Daniel José Forte Craveiro

Detection of Humans and Animals in a Forestry Environment Using edge computing

Dissertation in the context of the Master's in Electrical and Computer Engineering, Specialization in Robotics, Control and Artificial Intelligence, advised by Prof. Doctor A. Paulo Coimbra, and Professor Doctor Aníbal T. de Almeida and presented to the Department of Electrical and Computer Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

February of 2023

1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Đ
COIMBRA

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Daniel José Forte Craveiro

Detection of Humans and Animals in a Forestry Environment

Dissertation in the context of the Master in Electrical and Computer Engineering, Specialization in Robotics, Control and Artificial Intelligence, advised by Prof. Doctor A. Paulo Coimbra, and Professor Doctor Aníbal T. de Almeida.

February 2023

Acknowledgements

First and foremost I would like to thank my parents and brother, they gave me love, support and the strength to have the resilience to keep pushing forward in the hardest of times, during my journey through this graduation.

I want to address my most profound appreciation for my grandparents, uncles, and cousins who supported me in crucial moments with family gatherings. Building a solid foundation for our family union was essential throughout this phase.

To my friends from Souselas, I express my sincere thanks for providing comfort and strength in the pursuit of the conclusion of this work, as well as for understanding when I couldn’t be in festive moments, you were always on my mind.

To team Eletro, Gonçalo Lopes, João Duarte, Daniel Palaio, Hugo Figueiras, Guilherme Carvalho and Francisco Alves I want to thank you for your friendship, companionship, help and motivation throughout this graduation.

A special appreciation to Daniela Garcia and my Psychologist Catarina Calado that had strong participation in the improvement from a dark moment I faced in 2022, you were crucial to my recovery and the conclusion of this work despite all.

Lastly, I want to express my sincere gratitude to my supervisor Dr Paulo Coimbra for his guidance, help and mentorship throughout this dissertation.

Abstract

Wildfires are a rising problem in the present days, requiring urgent measures to suppress the escalating of these events. One of the measures taken is prevention, involving the clearing of forests by removing flammable material. This action becomes repetitive and can represent a risk for the human if not done cautiously.

A solution for the dangers of this hard labour job is forest cleaning robots, providing the aid that humans needed to perform these tasks. When operating these robots one should be aware that they can be dangerous to their surroundings (humans, wildlife encounters, etc). To prevent this risk conveys a need to have an alert system for these robots.

This dissertation proposes a solution for a detection system for humans and animals in a forest environment. The solution uses an edge device, NVIDIA Jetson Nano, a thermal camera, FLIR ADK, and a deep learning detection model, the SSD MobileNet V2. The system uses frameworks and tools to gather a software solution to process the data obtained from the thermal camera and detection model, outputting the results in a graphical interface.

Furthermore, the work presents real-world testing of the system, as well as a discussion of the results obtained. Moreover, it is presented future improvement ideas for the system.

Resumo

Os incêndios florestais são um problema crescente nos dias de hoje, exigindo medidas urgentes para suprimir a intensificação destes acontecimentos. Uma das medidas tomadas é a prevenção, envolvendo limpeza de florestas através da limpeza de material inflamável. Esta tarefa torna-se repetitiva após algum tempo, para além disso, também representa um risco para o ser humano se não for feita com cautela.

Uma solução para os perigos deste trabalho árduo são os robôs de limpeza florestal, auxiliando ou até substituindo os humanos para fazer estas tarefas. Ao operar estes robôs deve-se estar ciente de que os mesmos podem ser perigosos para o seu meio em redor (humanos, animais selvagens, etc.). Para prevenir este risco é necessário possuir um sistema de alerta para estes robôs.

Esta dissertação propõe uma solução para um sistema de deteção de humanos e animais num ambiente florestal. A solução proposta utiliza o dispositivo móvel de aceleração gráfica NVIDIA Jetson Nano, a câmara térmica FLIR ADK, e o modelo de deteção baseado em inteligência artificial SSD MobileNet V2. O sistema utiliza ferramentas de desenvolvimento para construir uma solução de software com o final de processar os dados obtidos pela câmara térmica e do modelo de deteção, mostrando os resultados obtidos numa interface gráfica.

Além disso, o trabalho apresenta testes num ambiente real, bem como uma discussão dos resultados obtidos. Inclusivamente são apresentadas futuras ideias de melhoria para o sistema em causa.

Contents

Acronyms

AI artificial intelligence.

ANNs Artificial Neural Networks.

AP Average Precision.

CNN Convolutional Neural Network.

COCO Common Objects in Context.

CPU central processing unit.

CUDA Compute Unified Device Architecture.

FPS frames per second.

GPU graphic processing unit.

IoT Internet of Things.

IoU Intersection Over Union.

mAP mean average precision.

ReLU Rectified Linear Unit.

SDK Software Development Kit.

SSD Single Shot MultiBox Detector.

VRAM video random access memory.

YOLO You Only Look Once.

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the context and motivation for this work, as well as the proposed solution and its technical requirements.

1.1 Context and Motivation

Forest fires are responsible for a significant abiotic disturbance in natural and planted forests, causing countless damage in large areas around the globe. In territories more prone to forest fires, particularly in the South Europe countries, the risk of uncontrol is constant and recurring.

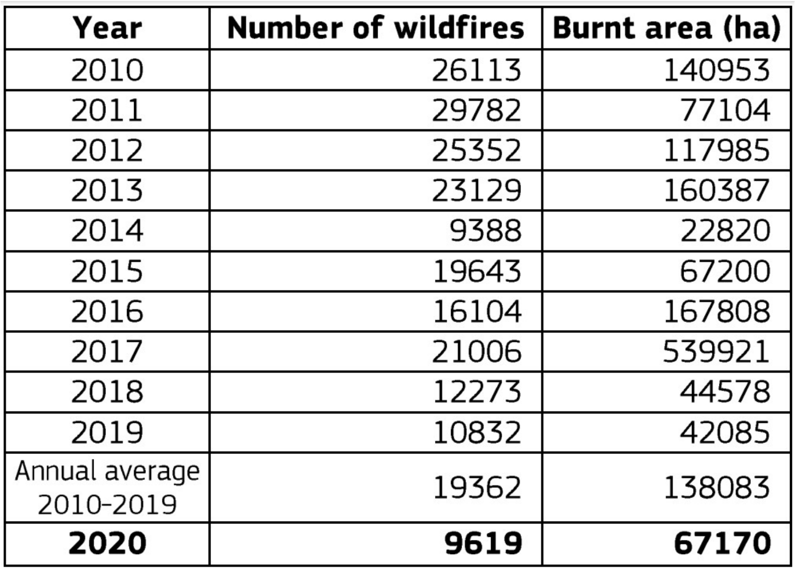
In 2020, 3400 km^2 of land was burnt in the European Union, from this land, 40% were protected areas of Europe Natura 2000, it will take years to restore and recover to what they were. Fires have ceased to affect the southern countries only, and are now a growing threat to Central and North Europe. Global warming is now producing fast-propagating wildfires that everyday firefighting can't control. This effect is seen in Europe and California(USA), Australia, Central and South America, Russia, Turkey, etc.

Efforts must be taken to mitigate this problem and minimize the impact of these wildfires and reduce their effects, not only in economic damage but also environmental impact and human losses. Today, with greater importance, we need to be more aware of the forest fire risk and fire prevention strategies at the European and National levels initiating strong cooperation between European countries and their nearest as they are all victims of this critical fire situation [1].

Portugal is part of the South Europe countries that have been suffering high losses due to rural fires, consequently losing part of its forestry. The Portuguese forest represents 3.2 million hectares which are equivalent to 36.2% of Portugal's mainland [2]. As shown in table 1.1, the burnt area annual average due to wildfires between 2010 and 2019 in Portugal was 138083 hectares which translate to 4.3% of the current forest territory.

Therefore it is crucial to develop new methods that prevent and control forest fires. One such method is the use of forest cleaning robots, which can effectively reduce the risk of fires by cutting flammable material such as dead vegetation, leaves, twigs and branches [3].

Table 1.1: Number of wildfires and burnt area in Portugal mainland from 2010 to 2020 [1].



Year	Number of wildfires	Burnt area (ha)
2010	26113	140953
2011	29782	77104
2012	25352	117985
2013	23129	160387
2014	9388	22820
2015	19643	67200
2016	16104	167808
2017	21006	539921
2018	12273	44578
2019	10832	42085
Annual average 2010-2019	19362	138083
2020	**9619**	**67170**

However, the deployment of forest cleaning robots also presents new risks and challenges, such as the potential for accidents and damage to the surrounding environment. To minimize these risks, it is essential to implement effective accident prevention systems on these robots. Accident prevention systems can provide real-time monitoring and analysis of hazardous situations, allowing for quick and accurate responses to prevent accidents.

In this work, we explore the importance of a solution for an accident prevention system with the ability to detect human and animal presence. This solution was met with the help of a thermal camera and an edge device powered by a deep learning algorithm.

This dissertation is part of the SafeForest project whose partners are the company Ingenarius, the Institute for Systems and Robotics of the University of Coimbra (ISR-UC), ADAI (Association for the Development of Industrial Aerodynamics), the company SILVAPOR and Carnegie Mellon University.

The SAFEFOREST project focuses on the creation of a semi-autonomous robotic system for the prevention of wildland and wildland-urban interface fires. Its goal is to significantly reduce the cost of maintaining forests, particularly near homes and critical infrastructure, and to control the spread of large forest fires.

To achieve this, SAFEFOREST aims to develop innovative monitoring and robotic systems that can semi-automatically clear and manage forest fuels in areas with challenging terrain. These systems will consist of semi-autonomous mobile platforms that use advanced drone technology to map the area and identify areas in need of land clearing. These platforms will then remove redundant vegetation and create necessary fuel breaks. This will be made possible through the integration of multiple sensors on a semi-autonomous all-terrain platform, capable of functioning in a variety of terrains. Advanced multi-sensor drone systems will monitor the forest and assist with terrain mapping, ensuring the efficient and effective implementation of the project's goals [4].

This dissertation is organized into four chapters. Chapter 2 covers the main theory behind the development of this work, mainly a deep learning introduction, graphic processing units (GPUs) overview, as well as state-of-the-art object detection algorithms and thermal cameras used in the work. Chapter 3 covers the methods followed in the execution of this work, and does an overall system overview of the solution developed, the software involved, as well as the procedure to evaluate it in a real-world environment. Then, in Chapter 4 we present the results obtained from the real-world testing of the system, as well as a discussion of the results. Finally, in Chapter 5, a conclusion is presented assessing the requirements of the solution as well as a possible future work section.

1.2 Proposed solution

The solution to solve the problem of detection of humans and animals in a forestry environment is composed of an edge device powered by a GPU called NVIDIA Jetson Nano, the FLIR ADK thermal camera, and the detection model SSD MobileNet V2. In section 3 we go into detail about this system.

1.3 Requirements

To evaluate the solution proposed and assess its good functionality a few requirements were elaborated on:

- Successful detection at a distance of 3 meters from the target.
- Different testing environments (forestry, plain grass).
- Human and dog detection in different poses.
- Detection with different light conditions (direct sunlight, shade).
- Detection at further distances than 3 meters from the target.
- Live inference and post-captured video inference as a benchmark.

Chapter 2

Fundamentals and concepts

The fundamental ideas of deep learning as well as related research in the fields of deep learning, edge machine learning, and thermal cameras are addressed in this chapter.

2.1 Deep learning

Deep learning is at the forefront of today's AI technology due to its recent achievements, which range from spotting cat photographs on the Internet to defeating chess and go masters, translating texts, and determining a protein's 3D shape from its amino-acid sequence. Artificial Neural Networks (ANNs) were the original title for deep learning methods, and as more layers of artificial neurons were added to ANNs, the name Convolutional Neural Network (CNN) gained popularity. Image classification and object detection are frequent uses of computer vision. The goal of image classification, which is primarily utilized for photographs of single objects, is to categorize the input image as a whole. Object detection is frequently employed when there are several objects present because the deep learning algorithm both classifies and locates each object that is found.[5]

2.1.1 Convolutional neural networks

One of the best learning algorithms for comprehending picture content, CNNs have demonstrated outstanding performance in tasks involving image segmentation, classification, detection, and retrieval. Multiple learning stages made up of a combination of convolutional layers, non-linear processing units, and subsampling layers make up CNN's topology as seen in figure 2.1 b).
Using a bank of convolutional kernels, each layer of CNN's multilayered hierarchical network executes several transformations. This convolution process aids in the extraction of valuable information from locally associated input points shown in figure 2.2 a).

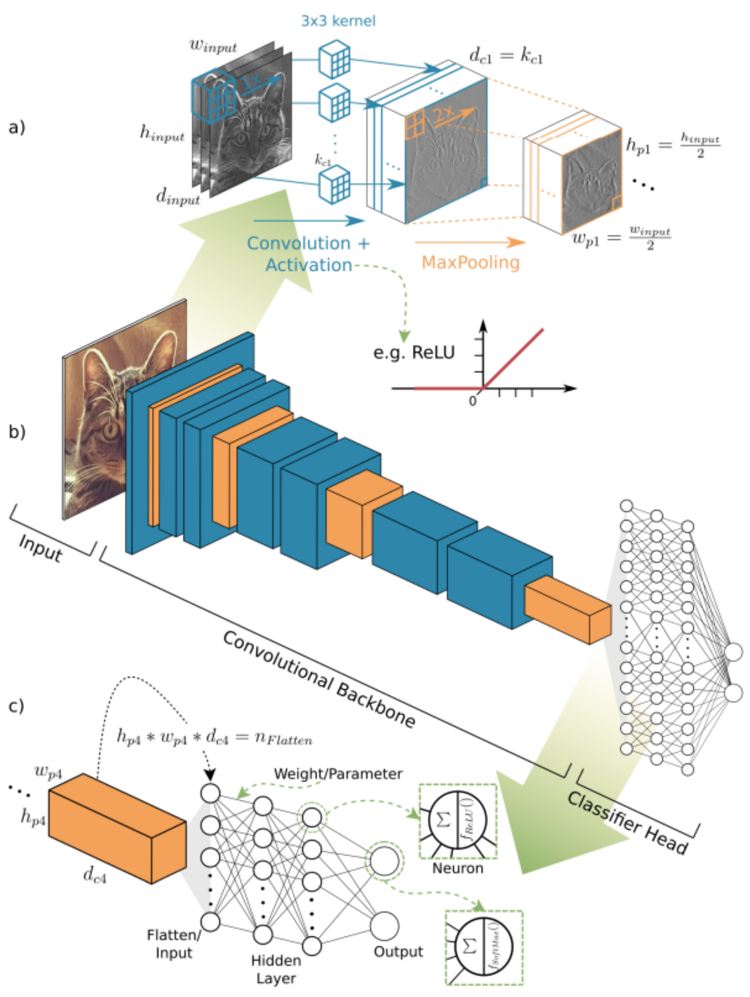


Figure 2.1: Overall overview of a Convolutional Neural Network (CNN) architecture. [6]

The activation function, which not only aids in learning abstractions but also embeds non-linearity in the feature space, receives the output of the convolutional kernels. This non-linearity produces various activation patterns for various responses, making it easier to learn the semantic differences between images. Another key stage in creating the neural network infrastructure is choosing the appropriate loss function. When using neural networks, our goal is to reduce error, which is represented by the loss function as the difference between the actual value and the value that was predicted. Both of these functions are shown in Figure 2.1 c) [7].

2.2 Edge machine learning

Machine learning is a very computationally demanding technology, but when working with specific problems or challenges we can move this technology to the edge and implement machine learning on specific devices (such as Internet of Things (IoT) devices, smartphones, etc.) as opposed to the cloud or centralized server, this process being referred to as edge machine learning.

On this matter, [8] published an article comparing 3 commonly used edge machine learning devices for high-performance deep learning applications: NVIDIA Jetson Nano and TX2, and the Raspberry Pi 4. The work evaluated the performance of the three devices by training a CNN on different amounts of data on a single dataset (13 different fashion products with 45K images) and assessing the accuracy, time of processing, memory and power, see table 2.1. The jetson TX2 showed to be the better device in every metric evaluated but at a cost of about 4 times the cost of the NVIDIA Jetson nano. The NVIDIA Jetson nano proved to be an attractive choice for high-performance deep learning applications.

Table 2.1: Benchmarking results for Jetson TX2, Nano and Raspberry Pi [8].

	Acc (%)			**Time(sec)**			**Memory (GB)**			**CPU (Power/W)**			**GPU (Power/W)**		
Dataset	**TX2**	**Nano**	**PI**	**TX2**	**Nano**	**PI**	**TX2**	**Nano**	**PI**	**TX2**	**Nano**	**PI**	**TX2**	**Nano**	**PI**
Idle	-	-	-	-	-	-	1,9	1,5	1,4	0,675	0,47	0,30	2,6	0,76	-
5K	87,6	87,5	87,2	23	32	173	2,6	2,0	2,1	2,23	1,50	3,5	5,27	2,23	-
10K	93,8	93,9	91,6	32	58	372	3,1	2,75	2,6	2,78	2,32	3,6	5,32	3,25	-
20K	94,6	94,5	-	52	-	462	4,5	ERR	4,0	3,76	-	3,9	5,22	-	-
30K	96,4	-	-	122	-	-	5,2	ERR	ERR	4,25	-	-	5,74	-	-
45K	97,8	-	-	235	-	-	6,5	ERR	ERR	4,92	-	-	6,29	-	-

2.2.1 Relevant work on edge machine learning

In 2022, Wang et al [9], developed a real-time monitoring system of ship targets for intelligent army weapon equipment. The jetson nano micro-computer and a smaller version of the network You Only Look Once (YOLO) v5 were used, showing a fast detection speed and short training time, meeting the requirements for real-time maritime battlefield monitoring.

Minh et al [10] presented a work based on performance evaluation of deep learning models on edge devices, specifically real-time traffic tracking and detecting applications, it was concluded that models such as MobileNet-SSD were capable of achieving good detection speed (40 FPS), high accuracy and high mean average precision (mAP) during the training session.

Furthermore, [11] developed a solution for real-time monitoring of agricultural land with crop prediciton and animal intrusion prevention using IoT and edge machine learning. This project had sensors (such as soil moisture, infrared, rain, infrared, etc.) connected to a small edge device called Raspberry Pi 3. The edge device had a CNN installed so it could detect animal intrusion and with this information fear the animal in question using a speaker connected to the device.

2.2.2 TensorRT

TensorRT is a Software Development Kit (SDK) that enables high-performance machine learning inference working in collaboration with training frameworks such as TensorFlow, PyTorch and MXNet. This framework allows us to quickly and effectively operate a trained network on NVIDIA hardware (Jetson Nano, Jetson Xavier, etc.) and performs up to 36 times faster than CPU-only platforms during inference (a process of using existing knowledge to make predictions on new, unseen data) [12].

The training of a network can be done on a full-size machine or GPU. After this training, all the network nodes are defined on a model file, along with their operations, inputs, and outputs. In addition, training creates a checkpoint file that captures the values of all the weights and biases. This pre-trained representation of the network is deployed using the TensorRT framework on the edge device (Jetson nano, Jetson TX2, etc) to be used for inference [13].

2.2.3 Detection Models

A detection model is made of a base network and a classifier head, see figure 2.2. The base network is a pre-trained neural network which is used to capture features from the input data of the detection model. The classifier head outputs the prediction result of the data intake.

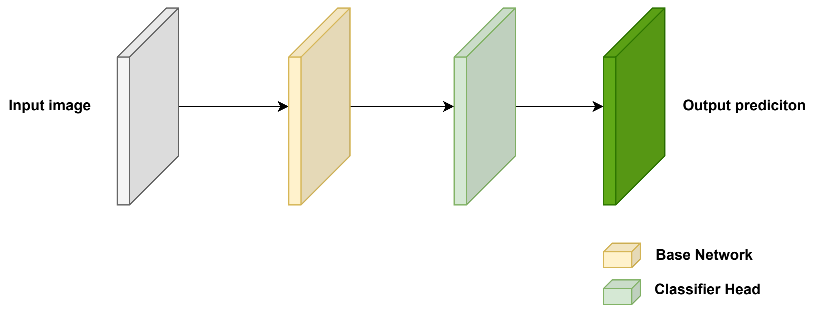


Figure 2.2: Overall overview of a detection model architecture.

The model's structure, size and complexity is determined by the use case of the challenge we face. Depending on the number of operations needed for the end goal, the choice of the model requires a few benchmarking and testing before determining which one to use.

Single shot multibox detector (SSD)

The Single Shot MultiBox Detector (SSD) is single deep neural network released in 2016, [14] that attained new records of performance and accuracy for tasks involving object detection. Seen in figure 2.3, the model takes advantage of the VGG-16 architecture discarding the fully connected layers of the network. After the VGG-16 convolutional layers the model SSD counts with additional convolutional layers with decreasing sizes. These layers happen to be seen as a pyramid representation of diverse scales. Large fine-grained feature maps at earlier levels capture small objects and small coarse-grained feature maps detect large objects efficiently. The detection occurs in every pyramidal layer, targeting objects of different sizes [15].

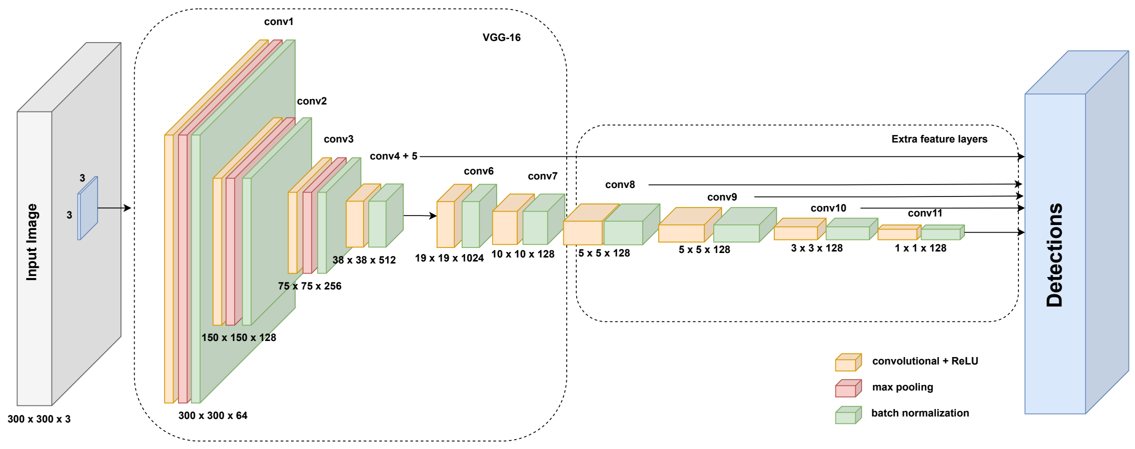


Figure 2.3: Overall overview of the SSD model architecture.

Some object detection systems use this method, which involves predicting bounding boxes, resampling pixels or features for each box, and then using a high-quality classifier. While this method remains accurate, this strategy is computationally heavy and doesn't serve the purpose of embedded devices and is slower in real-time appliances.

The SSD solves the issue of a high-quality classifier (computationally heavier) by lowering the output space into a cluster of default bounding boxes with different aspect ratios and dimensions per feature map. When predicting, the model generates scores for the existence of each object category in each default box to match accordingly with the shape of the object [14].

MobileNet V2

Mobilenet V2 is a highly efficient and lightweight CNN designed for mobile and embedded vision applications. This network uses shortcut connections between layers and depth-wise convolutions to maintain a low operational load. The model is composed of three convolutional layers stacked in a block as seen in figure 2.4. The first layer is a 1x1 expansion layer which has the function of expanding the number of channels that flow through it. The second layer is the depth-wise convolutional layer that is in charge of filtering the input, followed by the 1x1 projection layer or bottleneck layer. The projection layer reduces the number of channels and diminishes the amount of data that flows via the network [16].

Each layer of the network has a batch normalization layer and activation function (Rectified Linear Unit (ReLU) - 6) besides the projection layer. ReLU-6 is an extension of the standard ReLU function, where instead of having an upper bound of infinity, it has an upper bound of 6 (see equation 2.1), making the network learn sparse features earlier [17].

$$min(max(x,0),6) \tag{2.1}$$

The block's input and output are both low-dimensional tensors, but the block's internal filtering is carried out on a high-dimensional tensor.The residual link is only utilized when the number of channels entering and leaving the block are equal in order to aid in the flow of gradients through the network [16].

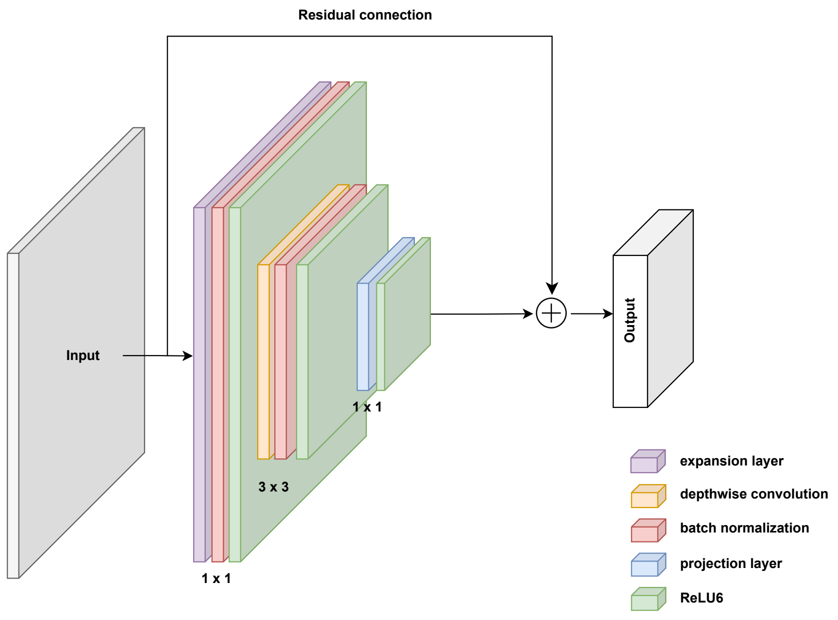


Figure 2.4: MobileNet V2 bottleneck residual block.

While SSD is conceived to work under the independence of the base network, one can stack SSD as the classifier head and MobileNet V2 as the base network combining both architectures to improve the overall performance of the model (see figure 2.5). SSD provides high accuracy and real-time performance, while Mobilenet V2 delivers low computational complexity and high accuracy [18].

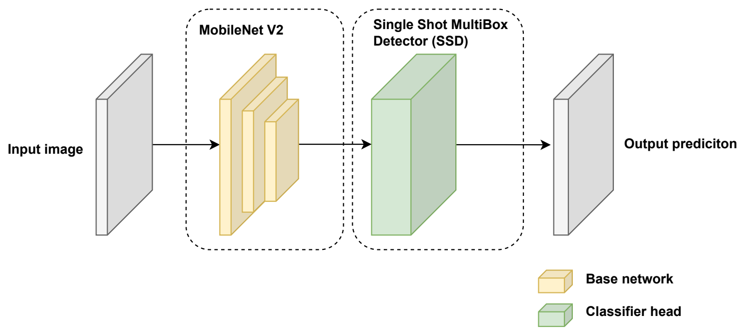


Figure 2.5: Detection model of MobileNet V2 as base network SSD as the classifier.

You Only Look Once (YOLO) version 7

At the current date, YOLO v7 surpassed every object detector both in speed and accuracy in the 5 to 160 frames per second (FPS) range, achieving 56.8% Average Precision (AP) among most real-time object detectors capable of attaining 30 FPS or more on NVIDIA V100 graphic processing unit (GPU)[19].

In 2022, Wang et al [20] published a research paper regarding the detection of defects in the steel strip production process. For this end, they used YOLO v7 to address the detection of these defects achieving 80,2% mAP on the GC10-DET dataset (a metallic surface defect dataset) accomplishing a result which is better than the other models at that time of publishing. In India, potholes account for the majority of fatal and injury-causing incidents. [21] developed a phone application with the ability to report potholes on roads. The system usedYOLO v7 network to detect the pothole, identify its location and determine the pothole count.

The YOLO v7 model architeture consists of (see figure 2.6)[20]:

- The **backbone network**, is a CNN that is used to extract the features from the input image. The backbone network in YOLO v7 uses a variation of the EfficientNet model, which is renowned for its excellent performance and high efficiency.
- The **neck network**, which links the backbone network to the head network refines the features extracted by the backbone network.

- The fully linked **head network**, which predicts the kind and placement of objects in a picture using the fine-tuned features from the neck network.
- A last layer, which is used as an output for classifying and locating objects in the image.

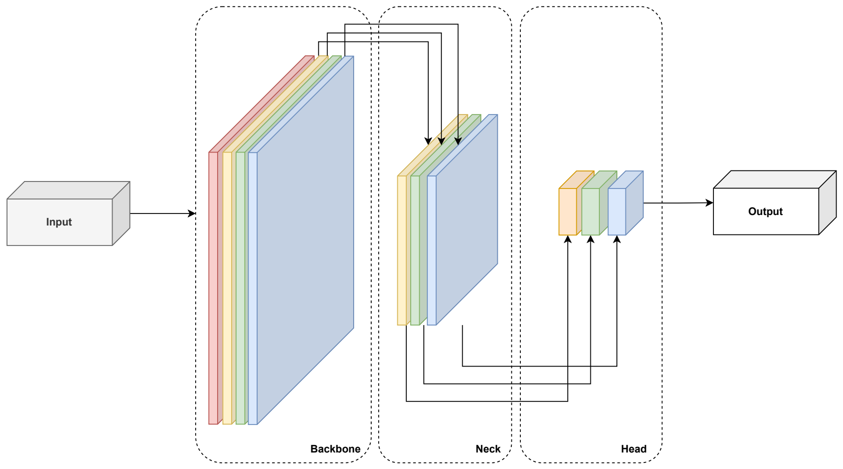


Figure 2.6: YOLO v7 model architecture diagram: input, backbone, neck, head and output.

While being a much more complex and computationally heavier network, the YOLO v7 remains a good model for benchmark reasons.

2.3 Detection model evaluation

In this section, we address the evaluation metrics used in object detection. We discuss typical evaluation metrics used in the machine learning industry, such as recall, precision, and Intersection Over Union (IoU).

2.3.1 Recall and precision

On binary classification applications, the confusion matrix can be used to define the discrimination evaluation of the optimal answer during the classification training, as shown in table 2.2. The predicted class is shown in the table's rows, while the ground truth class is shown in the columns.

As a practical demonstration of the terms using the human class detection application: true positive is when a human is detected as a human, a false negative is when the detection model fails to detect a human, the true negative is when the detection model effectively didn't detect an animal.

Table 2.2: Confusion matrix for binary classification [22].

	Actual Positive Class	**Actual Negative Class**
Predicted Positive Class	True positive (*tp*)	False positive (*fp*)
Predicited Negative Class	False negative (*fn*)	True negative (*tn*)

$$precision = \frac{tp}{tp + fp} \tag{2.2}$$

$$recall = \frac{tp}{tp + fn} = \frac{tp}{predicted\ positives} \tag{2.3}$$

Precision and recall can be calculated using equations 2.2 and 2.3 respectively, they are used to determine a model's average precision. In terms of the percentage of right predictions, precision expresses how precise the predictions are. Recall gauges how well a model finds every occurrence.

The Average Precision (AP) is a commonly used metric to evaluate object detection models, and it translates as the weighted mean of precision scores achieved at each precision-recall curve (figure 2.7) threshold. The general definition for the Average Precision (AP) is given by calculating the area under the precision-recall curve, this area is given by the integral in equation 2.4 (*r* being the recall and *p* the precision).

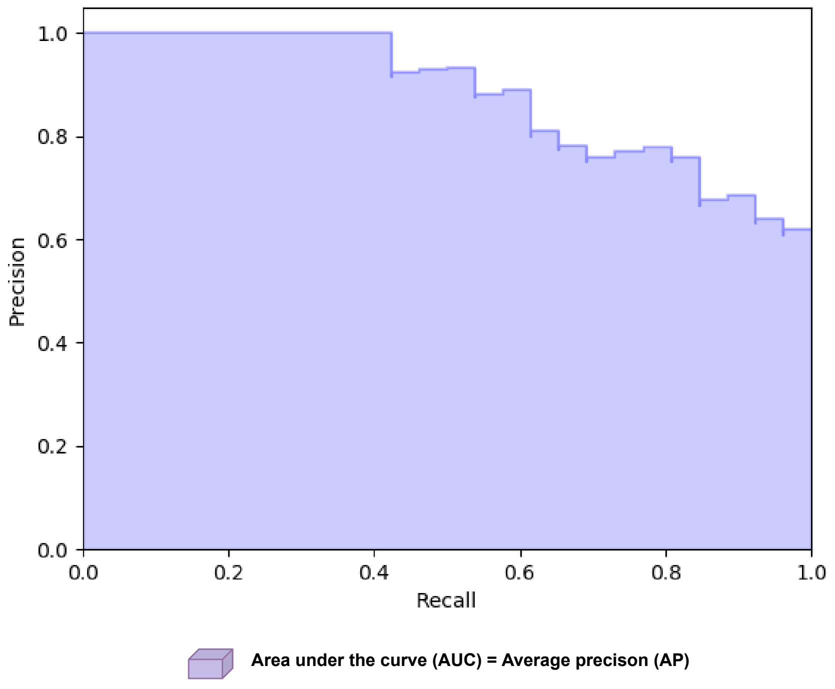


Figure 2.7: Precision-recall curve and the AP.

$$AP = \int_0^1 p(r)dr \tag{2.4}$$

Another metric used for evaluating a detector when multiple classes are present is the mean average precision (mAP), which measures the AP across multiple classes. To compute the mAP we add the average precisions for each class and divide them by the number of classes, see equation 2.5 (*N* stands for number of classes).

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{2.5}$$

2.3.2 Intersection over union

Intersection Over Union (IoU) is an evaluation metric that is used to assess how well object detection models are behaving. It calculates how closely an image's predicted bounding box and the actual bounding box match one another [23]. The area of intersection over the area of union of the two bounding boxes is used to calculate the intersection over union, or IoU, see equation 2.6.

$$IoU = \frac{area\ of\ overlap}{area\ of\ union} \tag{2.6}$$

The predicted bounding box perfectly overlaps the ground truth bounding box when the IoU value is 1, whereas a value of 0 indicates that there is no overlap between the bounding boxes. A common threshold value for IoU is 0.5, which means that if the predicted bounding box's IoU is greater than or equal to 0.5, it is considered a true positive, otherwise a false positive. A practical example of the metric can be seen in figure 2.8.

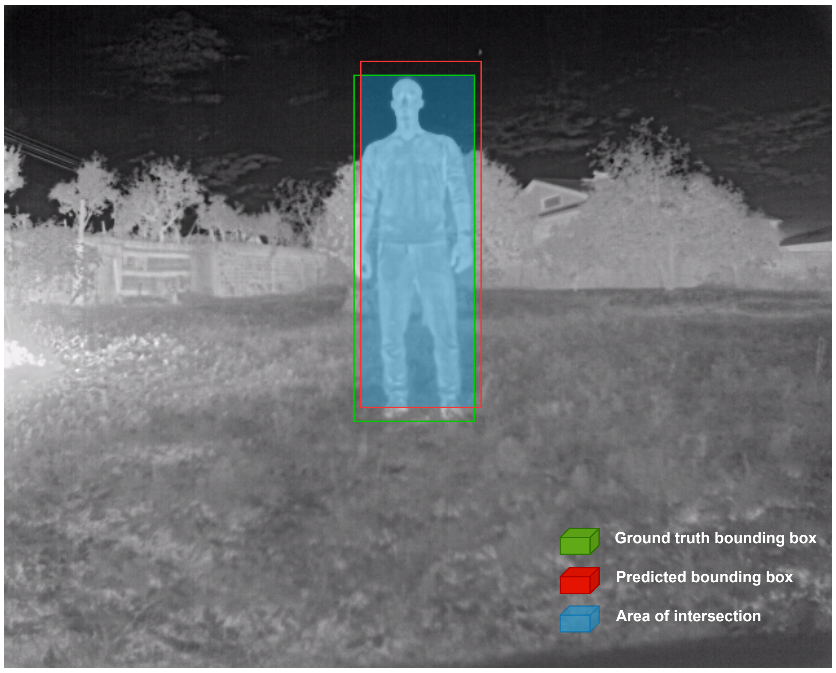


Figure 2.8: Intersection over union evaluation metric.

2.3.3 COCO dataset evaluation

The Microsoft team created the Common Objects in Context (COCO) dataset, which was originally made available in 2014. It is made to aid research into artificial intelligence, machine learning and computer vision [24].

To train and test object detection models, researchers frequently employ the COCO dataset. The COCO dataset offers a huge and varied set of images with a wide variety of items and contexts, and object detection models are trained to recognize and locate things inside an image.

The 80 object classes that make up the COCO dataset each have a different set of object examples. The collection of these 80 object classes contains details about the object type, location, size, and each instance of the object is identified by an ID. The collection also contains a number of captions that list the items in each image.

Additionally, the COCO dataset offers a leaderboard with 12 evaluation metrics that serve as benchmarks for object detection models. It provides a way to compare the performance of different models on the COCO dataset and track progress in the field of object detection [25].

2.4 Thermal cameras and human detection

Our focus in this work is on human and animal detection using thermal camera images. The importance of human detection for different applications, including self-driving automobiles and driver support systems, has led to extensive research on the subject [26]. For visible-range color cameras to detect humans, a variety of techniques have been suggested. Recent research attempts to solve challenges brought on by the visible-range camera sensor, including occlusions, person stature, and differences in lighting [27].

The majority of the current approaches for detecting humans start to fail when visible-range cameras are unable to gather enough data from a scene because there is not enough light for illumination. The issue is exacerbated when there is no light source and it is pitch black.

The issue of low light conditions is solved by a different kind of sensor, specifically a thermal or long-wave infrared camera sensor, which can collect useful data from dimly illuminated or completely dark surroundings. These sensors are primarily made to measure the temperature of a scene, making it possible to take photographs of people that are plainly apparent due to their body temperatures. The performance of thermal cameras is nevertheless impacted by other issues. For instance, the weather may have an impact on the pixel values in a thermal image. More, on a chilly day, humans might stand out significantly against the background. On warmer days, however, there may be less contrast between people and the background, which can make it harder to detect humans. Another challenge is that it can be difficult to discern between a person and another hot item when there are various objects close to a human body [28][29].

More recently, sensors with higher resolution thermal image capture, like the FLIR A-35, FLIR ADK and Raytheon 300D, are performing better at detecting humans in warmer conditions, doing a better job at differentiating humans from the background.

Chapter 3

Method

The following chapter describes the methodologies, tools, hardware and software used to address the challenge of human and animal detection.

3.1 Hardware

This section presents the software and hardware used in the development of this work. It includes a brief overview of graphic processing units (GPUs) and cameras.

3.1.1 Graphic processing units (GPUs)

A graphic processing unit (GPU) is a specialized type of processor designed for handling graphics operations and tasks that involve massive parallel processing. Deep learning applications involve large amounts of matrix and vector operations and GPUs can perform these operations more efficiently than central processing units (CPUs). Furthermore, GPUs usually have more cores than CPUs, allowing them to complete more operations in parallel, resulting in faster neural network training times and detection.

As shown in section 2.2 the NVIDIA Jetson Nano proved to be a small powerful edge device powered by a small GPU capable of achieving high accuracy results in the training of a CNN as well as a small power consumption when operating. With this in mind, this edge device was determined to be our choice to develop this work. Some characteristics of this device can be seen in table 3.1.

Table 3.1: NVIDIA Jetson Nano characteristics [30].

	NVIDIA Jetson Nano
Performance	472 GFLOPS
CPU	Quad-Core ARM Cortex-A57 64-bit @ 1.42 GHz
GPU	NVIDIA Maxwell w/ 128 CUDA cores @ 921 MHz
Memory	4 GB LPDDR4 @ 1600MHz, 25.6 GB/s
Networking	Gigabit Ethernet / M.2 Key E
Display	HDMI 2.0 and eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Other	40-pin GPIO
Video encode	H.264/H.265 (4Kp30)
Video decode	H.264/H.265 (4Kp60, 2x 4Kp30)
Camera	MIPI CSI port
Storage	16 GB eMMC
Power under load	5W-10W
Price	$89

For evaluation and benchmark purposes, another GPU was chosen to take part in this work, in this case, a full-size GPU which is a GPU that is designed to fit into a standard desktop computer. Additionally, a full-size GPU typically has a higher number of cores, video random access memory (VRAM), and a wider memory bus providing improved performance for demanding tasks such as CNN training, or object detection. The GPU used was an NVIDIA RTX 2060, and some of its characteristics can be seen in table 3.2.

Table 3.2: NVIDIA RTX 2060 characteristics [31].

	NVIDIA RTX 2060
Performance	7.5 GFLOPS
CPU	-
GPU	NVIDIA Turing w/ 1920 CUDA cores @ 1365 MHz
Memory	6 GB GDDR6 @ 14GHz, 336 GB/s
Networking	Gigabit Ethernet / M.2 Key E
Display	3x DisplayPort 1.4, HDMI 2.0b
USB	-
Other	-
Video encode	H.264/H.265 (4Kp60), VP9 (8Kp30), NVENC
Video decode	H.264/H.265 (4Kp60), VP9 (8Kp30), NVENC
Camera	-
Storage	-
Power under load	160W
Price	$347.99

3.1.2 Cameras

In this section, we present the cameras used for the development of this work. It was used a thermal infrared camera, and for benchmark reasons, it was chosen to use an RGB camera.

Thermal camera

The thermal infrared camera used in this work is the FLIR ADK by FLIR [32]. FLIR ADK is an outdoor/indoor long-wave infrared (LWIR) that was developed with an emphasis on pedestrian recognition. It accurately identifies people in crowded settings and provides vital data for automated decision-making. See table 3.3 for some of the main features of the FLIR ADK.

Table 3.3: FLIR ADK main features [33].

	FLIR ADK
Sensor technology	Boson™ – Uncooled VOx microbolometer
Spectral range	8-14 microns - Longwave infrared (LWIR)
Arctral range	8-14 microns - Longwave infrared (LWIR)
Array format	640 × 512
Pixel Pitch	12µm
Effective frame rate	Full Frame (30 & 60 Hz selectable), 9 Hz optional
FOV – horizontal	75°
Output format	16-bit TIFF (raw sensor format) or compressed 8-bit
Environmental Protection	IP67
Solar protection	Yes (sun will not damage sensor)
Dimensions (W x H x D)	35 × 40 × 47 mm
Weight	100 g
Operating Temperature	-40°C to +85°C
Power Consumption	1 W (without heater); ∼4 W average and 12 W maximum(with heater)

RGB camera

An RGB camera is a type of camera that captures coloured images by separating frames into red, green, and blue components, as it operates within the visible light spectrum with wavelengths ranging from 0.4 microns to 0.7 microns. This imaging device processes the incoming light and records the intensity of each colour channel, producing a final colour representation of the captured scene.

The RGB camera used in this work is the iPhone 11 Pro (Camera) by Apple [34]. The iPhone 11 Pro is a smartphone developed by Apple and released in September 2019. The smartphone is known for its advanced camera system and fine build quality. See table 3.4 for the iPhone 11 Pro camera's main features.

Table 3.4: Iphone 11 Pro camera main features [34].

	iPhone 11 Pro
Sensor technology	Back-side illuminated (BSI) CMOS sensor
Spectral range	0.4-0.7 microns - Visible light
Arctral range	0.4-0.7 microns - Visible light
Array format	4K, 1080p
Pixel Pitch	-
Effective frame rate	4K at 60 FPS, 1080p at 120 FPS
FOV – horizontal	83°
Output format	HEIF, H.264
Environmental Protection	IP68
Solar protection	No (sun might damage sensor)
Dimensions (W x H x D)	144 × 71.4 × 8.1 mm
Weight	188 g
Operating Temperature	0°C to +35°C
Power Consumption	-

3.2 Systems overview

The research on the challenge of detecting animals in a forestry environment involved two systems: The main system which is composed of the NVIDIA Jetson Nano, the FLIR ADK camera and the SSD MobileNet V2; A benchmark system composed of a standard desktop computer and an NVIDIA RTX 2060 with the latest YOLO v7 detection model.

This section describes both systems' architecture as well as the software required to run both systems and their respective detection models.

Main system

Figure 3.1 illustrates the workflow of the system proposed to solve the challenge of detecting humans and animals in a forestry environment. The system is composed of:

- **FLIR ADK** thermal camera.
- **NVIDIA Jetson Nano** edge device.
- **Single Shot MultiBox Detector (SSD) MobileNet V2** detection model.

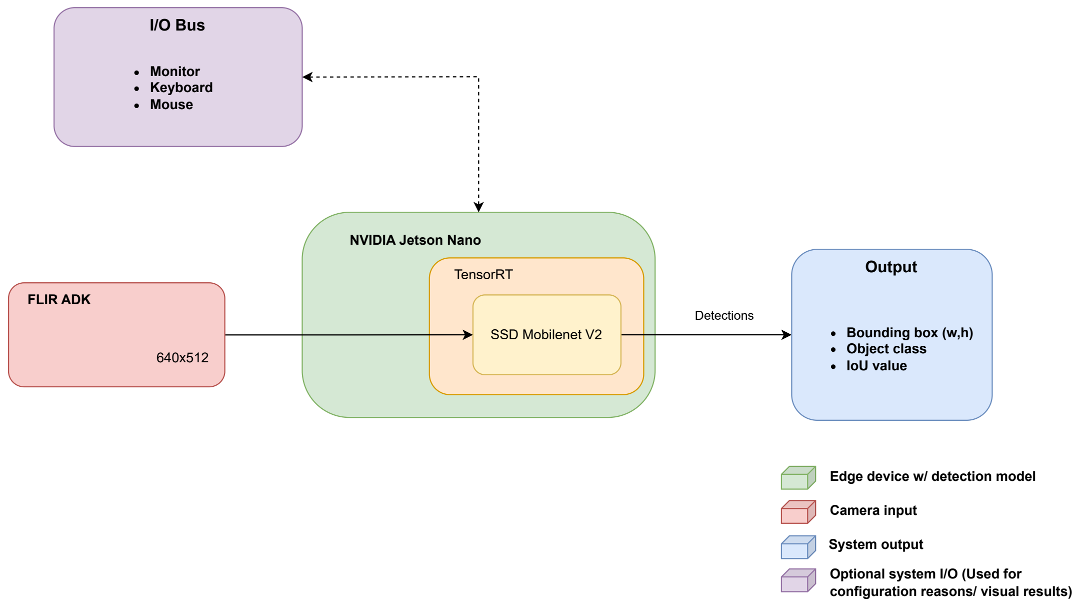


Figure 3.1: Main system overview.

Operation

The FLIR thermal camera is connected to the NVIDIA Jetson Nano as the input image device, feeding the captured images to the SSD Mobilenet V2 detection model. The output can be received through a screen or through a terminal, as seen in figure 3.1.

When operating the NVIDIA Jetson Nano, the device required some prerequisites in terms of software. The steps to achieve the functionality of the system are listed below:

1. Installation of the **NVIDIA Jetpack SDK** [35].

   A Software Development Kit from NVIDIA that provides a comprehensive solution for developing artificial intelligence (AI) and robotics applications on NVIDIA's Jetson platforms. The software includes the framework TensorRT mentioned in the section 2.2.2, taking a role in deploying the detection model SSD MobileNet V2. Moreover, the Jetpack SDK includes a bootloader, Linux kernel and an Ubuntu desktop environment that works as a graphic user interface facilitating the configuration, programming and operation of the system.

2. Clone and installation of the **NVIDIA Jetson Inference toolkit** [36] inside a **Docker container** [37].

   The NVIDIA Jetson inference toolkit is a set of libraries and tools that offer utilities, pre-trained models, sample code, and deployment options for Jetson hardware.

   Docker container is an open-source platform that enables the creation, deployment, and running of applications inside virtualized containers. Containers are isolated units that include the application code, runtime and settings required to run the application, providing a way for developers to easily deploy these applications in any infrastructure.

Following the prerequired software, a python script seen in algorithm 1 was written with the following characteristics:

1. Stream the display output from the FLIR ADK including:
    (a) Bounding box around the object detected.
    (b) Confidence score of the detection (Intersection Over Union (IoU)).
    (c) Class of the object detected.
2. Print in the command line terminal the following details:
    (a) Number of entities in the scene.
    (b) Number of humans in the scene.
    (c) Confidence score of the detection (IoU).
    (d) Coordinates of the centre of the bounding box in the scene.

.

Algorithm 1 Python script pseudocode for the detection.

```
Import jetson.inference
Import jetson.utils

net = jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.5)
camera = jetson.utils.gstCamera(640, 512, "/dev/video0")
display = jetson.utils.glDisplay()

while display.IsOpen() do
    img, width, height = camera.CaptureRGBA()
    detections = net.Detect(img, width, height)
    objcount = length of detections
    if objcount > 0 then
        personcount = 0
        tnow = current date and time
        for i = 0 to objcount - 1 do
            if detections[i].ClassID = 1 then
                conf = 100 * detections[i].Confidence (as integer)
                cx = detections[i].Center[0] (as integer)
                cy = detections[i].Center[1] (as integer)
                Print tnow, " OBJ", i + 1, "/", objcount, " Conf%", conf, " @", cx,
",", cy
            end if
        end for
    end if
end while
```

3.2.1 Benchmark system

Figure 3.2 illustrates the workflow of the system proposed to benchmark the main system (proposed solution) for detecting humans and animals in a forestry environment. The benchmark system is composed of:

- **Desktop** personal computer.
- **NVIDIA RTX 2060** full-size GPU.
- **YOLO v7** detection model.

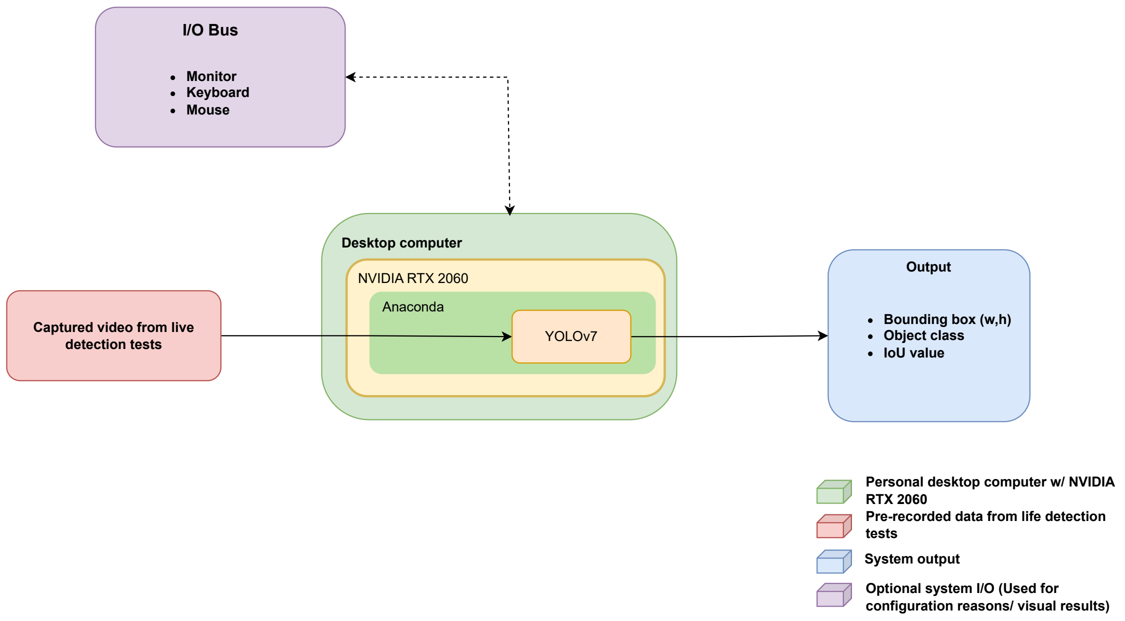

I/O Bus
Monitor
Keyboard
Mouse
Desktop computer
NVIDIA RTX 2060
Anaconda
YOLOv7
Captured video from live detection tests
Output
Bounding box (w,h)
Object class
IoU value
Personal desktop computer w/ NVIDIA RTX 2060
Pre-recorded data from life detection tests
System output
Optional system I/O (Used for configuration reasons/ visual results)

Figure 3.2: Benchmark system overview.

Operation

The present system works as a benchmark system, receiving as input recorded data from the detection testing. The system has installed the Anaconda deep learning framework with the latest YOLOv7 pre-trained model. The output result is used to evaluate the detection ability and compare with the NVIDIA Jetson Nano solution.

When operating the benchmark system with the YOLO v7 detection model, the device required some prerequisites in terms of software. The steps to achieve the functionality of the system are listed below:

1. Installation of the **Anaconda** environment [38].

   Anaconda is a free open-source distribution of Python and R programming languages for data science, scientific computing, and machine learning. It includes a collection of packages and tools, including popular libraries and tools for data analysis and visualization. Anaconda possesses environments that allow managing multiple versions of packages and dependencies and creating isolated environments for specific projects to avoid conflicts.

On this occasion, the Anaconda was needed to create an environment with the purpose of deploying the detection model YOLO v7.

2. Installation of the **Compute Unified Device Architecture (CUDA) toolkit** by NVIDIA [39]

   CUDA is an SDK developed by NVIDIA for programming GPUs for high-performance parallel computing. It provides a platform for developing, optimizing, and deploying GPU-accelerated applications, allowing developers to speed up their computationally intensive applications. The CUDA toolkit allows YOLO v7 to efficiently utilize the GPU NVIDIA RTX 2060 parallel processing power.

3. Clone and installation of the **YOLO version 7** repository [40].

   A repository with the YOLO version 7 that includes pre-trained models, and pre-developed scripts to run the software on custom data or live inference video.

Following the prerequired software, a python script is found in the YOLO v7 repository. This script allows us to process data in video format in the YOLO v7, and the detection model outputs the post-processed video with the following details as video overlay:

- Class of the object detected.
- Confidence score of the detection (Intersection Over Union (IoU)).
- Bounding box around the object detected.

The script allows us to set a custom threshold when performing the inference.

3.3 Systems evaluation

To evaluate the systems in a real-world environment, both were tested in two environments (forest and plain grass) and for 3 different distances to the target, these targets being a human and a human walking a dog. The methodology for the testing of the system is as follows:

- The testing scene has 3 positions with 3 meters between each other, in this way: target standing 3 meters from the camera, 6 meters from the camera and finally 9 meters from the camera as seen in figure 3.3.
- All the positions were measured using a measuring tape and the spots were marked on the ground with a stone so the target knew where to stand when testing.

- The two cameras were positioned standing 50 centimetres from the ground pointing in the direction planned for the test in question.
- After starting the recording the first shot is taken at 3 meters as a person standing, crouching, and standing with a dog, repeating the process at the 6-meter mark and 9-meter mark.

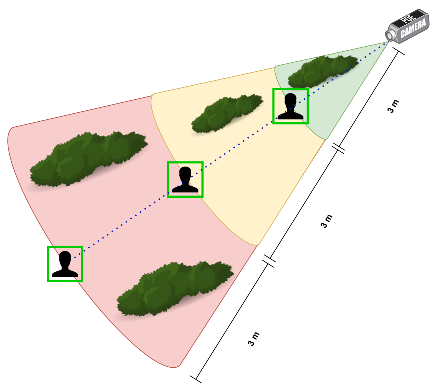


Figure 3.3: Testing scene diagram.

Chapter 4

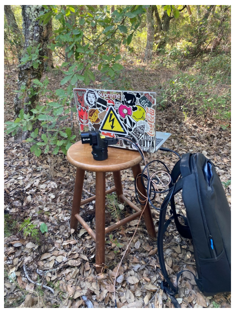
Results

4.1 Results

The developed system was tested in two environments (forest and plain grass) and for 3 different distances to the target, these targets being a human and a human walking a dog. In the forest test, the environment was shady because of the populated terrain with trees; In the case of plain grass, the test was done in a low-light environment in the evening. The forest test was done mainly with a 50% detection threshold and the plain grass test presented a result of a 25% detection threshold aswell as a 50% detection threshold detection result.

4.1.1 Forest

This test was executed in Souselas, Coimbra (40.305529, -8.428913). The recording direction on the forest environment was 150º Southeast in the shade, at 4 PM on 2 of October of 2022. The following two images (figure 4.13) show the testing setup for a human walking a dog.


(a) Recording setup.

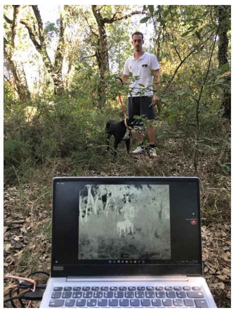
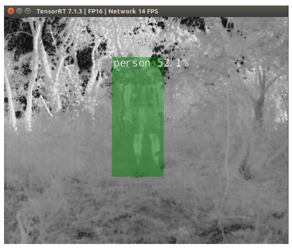

(b) Human walking an animal setup and output.

Figure 4.1: System being tested in the forest.

3 Meter

Next we present the detection for the 3-meter distance target from the camera results. Being the closest area to the robot operational area, this distance is the most important to be detected so we can prevent accidents efficiently.



(a) SSD Mobilenet V2, with a 52.1% detection confidence.

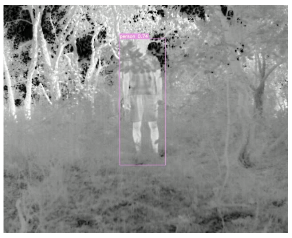
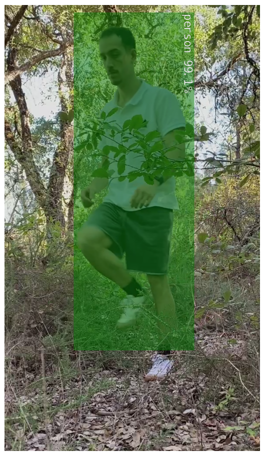

(b) YOLOv7, with a 74.0% detection confidence.

Figure 4.2: Standing human at 3 meters and 50% threshold.



(a) SSD Mobilenet V2, with a 99.1% detection confidence.

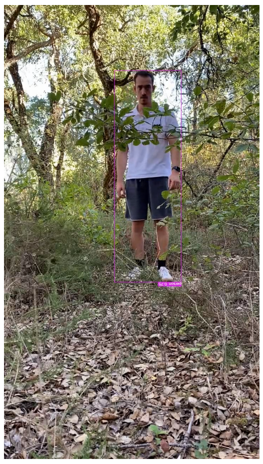
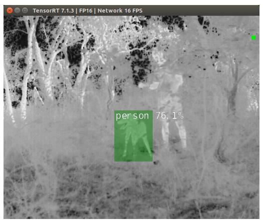
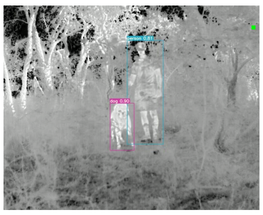

(b) YOLOv7, with a 75.0% detection confidence.

Figure 4.3: RGB image of a standing human at 3 meters and 50% threshold.

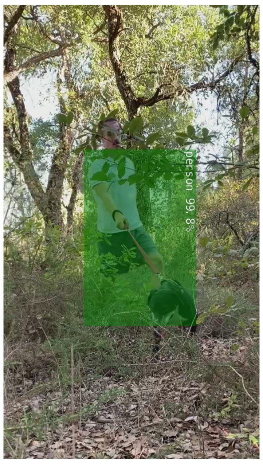


(a) SSD MBnet V2, with a 76.1% dog detection.

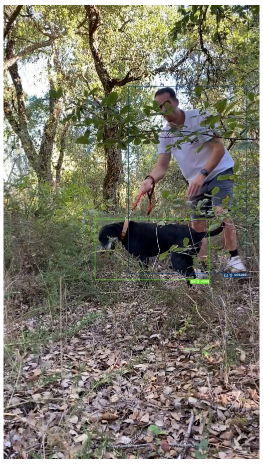


(b) YOLOv7, with 81.0% detection confidence for the human and 90.0% for the dog.

Figure 4.4: Human walking a dog at 3 meters and 50% threshold.



(a) SSD Mobilenet V2, with 99.8% human detection confidence.

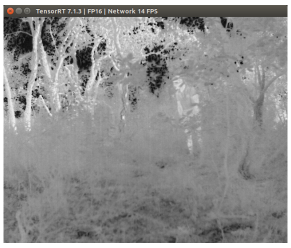


(b) YOLOv7, with 73.0% detecion confidence for the human and 81.0% for the dog.

Figure 4.5: RGB image of a Human walking a dog at 3 meters and 50% threshold.

6 Meters

Next we present the results referring to the 6-meter distance from the target.


(a) SSD Mobilenet V2, with a miss detection.

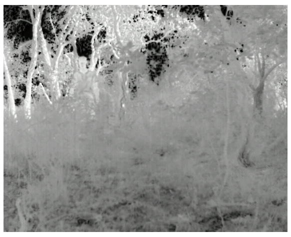
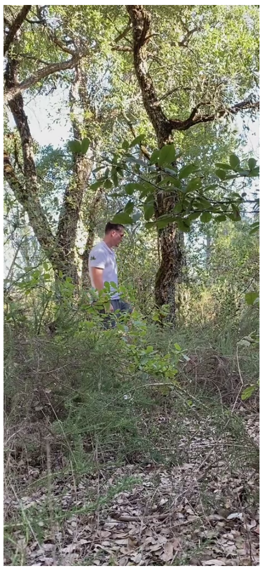

(b) YOLOv7, with a miss detection.

Figure 4.6: Standing human at 6 meters and 50% threshold.


(a) SSD Mobilenet V2, with a miss detection.

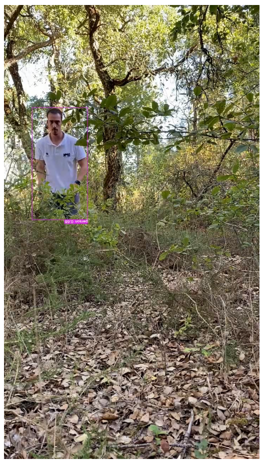
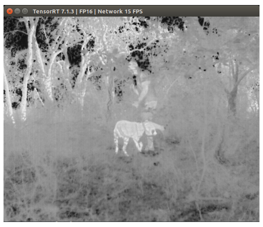

(b) YOLOv7, with a 68.0% of detection confidence.

Figure 4.7: RGB image of a standing human at 6 meters and 50% threshold.


(a) SSD Mobilenet V2, with a miss detection.

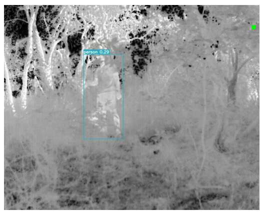
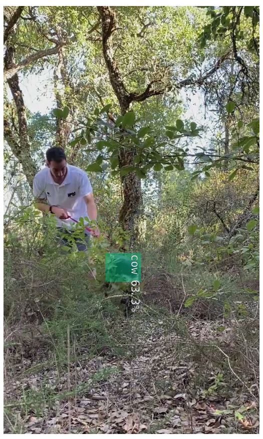
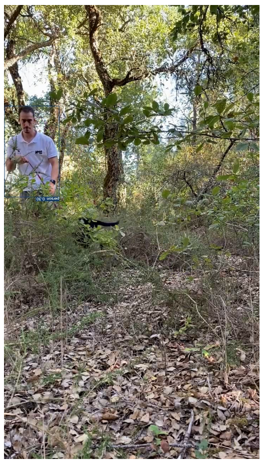

(b) YOLOv7, with a 29.0% detection confidence

Figure 4.8: Human walking a dog at 6 meters and 50% threshold for the the SSD Mobilenet V2 and 25% threshold for the YOLOv7.



(a) SSD Mobilenet V2, with a 63.3% detection confidence for the dog.

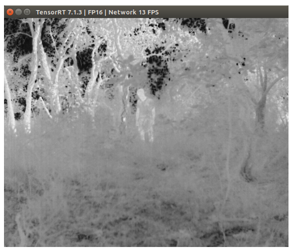


(b) YOLOv7, with a 30.0% detection confidence for the human.

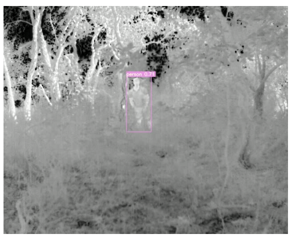
Figure 4.9: Human walking a dog at 6 meters and 50% threshold for the the SSD Mobilenet V2 and 25% threshold for the YOLOv7.

9 Meters

Next we present the results referring to the 9-meter distance from the target.

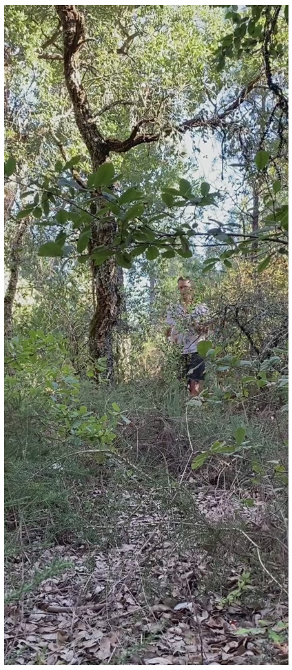


(a) SSD Mobilenet V2 miss detection.

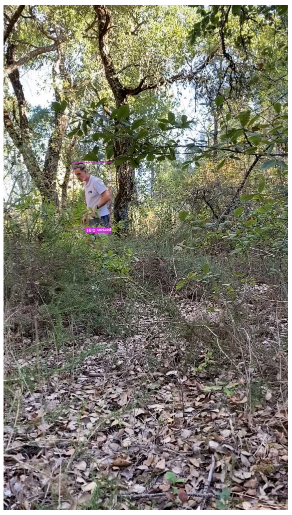


(b) YOLOv7, with a 71.0% of detection confidence for the human.

Figure 4.10: Standing human at 9 meters and 50% threshold.

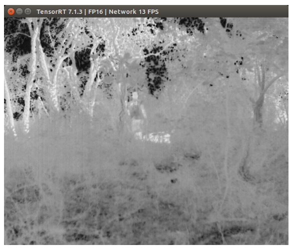


(a) SSD Mobilenet V2, miss detection.

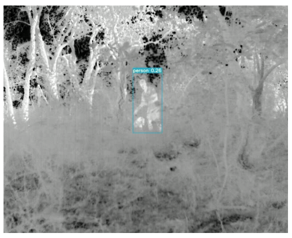


(b) YOLOv7, with a 91.0% detection confidence.

Figure 4.11: RGB image of a standing human at 9 meters and 50% threshold.



(a) SSD Mobilenet V2 with a miss detection.



(b) YOLOv7, with a 26.0% detection confidence for the human.

Figure 4.12: Human walking a dog at 9 meters and 50% threshold for the the SSD Mobilenet V2 and 25% threshold for the YOLOv7.

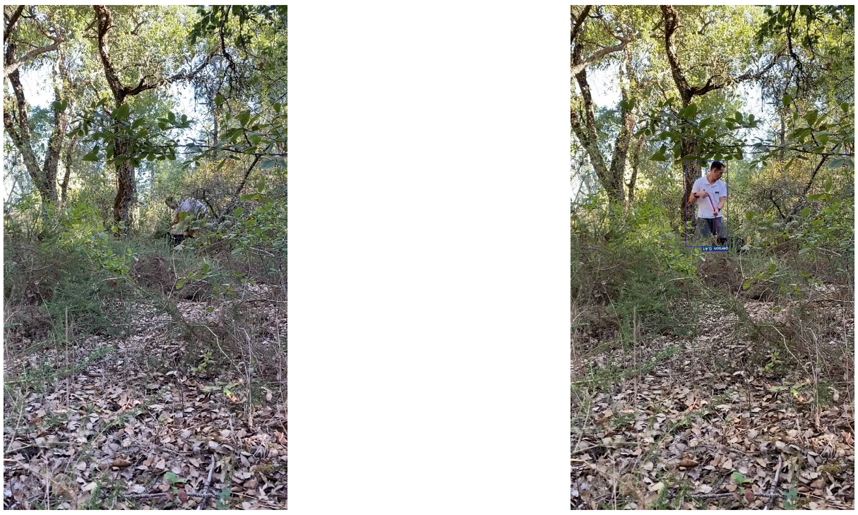

(a) SSD Mobilenet V2, with a miss detection.(b) YOLOv7, 41.0% detection confidence for the human.

Figure 4.13: RGB image of a human walking a dog at 9 meters and 50% threshold for the the SSD Mobilenet V2 and 25% threshold for the YOLOv7.

Table 4.1: SSD Mobilenet V2 results in forestry environment.

	Conditions							
Distance from the camera	**Thermal**	**RGB**	**Dog**	**Standing Human**	**Threshold**	**Human IoU**	**Dog IoU**	**Miss-detection**
3m	x			x		52,3%		
		x		x		99.1%		
	x		x	x			76.1%	
		x	x	x		99.8%		
6m	x			x				x
		x		x	50%			x
	x		x	x			63.3%	x
		x	x	x				x
9m	x			x				x
		x		x				x
	x		x	x				x
		x	x	x				x
Average						83,73%	69,70%	

As shown in table 4.1, the SSD Mobilenet V2 detection model showed good results in at 3 meters from the camera. For lengths longer than 3 meters the detection model had some difficulties in achieving the detections. The average intersection over union (IoU) for the human and dog were 83,73% and 69,70% respectively.

Table 4.2: YOLOv7 results in forestry environment.

	Conditions							
Distance from the camera	**Thermal**	**RGB**	**Dog**	**Standing Human**	**Threshold**	**Human IoU**	**Dog IoU**	**Miss-detection**
3m	x			x	50%	74,00%		
		x		x	50%	75,00%		
	x		x	x	50%	81,00%	90,00%	
		x	x	x	50%	73,00%	81,00%	
6m	x			x	50%	68,00%		
		x		x	50%			x
	x		x	x	25%	29,00%	63.3%	
		x	x	x	25%	30,00%		
9m	x			x	50%	71,00%		
		x		x	50%	91,00%		
	x		x	x	50%	26,00%		
		x	x	x	25%	41,00%		
Average						59,91%	85,50%	

As shown in table 4.2, the YOLOv7 model showed better results than SSD Mobilenet V2 which was to expect because of the different complexity of the models. The detection model YOLOv7 only showed one miss-detection and showed a lower average Human IoU (59,91%) than SSD Mobilenet V2 but a higher dog detection IoU with 85,50%.

4.1.2 Plain grass scenario

The garden of a house (40.304544, -8.427840), which was turned directly 192 degrees South at 5 PM on October 30, 2022, served as the background and testing site for the local installation of the plain grass arrangement. In this instance, we performed live inference detection utilizing the FLIR camera placed at 50 centimetres from the ground attached to the Jetson Nano and RGB video recording using the iPhone 11 camera. Below we can find a photo of the setup achieved (Figure 4.14).

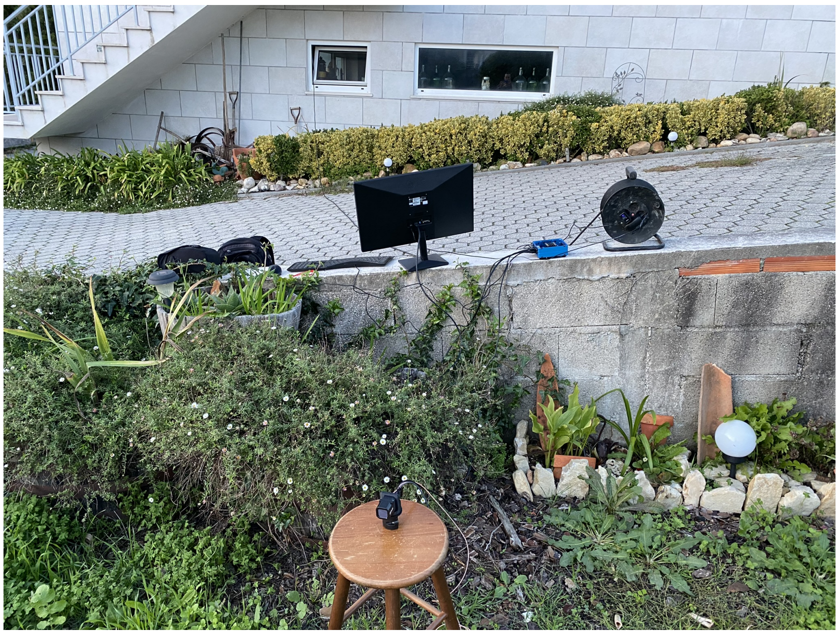
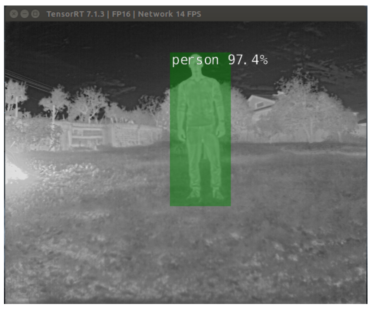


Figure 4.14: Plain grass live inference setup.

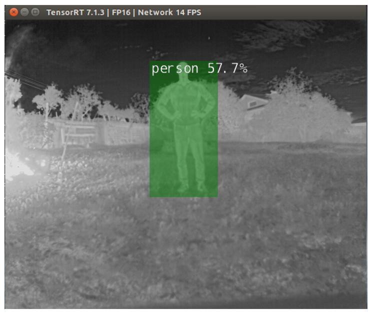
We performed live detection this time, and it was decided to change the threshold of confidence required to be considered a detection. The threshold was varied between 25% and 50%, and the results are shown below.

3 Meters

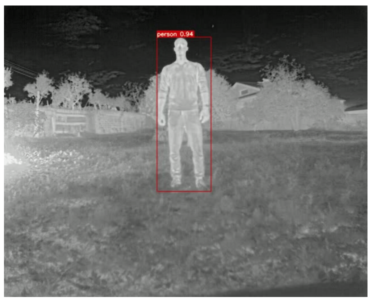
Next are shown the detection results at the 3-meter distance between the camera and the target.



(a) SSD Mobilenet V2, with a 97.4% of confidence and a 25% threshold.

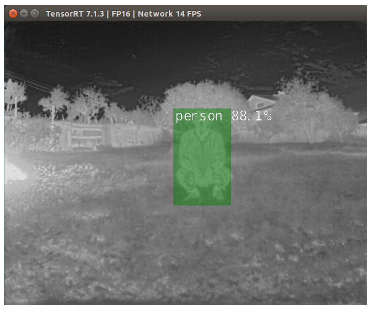


(b) SSD Mobilenet V2, with a 57.4% of confidence and a 50% threshold.

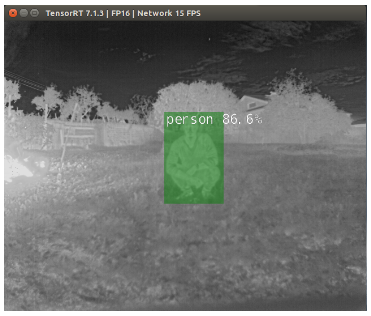


(c) YOLOv7, with a 94.4% of confidence and a 50% threshold.

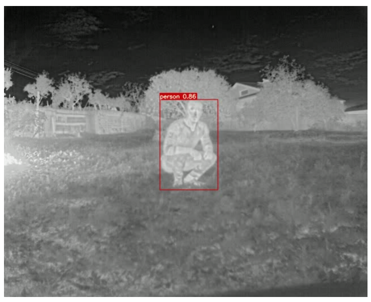
Figure 4.15: Standing human.



(a) SSD Mobilenet V2, with a 88.1% of confidence and a 25% threshold.

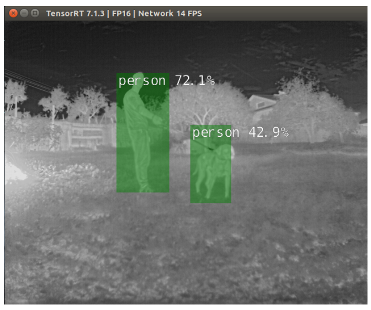


(b) SSD Mobilenet V2, with a 86.6% of confidence and a 50% threshold.

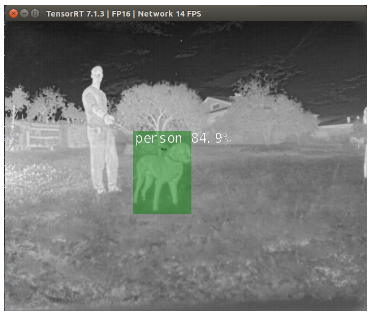


(c) YOLOv7, with a 86.0% of confidence and a 50% threshold.

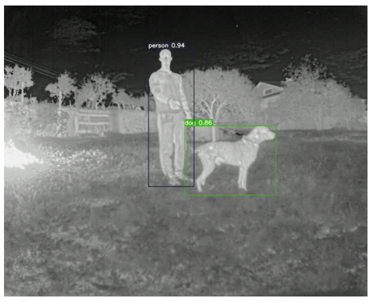
Figure 4.16: Crouching human.



(a) SSD Mobilenet V2, with a 72.1% degree for the human and 42.9% for the dog featuring 25% threshold.



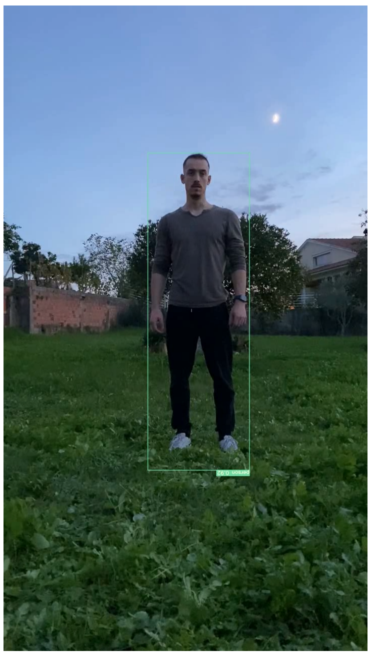
(b) SSD Mobilenet V2, with a miss detection for the human and 84.9% for the dog featuring 50% threshold.

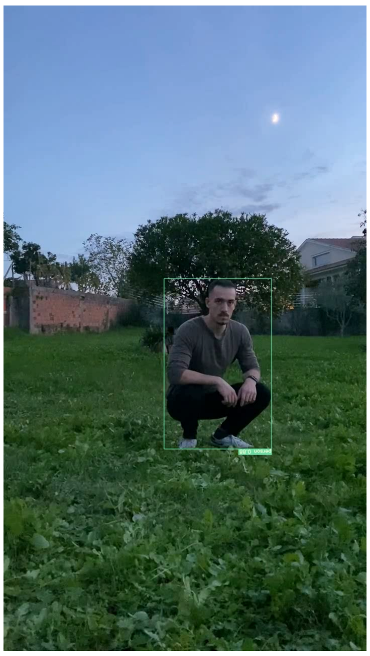


(c) YOLOv7, with a 94.0% degree for the human and 86.0% for the dog featuring 50% threshold.

Figure 4.17: Human and dog.

To serve as a term of comparison, we show the results of the detection in RGB using the YOLOv7 network.


(a) YOLOv7, with a 92.0% of confidence.


(b) YOLOv7, with a 88.0% of confidence.

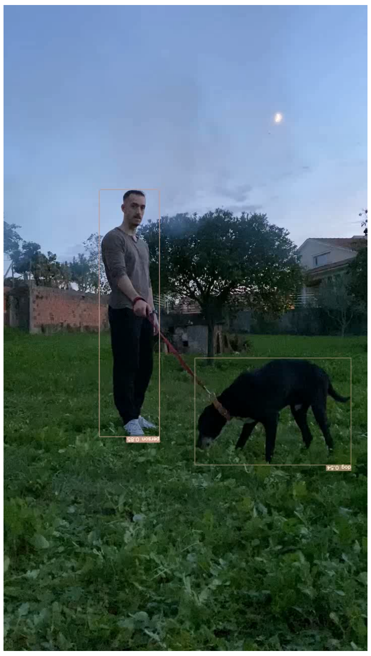
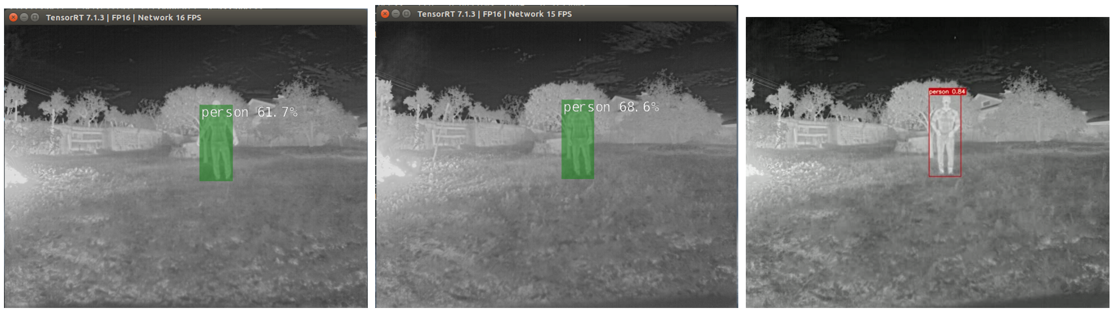

(c) YOLOv7, with a 85.0% degree for the human and 54.0% for the dog.

Figure 4.18: Human and dog.

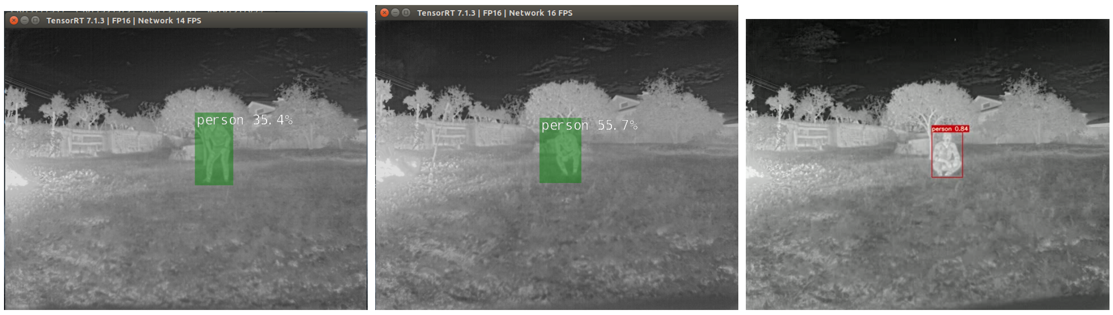
6 Meters

Next are shown the detection results at the 6-meter distance between the camera and the target.


TensorRT 7.1.3 | FP16 | Network 16 FPS
person 61.7%
TensorRT 7.1.3 | FP16 | Network 15 FPS
person 68.6%
person 0.84

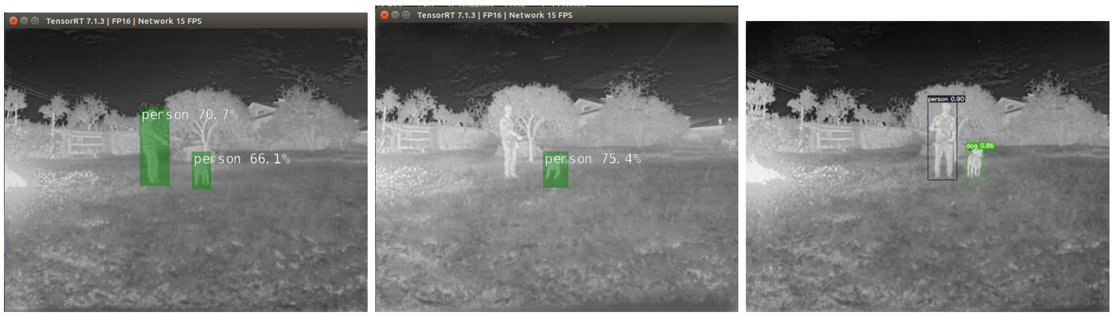
(a) SSD Mobilenet V2, with a 61.7% of confidence for 25% threshold.
(b) SSD Mobilenet V2, with a 68.6% of confidence for 50% threshold.
(c) YOLOv7, with a 84.0% of confidence for 50.0% threshold.

Figure 4.19: Standing human detection results in plain grass.


TensorRT 7.1.3 | FP16 | Network 14 FPS
person 35.4%
TensorRT 7.1.3 | FP16 | Network 16 FPS
person 55.7%
person 0.84

(a) SSD Mobilenet V2, with a 35.4% of confidence for 25% threshold.
(b) SSD Mobilenet V2, with a 55.7% of confidence for 50% threshold.
(c) YOLOv7, with a 84.0% of confidence for 50.0% threshold.

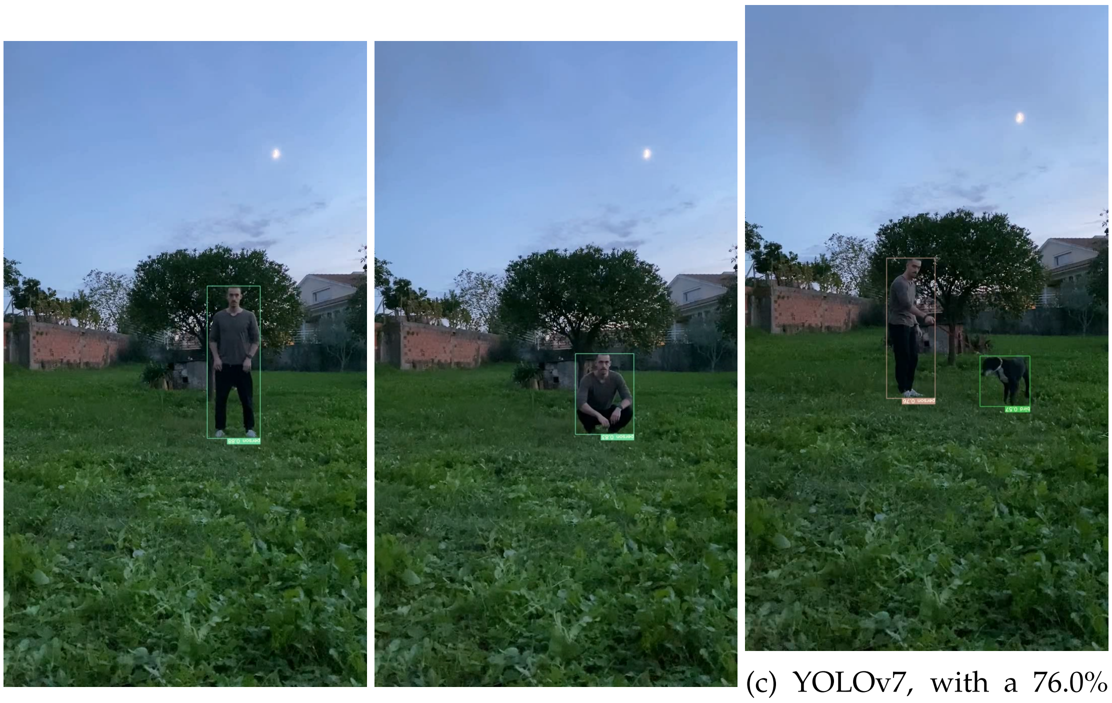
Figure 4.20: Crouching human detection results in plain grass.


TensorRT 7.1.3 | FP16 | Network 15 FPS
person 70.7%
person 66.1%
TensorRT 7.1.3 | FP16 | Network 15 FPS
person 75.4%
person 0.90
dog 0.86

(a) SSD Mobilenet V2, with a 70.7% degree for the human and 66.1% for the dog featuring 25% threshold.
(b) SSD Mobilenet V2, with miss detection for the human and 75.4% for the dog featuring 50% threshold.
(c) YOLOv7, with a 90.0% degree for the human and 86.0% for the dog featuring 50% threshold.

Figure 4.21: Human and dog detection results in plain grass.

Again, as a term of comparison, we show the results of the detection in RGB using the YOLOv7 network.



(a) YOLOv7, with a 88.0% of confidence.

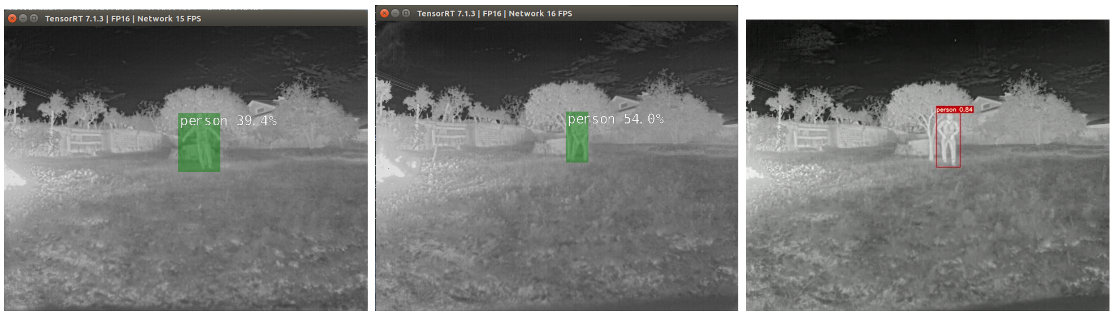
(b) YOLOv7, with a 83.0% of confidence.

(c) YOLOv7, with a 76.0% degree for the human and 57.0% for the dog.

Figure 4.22: YOLOv7 detection at 6 meters.

9 meters

Next are shown the detection results at the 9-meter distance between the camera and the target.

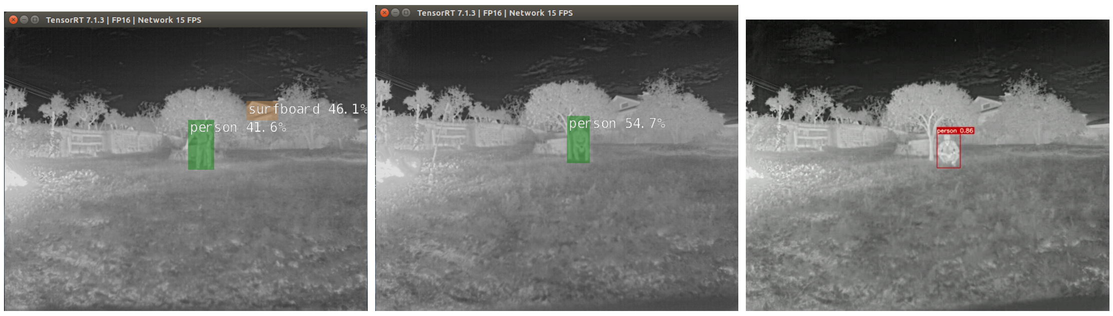

TensorRT 7.1.3 | FP16 | Network 15 FPS
person 39.4%
TensorRT 7.1.3 | FP16 | Network 16 FPS
person 54.0%
person 0.84

(a) SSD Mobilenet V2, with a 39.4% of confidence for 25% threshold.

(b) SSD Mobilenet V2, with a 54.0% of confidence for 50% threshold.

(c) YOLOv7, with a 84.0% of confidence for 50% threshold.

Figure 4.23: Standing human detection results in plain grass.

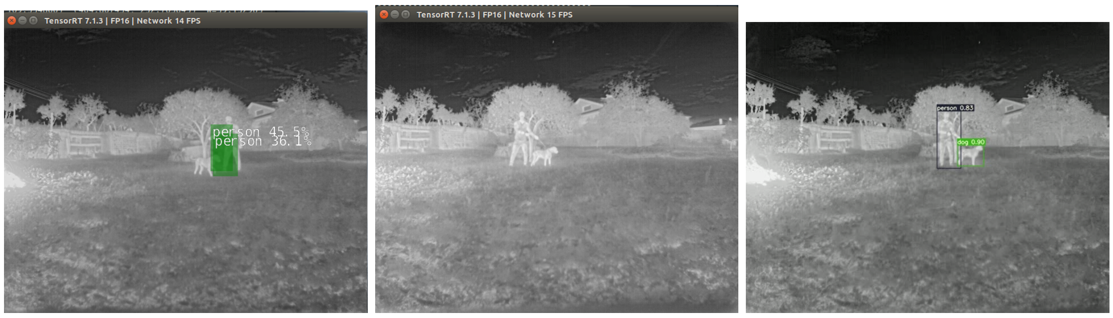

TensorRT 7.1.3 | FP16 | Network 15 FPS
surfboard 46.1%
person 41.6%
TensorRT 7.1.3 | FP16 | Network 15 FPS
person 54.7%
person 0.86

(a) SSD Mobilenet V2, with a 41.6% of confidence for 25% threshold.

(b) SSD Mobilenet V2, with a 54.7% of confidence for 50% threshold.

(c) YOLOv7, with a 86.0% of confidence for 50% threshold.

Figure 4.24: Crouching human detection results in plain grass.


TensorRT 7.1.3 | FP16 | Network 14 FPS
person 45.5%
person 36.1%
TensorRT 7.1.3 | FP16 | Network 15 FPS
person 0.83
dog 0.90

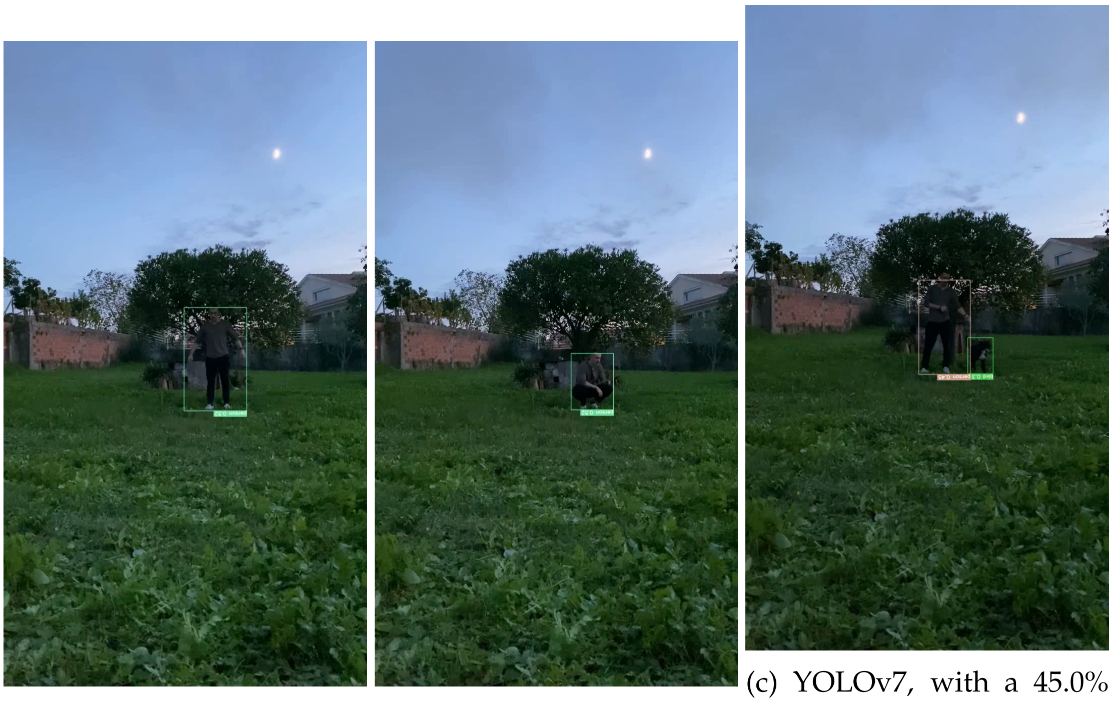
(a) SSD Mobilenet V2, with a 45.5% degree for the human and 36.1% for the dog featuring 25% threshold.

(b) SSD Mobilenet V2, with a miss detection on both objects featuring 25% threshold.

(c) YOLOv7, with a 83.0% degree for the human and 90.0% for the dog featuring 50% threshold.

Figure 4.25: Human and dog detection results in plain grass.

Finally, we show the results of the detection in RGB using the YOLOv7 network at the 9-meter from the target.



(a) YOLOv7, with a 52.0% of confidence.

(b) YOLOv7, with a 52.0% of confidence.

(c) YOLOv7, with a 45.0% for the human and 35.0% for the dog.

Figure 4.26: YOLOv7 detection at 9 meters.

Table 4.3: SSD Mobilenet V2 results in plain grass environment.

				Threshold					
	Conditions			25%		50%		Miss-detection	
Distance from the camera	**Crouching**	**Standing Human**	**W/ dog**	**Human IoU**	**Dog IoU**	**Human IoU**	**Dog IoU**	25%	50%
3m		x		97,40%		57,40%			
	x			88,10%		86,60%			
			x	72,10%	42,90%		84,90%		
6m		x		61,70%		68,60%			
	x			35,40%		55,70%			
			x	70,70%	66,10%		75,40%		
9m		x		39,40%		54,00%			
	x			41,60%		54,70%			
			x	45,50%	36,10%				x
Average				61,32%	48,37%	62,83%	80,15%		

From table 4.3 it can be seen that the SSD Mobilenet V2 detection model showed better results in plain grass than in the forest, achieving only one miss detection being the 9-meter mark with the dog at 50% threshold to be considered a detection. Nevertheless, the model showed 61,32% human IoU at 25% threshold and 62,83% at 50% threshold, in the dog case it showed an average of 62,83% at 25% threshold and 80,15% at 50% threshold.

Table 4.4: YOLOv7 results in plain grass environment.

	Conditions						
Distance from the camera	**Crouching**	**Standing Human**	**W/ dog**	**Threshold**	**Human IoU**	**Dog IoU**	**Miss-detection**
3m		x			94,40%		
	x				86,00%		
			x		94,00%	86,00%	
6m		x			84,00%		
	x			50%	84,00%		
			x		90,00%	86,00%	
9m		x			84,00%		
	x				86,00%		
			x		83,00%	90,00%	
Average					87,27%	87,33%	

The YOLOv7 thermal camera results in plain grass showed to be very good as well, averaging 87,27% for human IoU and 87,33% for the dog IoU, with no miss detections in this case.

Table 4.5: YOLOv7 RGB results in plain grass environment.

	Conditions						
Distance from the camera	**Crouching**	**Standing Human**	**W/ dog**	**Threshold**	**Human IoU**	**Dog IoU**	**Miss-detection**
3m		x		50%	92,00%		
	x			50%	88,00%		
			x	50%	85,00%	54,00%	
6m		x		50%	88,00%		
	x			50%	83,00%		
			x	50%	76,00%	57,00%	
9m		x		50%	52,00%		
	x			50%	52,00%		
			x	25%	45,00%	35,00%	
Average					73,44%	48,67%	

YOLOv7 RGB camera results were good but not as good as the thermal camera results which prove how good the thermal camera can be when used for live-detection purposes. Averaging 73,44% for the human IoU and 45,67% for the dog IoU, with no miss-detections.

Chapter 5

Conclusion

The aim to develop a detection system for human and animal presence was fulfilled. The testing of the system in a real-world environment was successfully met accordingly. This chapter compares the results and goals met to the requirements presented in chapter 1.

5.1 Requirement analysis

1. **Successful detection at a distance of 3 meters from the target**

   The main system proved to efficiently detect the target at 3 meters from the camera in both environments (forestry and plain grass). Showing better results in plain grass because of the lack of bushes and trees.

2. **Different testing environments (forestry, plain grass)**

   The system was tested in the forest environment and the plain grass environment. Providing results for better comparison between environments as well as serving as a reference for future studies.

3. **Human and dog detection in different poses.**

   In this work, the effort of using an animal as a target was made possible, providing two common targets for accidents using forest cleaning robots.

4. **Detection with different light conditions (direct sunlight, shade).**

   The tests of the system were made on two different days with weeks between each other, providing a sunlight and shady environment in the forest environment and a reduced lighting environment in the plain grass.

5. **Detection with different distances from the target to the camera.**

   Detection with different distances from the camera to the target was presented in the results, providing a better understanding of how far the system can detect consistently.

6. **Live inference and post-captured video inference as a benchmark.**

   The forest detection test was made using post-captured video and the plain grass test was made using live inference for the SSD MobileNet V2, and the post-captured video for the YOLO v7.

5.2 Future work

This work has been developed towards a target application, and for this application in particular, some future adjustments are suggested. The performance of the detector might be limited due to the quality of the currently available data, and thus a quick fix will be to gather more diversified data from different environments, as well as execute the respective detection tests.

To improve the detection quality, the use of transfer learning is suggested. Retraining the SSD MobileNet V2 with thermal images can prove to be a strong improvement to the system.

It would be interesting to further investigate how the system could be combined with other sensors, such as sound or LIDAR.

The implementation of this work on the SafeForest robot and executing live detection tests using the system mounted on the robot is a further suggestion.

References

[1] S.-M.-A. J, D. T, B. R, *et al.*, "Forest fires in europe, middle east and north africa 2020," no. KJ-NA-30862-EN-N (online),KJ-NA-30862-EN-C (print), 2021, ISSN: 1831-9424 (online),1018-5593 (print). DOI: 10.2760/216446(online) ,10.2760/059331(print).

[2] C. A. Felsemburgh, "Empreendedorismo e inovação na engenharia florestal 3," Ponta Grossa – Paraná – Brasil: Atena Editora, 2021, ISBN: 978-65-5706-963-9. DOI: 10.22533/at.ed.639211404.

[3] D. Topolsky, I. Topolskaya, I. Plaksina, *et al.*, "Development of a mobile robot for mine exploration," *Processes*, vol. 10, no. 5, 2022, ISSN: 2227-9717. DOI: 10.3390/pr10050865. [Online]. Available: https://www.mdpi.com/2227-9717/10/5/865.

[4] SAFEFOREST. "Semi-autonomous robotic system for forest cleaning and fire prevention." (2023), [Online]. Available: %5Curl%7Bhttps://safeforest.ingeniarius.pt/#overview%7D (visited on 03/18/2022).

[5] G. O. Strawn, "Masterminds of deep learning," *IT Professional*, vol. 24, no. 3, pp. 13–15, 2022. DOI: 10.1109/MITP.2022.3172838.

[6] T. Hoeser and C. Kuenzer, "Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends," *Remote Sensing*, vol. 12, no. 10, 2020, ISSN: 2072-4292. DOI: 10.3390/rs12101667. [Online]. Available: https://www.mdpi.com/2072-4292/12/10/1667.

[7] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *CoRR*, vol. abs/1901.06032, 2019. arXiv: 1901.06032. [Online]. Available: http://arxiv.org/abs/1901.06032.

[8] A. A. Süzen, B. Duman, and B. Şen, "Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn," *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1–5, 2020.

[9] Y. Wang, Y. Zhang, and K. Chen, "Real-time monitoring of ship targets at sea based on jetson nano," in *2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2022, pp. 166–169. DOI: 10.1109/ICAICA54878.2022.9844502.

[10] H. T. Minh, L. Mai, and T. V. Minh, "Performance evaluation of deep learning models on embedded platform for edge ai-based real time traffic tracking and detecting applications," in *2021 15th International Conference on Advanced Computing and Applications (ACOMP)*, 2021, pp. 128–135. DOI: `10.1109/ACOMP53746.2021.00024`.

[11] R. Nikhil, B. Anisha, and R. Kumar P., "Real-time monitoring of agricultural land with crop prediction and animal intrusion prevention using internet of things and machine learning at edge," in *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2020, pp. 1–6. DOI: `10.1109/CONECCT50063.2020.9198508`.

[12] C. Rau. "NVIDIA tensorrt documentation." (2022), [Online]. Available: `https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html#overview` (visited on 01/03/2023).

[13] M. Abadi, P. Barham, J. Chen, *et al.*, "{Tensorflow}: A system for {large-scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[14] W. Liu, D. Anguelov, D. Erhan, *et al.*, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*, Springer International Publishing, 2016, pp. 21–37. DOI: `10.1007/978-3-319-46448-0_2`. [Online]. Available: `https://doi.org/10.1007%2F978-3-319-46448-0_2`.

[15] L. Weng, "Object detection part 4: Fast detection models," *lilianweng.github.io*, 2018. [Online]. Available: `https://lilianweng.github.io/posts/2018-12-27-object-recognition-part-4/`.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018. DOI: `10.48550/ARXIV.1801.04381`. [Online]. Available: `https://arxiv.org/abs/1801.04381`.

[17] A. Krizhevsky, "Convolutional deep belief networks on cifar-10," May 2012.

[18] A. Srivastava, A. Dalvi, C. D. Britto, H. Rai, and K. Shelke, "Explicit content detection using faster r-cnn and ssd mobilenet v2," 2020.

[19] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, *Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, 2022. DOI: `10.48550/ARXIV.2207.02696`. [Online]. Available: `https://arxiv.org/abs/2207.02696`.

[20] Y. Wang, H. Wang, and Z. Xin, "Efficient detection model of steel strip surface defects based on yolo-v7," *IEEE Access*, vol. 10, pp. 133 936–133 944, 2022. DOI: `10.1109/ACCESS.2022.3230894`.

[21] E. S. T. K. Reddy and R. V, "Pothole detection using cnn and yolo v7 algorithm," in *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, 2022, pp. 1255–1260. DOI: `10.1109/ICECA55336.2022.10009324`.

[22] 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. 2015. DOI: `10.5121/ijdkp.2015.5201`.

[23] 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," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.

[24] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, *Microsoft coco: Common objects in context*, 2014. DOI: `10.48550/ARXIV.1405.0312`. [Online]. Available: `https://arxiv.org/abs/1405.0312`.

[25] Microsoft. "Common objects in context." (2023), [Online]. Available: `https://cocodataset.org/#detection-leaderboard` (visited on 01/30/2023).

[26] G. Brazil, X. Yin, and X. Liu, *Illuminating pedestrians via simultaneous detection amp; segmentation*, 2017. DOI: `10.48550/ARXIV.1706.08564`. [Online]. Available: `https://arxiv.org/abs/1706.08564`.

[27] S. Zhang, J. Yang, and B. Schiele, "Occluded pedestrian detection through guided attention in cnns," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6995–7003. DOI: `10.1109/CVPR.2018.00731`.

[28] F. Altay and S. Velipasalar, "The use of thermal cameras for pedestrian detection," *IEEE Sensors Journal*, vol. 22, no. 12, pp. 11 489–11 498, 2022. DOI: `10.1109/JSEN.2022.3172386`.

[29] M. Krišto, M. Ivasic-Kos, and M. Pobar, "Thermal object detection in difficult weather conditions using yolo," *IEEE Access*, vol. 8, pp. 125 459–125 476, 2020. DOI: `10.1109/ACCESS.2020.3007481`.

[30] D. Franklin. "Jetson nano brings ai computing to everyone." (2019), [Online]. Available: `https://developer.nvidia.com/blog/jetson-nano-ai-computing/` (visited on 03/18/2019).

[31] NVIDIA. "Nvidia turing gpu architecture." (2018), [Online]. Available: `https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf`.

[32] F. ADK™. "Flir adk thermal vision automotive development kit | teledyne flir." (2022), [Online]. Available: `https://www.flir.eu/products/adk/` (visited on 03/18/2019).

[33] Hexagon. "Flir adk specifications." (2022), [Online]. Available: `https://autonomoustuff.com/products/flir-adk`.

[34] Apple. "Iphone 11 pro - technical specifications." (2019), [Online]. Available: `https://support.apple.com/kb/SP805?viewlocale=en_US&locale=pt_PT` (visited on 01/18/2023).

[35] NVIDIA. "Nvidia jetpack documentation." (2023), [Online]. Available: `https://docs.nvidia.com/jetson/jetpack/introduction/index.html`.

[36] D. Franklin. "Jetson nano brings ai computing to everyone." (2019), [Online]. Available: `https://developer.nvidia.com/blog/jetson-nano-ai-computing/` (visited on 03/18/2019).

[37] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[38] C. analytics. "Data science platform." (2012), [Online]. Available: `https://www.anaconda.com/` (visited on 10/18/2022).

[39] NVIDIA. "Nvidia cuda toolkit." (2007), [Online]. Available: `https://docs.nvidia.com/jetson/jetpack/introduction/index.html`.

[40] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.