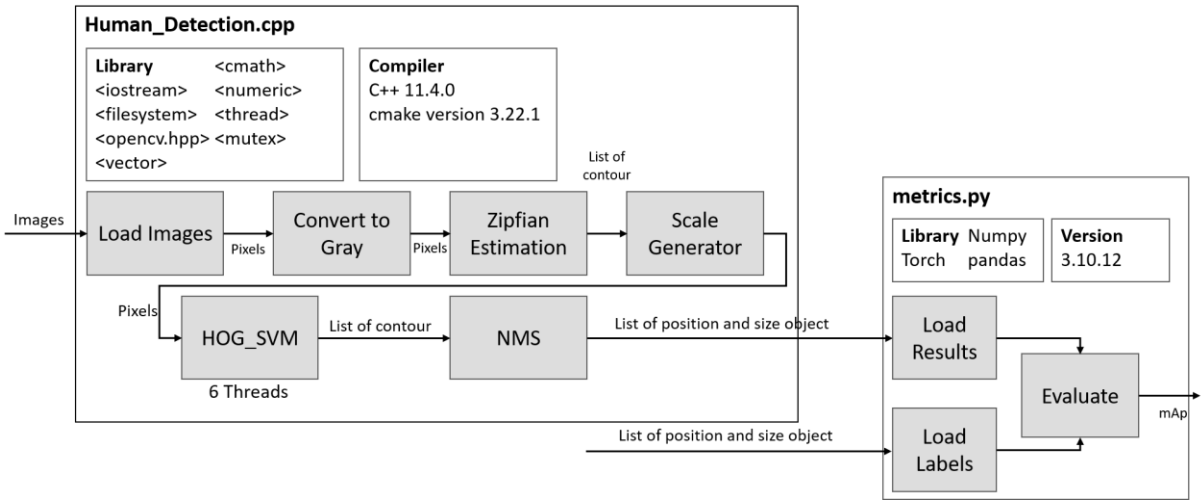


Figure 18 – Test System

Design a software testing program using the C++ language. The input will be images from the MOT15 dataset, which will be loaded using the OpenCV2 library. Next, the images will be fed into the proposed object recognition system, as described in the previous chapter. The program will record the position and size of the object detection areas on each frame. Also, record the processing time of each frame to determine the system's real-time response. This data will then be passed to a Python

program that uses the torch, pandas, and numpy libraries to compare it with the True Positive data file provided by the dataset. The results of this comparison will be used to calculate Precision, Recall, and mAP.

To set up the experiment environment, use a computer with an Intel® Core™ i5-1035G4 processor, 8GB of RAM, 128 GB SSD, and an Intel® Iris™ Plus Graphics card. Then, use the Windows Subsystem for Linux 2 to run Ubuntu 22.04.3 LTS on a Windows 11 OS. Evaluation Metric

We consider processing accuracy and speed to evaluate the proposed system's effectiveness. Accuracy is measured by mAP, while the number of frames per second measures speed.

3.1.1. Mean Average Precision

The Mean Average Precision (mAP) is the average AP over multiple IoU thresholds. To calculate mAP, two key concepts must be understood: precision and recall. These concepts are commonly used in classification problems. An overview of some essential concepts related to precision and recall in classification problems. If the model identifies an image as a human, it will be considered positive; if it determines it as a non-human, it will be regarded as negative. We will use True and False to indicate whether the model's classification matches the label.

Table 2 - Truth Table Confusion Matrix.

	Positive	***Negative***
Positive	True Positive	False Positive
Negative	False Negative	True Negative

(1) Intersection over Union (IoU)

The Intersection over Union (IoU) metric measures the overlap between the predicted and ground truth bounding boxes in object detection. A higher IoU score means the predicted box matches the ground truth box more closely.

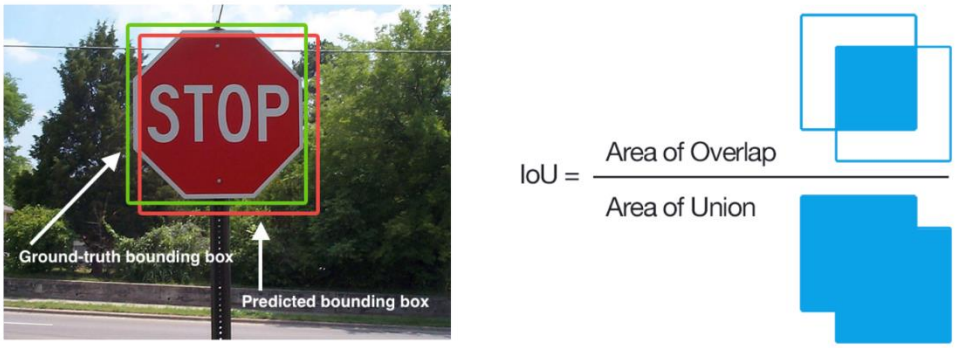


Figure 19 - Intersection over Union [38]

(2) Precision

The precision is the ratio of true positives to total positive predictions. It indicates the accuracy of a model in detecting an object.

$$precision = \frac{TP}{TP + FP}$$

(3) Recall

The recall ratio is the ratio of true positives to the total number of actual objects, assuming that the model can detect all relevant objects.

$$recall = \frac{TP}{TP + FN}$$

(4) Mean Average Precision (mAP)

Average Precision (AP) summarizes an object detection model's precision across all recall values by calculating the weighted mean precision at each recall threshold. It penalizes drops in precision at higher recalls.

Here is a summary of the steps to calculate the AP:

1. *Generate the prediction scores using the model.*
2. *Convert the prediction scores to class labels.*
3. *Calculate the confusion matrix—TP, FP, TN, FN.*
4. *Calculate the precision and recall metrics.*
5. *Calculate the area under the precision-recall curve.*
6. *Measure the average precision.*

The mAP is calculated by finding Average Precision(AP) for each class and then averaging over several classes.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi$$

3.1.2. Frame rate

The frame rate represents the number of frames that can be processed per second, with the unit being frames per second. This thesis uses two variables to measure each frame's processing time. The first variable is initialized when a new frame is loaded into the system for processing. The second variable captures the time when the frame processing is complete. Subtracting the first variable from the second gives the processing time for that individual frame. The average processing time across all frames is calculated. Dividing one by this average processing time results is the overall scanning frequency of the system for that dataset.

Mean Average Precision (mAP) and frame rate are important metrics for evaluating object detection systems. mAP is based on precision and recall, while frame rate indicates real-time performance. Using these standardized metrics makes it possible to compare different approaches and reasonably identify the system’s strengths and weaknesses.

3.2. Experimental results

3.2.1. Input: PETS09-S2L1 [39]

(1) Input description

The PETS09-S2L1 input includes images taken at a resolution of 768x578 and a speed of 7fps. The pictures appear to be captured from a high angle, overlooking a campus or an outdoor facility. The only motion in these images is from humans, often walking in unusual patterns.

Figure 20 - The image extracted from input PETS209-S2L.

(2) Result

In the results, the processing time for a single frame was found to be 55 milliseconds, which corresponds to an average frame rate of 18 frames per second. The analysis results indicate that the system's performance was moderate, with an average precision of 55%, an average recall of 17%, and an mAP score of 4%.

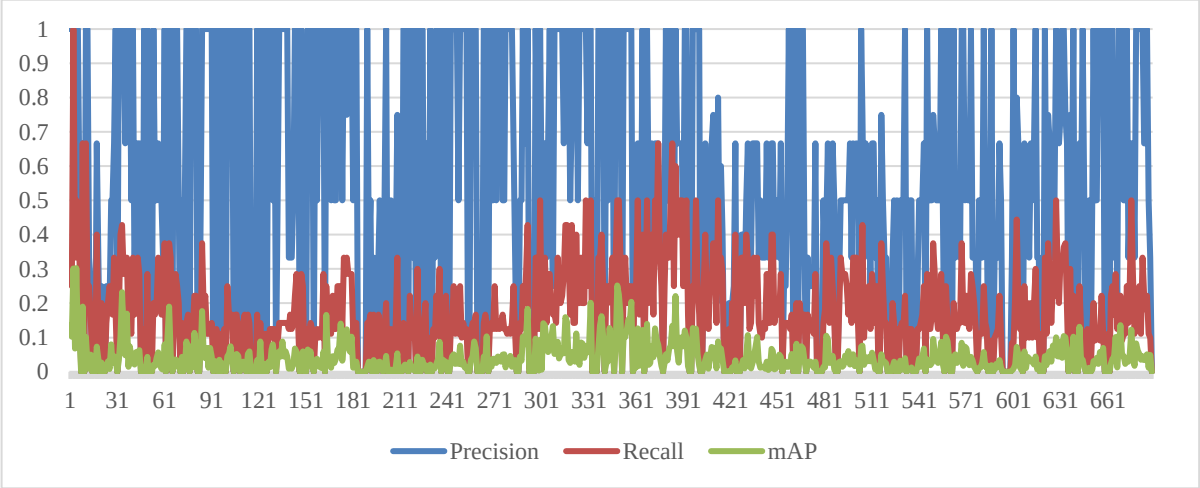


Figure 21 - Precision, recall, and mAP graphs per frame of input PETS209-S2L.

(3) Analysis

The system can process and analyze frames at a reasonable speed. However, it faces the challenge of low accuracy in identifying human movements correctly. This issue is demonstrated by the moderate precision, low recall, and very low mAP score.

3.2.2. Input: TUD-Stadtmitte [40]

(1) Input description

Input TUD-Stadtmitte is a series of images with a resolution of 640x480 and a speed of 25fps. This image depicts an urban pedestrian scene, likely a shopping area or street. The characteristic in the photo is that the moving object is only human moves in groups.

Figure 22 - The image extracted from the input TUD-Stadtmitte.

(2) Result

In the results, the processing time for a single frame was found to be 41 milliseconds, which corresponds to an average frame rate of 24 frames per second. The analysis results indicate that the system's performance was moderate, with an average precision of 51%, an average recall of 20%, and an mAP score of 7.7%.

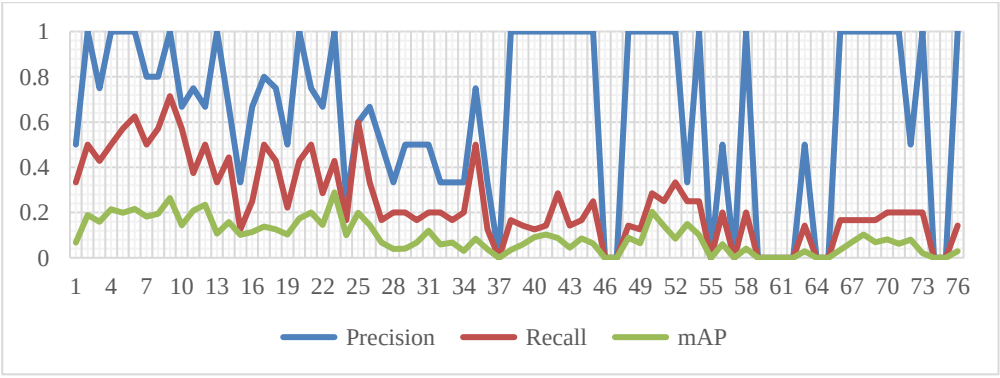


Figure 23 - Precision, recall, and mAp graphs per frame of input TUD-Stadtmitte.

(3) Analysis

The system can process images in near-real-time but faces significant challenges in accurately identifying and tracking human groups in urban pedestrian settings. The moderate precision combined with low recall and mAP scores suggests that the system might misidentify groups or miss them altogether.

3.2.3. Input: TUD-Campus [41]

(1) Input description

Input TUD-Campus is a series of images with a resolution of 640x480 and a speed of 25fps. The image captures a group of people walking outside a modern building with large windows reflecting the opposite scene.

Figure 24 - The image extracted from the input TUD-Campus.

(2) Results

In the results, the processing time for a single frame was found to be 84 milliseconds, which corresponds to an average frame rate of 12 frames per second. The analysis results indicate that the system's performance was moderate, with an average precision of 23%, an average recall of 21%, and an mAP score of 3.3%.

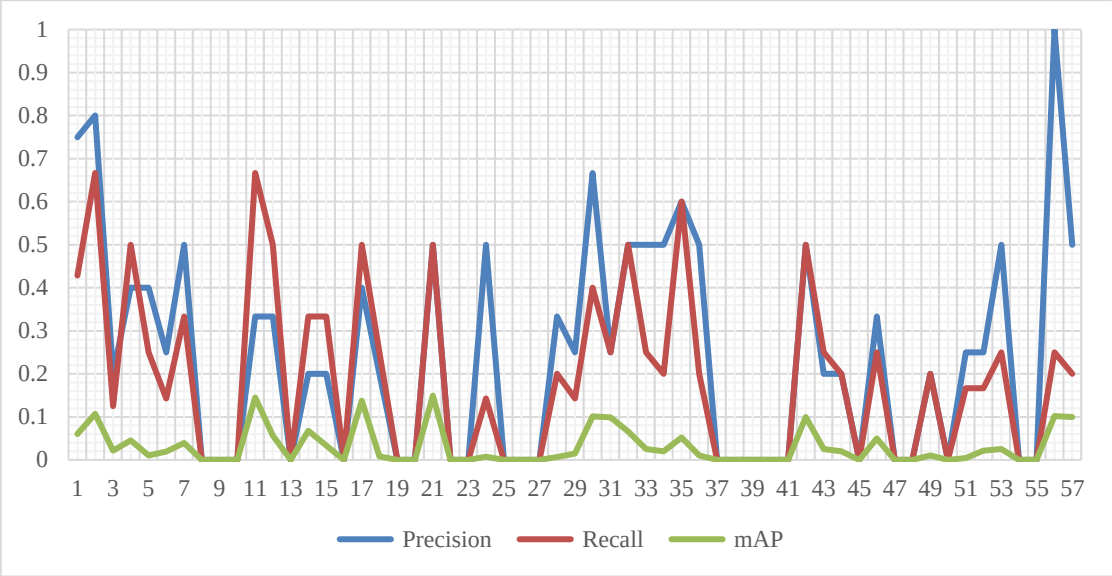


Figure 25 - Precision, recall, and mAp graphs per frame of input TUD-Campus.

(3) Analysis

The system analyzing the TUD-Campus dataset has some processing speed and accuracy limitations. The low precision, recall, and mAP values suggest that the algorithm might find identifying humans in the frames hard. The complexity of the photos, which often feature multiple overlapping humans and mirror reflections, can contribute to the lower quality performance of the system.

During the testing process, the proposed system encountered several limitations that led to relatively low precision, recall, and especially mAP coefficients. These limitations were caused by a variety of factors, including complex objects moving in groups, leading to encroachment, as well as objects with color tones similar to the background, which fragmented the object input for human detection algorithms. Additionally, recognition results were distorted due to data such as mirrors and shadows.

Moreover, the system lacked an algorithm to track objects when they appeared in the frame. Due to the limitation of the SVM training set, objects from various angles and directions could not be recognized. Despite these limitations, the system's strength lies in its speed, which is almost guaranteed.

The MOT15 dataset is an excellent benchmark for evaluating multiple object detection algorithms, particularly in human detection. The dataset presents a wide range of challenges such as occlusions, illumination variability, and scale of human figures. Despite advancements in technology, the dataset remains a challenging benchmark that tests the limits of current detection and tracking technologies. Even neural network algorithms find this dataset particularly challenging for accuracy, let alone systems that use simple feature extraction algorithms like the proposed one.

3.3. Conclusions

In this chapter, we conducted experiments to evaluate the performance of our proposed object detection system. We measured key metrics such as Mean Average Precision and frame rate using video inputs like PETS09, TUD-Stadtmitte, and TUD-Campus in the MOT15 Dataset.

The results demonstrate that while the system can process frames at reasonable speeds of up to 24 FPS, the accuracy of detection is limited, with medium precision (23-55%), relatively low recall (17-21%), and low mAP scores (3.3-7.7%). Factors like complex backgrounds, reflections, overlapped objects, limitations of the SVM classifier, not selecting suitable thresholds for each step like Zipfian, HOG_SVM, or NMS, and quantized values to optimize speed in steps contribute to these issues.

Although there are some challenges, the system has a positive aspect. It has low complexity and has been efficiently optimized for future hardware platforms, such as Xilinx's FPGA platform, to speed up the proposed system. Furthermore, the problems mentioned above can be significantly addressed by placing our system in a specific situation, providing a suitable training set of SVM weights, and conducting numerous corresponding experiments on several datasets with comparable conditions to the proposed problem. This will enable us to select appropriate thresholds for each step and ultimately complete the system. To improve object detection accuracy, we can use a tracking algorithm to follow an object. By doing this, we only need to run the HOG-SVM algorithm once on that object until it leaves the frame. This approach can help eliminate errors during the transition recognition process.

To summarize, the simulation system presented in this chapter has shown the fundamental weaknesses of a real-time object detection system. However, by calibrating the components in this proposed system, it is possible to improve the accuracy and verify that the algorithm's complexity is suitable to be deployed to hardware in the future.

Conclusions and Perspective

Real-time object detection here refers to quickly and accurately identifying objects in a digital image input data stream. It is one of the most important applications of computer vision engineering. It has been applied in many fields, from essential to military applications. However, balancing speed and accuracy, performance, and hardware configuration is considered one of the biggest challenges when researching this issue.

There are two main approaches: classical feature extraction algorithms and deep learning neural networks. Although the achievements of the deep learning approach in recent years have shown significant improvements in quality, accuracy, and processing speed, with its complexity, this method needs to be deployed on servers with powerful GPUs, which is a big challenge for edge device deployment. On the other hand, algorithms using classical feature extraction methods have lower performance but are lightweight, simple, and easy to deploy on edge devices. These algorithms also rely more on the designer than on training data like deep learning methods, which is quite difficult when collecting large amounts of data and optimizing the complex parameters of deep neural networks. And because it relies on the designer, it is possible to gradually improve its accuracy by calibrating the parameters, with far fewer parameters than deep neural networks. From state-of-the-art research, this thesis proposes a system using lightweight algorithms for deployment. With the case here being human detection with static camera input, the thesis proposes using the Zipfian Estimation technique combined with the HOG-SVM algorithm.

The proposed system is designed to detect objects specifically focusing on humans. It does this using a sequence of pre-processed images, which have been processed with Zipfian Estimation Techniques. The Object Detection module includes a Scale Generator module, six HOG-SVM modules that run in parallel using a multi-threaded architecture, and an NMS algorithm module. Using the Scale Generator and multi-threaded HOG-SVM computation modules, the system can detect all humans in multiple scales, reducing processing time. Finally, the results are merged with the

original image and can be viewed, stored, or transmitted to a server for further applications.

Finally, the experimental results on dataset MOT15 highlighted the current limitations of the proposed system. On the positive side, frame rates between 15-24 FPS demonstrated efficiency and hardware suitability. However, the accuracy metric mAP exposed the tradeoffs using simpler non-neural methods. In summary, while further development is still needed, this thesis provided a promising foundation for real-time detection that balances efficiency and accuracy. The system design and algorithms set the groundwork for future optimization on the hardware platform. Enhancements to components like the classifier, calibrated threshold parameters, and more controlled testing conditions can help overcome some of the accuracy limitations observed. Additionally, One idea is to use a tracking algorithm to follow an object, so that we only need to run the HOG-SVM algorithm once on that object until it leaves the frame. This approach can help eliminate errors during the transition recognition process. Overall, the proposed system shows the first step that, with improvement, could enable high-speed video analytics in real-time object detection.

Overall, this thesis has proposed a system application in real-time object detection. However, this system still has many limitations that need improvement. In the future, improving this system is necessary, in addition to improving the core components of the system, such as collecting and selecting the training SVM weights that are suitable for classification, aligning the threshold values of each step in the system, testing other algorithms and testing when comparing and combining different algorithms, combining non-neural and neural methods, etc. The second job that needs to be done is to deploy it to hardware. The platform here chooses an FPGA from Xilinx. Some methods can be deployed to hardware, such as running this system directly on the Linux operating system. of an embedded computer, using Matlab's HDL toolbox to package this model, using HLS to package the IP Core to hardware, or even directly deploying this IP Core using Verilog, etc. Finally, as Laozi said, "A journey of a thousand miles begins with a single step." this proposed system is the first step in bringing this system into reality.

References

[1] S. Dasiopoulou, "Knowledge-assisted semantic video object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, p. 1210–1224, 2005.

[2] [Online]. Available: https://en.wikipedia.org/wiki/ZALA_Lancet.

[3] "AI expands capabilities of surveillance and public safety tech," [Online]. Available: https://www.smartcitiesworld.net/ai-and-machine-learning/ai-expands-capabilities-of-surveillance-and-public-safety-tech.

[4] [Online]. Available: https://smcorridornews.com/cbps-autonomous-surveillance-towers-declared-a-program-of-record-along-the-southwest-border/.

[5] Haltakov, V., Belzner, H., & Ilic, S., "Scene understanding from a moving camera for object detection and free space estimation," *IEEE Intelligent Vehicles Symposium*, pp. 105-110, 2012.

[6] [Online]. Available: https://dashcam.70mai.com/a800s/.

[7] [Online]. Available: https://www.militaryaerospace.com/commercial-aerospace/article/14229347/raytheon-gets-order-for-180-aim9x-infraredguided-missiles-for-us-and-allied-forces.

[8] R. Girshick, "Rich feature hierarchies for accurate object detection and semantic segmentation.," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.

[9] R. Girshick, "Fast r-cnn," *Proceedings of the IEEE international conference on computer vision*, 2015.

[10] S. Ren, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems 28*, 2015.

[11] J. Redmon, "You only look once: Unified, real-time object detection,"

Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[12] W. Liu, "SSD: Single Shot MultiBox Detector," *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands*, 2016.

[13] Dalal, N., & Triggs, B., "Histograms of oriented gradients for human detection," *IEEE computer society conference on computer vision and pattern recognition*, 2005.

[14] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the seventh IEEE international conference on computer vision*, 1999.

[15] Viola, Paul, and Michael Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1, 2001.

[16] Wang, J., Ma, Y., Zhang, L., Gao, R. X., & Wu, D. , "Deep learning for smart manufacturing: Methods and applications.," *Journal of manufacturing systems 48*, pp. 144-156, 2018.

[17] Fang, Wei, Lin Wang, and Peiming Ren, "Tinier-YOLO: A real-time object detection method for constrained environments.," *IEEE Access 8*, pp. 1935-1944, 2019.

[18] Hoang, V. P., Ninh, H., & Hai, T. T. , "CBAM-YOLOv5 for infrared image object detection," *Artificial intelligence and machine learning in defense applications IV*, vol. 12276, pp. 116-127, 2022.

[19] C.-Y. A. B. a. H.-Y. M. L. Wang, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[20] Suleiman, A., Chen, Y. H., Emer, J., & Sze, V. , "Towards closing the energy gap between HOG and CNN features for embedded vision," *IEEE International*

Symposium on Circuits and Systems, pp. 1-4, 2017.

[21] O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., ... & Walsh, J, "Deep learning vs. traditional computer vision," *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC),* vol. 1, pp. 128-144, 2020.

[22] K. Takagi, K. Mizuno, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A sub-100-milliwatt dualcore hog accelerator vlsi for real-time multiple object detection," *IEEE-ICASSP*, p. 2533–2537, 2013.

[23] A. Suleiman, "An energy-efficient hardware implementation of HOG-based object detection at 1080HD 60 fps with multi-scale support.," 2016.

[24] Nguyen, N. D., Bui, D. H., & Tran, X. T., "A novel hardware architecture for human detection using HOG-SVM co-optimization," *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05).,* vol. 1, pp. 886-893, 2019.

[25] J. Li, X. Liu, F. Liu, D. Xu, Q. Gu, and I. Ishii, "A hardware-oriented algorithm for ultra-high-speed," vol. 19, 2019.

[26] Ishii I., Tatebe T., Gu Q., Takaki T., "Color-histogram-based tracking at 2000 fps," *J. Electron.Imaging. 2012;21:013010. doi: 10.1117/1.JEI.21.1.013010.,* 2012.

[27] Wasala, M., & Kryjak, T., "Real-time HOG+ SVM based object detection using SoC FPGA for a UHD video stream," *11th Mediterranean Conference on Embedded Computing (MECO),* pp. 1-6, 2022.

[28] Nguyen, N. S., Bui, D. H., & Tran, X. T., "Reducing temporal redundancy in MJPEG using Zipfian estimation techniques," *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS). IEEE,* pp. 65-68, 2014.

[29] A. Manzanera, "σ-δ background subtraction and the Zipf law," *Progress in*

Pattern Recognition, Image Analysis and Applications: 12th Iberoamericann Congress on Pattern Recognition, CIARP 2007, pp. 42-51, 2007.

[30] Lacassagne, L., Manzanera, A., & Dupret, A, "Motion detection: Fast and robust algorithms for embedded systems," *IEEE international conference on image processing (ICIP)*, pp. 3265-3268, 2009.

[31] Richefeu, A. M. J., "A robust and computationally efficient motion detection algorithm based on σ-δ background estimation," *Indian Conference on Computer Vision, Graphics & amp; Image Processing*, vol. 9, pp. 16-18, 2004.

[32] Dollár, P., Belongie, S., & Perona, P., "The fastest pedestrian detector in the west," 2010.

[33] Dollar, P., Wojek, C., Schiele, B., & Perona, P., "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence, 34(4)*, pp. 743-761, 2011.

[34] Cristianini, N., & Shawe-Taylor, J., An introduction to support vector machines and other kernel-based learning methods, Cambridge university press, 2000.

[35] Ho, H. H., Nguyen, N. S., Bui, D. H., & Tran, X. T, "Accurate and low complex cell histogram generation by bypass the gradient of pixel computation," *4th NAFOSTED Conference on Information and Computer Science*, pp. 201-206, 2017.

[36] Nguyen, T. A., Tran-Thi, T. Q., Bui, D. H., & Tran, X. T, "FPGA-Based Human Detection System using HOG-SVM Algorithm," *International Conference on Advanced Technologies for Communications (ATC)*, pp. 72-77, 2023.

[37] [Online]. Available: https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/.

[38] [Online]. Available: https://www.v7labs.com/blog/mean-average-precision.

[39] Ferryman, J. & Shahrokni, A., "PETS2009: Dataset and challenge.," *IEEE*

International Workshop on Performance Evaluation of Tracking and Surveillance, 2009.

[40] Andriluka, M., Roth, S. & Schiele, B. , "Monocular 3D Pose Estimation and Tracking by Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[41] Andriluka, M., Roth, S. & Schiele, B., "People-Tracking-by-Detection and People-Detection-by-Tracking," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[42] Ren, S., He, K., Girshick, R., & Sun, J. , "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems 28*, 2015.

[43] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. , "You only look once: Unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[44] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C., "SSD: Single Shot MultiBox Detector," *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands*, 2016.

[45] N. Dalal, "Histograms of oriented gradients for human detection," 2005.

[46] J. Wang, "Deep learning for smart manufacturing: Methods and applications.," *Journal of manufacturing systems 48*, pp. 144-156, 2018.

[47] Leal-Taixé, L., Milan, A., Reid, I., Roth, S. & Schindler, K., "MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking," 2015.