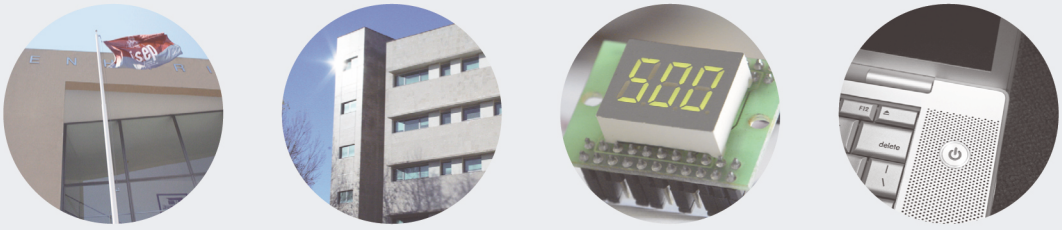
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Deep Learning Approach for UAV Visual Electrical Assets Inspection

JOEL ADÃO PACHECO BARBOSA
novembro de 2020

POLITÉCNICO
DO PORTO

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Deep Learning Approach for UAV Visual Electrical Assets Inspection

JOEL ADÃO PACHECO BARBOSA
Novembro de 2020

POLITÉCNICO
DO PORTO

Deep Learning Approach for UAV Visual Electrical Assets Inspection

Joel Adão Pacheco Barbosa
Nº 1131197

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Sistemas Autónomos
Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

2020

Dissertação, para satisfação parcial dos requisitos do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Joel Adão Pacheco Barbosa

Nº 1131197

Orientador: André Miguel Pinheiro Dias

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização de Sistemas Autónomos

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

November 27, 2020

Abstract

The growth in the electrical demand by most countries around the world requires bigger and more complex energy systems, which leads to the requirement of having even more monitoring, inspection and maintenance of those systems. To respond to this need, inspection methods based on Unmanned Aerial Vehicles (UAV) have emerged which, when equipped with the appropriate sensors, allow a greater reduction of costs and risks and an increase in efficiency and effectiveness compared to traditional methods, such as inspection with foot patrols or helicopter-assisted. To make the inspection process more autonomous and reliable, most of the methods apply visual detection methods that use highly complex Deep Learning based algorithms and that require a very large computational power.

This dissertation intends to present a system for inspection of electrical assets, able to be integrated onboard the UAV, based on Deep Learning, which allows to collect visual samples grouped and aggregated for each electrical asset detected. To this end, a perception system capable of detecting electrical insulators or structures, such as poles or transmission towers, was developed, using the Movidius Neural Compute Stick portable platform that is capable of processing lightweight object detection Convolutional Neural Networks, allowing a modular, low-cost system that meets real-time processing requirements. In addition to this perception system, an electrical asset monitoring system has been implemented that allows tracking and mapping each asset throughout the inspection process, based on the previous system's detections and a UAV navigation system. Finally, an autonomous inspection system is proposed, which consists of a set of trajectories that allow an efficient application of the monitoring system along a power line, through the mapping of structures and the gathering of insulator samples around that structure.

Keywords: Electrical Assets Inspection, Deep Learning, Object Detection,

Multi-Object Tracking, UAV

Resumo

O grande crescimento da exigência elétrica pela maioria dos países por todo o mundo, requer que os sistemas de energia sejam maiores e mais complexos, o que conduz a uma maior necessidade de monitorização, inspeção e manutenção desses sistemas. Para responder a esta necessidade, surgiram métodos de inspeção baseados em Veículos Aéreos Não Tripulados (VANT) que, quando equipados com os sensores apropriados, permitem uma maior redução de custos e riscos e um grande aumento de eficiência e eficácia em comparação com os métodos tradicionais, como a inspeção com patrulhas pedonais ou assistida por helicóptero. Para tornar processo de inspeção mais autónomo e confiável, a maioria dos métodos realiza método de deteção visuais que utilizam algoritmos baseados em Deep Learning de elevada complexidade e que requerem um poder computacional muito grande.

Nesta dissertação pretende-se apresentar um sistema de inspeção de ativos elétricos, para integração em VANTs, baseado em Apredizagem Profunda, que permite recolher amostras visuais agrupadas e agregadas por cada ativo elétrico detetado. Para tal foi desenvolvido um sistema de perceção capaz de detetar isoladores elétricos ou estruturas, como postes ou torres de transmissão, com recurso à plataforma portátil Movidius Neural Compute Stick que é capaz de processar Redes Neuronais Convolucionais leves de deteção de objetos, permitindo assim um sistema modular, de baixo custo e que cumpre requisitos de processamente em tempo real. Para além deste sistema de perceção, foi implementado um sistema de monitorização de ativos elétricos que permite seguir e mapear cada ativo ao longo do processo de inspeção, com base nas deteções do sistema anterior e no sistema de navegação do VANT. Por fim, é proposto um sistema de inspeção autónomo que consiste num conjunto de trajetórias que permitem aplicar o sistema de monitorização de ativos elétricos ao longo de uma linha elétrica, através do mapeamento de estruturas e na recolha de amostras de isoladores em torno dessa estrutura.

Keywords: Inspeção de Ativos Elétricos, Deep Learning, Deteção de Objetos, Seguimento de Múltiplos Objetos, VANT

Contents

This page was intentionally left blank.

List of Figures

This page was intentionally left blank.

List of Tables

This page was intentionally left blank.

Acronyms

2D Two Dimensional

3D Three Dimensional

ANN Artificial Neural Network

API Application Programming Interface

CNN Convolutional Neural Networks

DLT Direct Linear Transformation

GPS Global Positioning System

GPU Graphics Processing Unit

IR Intermediate Representation

ISEP Instituto Superior de Engenharia do Porto

KCF Kernelized Correlation Filters

LIDAR Light Detection And Ranging

LMDB Lightning Memory-mapped Database

LSA Laboratório de Sistemas Autónomos

LSD Line Segment Detector

MORSE Modular Open Robots Simulator Engine

NCS Neural Compute Stick

RCNN Region-based Convolutional Neural Network

ReLU Rectified Linear Unit

RI-LDP Rotation Invariant Local Directional Pattern

ROI Region of Interest

ROS Robot Operating System

RPN Region Proposal Network

SSD Single shot multibox detector

SVD Singular Value Decomposition

SVM Support Vector Machine

TIR Thermal Infrared

UAV Unmanned Aerial Vehicles

VPU Vision Processing Unit

YOLO You Only Look Once

Chapter 1

Introduction

In the recent years, we have witnessed a big growth of the electrical demand by the countries due to its demographic and economic expansion, requiring larger and more complex power systems. This complexity and growth have led to a greater need for monitoring, inspection and maintenance of these systems in order to reduce their vulnerabilities and avoid interruptions of the electricity flow. Power failures, both short and long-term, can have catastrophic effects on critical infrastructures, such as hospitals, telecommunications networks and schools, or on some companies, resulting in shutting down their productions leading to economic losses.

The inspection of electrical assets, such as insulators, pylons, dams, among others, was initially performed by specialized human labour on foot patrols or manned helicopters. These solutions are expensive, inefficient, can take a lot of time and are potentially dangerous. Over the past few years, the inspection process evolved to the used of robotic systems, such as automated helicopters, flying robots and climbing robots, replacing the human being in high-risk tasks, improving the cost-efficiency ratio as well as increasing the speed, accuracy and safety of the inspection tasks. Among these robotic systems, UAV are presented as one of the best options, as they can carry state-of-the-art sensors and fly close to power lines, which can significantly improve the inspection accuracy, obtaining more detailed information on conductors, pylons and energy assets.

The development of UAVs together with the great advance of the deep learning technologies, particularly in the detection of objects, allowed to increase the level of autonomy of the inspection process. But currently these systems require large computational resources to run the proposed algorithms and, consequently, they present an high cost in real time requirements, power consumption, portability, payload for the UAV's and price.

This thesis aims to overcome the drawbacks of the current inspection systems by developing a modular, low-cost and capable of run in real-time system for electrical assets inspection, for integration in UAV, based on Deep Learning, which allows to collect visual samples grouped and aggregated for each electrical asset detected.

1.1 Motivation

The Autonomous Systems Laboratory (LSA) is a R&D unit from Instituto Superior de Engenharia do Porto (ISEP) which conducts research in autonomous systems, especially in the areas of marine and aerial robotics, with the development of multiple land, air and sea autonomous robots. One of the projects that this laboratory was involved in partnership with EDP Labelec is the Electrical asset inspection project. This project arises as a catalyst for this thesis, having as main objective to add a set of cutting-edge methods to improve the current state of the visual inspection of electrical assets, as well as make the inspection process more autonomous.

1.1.1 EDP Labelec Project - Electrical Asset Inspection

This project consisted in developing a UAV, with operational capacity to address the requirements of inspection and monitoring of electrical assets, in order to guarantee an optimization of the inspection process in EDP assets such as lines, substations and wind turbines. This UAV contains:

- A sensory payload composed of calibrated thermal imaging camera, high resolution camera and LiDAR system;
- Navigation system with high accuracy of positioning and attitude, with redundancy and tolerance to strong magnetic fields;
- Drone control maneuvers using on-board sensory information from navigation and perception systems;
- Real-time sensing processing and fusion on-board, allowing the automatic generation of preliminary reports with the geo-referenced position and image of points of interest.
- Operator software tool with design and mission specification capability, operation parameterization, operation supervision, problem diagnosis, data export/access capability and preliminary report generation tool for the inspection process;

This project allowed a greater reduction of the human risk/equipment inherent in the inspection, the operational costs, not requiring specialized teams for piloting and the time of operation allowing the execution of more services.

1.2 Goals

This thesis aims to develop an autonomous system, as an alternative to the current traditional methods, capable of performing electrical asset inspection tasks in a faster and cheaper way, using a UAV equipped with a vision system. This system must be able to detect the assets that arise in its field of view and affect the behavior of the UAV in order to obtain the best representative images of the current state of the asset. In this way, the development of the project implies the accomplishment of the following objectives:

- Identification of the electrical assets most relevant to the inspection process;
- Creation of a dataset with different images of the assets;
- Development of a data augmentation system to reduce some noise of the dataset and increase the accuracy of the object detection system;
- Implementation of an object detection system using the Movidius Neural Compute Stick
- Comparison of different Convolutional Neural Networks (CNN) for electrical assets detection;
- Development of a system that tracks the electrical assets during the inspection process;
- Definition of an autonomous navigation system capable of perform movements along the transmission line and around electrical structures;

1.3 Thesis Structure

In Chapter 3 it is presented the theoretical concepts on which all systems developed in this dissertation were based.

The Chapter 4 presents the high level architecture of the proposed system.

The Chapter 5 presents the perception system based on Deep Leaning to detect electrical insulators or structures.

The Chapter 6 describes the monitoring system for electrical assets, which consists of following the different assets detected in the image sequence throughout the inspection process.

Chapter 7 consists of the presentation of the autonomous inspection system based on a state machine that allows the generation of a set of trajectories in order to take advantage of the monitoring system of electrical assets along a power line.

Chapter 8 presents the results obtained in the detection of electrical assets, as well as the evaluation of the performance of the monitoring system when the autonomous inspection system is applied.

Finally, in chapter 9 the conclusions are presented about all the other chapters and some suggestions for future work are made that allow to improve the system obtained in this dissertation.

Chapter 2

State of the Art

In this chapter it will be exposed some works that addressed the inspection of electrical assets using computer vision which innovated by the use of UAVs in the automation of the inspection process. Among these, it is addressed the detection of assets, from projects that used traditional methods to the latest ones that explored deep learning techniques.

Nowadays, due to the electrical demand all over the world, the maintenance of electricity transmission and distribution networks has become especially relevant and critical. For such, different types of visual inspection methods have emerged in order to maintain the availability, reliability and sustainability of these systems by electricity companies.

Among these methods, there are some that require direct human intervention, such as the foot patrol or the helicopter-assisted inspection. The foot patrol inspection is performed by a team of inspectors that travel on foot to visually inspect the power lines, either using binoculars and/or cameras [1]. The helicopter-assisted inspection, consists on a multiple people team, usually a pilot, an inspector, and/or a camera operator, that, on board of a helicopter, performs the monitoring of the electrical assets and the data acquisition for offline analysis [2]. These solutions are slow, expensive, potentially dangerous and present a lack of efficiency, since they are highly dependent on terrain and weather conditions.

In recent years, the visual inspection methods aim to replace the direct intervention of specialized human labor, by using robots such as climbing robots and/or UAVs [1]. In climbing robots inspection, the inspection is conducted by a robot that moves along power lines, which usually carries a variety of sensors for line navigating, crossing obstacles and inspecting the lines and power components [3, 4]. In Figure 2.1, it is possible to see the climbing robot used in [4] with some examples of detections. Besides inspec-

Figure 2.1: Insulator inspection using a climbing robot, with examples of the data provided by it (Source: [4]).

tion, these robots can also be used to clean some power line components [5, 6]. This kind of robots are relatively slower than UAVs and are in contact with the power lines which could damage the cable or may not be able to pass across some obstacles on the cable. Despite these limitations, the proximity to the power lines can also be taken as an advantage due to the consequent increase of inspection accuracy [1].

The inspection using UAVs is presented as one of the best options taking into account that the cost benefit ratio is higher comparing to the other methods, as it presents a lower cost for high accuracy, efficiency and safety. These platforms are equipped with the state of the art sensors, such as Light Detection And Ranging (LIDAR) sensors and cameras (e.g. video, photo and thermal) which can be used navigate along power lines or gather real time footage and data. In addition, UAVs can fly close to power lines improving significantly the inspection accuracy, taking more detailed information of the conductors, pylons, and power component. Malveiro et al. [7] presented their system for inspection of high voltage power line using UAVs, offering services such as image and video data, thermal inspection, corridor mapping and creation of a Digital Terrain Model, troubleshooting reports, acquisition of LIDAR data and report of the major risks to the electrical assets, like vegetation. Deng et al. [8] proposed a multi-platform of UAV systems with different functionalities in the whole inspection task, which comprised a fixed wing UAV for long range imaging, a hexarotor UAV for short distance imaging and a tethered multirotor UAV which carries a communication module for signal relay between the aircrafts and the ground station, Figure 2.2. This approach has shown a much higher efficiency than traditional methods, being capable to perform the inspection process in less than three hours and cover a bigger area.

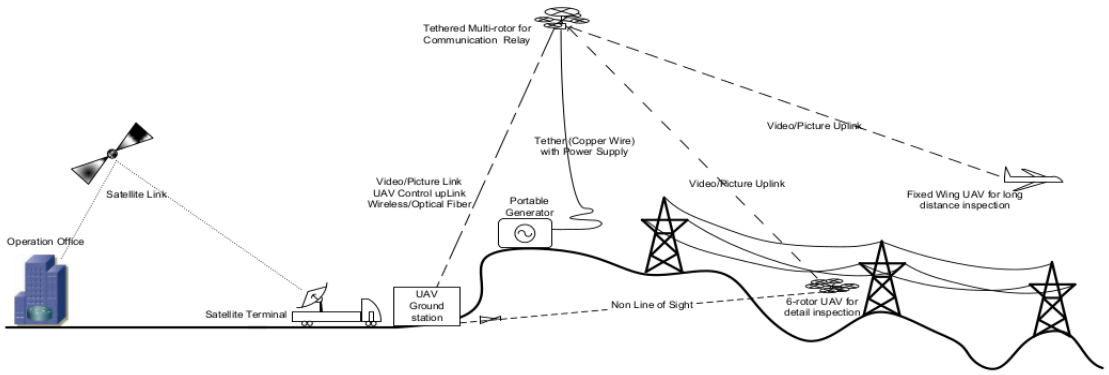


Figure 2.2: Cooperative UAV systems for power line inspection (Source: [8]).

The research about automating the task of visually inspecting power transmission systems using UAVs has also increased in the last years. Some projects, such as [2, 9, 10], have explored the UAVs to these tasks based on autonomous navigation through the power lines applying traditional computer vision methods to offer important features to the user, such as detection of faults in the electrical components or navigation information.

Luque et al. [2] developed a quadrotor helicopter capable to perform three types of flight modes: manual flight - the quadrotor is fully piloted by an user, Global Positioning System (GPS) fixed location - the quadrotor attempts to maintain the current attitude and GPS location, while is flying in manual flight, and GPS navigation - the quadrotor follows autonomously a path consisting in a set of GPS waypoints defined by the user in a ground control station. In Figure 2.3, it is presented the common flight trajectory used in the inspection of power lines. During the inspection flight, the quadrotor, equipped with a vision system based on a color camera and a Thermal Infrared (TIR) camera, transmits the aerial videos to the Ground Control Station, where the image is processed in order to perform a qualitative fault diagnosis. The image processing consists in initially apply a background subtraction method using as input the stereoscopic system (TIR + color cameras), from where the infrared image foreground is extracted. Next, the localization of the power line joints is performed, by detecting points of the poles with the highest temperature, called hot spots. Finally the mean of the temperature of each hot spot is obtained, and if a difference in temperature between all spots is found, it is considered that there is a potential fault which must be followed to further investigation.

Xie et al. [9] developed a multiple sensors platform using LIDAR, thermal camera,

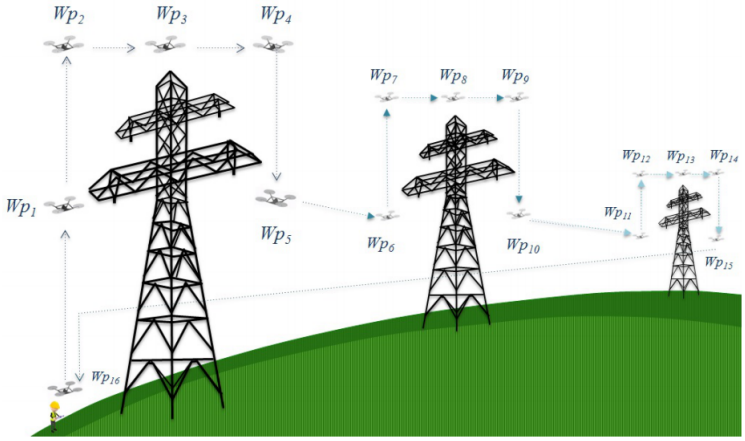


Figure 2.3: Common flight trajectory used in the inspection of power lines (Source: [2]).

an ultraviolet camera, and cameras to acquire information about power line components and surrounding objects, based on a large unmanned helicopter. They presented the planning method for the flight path and the tasks of the sensors before the inspection and the method used for tracking power lines and insulators during the inspection.

Due to the importance of insulators in the power distribution systems, Jabid et al. [10] address their studies to the detection of insulators and analysis of their defects. They proposed a new method based on Rotation Invariant Local Directional Pattern (RI-LDP) descriptors to represent the insulator image as a feature vector, which is used to feed a Support Vector Machine (SVM) classifier in order to differentiate whether a region of a sliding window framework is an insulator region or an non-insulator region. Each insulator region is then partitioned through the detection of elliptical shapes which characterize each cap of the insulator, followed by a filtering process which clutters the ellipses based on their size and orientation. Finally, for each cap, they apply a simple categorization technique by differentiate the defect as cracks, contamination, whitening, bullet damage, and alligatoring effects, taking in consideration the area of each cap.

These works based on traditional computer vision methods are highly dependent on specific detection conditions, such as good illumination, low background interference and knowledge about the scale and orientation of the electrical components. This requires a prior system's adaptation to the different conditions presented, which always requires a high knowledge about the conditions of the site under inspection, and thus making it highly reliant on specific threshold values.

To overcome this issues, the development of deep learning technology and, consequently, the rise of CNN resulting on a breakthrough in the field of object detection,

some projects have used it to improve the detection of the electrical assets.

Siddiqui et al. [11] is one of these projects, where they proposed a real-time electrical equipment detection and defect analysis. Using Darknet's open source framework and object detection system, You Only Look Once (YOLO) version 2, it was possible to differentiate 17 types of the insulator with 98% of accuracy. Additionally, it was developed a defect analyzer, using a rotation normalization and ellipse detection method, capable of detecting gunshot defects in the equipment.

Tao et al. [12] also addressed their research to insulator detector and defect analysis using aerial images. They proposed a two-stage cascading network to perform, in the first stage, localization of insulators and in the second stage defect detection. The insulator detection is based on a VGG classifier which generates a set of feature maps that are fed to a Region Proposal Network (RPN) for regions of interest proposals. This insulator location network is followed by a crop module which consists in crops the input images at the coordinates of the region proposals resultant from the RPN. For each cropped region, the second network is applied in order to detect possible defects, by initially generate a set of feature maps, this time using a more accurate network, the ResNet-101, because the defects are smaller than the insulator patches. The generated feature maps are then fed to a new RPN to locate the defects in each insulator patch. This cascade network can be visualized in Figure 2.4. They have created a data augmentation method to generate a bigger dataset to train the network, in order to prevent the scarcity of defect images, allowing a big increase in precision and recall, as well as, detection of a defect under various conditions.

Hui et al. [13, 14] presented a solution to continuous navigation along one side of overhead transmission lines using deep learning. They developed a system capable to detect and track in real time tower transmission, to provide their localization, based

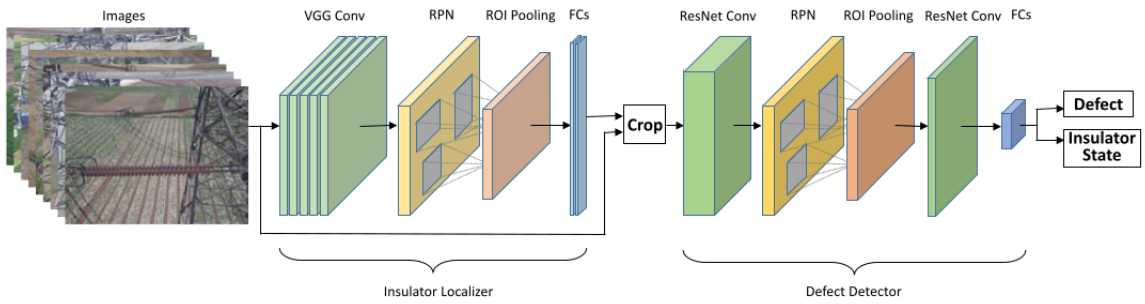


Figure 2.4: Convolutional Neural Network for insulator inspection and defect analysis (Source: [12]).

on Faster Region-based Convolutional Neural Network (RCNN) to reliably detect the transmission towers and Kernelized Correlation Filters (KCF) to continuously track their localization in the image. To continually navigate along the power lines, they computed and optimized their vanishing point to provide UAV with a robust heading, using the Line Segment Detector (LSD) to detect the lines. Finally, to measure the distance from transmission lines, a distance estimation process from UAV to the tower, by triangulation, was performed, following a multiple view strategy.

In most of the presented works, the systems require large computational resources to run the proposed algorithms and, consequently, they present an high cost in real time requirements, power consumption, portability, UAV's payload and price.

Chapter 3

Fundamentals

This chapter provides a brief overview of some concepts that are considered important for helping in the understanding of the developed work.

3.1 Deep Learning

Deep learning is as a subset of machine learning, that trains a computer to perform human-like tasks, such as speech recognition or visual object recognition, by using artificial neural networks that are capable to learn task related features from large unstructured data sets by repeatedly change its internal parameters [15].

3.1.1 Artificial Neural Networks

An Artificial Neural Network (ANN) is a series of algorithms that aim to recognize underlying relationships in a data set through a process similar to the functioning of the human brain. These networks can adapt to input changes, allowing to generate the best possible result without having to redesign the output criteria [16].

3.1.1.1 Neurons

The ANN is compose by a collection of interconnected neurons, often called nodes or units, that can transmit signals from one to another and operate in parallel according to the given input. Each neuron takes the signals from other neurons as inputs (e.g. x_0, x_1, x_2) and interacts multiplicatively with them (e.g. w_0x_0, w_1x_1, w_2x_2) to control the weight/strength (e.g. w_0, w_1, w_2) of influence between neurons. After, all of the

weighted signals get summed and it's added some bias offset b [17]. Finally, the result of this process is applied to an activation function to compute the neuron output. The mathematical model of a neuron is represented in Figure 3.1.

Activation Functions: An activation function is used to make the network more powerfull and add the ability to the networks of learning how to deal with complex data and represent non-linear complex arbitrary functional mappings between inputs and outputs.

The most common activation functions are the Sigmoid,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

the Tanh or Hyperbolic Tangent,

$$f(x) = \tanh(x) = \frac{\sinh x}{\cosh x} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.2}$$

and the Rectified Linear Unit (ReLU),

$$f(x) = \max(0, x) \tag{3.3}$$

In Fig. 3.2 it is possible to analyze the different representations of the activation functions.

3.1.1.2 Layers

Depending on their inputs and outputs, neurons are generally arranged into three different types of layers, as represented in Figure 3.3:

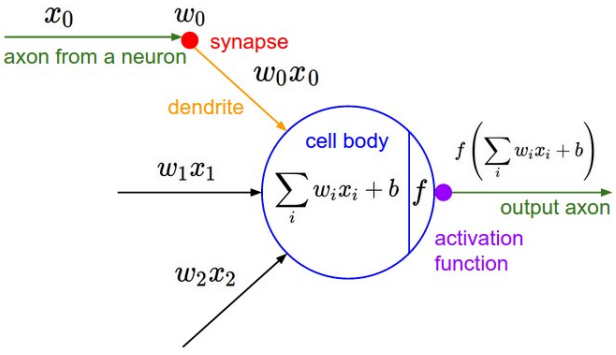


Figure 3.1: Mathematical model of a neuron (Source: [17]).

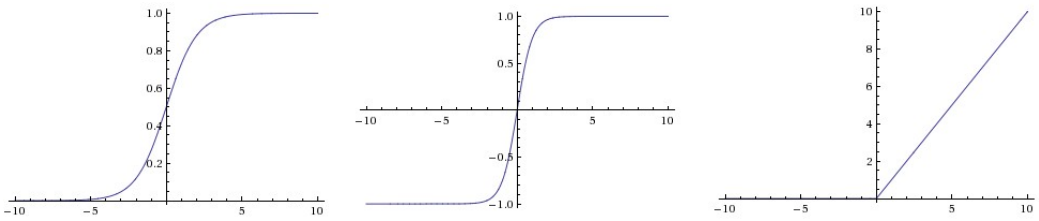


Figure 3.2: Representation of activation functions graphically. On the left is the Sigmoid, the Hyperbolic Tangent on the center and the ReLU on the right

- **Input Layer:** The Input layer communicates with the external environment that presents a pattern to the neural network. Its job is to deal with all the inputs only and pass the information to the next layer.
- **Hidden Layer:** The Hidden layer is where intermediate processing is done, it performs computations and then transfers the weights (signals or information) from the input layer to the following layer. This layer is responsible for the extraction of required features from the input data.
- **Output Layer:** The output layer of the neural network collects and transmits the information mapped to the desired output format.

3.1.1.3 Learning Process

To train an ANN is necessary to feed it with a lot of data. The learning process is based on two types depending on this input data: supervised learning and unsupervised learning. The supervised learning uses a training dataset as a ground-truth, behaving as

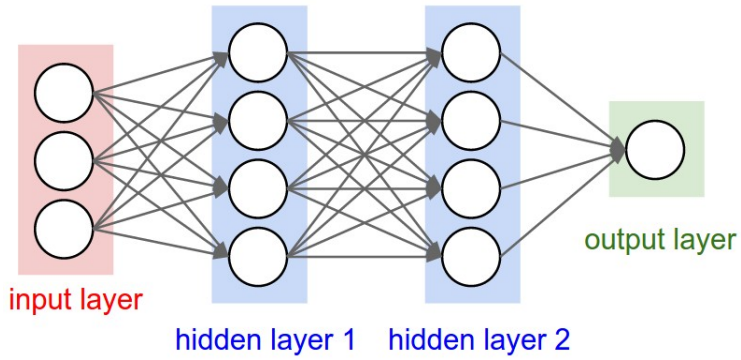


Figure 3.3: Four-layer neural network (Source: [17]).

a teacher supervising the process. Since the correct answers are known, the algorithm iteratively makes predictions on the training data and is corrected by the ground-truth data. The unsupervised learning doesn't have the correct answers and there is no teacher. The neural network is capable to automatically find structure in the data by extracting useful features and analyzing its structure. In the context of this thesis it will be only considered the supervised learning.

The training of a network consists in minimizing its cost function, which is based on the principle of maximum likelihood using the cross-entropy between the training data and the network's prediction [16]. One of the methods used to minimize the cost function is backpropagation. This method consists in propagate the error backwards through the network modifying the weights between consecutive layers and the biases of the neurons, based on gradient descent technique. The gradient descent consist in deriving the cost function in order to the weight which needs to be minimised.

This learning process is responsible for learning the weights and bias parameters during the training time. However, there are some parameters that influence the quality of learning and that can be assigned or changed before training the model. These parameters are called hyperparameters and may include the number of iterations, the number of hidden layers, the number of neurons per layer, among others...

3.1.2 Convolutional Neural Networks

A CNN is a special type of multi-layer neural network that was design to take advantage of the Two Dimensional (2D) structure of an input such as image data or speech signal. Unlike a regular Neural Network, the layers of a CNN have neurons arranged in 3 dimensions: width, height, depth. As the name implies, CNN employ a mathematical operation called convolution.

The architecture of these networks is composed by three main types of layers: Convolutional Layer, Pooling Layer and Fully-Connected Layer (Fig. 3.4).

The Convolutional layer computes the convolution between the input and a filter resulting on and pass the output (feature maps) to the next layers. The execution of convolution, as depicted in figure 3.5 is done by sliding the filter (or kernel) over the input, where at every location a matrix multiplication is performed, followed by the sum of each multiplication element. This process depends on some hyperparameters, such as the number of filters, the stride, the padding values. Stride is the size of the step that the convolution filter moves each time. By increasing the stride size, the filter is sliding over the input with a larger interval and thus has less overlap between the cells. The

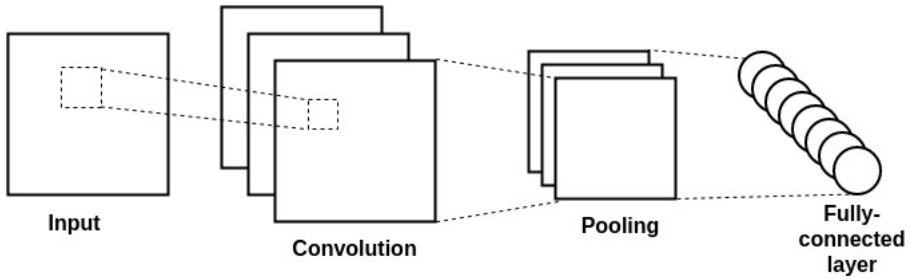


Figure 3.4: Basic CNN Architecture

padding process consists in to pad the input with zeros around the border, which allows to control the spatial size of the output. The depth of the output is equal to the number of filters. The spacial arrangement of the output produced by the convolutional layer is:

- $Width(W) = Height(H) = \dfrac{W0 - F + 2P}{S} + 1$
- Depth = K

where K is the number of filters, F is the filter size, S is the stride value, P is the padding and W0×H0×D0 the dimension of the input.

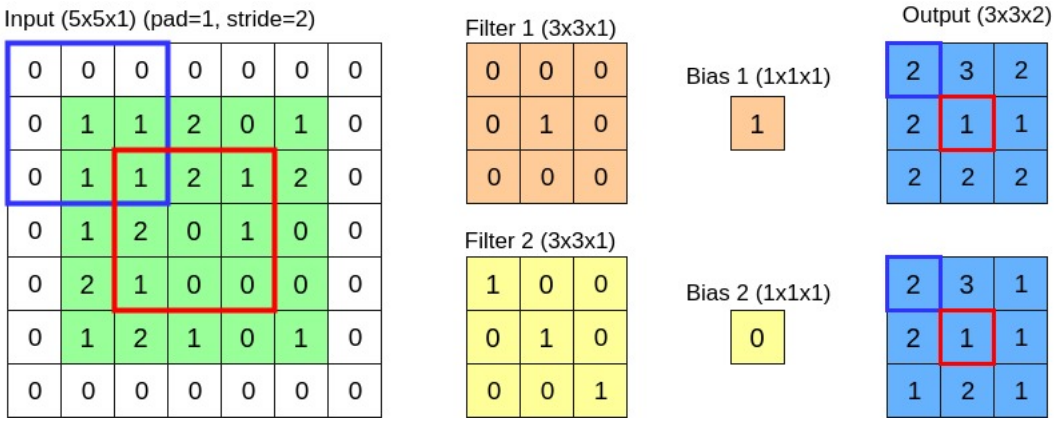


Figure 3.5: Convolutional Layer Process. In this example, the input volume is of size W0=5, H0=5, D0=1 and hyperparameters K=2, F=3, S=2, P=1. Therefore, the output volume has spatial size ((5 - 3 + 2×1)/2) + 1 = 3 . Each element of the output is computed by elementwise multiplying the input with the filter, summing it up, and then adding the bias to the result.

After the convolution an activation (e.g. ReLu, sigmoid, tanh) is performed. The purpose of activation is, as explained in Section 3.1.1.1, to introduce non-linearity.

The pooling layer reduces the spatial size of the feature map to decrease the amount of parameters and computation in the network, hence, to control overfitting. The most common option is the max-pooling, which consists in selecting the highest value in a 2 × 2 input region (figure 3.6). The depth dimension remains unchanged.

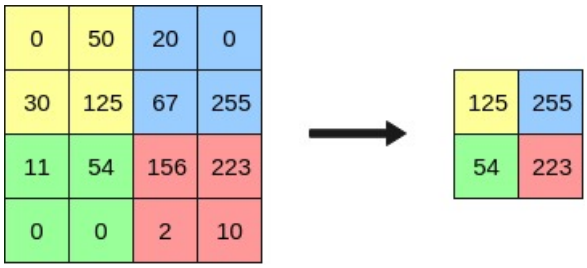


Figure 3.6: Max pooling operation

To complete the CNN, it is necessary to give it the ability to actually make predictions. That is done using fully-connected layers which have every node connected to every output from the previous layer, followed by the activation function (e.g. Softmax for a multiclass classification problem).

In order to get the best possible performance, CNN models were tending to get bigger and bigger, which required even more powerful Graphics Processing Unit (GPU). With the appearance of edge-devices, which have extreme memory and computation constraints, arose the need of use lighter CNN models that could compete with the state-of-the-art models in terms of performance.

3.1.2.1 Lightweight CNN Architectures for Classification

MobileNet V1: In 2017, Google proposed the first generation of the MobileNet [18], particularly useful for mobile and embedded vision applications, which introduced a new type of convolution named depthwise separable convolution.

A depthwise separable convolution consists of 2 parts:

- **Depthwise convolution**

  A one channel filter is convolved independently over each channel of the input, which means that if the input has M channels it should be used M one channel filters. The result of each convolution is concatenated with the other ones.

- **Point wise convolution**

  The purpose of this operation is only to reduce or increase the feature map depth. Basically, each pixel of the intermediate result from depthwise convolution, with

size WxHxM, is convolved with a 1x1xM filter, generating a feature map with WxHx1 volume. If the desired output is N, it should be convolved N 1x1 filters rather than just one.

In figure 3.1.2.1 it is possible to see an example of this process.

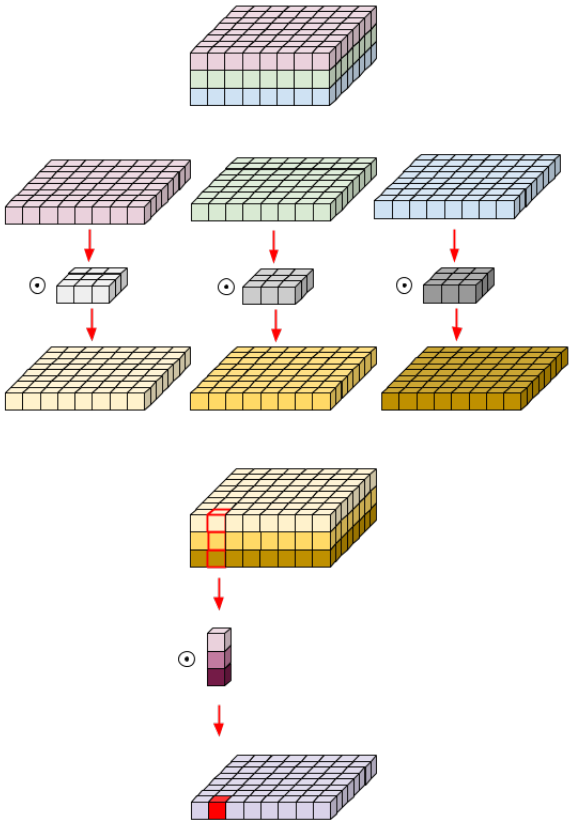


Figure 3.7: Depthwise Separable Convolution Process
(Source [19])

Comparatively with the standard convolution, the depthwise separable convolution computational cost is considerably lower, because it requires less operations than the standard convolution:

- Standard Convolution: $D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$
- Separable Depthwise Convolution: $D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$

where, M is the number of input channels, N the number of output channels, $D_K \times D_K$ the kernel size and the feature map size $D_F \times D_F$.

MobileNet V2: In 2018, Google proposed its second version of MobileNet [20]. It was built upon the ideas from MobileNet V1, using depthwise separable convolution as efficient building blocks. However, the second version introduces two new features to the architecture: linear bottlenecks between layers and residual connections connections between bottlenecks, building a block called Bottleneck Residual Block. Its representation is depicted in Figure 3.8.

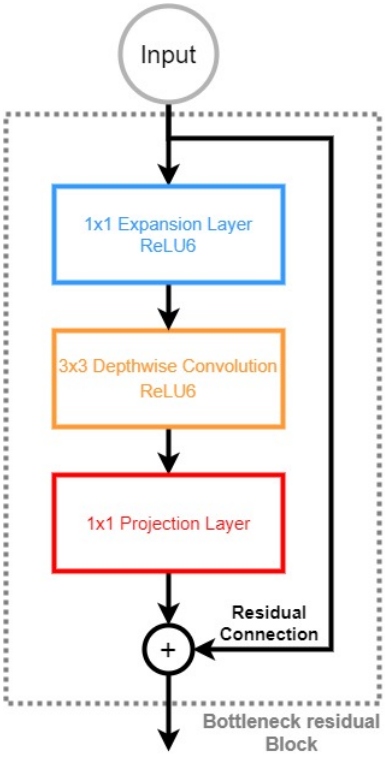


Figure 3.8: Bottleneck Residual Block (adapted from [20])

The first layer of this block is a 1×1 convolution, known as expansion layer, whose purpose is to expand the number of channels in the data before it goes into the depthwise convolution. This layer acts as a data decompressor, restoring the data to its previous full form, since information flows between blocks like a compressed version of the real one. This expansion is defined by a hyperparameter called expansion factor, whose default value is 6.

The last two layers represent a depthwise separable convolution: a depthwise convolution that filters the inputs, followed by a 1×1 pointwise convolution layer. However, in this version, the pointwise convolution has a new task: unlike the first version of MobileNet, the number of channels gets smaller, projecting data with a high number of

channels into a feature map with lower number of channels. For this reason, this layer is now known as projection layer. In other words, taken the uncompressed data from the expansion layer, the depthwise convolution layer filters this data and the projection layer compresses it to the input data form.

When the number of channels going into the block is the same as the number of output channels, an inverted residual connection is used. This residual connection works just like in ResNet [21] and exists to help with the flow of gradients through the network during the training stage, improving the back-propagation process. Here, this block is called inverted residual because connects layers with a small number of channels while in the normal residual from ResNet it goes between layers that have many channels.

These layers, except the projection layer, use ReLU6 as the activation function. The projection layer generates low-dimensional data, since using a non-linearity after this layer destroys useful information, reducing its representational power.

PeleeNet The architecture of PeleeNet has been designed as a variation of DenseNet [22], with optimizations that consider the mobile devices with limited computing power and memory resource [23]. The three main parts that PeleeNet presents are:

- **Two-Way Dense Layer:** A two-way dense layer, Figure 3.9, based on the Inception module from GoogLeNet [24], gets information from the previous layer using two different ways by applying two different filter sizes making the network wider instead of deeper. This module also increases the capability for learning visual patterns for large objects.

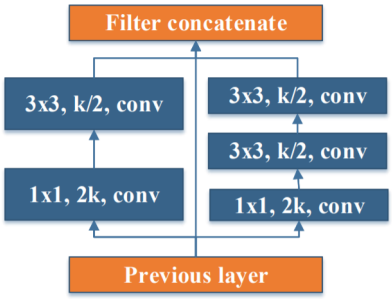


Figure 3.9: Two-Way Dense Layer structure(Source [23])

- **Stem Block:** A stem block is a stack of operations that precede the first Dense block, which in this case refers to the two-way dense layer. This block is inspired by Inception-V4 [25] and DSOD [26] and its purpose is to increase the feature

expressiveness without increasing too much the computational cost. In Figure 3.10 it is possible to see its structure.

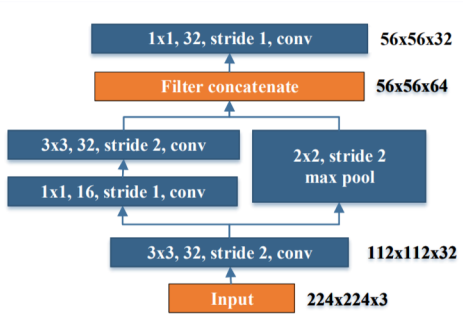


Figure 3.10: Stem Block structure (Source [23])

- **Transition Layer:** The Transition Layer in DenseNet is composed of a 1×1 convolution and a maxpool layer with an associated compression factor that can reduce number of feature map channels generated by the previous dense block. In PeleeNet, this compression does not occur and the number of output channels is the same as the number of input channels.

From the design of this architecture, its authors proved that it is possible to build an efficient model using only standard convolutions instead of using the previously presented depthwise separable convolutions.

3.1.3 Object Detection

CNNs can be grouped according to their task in computer vision. The networks previously presented were developed only to predict the type or class of an object in an image and, therefore, are included in the image classification task. Object detection is responsible of locating objects in an image with a bounding box where each of these objects have their class or type predicted.

The object detection networks can be divided according to their type: two-stage network or single-stage network. In two-stage networks, like the R-CNN family [27], [28] and [29], the initial stage identifies region proposals or subsets of the image that might contain an object, whereas the second stage classifies the objects within the region proposals.The single-stage networks treat object detection as a simple regression problem by taking an image while learning both class probabilities and bounding box coordinates. In this project it will only be considered the single stage networks.

3.1.3.1 Single shot multibox detector (SSD)

The SSD approach [30] is based on a feed-forward convolutional neural network that uses an image classification network as a base network (originally the VGG-16) at the early stages. This network works as a feature extractor, whose features are followed by convolutional layers that decrease in size progressively, generating features at different scales. The multi-scale feature maps are divided into cells and for each cell, a fixed amount of default bounding boxes with different sizes and aspect ratios are generated. For each box, the network generates scores for the presence of each object class and produces adjustments to better match the object shape. This step generates a lot of bounding boxes, that's why it is necessary a non-maximum suppression step which removes the low confidence boxes and fuses highly overlapping ones. In the Figure 3.11 it is possible to observe the SSD architecture.

3.1.3.2 You Only Look Once (YOLO) family

The YOLO-based approach consists of a series of methodologies that have been gradually improving its original version, accompanying the development of object-detection technologies, which is currently in the third version [31]. In this version, initially, during the training phase, the network is fed with images to predict Three Dimensional (3D) tensors corresponding to a certain number of scales (three scales in Yolov3 and two scales in tiny-Yolov3), coming from the backbone network as the feature extractor, which aims to detect objects with different sizes. For each N scale, the image is divided into N×N grid cells and each grid cell corresponds to a voxel that contains the bounding box coordinate, objectness score, and class confidence. If the center of the object's ground truth falls inside a certain grid cell, it is assigned with three prior/anchor boxes of different sizes, choosing, in the training phase, the one that better overlaps with the ground truth

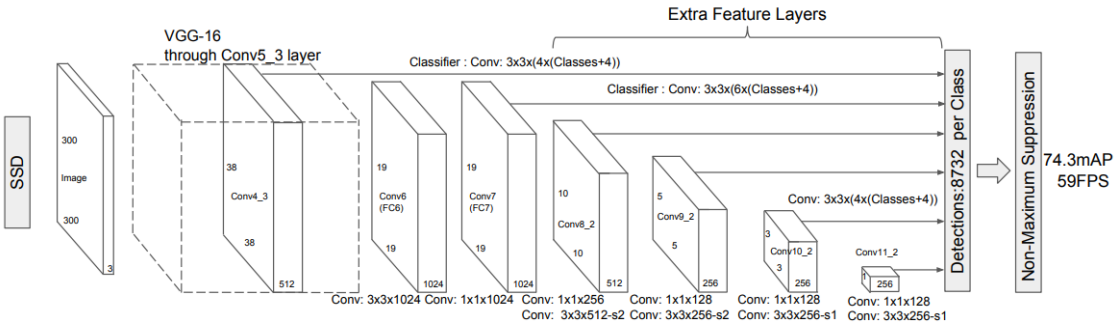


Figure 3.11: Single shot Detector Architecture (Source [30])

bounding box and predicts the corresponding offsets to the prior box. The main differences between the Yolov3 and the tiny-Yolov3 networks are the number of scales and the feature extractor, which are smaller in both cases in the tiny-Yolov3 network.

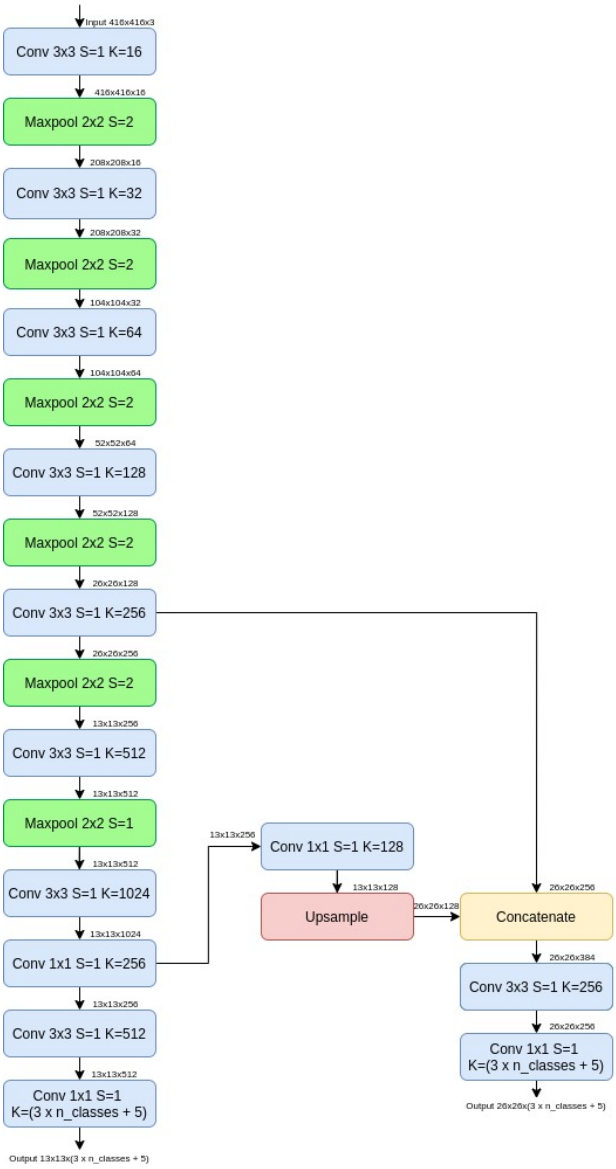


Figure 3.12: Tiny Yolov3 Architecture (Source [31])

3.2 Multiple Object Tracking

Multiple object tracking consists of keeping track of objects in each frame as they move around and provide a consistent labelling of them. The main idea is to, for each frame, detect the location of all possible objects of interest, then predict new locations of objects from previous frames and finally associate the objects in the current frame to the past frames using features such as location and appearance.

In the following subsections it will be described the approaches to the two tracking phases: the Bayesian estimation for prediction of objects location and the Hungarian Algorithm for matching of objects and predictions.

3.2.1 Bayesian Estimation

In order to estimate the state of multiple objects, it's necessary to estimate the state of a single object. The Bayesian estimation refers to the tasks of recursively estimating the single state, denoted by $\mathbf{x}_k$, which generates a single detection, denoted $\mathbf{z}_k$, at each discrete time step k [32]. The Bayesian estimation scheme for time step k is illustrated in Figure 3.13.

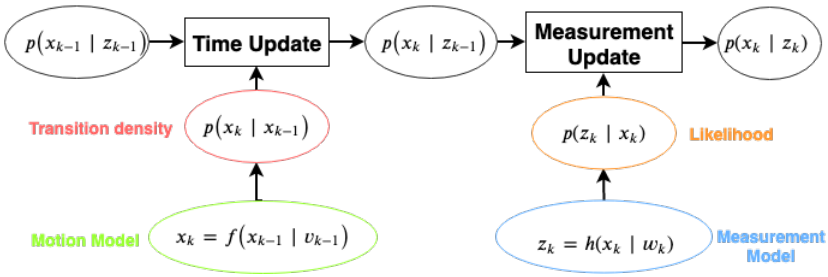


Figure 3.13: Bayesian estimation overview

This algorithm is divided into two steps: the prediction step (or time update) and the measurement update step.

Predict Step The Predict, or time update, step consists in predict the motion, given the measurements up to $k-1$, z_{k-1}, that the object performs between detections. This is usually described by the Chapman-Kolmogorov equation:

$$p(x_k|z_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{k-1})\, dx_{k-1} \tag{3.4}$$

The transition density $p(x_k|x_{k-1})$ is defined from a predefined motion model $x_k =$

$f(x_{k-1}, v_{k-1})$, usually a non-linear function, where $\mathbf{v}_{k-1}$ is a random process noise, included to handle the uncertainty and model errors.

Update Step The Update, or measurement update, step consists of using the sensor detections to update the object predicted state, which requires a measurement model $z_k = h(x_{k-1}, w_k)$, where w_k is sensor noise, meaning that the detection is corrupted by noise. The updated state follows the Bayes theorem:

$$\begin{aligned} p(x_k|z_k) &\propto p(z_k|x_k)p(x_k) \\ &\propto p(z_k|x_k)p(x_k|z_{k-1}) \end{aligned} \tag{3.5}$$

3.2.1.1 Kalman Filter

The Kalman filter [33] is a method used for estimating a discretized state vector x_k, based on the knowledge of the system and the measurement vector z_k, at time k. Thus, it is usually more accurate than filter that compute their estimates based on the current measurement. The goal is to estimate a linear state-space system, first is made a prediction using the Chapman–Kolmogorov equation 3.4, at time k. When a new measurement is received, that prediction is updated based on Bayes theorem 3.5. The *a posteriori* estimate is the estimate after the measurement z_k is taken into account and is given as

$$\hat{x}_{k|k} = E[x_k|z_1, z_2, ..., z_k] \tag{3.6}$$

where the operator E[·] denotes the expected value of (·). Based on the measurements and previous states, the goal is to approximate the states such that the error between the estimated and true states are minimized,

$$e_k \triangleq x_k - \hat{x}_{k|k-1} \tag{3.7}$$

Approach: The initialization of the estimation process starts with the a posteriori estimate of the initial state vector x_0. Since no measurements are available initially, it is reasonable to set this as the expected value of the initial states:

$$\hat{x}_0 = E[x_0] \tag{3.8}$$

During the predict step is performed a prediction of the mean and covariance using a motion model, which in the case of linear models and independent Gaussian noise can be written as:

$$\hat{x}_{k|k-1} = F \cdot \hat{x}_{k-1|k-1} + B \cdot u_k \tag{3.9}$$

$$P_{k|k-1} = F \cdot P_{k-1|k-1} \cdot F^T + Q \tag{3.10}$$

where F is the system's dynamic model which propagates the previous state in the current state, B and u_k are the control matrix and vector, respectively. Q is the process noise covariance and describes the noise associated with the propagation model.

Finally, it will be introduced the measurement update equations for the gain, state estimate and the covariance of the state estimation. The predicted state can be updated taking a new measurement z_k at time k with measurement covariance R.

In order to capture the new information that the new measurement brings, given a relation between the last prediction and the current measurement, the innovations is computed:

$$v_k = z_k - H\hat{x}_{k|k-1} \tag{3.11}$$

being H the measurement matrix, responsible convert the last predicted state to the measurement state space.

The measurement update equations for the linear Kalman filter are:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k v_k \tag{3.12}$$

$$P_{k|k} = (I - K_k \cdot H) \cdot P_{k|k-1} \tag{3.13}$$

where the Kalman gain K_k and innovation covariance S_k are obtained by the following equations

$$K_k = P_{k|k-1} H^T S_k^{-1} \tag{3.14}$$

$$S_k = H P_{k|k-1} H^T + R \tag{3.15}$$

The Kalman gain, K_k determines how much this measurement is reliable, which is a way to weight the two steps against each other, depending on their respective uncertainty. The matrix R represents the measurement model noise and describes the uncertainty related with source of such measurement,

3.2.2 Data Association

The data association task consists in determine the correspondence of new detections in the current frame with the estimations provided by the previous presented step or if this detection represents a new object.

Hungarian Algorithm The assignment problem goal is to find an optimal matching between two sets of objects, which in this case are the Kalman filter instances with the estimation of an object location, in the theory is called as *agent*, and current frame detections, which are also known as *tasks*. For each agent and task association there is a cost function that relates the agent i to the task j, and with this in account it is possible to find an assignment with the minimum cost, such that no agent is assigned more than one task, and no task is assigned to more than one agent. The Hungarian algorithm [34], also known as Kuhn–Munkres algorithm, was proposed to solve the assignment problem in polynomial time, which after some modifications presented in [35], achieves the optimal matches in four matrix manipulation steps with time complexity $O(n^3)$, being n the number of agents. The input is a cost matrix, where each element is the cost of the agent i to the task j, where i and j can describe, respectively, the element's line and column in the matrix. When the number of agents and tasks is not equal, large values are added to the remaining elements of the cost matrix to make it squared, which in this way, are discarded in the final matching.

3.3 Coordinate Systems

Different coordinate systems or frames are used to represent the different state spaces adopted in this thesis, whose representations are made with respect to a particular coordinate system called the frame of reference:

- The **World frame or Global frame** is used as a general reference. This system is usually represented as NED (North-East-Down) or ENU (East-North-Up) from the conversion of the geodesic coordinates to ECEF and from ECEF to NED or ENU. In Figure 3.14 it is represented with the subscript w.
- The **Body frame** is used do represent the platform and follows its movement. The origin is located in the center of gravity of the platform and the base vectors are pointing in the forward, left and up direction. The roll, pitch and yaw angles are defined as the rotation angles from the Global to the Body frame. In Figure 3.14 it is represented with the subscript b.

- The **Camera frame** follows the camera located on the platform. It is defined in the same way as the Body frame but relative to the camera. The origin is located at the focal point of the camera and the optic axis as the Z-axis. In Figure 3.14 it is represented with the subscript c.
- The **Image or retinal plane** is the plane on which the image is formed based on the model of the camera. Its origin is located at the point of intersection of the plane and the optical axis, called principle point and the base vectors are pointing to the right and down direction of the plane. In Figure 3.14 it is represented by the axes x and y.
- The **Image Frame** coordinate system measures pixel locations in the image plane. The origin is located at the top left corner of the image plane and the base vectors are pointing in the right and down direction of the image. In Figure 3.14 it is represented by the axes u and v.

Figure 3.14 summarizes these different coordinate systems.

3.3.1 Conversion Between Coordinate Systems

The conversion from one frame (**A**) to another (**B**) is represented by an homogeneous transformation (M), which consists on a rotation (R) and a translation (t) between both coordinate systems origins. The Transformation M is given by:

$$M_A^B = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{3.16}$$

The inverse homogeneous transformation, from coordinate system (**B**) to (**A**), is defined as:

$$M_B^A = (M_A^B)^{-1} = \begin{bmatrix} R_{3\times3} & -R_{3\times3} \times t_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{3.17}$$

Based on this idea, the transformation from global frame to body frame can be represented as:

$$\begin{bmatrix} X_b \\ Y_b \\ Z_b \\ 1 \end{bmatrix} = \begin{bmatrix} R_w^b & t_w^b \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.18}$$

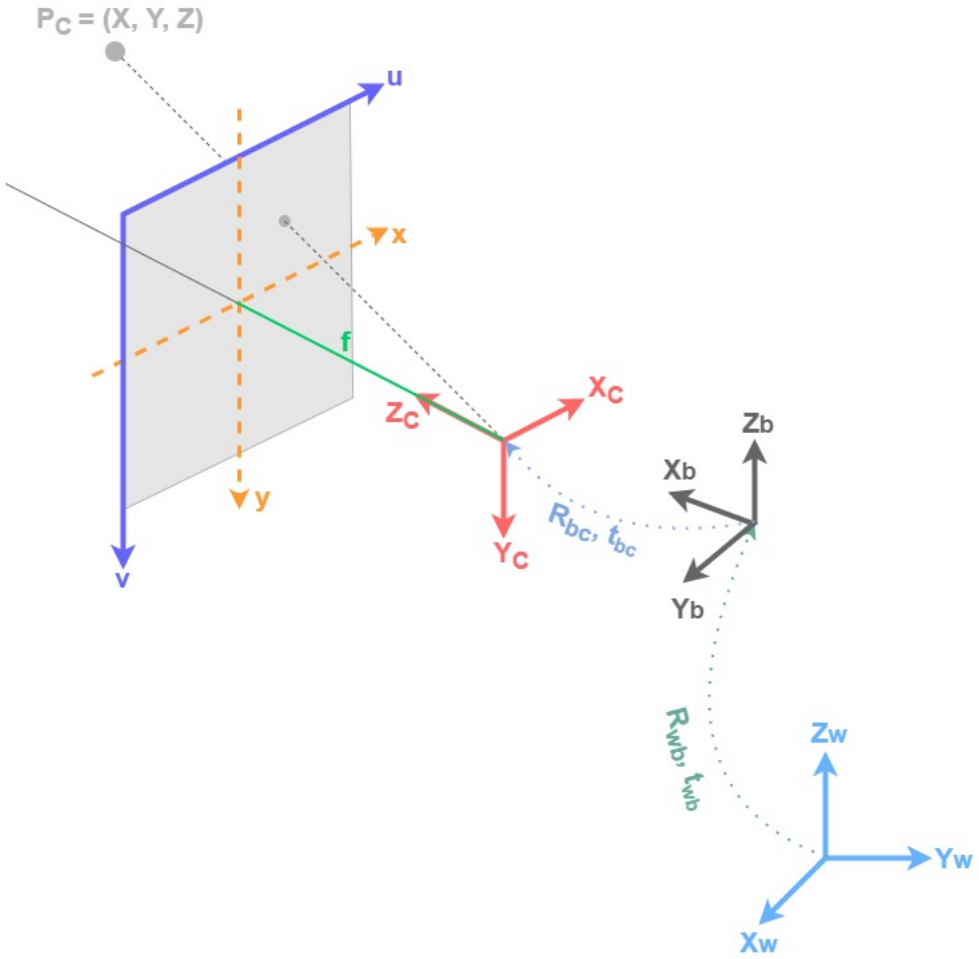


Figure 3.14: Coordinate systems representation from World to Image frame

Similarly, the transformation from body to camera frame is represented as:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_b^c & t_b^c \\ 0_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} X_b \\ Y_b \\ Z_b \\ 1 \end{bmatrix} \tag{3.19}$$

Thus, the transformation from global frame to camera frame is given by:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_b^c & t_b^c \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} R_w^b & t_w^b \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.20}$$

3.3.1.1 Global to image Frame Transformation

To convert between the Global and image frames, a model of the camera is required. Based on the pinhole camera model, which consists in placing a barrier with a small aperture between the 3D scene and the image sensor where only one (or a few) light rays emitted from a 3D Point passes through, as observed in Figure 3.15. The result is that the sensor is exposed by an inverted "image" of 3D object by a one-to-one mapping between both spaces. Sometimes, the image plane is placed between the camera frame origin and the 3D object at a distance f from origin. In this case, it is called the virtual image plane. Note that the projection of the object in the image plane and the image of the object in the virtual image plane are identical up to a scale (similarity) transformation.

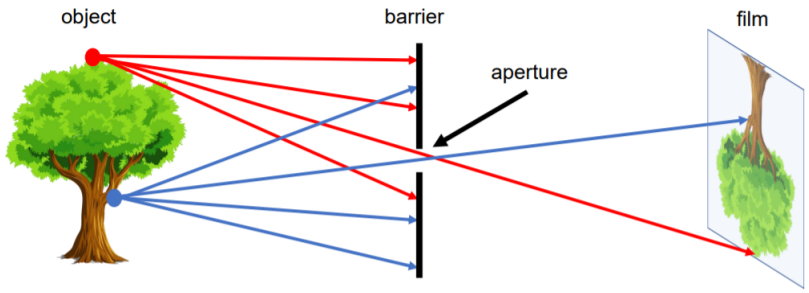


Figure 3.15: The pinhole camera model (Source [36])

This one-to-one mapping can be described using the triangle similarity:

$$\frac{f}{Z_C} = \frac{y_i}{Y_C} = \frac{x_i}{X_C} \tag{3.21}$$

and can be written in the form:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \frac{X_C \cdot f}{Z_C} \\ \frac{Y_C \cdot f}{Z_C} \end{bmatrix} \tag{3.22}$$

where f is the focal length, (X_c, Y_c, Z_c) are the coordinates of the 3D point in the camera frame and (x_i, y_i) are the coordinates of the 3D point projected in the image

plane.

The Pinhole model does not take into account that most cameras have only discrete image coordinates, also referred as pixels, it is necessary to map the coordinates in the image plane to image frame. As explained before, the image plane frame has its origin at the center of the image frame (c_x, c_y), also called principle point, and the image frame has its origin at the top-left corner, so the 2D coordinates (x_i, y_i) in the image plane and 2D coordinates (u, v) in the image frame are offset by a translation vector $[c_x, c_y]^T$. Thus the equation 3.22 becomes:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{X_C \cdot f}{Z_C} + c_x \\ \frac{Y_C \cdot f}{Z_C} + c_y \end{bmatrix} \tag{3.23}$$

Another effect that Pinhole model doesn't take in account is the shape of the pixels, since in not guaranteed that their aspect ratio is one (square pixels). To remove the assumption of square pixels, the model has to incorporate the description of different focal lengths in each direction, f_x and f_y:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{X_C \cdot f_x}{Z_C} + c_x \\ \frac{Y_C \cdot f_y}{Z_C} + c_y \end{bmatrix} \tag{3.24}$$

From Equation 3.24, it is possible to state that the projection from the 3D point in camera frame to the 2D point in image frame is not linear (division by Z_c). One way to make this equation linear is to change the coordinate systems to the homogeneous coordinate system [36]. Since any homogeneous vector $[Z_c u, Z_c v, Z_c]^T$, $x_3 \neq 0$, represents the 2D point (u, v). Converting the vector $[u, v]^T$ to its homogeneous representation with a scale Z_c:

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} X_C \cdot f_x + c_x \cdot Z_C \\ Y_C \cdot f_y + c_y \cdot Z_C \\ Z_C \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \tag{3.25}$$

Note that an arbitrary homogeneous vector $[x_1, x_2, x_3]^T$, $x_3 \neq 0$, represents the Euclidean point $(x_1/x_3, x_2/x_3)$.

Resulting in a linear transformation that can decomposed into:

$$Z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3\times3} & 0_{3\times1} \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \tag{3.26}$$

From the previous transformation presented in Equation 3.20, we know how to map a point in global frame to the camera frame. The homogeneous transformation between these two frames will be represented as E for simplicity sake.

$$Z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} E \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = KE \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.27}$$

As presented in Equation 3.27, the projection consists in two types of parameters, the intrinsic and extrinsic parameters. The parameters contained in intrinsic camera matrix K are the intrinsic parameters (f_x, f_y, c_x and c_y), which depend on the type of camera. The extrinsic parameters include the rotation and translation, which do not depend on the camera's build.

The model does not include, for example, geometric distortions or blurring of unfocused objects caused by lenses and finite sized apertures. The camera distortion is composed by two components, the radial distortion and tangential distortion [37].

The radial distortion occurs due to lens characteristics, where the light rays bend is almost null in the image plane center as it gets bigger near the edges. The radial distorted points are modeled by coefficients k_1, k_2 and k_3, and can be computed as:

$$x_{distorted}(radial) = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.28}$$

$$y_{distorted}(radial) = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.29}$$

where r is distance to the optical center ($r = \sqrt{x^2 + y^2}$) and x and y are the undistorted pixel locations and $x_{distorted}$ and $y_{distorted}$

The tangential distortion occurs when the lens and the image plane are not parallel, resulting in a misalignment in both position and orientation. The tangential distorted points are modeled by coefficients k_4, k_5, and can be computed as:

$$x_{distorted}(tangential) = x + (2k_4 y + k_5(r^2 + 2x)) \tag{3.30}$$

$$y_{distorted}(tangential) = y + (k_4(r^2 + 2y) + 2k_5 x) \tag{3.31}$$

With the use of estimation models, it is possible to obtain the distortion coefficients, allowing to correct the image.

3.4 Multi-view Depth Estimation by Triangulation

In order to estimate the 3D position of a point seen from multiple camera views, it was used a method presented in [38], which presents a statistical analysis of depth and derives a probabilistic approach to explicitly handle outliers. This method was also used in other works such as SVO [39] and REMODE [40] that proved the efficiency and effectiveness of this method when applied to UAV, both in the ability to detect outliers and in the depth estimation.

3.4.1 Triangulation

In order to determine a point in 3D space, given its projections onto two images, it can be used a linear triangulation method described in [41], which makes use of the Direct Linear Transformation (DLT) method idea.

Let C_{k-1} and C_k be two different points of view of the same camera C, represented as:

$$C_{k-1} = K \cdot E_{k-1} = \begin{bmatrix} (c_{k-1}^1)^T \\ (c_{k-1}^2)^T \\ (c_{k-1}^3)^T \end{bmatrix} \tag{3.32}$$

$$C_k = K \cdot E_k = \begin{bmatrix} (c_k^1)^T \\ (c_k^2)^T \\ (c_k^3)^T \end{bmatrix} \tag{3.33}$$

Where, K is the intrinsic matrix of the camera C, E_{k-1} is the extrinsic matrix of the view C_{k-1} and E_k is the extrinsic matrix of the view C_k. Also, $(c_n^i)^T$ is the i-th row of the camera view matrix n.

Being $x = [u_{k-1}, v_{k-1}, 1]^T$ and $x' = [u_k, v_k, 1]^T$ be the projection of a 3D point X, seen by both views, in camera view C_{k-1} and C_k, respectively, as represented in Figure 3.16. Considering the Equation 3.27, the projection of X to the image frame of camera view C_{k-1}, originating x, is given by:

$$\lambda_{k-1} \cdot x = C_{k-1} \cdot X \tag{3.34}$$

Similarly, the projection of X to the image frame of camera view C_k, originating x', is given by:

$$\lambda_k \cdot x' = C_k \cdot X \tag{3.35}$$

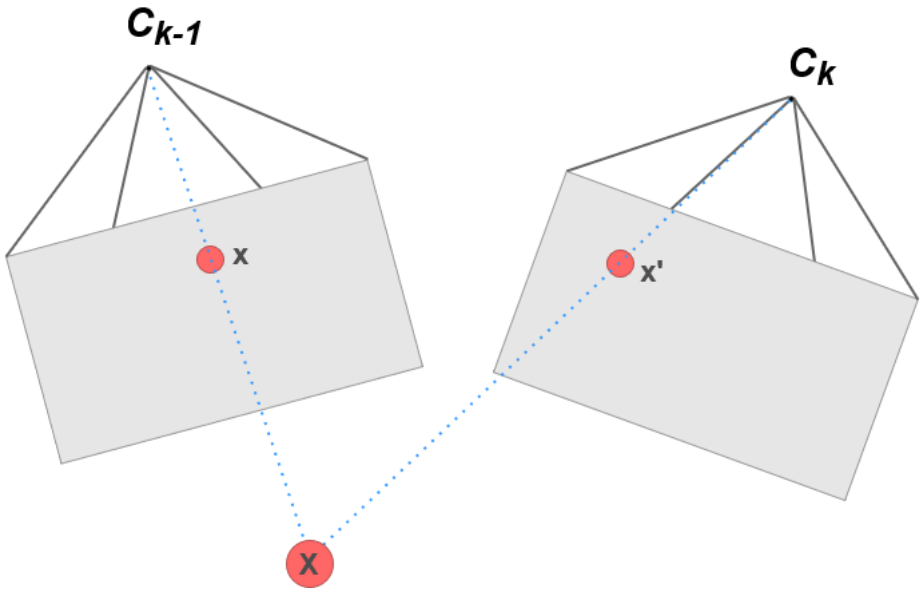


Figure 3.16: Two view Triangulation Representation

Now using the DLT idea that, the homogeneous scale factor, λ_{k-1} and λ_k, can be eliminated, since the cross product of two vectors that have the same direction is zero. Thus, for Equation 3.34, the vectors x and $C_k X$ have the same direction, so $x \times C_k X = 0$. Solving this cross product gives:

$$\begin{cases} u_{k-1} \cdot (c^3_{k-1})^T \cdot X - (c^1_{k-1})^T \cdot X = 0 \\ v_{k-1} \cdot (c^3_{k-1})^T \cdot X - (c^2_{k-1})^T \cdot X = 0 \\ u_{k-1} \cdot (c^2_{k-1})^T \cdot X - v_{k-1} \cdot (c^1_{k-1})^T \cdot X = 0 \end{cases} \tag{3.36}$$

Similarly, for Equation 3.35 the cross product gives:

$$\begin{cases} u_k \cdot (c^3_k)^T \cdot X = (c^1_k)^T \cdot X \\ v_k \cdot (c^3_k)^T \cdot X = (c^2_k)^T \cdot X \\ u_k \cdot (c^2_k)^T \cdot X - v_k \cdot (c^1_k)^T \cdot X = 0 \end{cases} \tag{3.37}$$

The above two systems of Equations 3.36 and 3.37 can be combined into a $AX = 0$ form, by selecting the two linear independent equation, giving a linear equation in X:

$$\begin{bmatrix} u_{k-1} \cdot (c^3_{k-1})^T - (c^1_{k-1})^T \\ v_{k-1} \cdot (c^3_{k-1})^T - (c^2_{k-1})^T \\ u_k \cdot (c^3_k)^T - (c^1_k)^T \\ v_k \cdot (c^3_k)^T - (c^2_k)^T \end{bmatrix} \cdot X = 0 \tag{3.38}$$

Using Singular Value Decomposition (SVD) [42] method for solving the equation 3.38, its possible to obtain the values of X.

3.4.2 Probabilistic Depth Filter

In order to estimate depth in a robust way, a probabilistic depth sensor was used based on [38], which the key idea is to update posterior depth distributions with every new frame.

The authors of [38] have shown that a true depth measurement is concentrated around a single depth value (good measurement) and there is a lot of noise (bad measurement) albeit uniformly distributed along different depth values. This distribution is depicted in Figure 3.17. Based on these observations, a probabilistic depth sensor was modeled as a combination of a Gaussian distribution (for good measurement) around the correct depth Z and an uniform distribution selected from the interval $[Z_{min}, Z_{max}]$ (for bad measurement), respectively weighted by the inlier ratio π which indicates the probability of the measurement being inlier and (1 - π) for the outlier ratio. Mathematically the depth after n^{th} measurement takes the form:

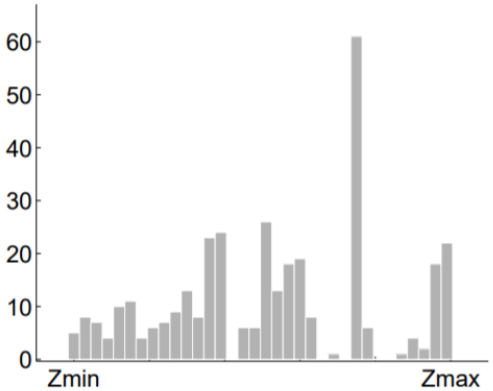


Figure 3.17: Depth distribution for 60 consecutive images measured along the optic ray (Source [38])

$$p(d_n|Z,\pi) = \pi N(d_n|Z,\tau_n^2) + (1-\pi)U(d_n|Z_{min},Z_{max}) \tag{3.39}$$

Where, τ_n^2 is the variance of a good measurement by assuming 1 pixel measurement noise in the image, and can be computed as:

$$\tau_k^2 = \left(||\mathbf{p}|| - ||\mathbf{p}^+||\right)^2 \tag{3.40}$$

Where, $\mathbf{p}$ is the point that corresponds to a good measurement, and $\mathbf{p}^+$ the point resultant from the 1 pixel measurement noise, being both points relative to the reference camera frame, C_{k-1}.

Considering the Figure 3.18, let $\mathbf{b}$ be the translation component, also called baseline, between the camera view C_{k-1}, the reference frame, and C_k, the pose of the current camera frame, $\mathbf{p}$ a point $\mathbf{d}$ units away from C_{k-1}, provided by the triangulation process, and $\mathbf{c}$ be the unit vector from C_{k-1} through the pixel correspondent to the projection of point $\mathbf{p}$ in the image frame. The point $\mathbf{p}^+$ can be computed by applying the law of the sines, giving:

$$||\mathbf{p}^+|| = ||\mathbf{b}||\frac{\sin\beta^+}{\sin\gamma} \tag{3.41}$$

Where,

$$\beta^+ = \beta + 2\arctan\left(\frac{1}{2f}\right) \tag{3.42}$$

and

$$\gamma = \pi - \alpha - \beta^+ \tag{3.43}$$

can be obtained from:

$$\mathbf{a} = \mathbf{p} - \mathbf{b} \tag{3.44}$$

$$\beta = \arccos\left(-\frac{\mathbf{b}\cdot\mathbf{a}}{||\mathbf{b}||\cdot||\mathbf{a}||}\right) \tag{3.45}$$

and

$$\alpha = \arccos(\frac{\mathbf{c}\cdot\mathbf{b}}{||\mathbf{b}||}) \tag{3.46}$$

being the term added to β in Equation 3.45 the angle generated by the ray for 1 pixel in the image. Equation 3.43 ensures that sum of all angles in a triangle is π.

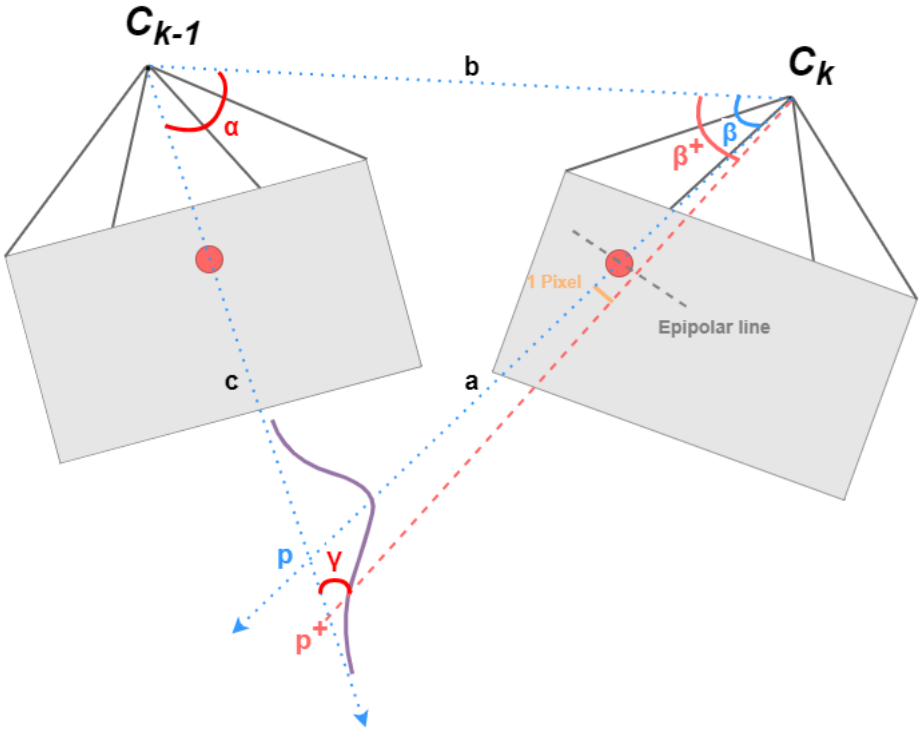


Figure 3.18: Representation of variance in triangulation.

Using the triangulation method described above, it is possible to generate a sequence of depth hypothesis $D = d_{r+1}, ..., d_{r+n}$, for the sequence of n frames $k = r, r+1, \ldots, r+n$ that observe the point p (first observed in the reference frame r), by triangulating views r and k.

The likelihood introduced in 3.39 is a typical mixture model and, as such, its parameters can be estimated using Sequential Bayesian updates, where it is defined a prior over depth and inlier ratio and then calculate the posterior distribution given all measurements. Assuming independent observations, the posterior is given by

$$p(Z, \pi|D) \propto p(Z, \pi) \prod_n p(d_n|Z, \pi) \tag{3.47}$$

Where $p(Z, \pi)$ is the prior on depth and inlier ratio which is assumed to be uniform.

In the supplementary material of [38] it is shown that a good approximation to the depth posterior is the product of a Gaussian for the depth with a Beta distribution for the inlier ratio (Gaussian × Beta):

$$q(Z, \pi | a_n, b_n, \mu_n, \sigma_n^2) = \beta(\pi | a_n, b_n) N(Z | \mu_n, \sigma_n^2) \tag{3.48}$$

where a_n and b_n are parameters of Beta distribution and can be interpreted as probabilistic counters of how many inlier and outlier measurements have occurred during the lifetime of the measurement. μ_n, σ_n^2 represent the mean and variance of the Gaussian depth estimate. If $q(Z, \pi | a_{n-1}, b_{n-1}, \mu_{n-1}, \sigma_{n-1}^2)$ was the true posterior after $n-1$ measurements, the new posterior after observing d_n would have the form

$$p(Z, \pi | D) \approx q(Z, \pi | a_{n-1}, b_{n-1}, \mu_{n-1}, \sigma_{n-1}^2) \times p(d_n | Z, \pi) \tag{3.49}$$

This distribution is no longer of the form Gaussian $\times$ Beta but, as the authors proved, it can be used a moment matching approximation, through the definition of parameters a_n, b_n, μ_n, σ_n^2 such that the posterior $p(Z, \pi | D)$ and the approximation $q(Z, \pi | a_n, b_n, \mu_n, \sigma_n^2)$ share the same first and second order moments for Z and π.

In the supplementary material provided by the authors of [38], it is presented how these parameters were derivated.

First, to simplify the notation, the subscripts are discarded. a', b', μ', σ'^2 and a, b, μ, σ^2 are the posterior and prior parameters respectively.

- **Moments of approximated posterior**

  The first and second order moment w.r.t $\boldsymbol{Z}$ for the Equation 3.48 are given respectively by:

$$E[Z] = \mu' \tag{3.50}$$

$$E[Z^2] = \mu'^2 + \sigma'^2 \tag{3.51}$$

  Similarly, the first and second order moment w.r.t π are given respectively by:

$$E[\pi] = \frac{a'}{a' + b'} \tag{3.52}$$

$$E[\pi^2] = \frac{a'(a'+1)}{(1 + a' + b')(a' + b')} \tag{3.53}$$

- **Moments of actual posterior**

  The first and second order moment w.r.t $\boldsymbol{Z}$ for the Equation 3.49 are given respectively by:

$$E[Z] = C_1 m + C_2 \mu \tag{3.54}$$

$$E[Z^2] = C_1(s^2 + m^2) + C_2(\mu^2 + \sigma^2) \tag{3.55}$$

Similarly, the first and second order moment w.r.t π are given respectively by:

$$E[\pi] = C_1 \frac{a+1}{a+b+1} + C_2 \frac{a}{a+b+1} \tag{3.56}$$

$$E[\pi^2] = C_1 \frac{(a+1)(a+2)}{(a+b+1)(a+b+2)} + C_2 \frac{a(a+1)}{(a+b+1)(a+b+2)} \tag{3.57}$$

Where m and s^2, are, respectively, given by:

$$s^2 = \frac{(\tau^2 \times \sigma^2)}{(\tau^2 + \sigma^2)} \tag{3.58}$$

$$m = s^2 \left(\frac{d}{\tau^2} + \frac{\mu}{\sigma^2} \right) \tag{3.59}$$

And, C_1 and C_2, can be computed as:

$$C_1 = \frac{a}{a+b} \mathcal{N}(d \mid \mu, \sigma^2 + \tau^2) \tag{3.60}$$

$$C_2 = \frac{b}{a+b} \mathcal{U}(d) \tag{3.61}$$

- **Moment Matching Approximation**

  As referred before, the moment matching approximation consists in the sharing of the same first and second order moments, for Z and π, of the posterior $p(Z, \pi|D)$ and the approximation $q(Z, \pi|a_n, b_n, \mu_n, \sigma_n^2)$.

  Mathematically, this approximation takes the form:

$$\mu' = C_1 m + C_2 \mu \tag{3.62}$$

$$\mu'^2 + \sigma'^2 = C_1(s^2 + m^2) + C_2(\mu^2 + \sigma^2) \tag{3.63}$$

$$\frac{a'}{a'+b'} = C_1\frac{a+1}{a+b+1} + C_2\frac{a}{a+b+1} \tag{3.64}$$

$$\frac{a'(a'+1)}{(1+a'+b')(a'+b')} = C_1\frac{(a+1)(a+2)}{(a+b+1)(a+b+2)} + C_2\frac{a(a+1)}{(a+b+1)(a+b+2)} \tag{3.65}$$

Solving the above equations, it is possible to get the updated parameters:

$$\mu' = C_1 m + C_2\mu \tag{3.66}$$

$$\sigma'^2 = C_1(s^2+m^2) + C_2(\mu^2+\sigma^2) - \mu'^2 \tag{3.67}$$

$$a' = \frac{e-f}{f-\frac{e}{f}} \tag{3.68}$$

$$b' = a'\frac{1-f}{f} \tag{3.69}$$

Where, e is given by:

$$e = C_1\frac{(a+1)(a+2)}{(a+b+1)(a+b+2)} + C_2\frac{a(a+1)}{(a+b+1)(a+b+2)} \tag{3.70}$$

and, f:

$$f = C_1\frac{a+1}{a+b+1} + C_2\frac{a}{a+b+1} \tag{3.71}$$

This page was intentionally left blank.

Chapter 4

Project

This section presents the architecture of the system proposed in this dissertation, presenting the different modules necessary to enable a UAV to perform an autonomous inspection of electrical assets around an electrical transmission structure.

The system for autonomous inspection of electrical assets is composed by the modules: Inspection System, Waypoint Generator, Electrical Assets Monitoring System and Electrical Assets Detection System. In figure 4.1 it is possible to observe the high-level architecture of the system, showing how the information flows between the different modules and how it interacts with external systems present in the UAV.

The proposed system aims to collect samples of electrical assets from a UAV, through its monitoring throughout the inspection process. For this, this system requires external information that comes from sensors existing in the UAV, more precisely the pose of the platform and a sequence of images from a monocle camera. The exchange of information with modules external to this system is done according to the publish-subscriber paradigm, using the Robot Operating System (ROS) [43] middleware, in which the system subscribes messages with the information it wants, which were previously published by the UAV system via broadcast.

During the inspection process, the Inspection System has the function of defining the behavior of the UAV, assigning a specific action and/or trajectory to each state of the system. For this purpose, this module consists of a state machine, in which each state corresponds to a thread that runs cyclically until a certain condition occurs. When this condition is met, the current state moves to the next state. In each state, this module interacts with the surrounding modules in order to trigger actions, or define the trajectories corresponding to each state.

Each trajectory defined in the Inspection System module is calculated by the Way-

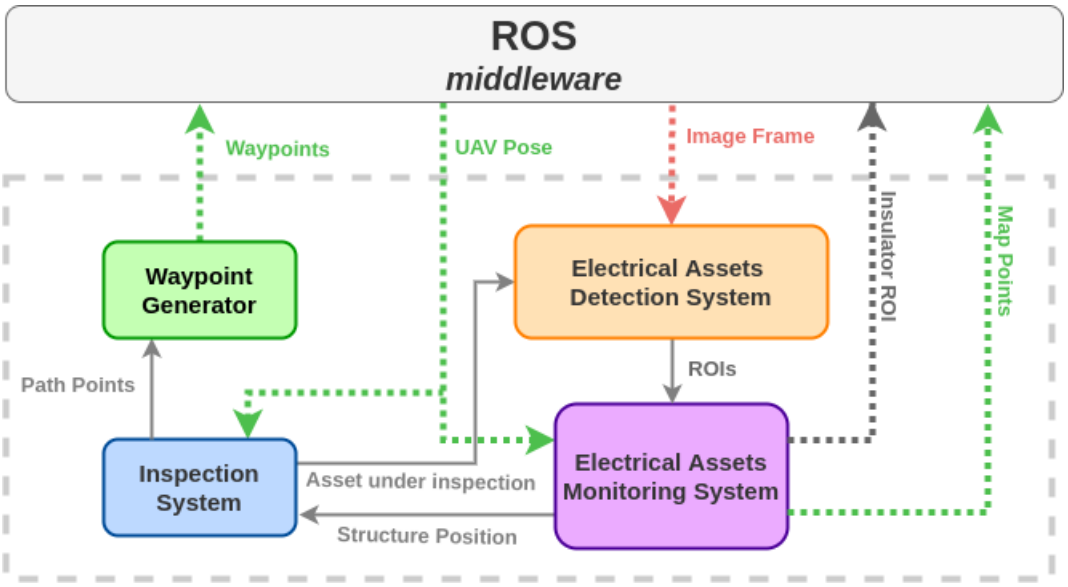


Figure 4.1: High-level architecture of the proposed system.

point Generator module by generating a set of points that define a path between the start and end points of each trajectory. Each of these points is then published in a ROS topic with a message that corresponds to the UAV pose planned at that point on the path, and which must be overwritten by the external UAV control system that must move the UAV through that path.

While the UAV performs each trajectory, the Electric Asset Detection System, which uses a Convolutional Neural Network to detect insulators and energy transmission structures, runs in parallel. This system receives each image from a monocular camera and processes it using a low-cost portable device, called Movidius Neural Compute Stick, which provides the location and type of the different electrical assets in each image. These detections, together with the UAV pose, feed the Electric Asset Monitoring system, which aims to track, map and collect samples of the different assets that appear in the field of view of the UAV camera. During the inspection process, this module publishes on a topic a message with the regions of the image that contain a sample of an asset whose position has been estimated in the global framework. At the end of the inspection process, the map of estimated electrical assets is also published in a topic that contains the position in the global reference of each asset.

The programming language used was C++, always considering a reusable, modular code structure and good programming practices. For solutions supported by the vision

system, the OpenCV library was used, and for linear algebra operations, it was used the Eigen library, due to its versatility and processing speed on different platform's processing units. As previously mentioned, the detection system uses a Movidius Neural Compute Stick to process the CNN corresponding to the electrical asset detector. In order to integrate this functionality, the OpenVino toolkit was used to generate, after optimization, and to process a set of CNNs compatible with Movidius NCS.

This page was intentionally left blank.

Chapter 5

Deep Learning based Electrical Assets Detection System

In this chapter it will be presented the proposed perception approach to detect the electrical assets of interest. This detection system is capable of detect insulators and structures (poles and pylons) from images captured through a UAV, based on lightweight Convolutional Neural Networks and it is able to run on a portable device, aiming for a low cost, accurate and modular system, capable of running in real time.

5.1 Electrical Assets Detection System High-level Architecture

The proposed detection system uses as its core the OpenVINO toolkit, which provides two APIs: the Inference Engine and the Model Optimizer. The Inference Engine, takes as input the image to process and outputs the regions proposals represented by the detected bounding boxes and confidences. This module can be configurable in the initialization stage, where it is possible to get the network optimized model, created by the Model Optimizer API, via its Intermediate Representation, and the Movidius NCS as processing platform. At processing stage, the Inference Engine is fed with the image resized to the size defined by the network, sends the input data to Movidius NCS and receives the detection proposals as an inference response.

At the end, each detection proposal is filtered by its confidence using a certain confidence threshold, being then converted to a Region of Interest (ROI) representation. Each ROI is represented by the class and bounding box of the detected object, and a keypoint. The keypoint for insulators is the center of the detected bounding box, while

for structures is the top-center point the bounding box.

In Figure 5.1, it is depicted the high-level architecture of the system.

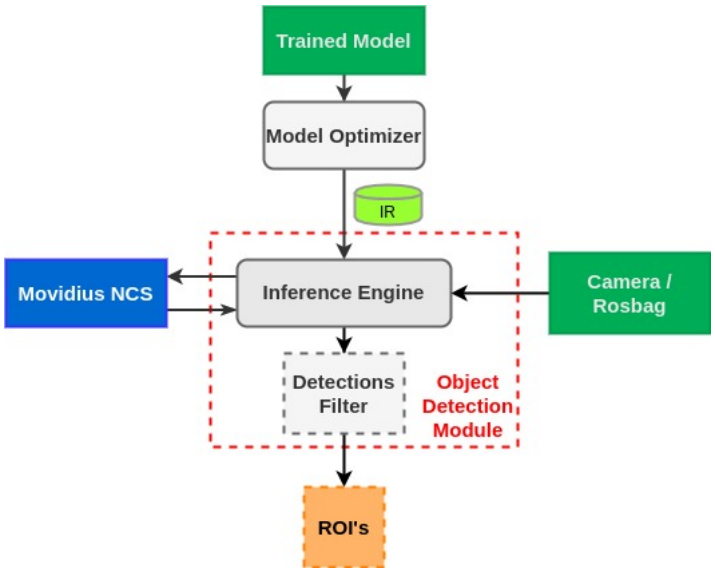


Figure 5.1: Electrical Assets Detection System High-level architecture.

5.2 Dataset and Data Augmentation

To train the networks a dataset was created with 585 images gathered from the INESC TEC's UAV STORK I, Figure 5.2, on its multiple inspection missions. This hexacopter UAV has been developed and used for applications such as search and rescue operations, environmental monitoring, 3D mapping, inspection, and surveillance and patrol.

The initial dataset consists in multiple samples of pylons, unity poles and different types of insulators with different resolutions and, for each of these samples, an annotation file in PASCAL VOC format containing the location of the object and its class, structure or insulator, was created using the graphical image annotation tool, LabelImg[1]. In Fig. 5.3, it's possible to see some examples of these images.

The main idea behind the dataset creation is to represent the objects of interest, referred as classes, in a representative and distinctive way, taking in account the multiple possible shapes and poses, the different conditions where they can appear and the balance between the number of samples for each class.

[1] `https://github.com/tzutalin/labelImg`

Figure 5.2: UAV STORK I

Figure 5.3: Example of images collected by the UAV during the visual inspection.

The first version of the dataset did not have much of these features, and therefore, it was not sufficient, neither quantitatively nor qualitatively, to train the networks and for that reason, a process of data augmentation was applied. This process, besides increasing the number of examples per class, allows the reduction of overfitting and improves the immunity to some conditions of the environment, such as fog, blur, noise and scale variation, and consists in the following augmentation techniques:

- **Rescale**: 300x300 and 512x512;
- **Rotation**: 35 degree steps clock-wise;
- **Hue and Saturation**: Change of hue and saturation components;
- **Blur**: Two different intensities of blur;
- **Contrast Normalization**: Normalization of the image contrast;
- **Fog**: Fog simulation;

- **Gaussian Noise**: Two different intensities of Gaussian Noise;
- **Salt and pepper**: Random black and white pixels;
- **Elastic Transformation**: Image quality reduction;

The final dataset contained a total of 15795 images, which resulted from the augmentation transformations to each image and respective annotation file. In Fig. 5.4, it's depicted an example of the data augmentation process applied to an image of the original dataset.

Finally, this dataset was randomly divided into two sets, 70% of the images for the training set and the remaining 30% for test/validation set.

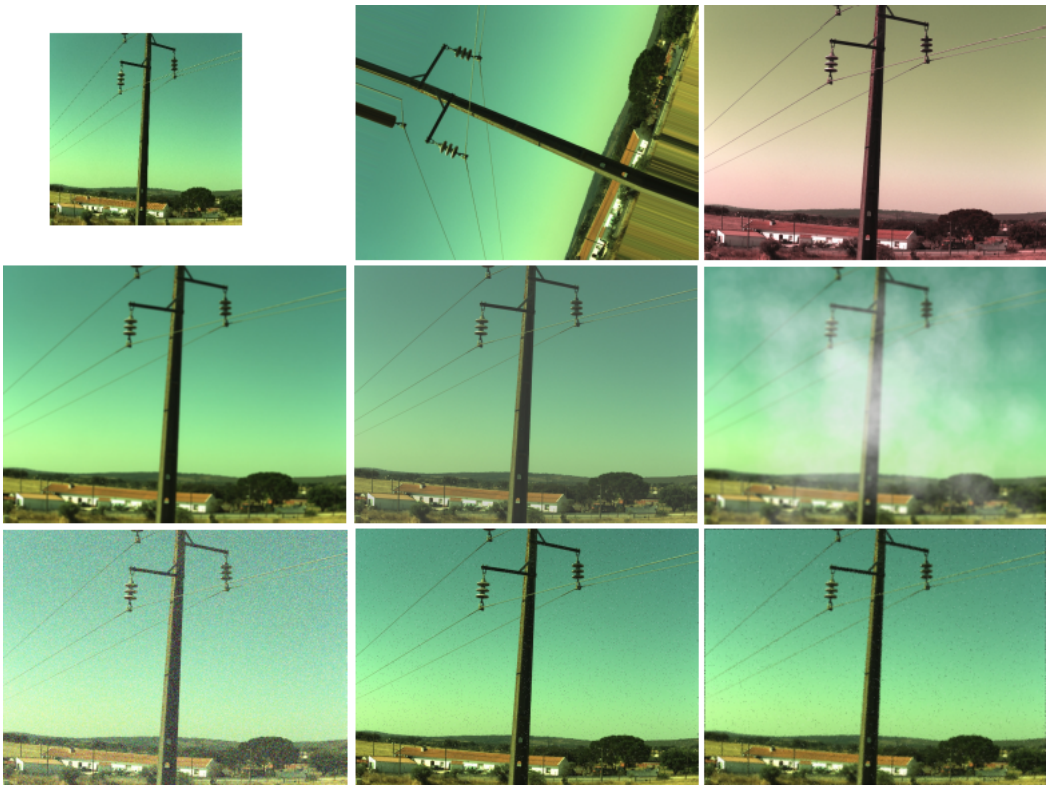


Figure 5.4: Example of the transformations applied in the data augmentation process

5.3 Movidius™Neural Compute Stick and OpenVINO™toolkit

The Movidius Neural Compute Stick (NCS), Figure 5.5, is a low-cost and low-power USB device based on Myriad 2 Vision Processing Unit (VPU). This device allows rapid prototyping, validation, and deployment of deep neural network inference applications at the edge.

To take advantage of Movidius™, it has been used the toolkit OpenVINO™. The OpenVINO™toolkit provides the ability of CNN-based deep learning inference and helps further unlock cost-effective, real-time vision applications. OpenVINO™supports heterogeneous execution across computer vision accelerators, CPU, GPU, Intel® Movidius™Neural Compute Stick, an FPGA, using a common Application Programming Interface (API) and supports models in popular formats such as Caffe, Tensorflow, MXNet, and ONNX.

This toolkit includes two components, named Model Optimizer and Inference Engine. The Model Optimizer is a cross-platform command-line tool that facilitates the transition between the training and deployment environment, performs static model analysis, and adjusts deep learning models for optimal execution on end-point target devices. Model Optimizer produces an Intermediate Representation (IR) of the network as output. The Inference Engine is a C++ library with a set of C++ classes to infer input data (images) and get a result. This library provides an API to read the Intermediate Representation, set the input and output formats, and execute the model on devices.

5.4 Lightweight Object Detection Convolutional Neural Networks

The Movidius™ Neural Compute Stick and OpenVINO ™ toolkit suite does not offer compatibility for all CNN architectures, or even all layers, mainly due to the processing power of Movidius™ NCS. As such, it was necessary to evaluate among a set of CNN architectures for object detection which of these would be able to be integrated into this system. This choice included the evaluation of the networks as to their speed of inference, their performance relative to the precision and size and type of blocks of the architecture itself. Thus, it was decided to choose object detection CNNs that can be classified as SSD based or Yolo based, due to the type of detector. This type of detectors have been properly explained in 3.1.3.1 and 3.1.3.2.

Figure 5.5: Movidius Neural Compute Stick

5.4.1 SSD-based Models

As mentioned previously in Section 3.1.3.1, SSD uses a CNN for image classification as base network only to extract features in the image at the early stages which constitute an important factor for the performance of this type of detectors. Among the image classification networks compatible with Movidius NCS are MobileNet-V1, MobileNet-V2 and PeleeNet, which have already been covered in Section 3.1.2.1. The choice of these networks was also due to the impact that these networks have on state-of-the-art detection systems as they present a good precision-speed trade-off within lightweight networks. These approaches will be from now on referred as MobileNetV1-SSD, MobileNetV2-SSD and PeleeNet-SSD.

In order to generate the trained models to deploy these networks, the open-source framework Caffe[2] [44] was used, which presents fast performances for CNN for images and compatibility with SSD layers.

This process was divided in four steps: dataset conversion to Lightning Memory-mapped Database (LMDB) format, generation of model configuration files, the training process and the intermediate representation generation using the OpenVINO's Model Optimizer API. This workflow can be seen in Figure 5.6.

To obtain the best performance when reading the dataset information, Caffe provides the capability to use the LMDB database format due to its fast access to disk content functionality. Since the created dataset is in PASCAL VOC format, this framework already contains a helper script that allows to convert from this format to LMDB format.

The second step is the creation of the model configuration files, for each network, describing the networks structure, with the definition of the different layers and respective parameters, as mentioned in Section 3.1.2. There are a set of hyperparameters that can be configured in this file that influence the training performance, such as the number of classes (two in this work), the input size (300x300) and the batch size, which is the

[2] `https://github.com/weiliu89/caffe/tree/ssd`

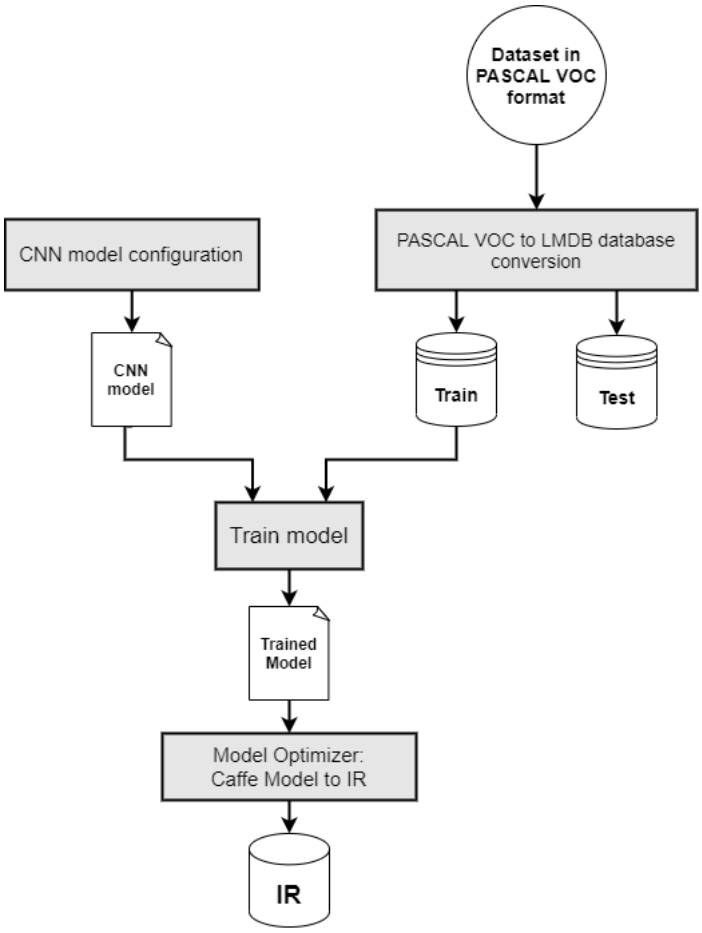


Figure 5.6: Workflow to train a Caffe CNN and generate an OpenVINO's Intermediate Representation

number of dataset samples processed before the model (weights and biases) being updated, and hence, depends on the computational resources available, because the higher the batch size more memory space is required.

The third step is the training of the CNN models using the files from the two previous steps. Caffe provides a set of bash scripts for training or test which require the configuration of the input files, the network configuration files and the definition of some hyperparameters, such as:

- **Learning rate:** Controls how much adjusting the network weights requires with respect to the loss gradient.

- **Optimizer:** Stochastic gradient descent. This method addresses the general optimization problem of loss minimization

After running the training script, the training API starts the presentation of the training loss per iteration. This information provides the current state of the training process and it is used as a criteria to validate when this process should stop. A simple criteria consists in evaluation of this loss which after a large number of iterations its value does not longer decrease, stabilizing around a certain value. In Figures 5.7, 5.8 and 5.9 it is possible to verify the evolution of the training loss per iteration for MobileNetV1-SSD, MobileNetV2-SSD and PeleeNet-SSD, respectively. MobileNetV1-SSD took two days and twenty three hours and 60000 iterations to reach the loss stabilization, MobileNetV2-SSD took four days and twenty hours and 120000 iterations and PeleeNet-SSD took one day and twenty two hours and 70000 iterations.

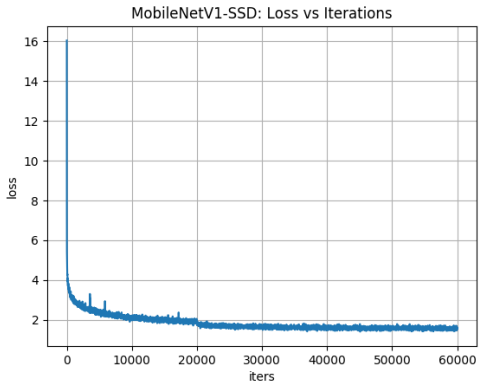


Figure 5.7: Training loss over iterations for MobileNetV1-SSD

Finally, the forth and final step consists in the generation of the intermediate representation of each trained model using the Model Optimizer API. As mentioned previously, this API already provides compatibility with Caffe models which allows a direct conversion to the IR model to be inferred by the Movidius NCS.

5.4.2 YOLO-based Models

In the section 3.1.3.2, it was presented the most recent versions of YOLO family, the Yolov3 and tiny-Yolov3 networks. Their authors developed a framework called Darknet[3] [45] written in C language, allowing a good performance and support for GPU

[3] `https://github.com/pjreddie/darknet`

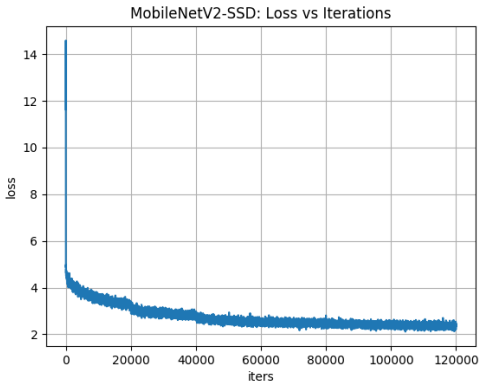


Figure 5.8: Training loss over iterations for MobileNetV2-SSD

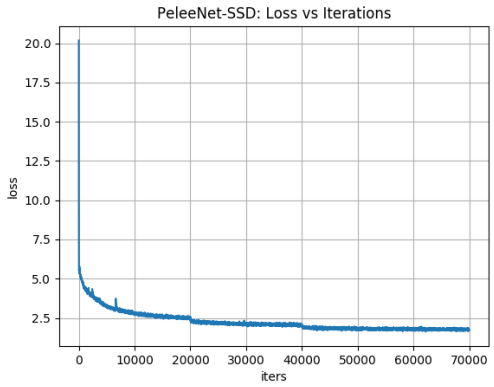


Figure 5.9: Training loss over iterations for PeleeNet-SSD

computation and therefore has been used to train these networks.

This process took five steps to generate the Intermediate Representation to be deployed in the detection system: the dataset conversion to Darknet format, network structure file configuration, training and validation process, conversion from Darknet trained model to Tensorflow model and finally, the Intermediate Representation generation using the Model Optimizer. In Figure 5.10 it is possible to see the workflow of this process.

First it was necessary to prepare the dataset to be compatible with this framework. The way this framework loads the dataset is more straightforward than in Caffe, since the training set is read in the original format, images and annotation files. However, Darknet uses a different format of annotation files, so the first task was to perform the

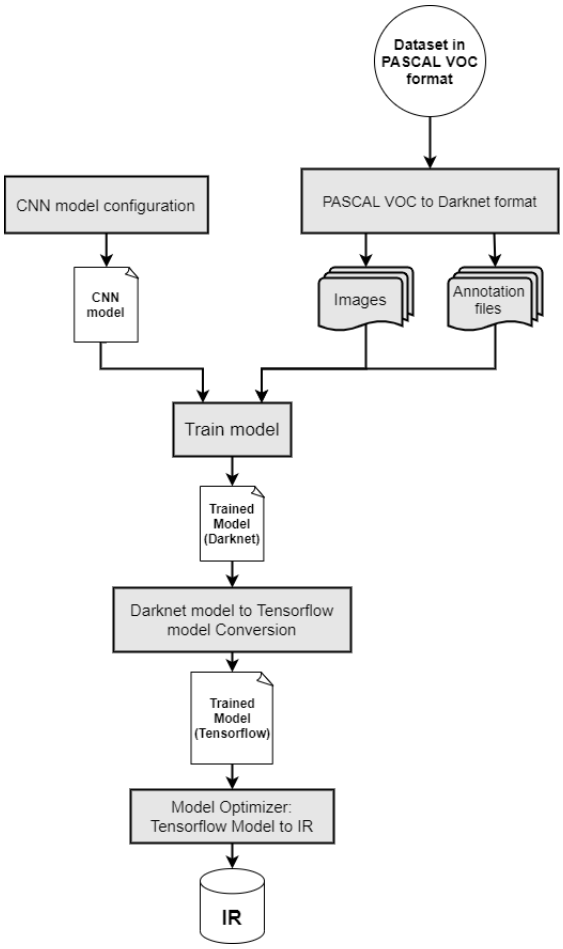


Figure 5.10: Workflow to train a Darknet CNN and generate an OpenVINO's Intermediate Representation

conversion from the PASCAL VOC format to Darknet format.

Each network is represented by a configuration file that contains the structure of the network and its hyperparameters. In Darknet implementation, the configuration files were already provided being required to change the networks input size for 416×416, which has a major impact on the computational resources required for the training phase and on the network performance, and the number of classes and the number of filters used in the last convolutional layer that depends in the number of classes. Besides this two hyperparameters, it is also necessary to adapt the batch size and the learning rate, like in Caffe approach, for the computational resources available where the training will be performed.

After the configuration it was possible to start the training process. The training process works in the same way as presented for the SSD-based networks and the same validation technique was used. In Figures 5.11 and 5.12 it is possible to verify the training loss evolution over iteration for YoloV3 and tiny-Yolov3, respectively. The YoloV3 took two days and one hour and 20000 iterations to reach loss stability, while tiny-Yolov3 training last two days and sixteen hours and 75000 iterations.

Since OpenVINO does not offer compatibility for Darknet models, it was performed a conversion to a Tensorflow model using a YoloV3 implementaion in Tensorflow[4]. Finally, the Tensorflow weights model is then converted to the corresponding Intermediate Representation.

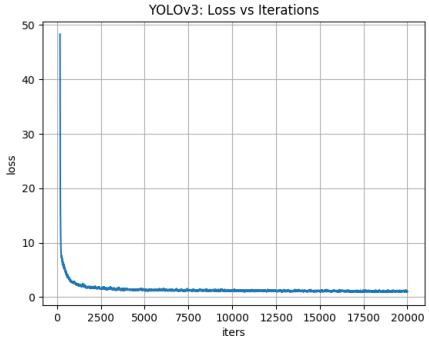


Figure 5.11: Training loss over iterations for YOLOv3

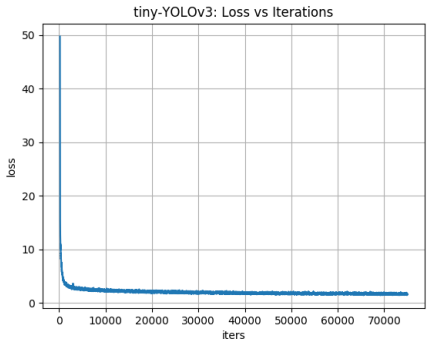


Figure 5.12: Training loss over iterations for tiny-YOLOv3

[4]`https://github.com/mystic123/tensorflow-yolo-v3`

This page was intentionally left blank.

Chapter 6

Electrical Assets Monitoring System

In the previous chapter, the system that allows the detection of the electrical assets that are considered most relevant to the inspection process was presented. However, during this process, it becomes important to associate a certain detection to its respective electrical asset, in order to gather multiple samples of that sample. For such, the Electrical Assets Monitoring System was developed, whose purpose is to get samples of electrical assets during the inspection process.

6.1 Electrical Assets Monitoring Algorithm Overview

The Electrical Asset Monitoring Algorithm is composed of three interconnected components: the Multi-Object Tracker, the Depth Estimator and the Global Map. Having as input the current camera pose and the ROIs coming from the Electrical Asset Detection System, the proposed algorithm aims to estimate the electrical assets location in the image frame and in the global frame. The Figure 6.1 shows the architecture of the Electrical Assets Monitoring System.

The main idea of the Multi-Object Tracker is to estimate the position and dimensions of the electrical asset, in each image, using a Kalman Filter. This Kalman Filter instance of the electrical asset, hereinafter called track, is then associated with the respective ROI proposal. If no association occurs, a new track is created, which means that the system may have detected a new potential electrical asset. Each track will be used to represent an electrical asset in the image frame, being described by the keypoint position and bounding box dimensions.

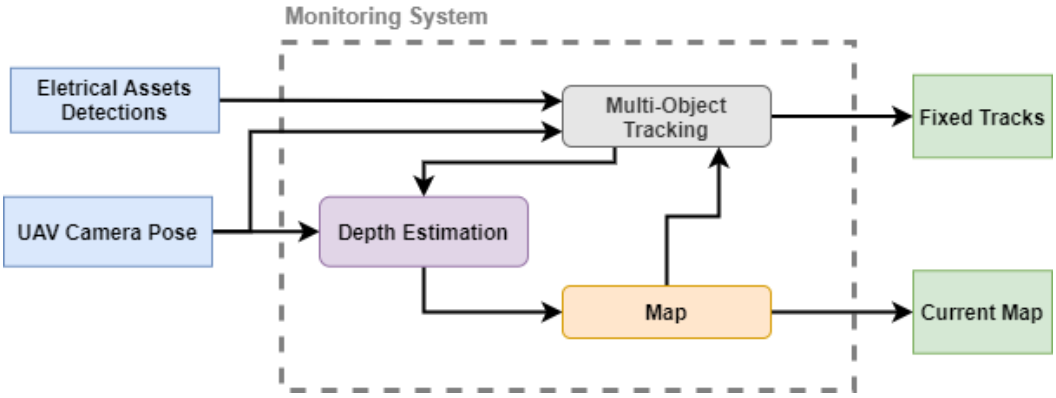


Figure 6.1: High-level Architecture of Electrical Assets Monitoring Algorithm

The Depth Estimator uses a probabilistic filter to estimate the distance from a reference camera location to a certain electrical asset, also called depth, from which it is possible to estimate, in the global reference frame, the electrical asset's: height, width, area and 3D position of an reference point. Each electrical asset will be represented, in global frame, by the tangent plane seen from the camera, correspondent to the projection of the respective track to the global frame. Thus, the 3D reference point will be the track's keypoint projected to global frame coordinates. Each reference point has an associated probabilistic filter, and as long as it does not converge, that point will be called as point candidate.

The Global Map is the structure that contains the 3D points corresponding to the position of each electrical asset estimated by the probabilistic depth filter. After the moment that a point is added to this structure, the association, performed in the Multi-Object Tracker, will depend on its position. A track whose reference 3D Point is present in Global Map will be called as fixed track.

This system provides the current Global Map and the fixed tracks that are in the camera's field of view.

6.2 Multi-Object Tracker

The Multi-Object Tracker formulation used in this project is based on the 2D multi object tracker, SORT [46], since it presented a good trade-off between precision and processing speed as evidenced by its results, among other online trackers, in the MOTChallenge 2016 [47], a yearly competition used to benchmark multiple object tracking models.

As mentioned in Subsection 6.1, the Multi-Object Tracker key idea is to follow each

electrical asset by modelling its movement and dimensions across a sequence of images using a Kalman Filter formulation.

For each image, the following happens:

1. The detected ROIs that are located inside an image border equal to **border_size** are removed.
2. For each track, its state is predicted in a Kalman Filter Predict step.
3. Fixed tracks that re-appeared in the current image are set to active.
4. Fixed tracks that have left the camera's field of view are set to inactive.
5. The active tracks are associated to the respective ROI detection.
6. Each associated track and the respective candidate point are updated, in a Kalman Filter Update step and in the Depth Estimator. A **missedAssociations** counter is set to zero and the **Associations** counter is incremented.
7. For each track without association, the **missedAssociations** counter is incremented and the **Associations** counter is set to zero.
8. For each detection without track association, a new Kalman filter track is created, initialized and set to active.
9. Each non fixed track with **missedAssociations** greater than ***maxAge*** is deleted.
10. When a new candidate to point probabilistic filter converged, if there is a point already in map at less than 1.5 meters from its, the candidate is considered outlier and is deleted.

In the Algorithm 1 it is possible to see the designed algorithm with the Multi-Object tracker procedure.

6.2.1 Kalman Filter

In order to estimate the tracks position and dimensions in pixel coordinates on every image, it was used a Kalman Filter formulation, whose theory is presented in detail in the Section 3.2.1.1.

A track is represented by the state space vector **x**, and is defined by:

$$\hat{x} = \left[x, y, A, ar, v_x, v_y, v_A, v_{ar}\right] \tag{6.1}$$

Algorithm 1 Multi Object Tracker algorithm

```
Filter detections near image borders
foreach track in tracks do
    PredictKalmanTrack(track)
end
Get tracks that re-appeared in the current frame
Inactivate track outside current frame
Associate detections and tracks based on cost function
foreach track in active tracks do
    if track has detection associated then
        UpdateDepthEstimator(CameraPose, detection)
        if Map has track 3D point then
            Project 3D point to image plane
            UpdateKalmanTrack(track, P3DOnimage)
        else
            if DepthEstimator converged then
                Add candidate to Map and set track as fixed
            end
        end
        UpdateTrack(track, detection)
    else
        if Map has track 3D point then
            Project 3D point to image plane
            if 3D point projection is in current frame then
                UpdateKalmanTrack(track, P3DOnimage)
            end
        end
    end
end
foreach detection in unmatchedDetections do
    Initialize new track with detection
end
Tracks Filter
Map outliers Removal
return Tracks with 3D Point in Map
```

Where $[\boldsymbol{x}, \boldsymbol{y}]$ are the keypoint's position in the image, $\boldsymbol{A}$ and $\boldsymbol{ar}$ are, respectively, the area and aspect ratio ($\frac{width}{height}$) of the bounding box that delimits the region of the image that contains a possible electrical asset, $[\boldsymbol{v_x}, \boldsymbol{v_y}]$ are the velocities of the track keypoint, $\boldsymbol{v_A}$ is the bounding box's change in area over time and $\boldsymbol{v_{ar}}$ is the bounding box's change of the aspect ratio over time.

The filter initialisation can be given by the information contained in the first ROI measurement, but setting the velocities to zero. The initial state $\hat{x}_0$ can be written as:

$$\hat{x}_0 = \Big[x_i, y_i, A, ar, 0, 0, 0, 0\Big] \tag{6.2}$$

In Predict step, the previous state and covariance need to be propagated to the current state. The prediction of the new state, that relies on Equation 3.9, consists in the usage of the track motion model in the 2D image frame represented by the transition matrix F_k. Since each electrical asset is static and the movement of the UAV, during the inspection process, must have constant velocity, a track can be modeled as a constant velocity model, and therefore, F_k can be written as:

$$F_k = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

The computation of the new covariance, as we can see in 3.10, requires a Process Noise matrix, represented as Q_k, given by:

$$Q_k = \begin{bmatrix} \sigma_x & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_A & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{ar} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{v_x} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{v_y} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{v_A} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{v_{ar}} \end{bmatrix} \tag{6.4}$$

Where each value in the matrix diagonal corresponds to the uncertainty of each state.

Whenever a new ROI detection arrives and the association with an existing track occurs, the predicted state can be updated, in the Update step. As explained by the equations 3.12 and 3.13, the update makes use of the matrices H_k, the measurement matrix, for state space update and R_k, the measurement noise matrix, for covariance update. In this system, it is considered two types of sensors: the Electrical Assets Detection System and the virtual sensor corresponding to the projection of the track 3D point in Global Map.

Thus, an Electrical Assets Detection observation is represented by:

$$z_1 = \begin{bmatrix} x_1, y_1, A_1, ar_1 \end{bmatrix} \tag{6.5}$$

Where x_1 and y_1 are the coordinates, in the image frame, of the detection ROI keypoint, A_1 is the area of the detection ROI bounding box and ar_1 is the aspect ratio of the detection ROI bounding box. The corresponding measurement matrix H_1 and measurement noise matrix are given, respectively, by:

$$H_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.6}$$

$$R_1 = \begin{bmatrix} \sigma_{x_1} & 0 & 0 & 0 \\ 0 & \sigma_{y_1} & 0 & 0 \\ 0 & 0 & \sigma_{A_1} & 0 \\ 0 & 0 & 0 & \sigma_{ar_1} \end{bmatrix} \tag{6.7}$$

In the Subsection 6.1, it was explained that from the estimated depth it is possible to estimate the electrical asset reference 3D location and the approximated dimensions, width and height, and consequently its area in global reference frame. All of this data can be projected to the current frame, which makes the observation, z_2, equal to the previous sensor:

$$z_2 = \begin{bmatrix} x_2, y_2, A_2, ar_2 \end{bmatrix} \tag{6.8}$$

where x_2 and y_2 are the projection of the 3D world point onto the image in pixel coordinates, A_2 is its projected area and ar_2 the projected aspect ratio. The corresponding measurement matrix H_2 and measurement noise matrix, R_2 are given, respectively, by:

$$H_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.9}$$

$$R_2 = \begin{bmatrix} \sigma_{x_2} & 0 & 0 & 0 \\ 0 & \sigma_{y_2} & 0 & 0 \\ 0 & 0 & \sigma_{A_2} & 0 \\ 0 & 0 & 0 & \sigma_{ar_2} \end{bmatrix} \tag{6.10}$$

Since there are two types of measurements, a data fusion method is required. To this end, it is proposed to use a multi-sensor fusion architecture applying the Sequential-Sensor Update method [48] in the Update step due to its capacity to easily deal with asynchronous measurements. This approach considers each sensor measurement as an independent, sequential update to the states estimate, as it is possible to see in Figure 6.2. In the Algorithm 2 its possible to observe the procedure followed by Sequential-Sensor Update.

Figure 6.2: Sequential-Sensor Update method

Algorithm 2 SEQUENTIAL-SENSOR UPDATE

Input: $\hat{x}_{k|k-1}$, $P_{K|k-1}$, z_1, z_2

$\hat{x}_{k|k,0} = \hat{x}_{k|k-1}$

$P_{k|k,0} = P_{k|k-1}$

for $p \leftarrow 1$ *to* N **do**

$S_{k,p} = H_{k,p} \cdot P_{k|k,p-1} \cdot H_{k,p}^T + R_{k,p}$

$K_{k,p} = P_{k|k,p-1} \cdot H_{k,p}^T \cdot S_{k,p}^{-1}$

$v_{k,p} = z_{k,p} - H_{k,p} \cdot \hat{x}_{k|k,p-1}$

$\hat{x}_{k|k,p} = \hat{x}_{k|k,p-1} + K_{k,p} \cdot v_{k,p}$

$P_{k|k,p} = P_{k|k,p-1} - K_{k,p} \cdot S_{k,p} \cdot K_{k,p}^T$

end

return $\hat{x}_{k|k,p}$, $P_{k|k,p}$

6.2.2 Association

The goal in the association step is to find an optimal match between the estimated tracks and detections. The first step when finding an optimal match is generating the cost matrix. This is a matrix where each element $c_{i,j}$ is the result of the application of a cost function between the track j and detection i, which indicates the level of similarity between them. Given the cost matrix, the Hungarian algorithm can find the optimal match between predictors and detections. It is, however, beneficial to have a threshold value used to reject associations if the cost is too high or too low.

It is also considered the situation where for the same detection has similar cost other different tracks and an association was found for at least one of these tracks. When this happens, these tracks are all rejected but not removed. In the algorithm 3 it is possible to verify in detail how associations are performed and how they are handled.

Algorithm 3 Association Algorithm

```
Input: tracks, detections
foreach track in tracks do
    foreach detection in detections do
        ComputeCost(detection, track)
        Add Cost to Matrix
    end
end
associations_raw = HungarianAlgorithm()
foreach association_raw in associations_raw do
    if association cost ≤ Threshold then
        association(det,track) = true;
    end
end
Check close tracks for same detection and reject both
return associations, unmatchedDetections;
```

Similarity Cost function Since we have two types of sensors and initially only the Electrical Assets Detection System provides data, it was considered two different functions to quantify the match between the measure and the track. These functions are based on the [49], which consists in combined the measured the difference in position 2D, shape or Position 3D, exponentially.

Let A be the detected ROI, B a track and C the position 3D of the track B in the Global Map. The cost function for the detections only sensor is represented by $Cost(A, B)$, which consists in add the position cost, $c_{pos}(A, B)$ multiplied by the weight

k_1, with the shape cost, $c_{shape}(A, B)$, multiplied by the weight $(1-k_1)$. The cost function for the detections + Global Map sensor is represented by $Cost(A, B, C)$, which consists in add the 2D position cost, $c_{pos}(A, B)$ multiplied by the weight k_1, with the 3D position cost, $c_{3D}(A, C)$, multiplied by the weight $(1 - k_1)$. Both cost functions are given by:

1. **Electrical Assets Detection System only:**

$$Cost(A, B) = k_1 \cdot c_{pos}(A, B) + (1 - k_1) \cdot c_{shape}(A, B) \tag{6.11}$$

$$c_{pos}(A, B) = 1 - e^{-k_2 \cdot \left(\frac{X_A - X_B}{W_A}\right)^2 + \left(\frac{Y_A - Y_B}{H_A}\right)^2} \tag{6.12}$$

$$c_{shape}(A, B) = 1 - e^{-k_3 \cdot (\frac{|H_A - H_B|}{H_A + H_B} + \frac{|W_A - W_B|}{W_A + W_B})} \tag{6.13}$$

2. **Electrical Assets Detection System + Global Map:**

$$Cost(A, B, C) = k_1 \cdot c_{pos}(A, B) + (1 - k_1) \cdot c_{3D}(A, C) \tag{6.14}$$

$$c_{pos}(A, B) = 1 - e^{-k_2 \cdot \left(\frac{X_A - X_B}{W_A}\right)^2 + \left(\frac{Y_A - Y_B}{H_A}\right)^2} \tag{6.15}$$

$$c_{3D}(A, C) = 1 - e^{-k_3 ||P3D_A - C||} \tag{6.16}$$

With,

$$P3D_A = Pose^W_{cur} \cdot (depth_{cur} \cdot K^{-1} \cdot [X_A, Y_A, 1.0]^T) \tag{6.17}$$

$$depth_{cur} = ||t^W_{cur} - C|| \tag{6.18}$$

Where, (X, Y) is the keypoint position, of either A or B, depending on the subscript. W and H are the width and height, respectively of the bounding box of A or B. $Pose^W_{cur}$ is the current pose of camera and t^W_{cur} its respective translation component, in the global reference frame, K is the Intrinsics matrix, $[X_A, Y_A, 1.0]^T$ is the homogeneous vector of the keypoint position of A, $depth_{cur}$ is the euclidean distance from t^W_{cur} to the point C, and finally, $P3D_A$ is the keypoint position of A projected to the global frame, by considering that have the same depth as C from the current camera position.

6.3 Multi-view Depth Estimator

As referred in the Subsection 6.1, the Depth Estimator aims to estimate the depth from reference point of an electrical asset to a reference camera pose, using a probabilistic filter. This module relies on the method presented in Section 3.4, whose idea is to update the posterior depth distribution every frame, using the probabilistic depth filter, proposed in the Subsection 3.4.2, by taking depth measurements provided by the triangulation method, pointed in subsection 3.4.1. From the estimated depth it is possible to get an approximation of the electrical asset's 3D reference point, height, width and area in the global reference frame. Since the two types of objects of interest are static in the world frame, the transformation of their 3D location to the image frame allows a long term tracking of each object making the tracks immune to scale variation, occlusions and temporary disappearance from the field of view of the camera.

6.3.1 Update Depth Estimator

In the Algorithm 4 it is described the process followed to estimate the candidate, which corresponds to *UpdateDepthEstimator* in the Algorithm 1.

In *UpdateDepthEstimator* method, after create a new track, the filter is only initialized as soon as five detections are correctly associated, with the current detection, $Detection_{ref}^{img}$, and camera pose, $Pose_{ref}^{C}$, which will be used as reference in the triangulation process. It is important to refer that $Detection_{ref}^{img}$ is the raw ROI provided by the detection system, so it has the keypoint, bounding box and the class of the detection. Then, the inlier ($\boldsymbol{a}$) and oultier ($\boldsymbol{b}$) counters are set to 10, since, as referred in [38], this values correspond to a prior for the inlier ratio centered on 0.5. Another value that was defined is the depth range, **ZRange**, that will be used to model the outliers in the distribution, which depend on the type of detection. If the detection is an insulator the value as set to 40 m, and for structures was defined to 90 m. The same occurs for minimum depth value, **minDepth**, because it also depends on the type of detection. This value was set to 5 m for insulator, and 25 meters for structures.

From this moment on, for each track association, the respective detection and current UAV's camera pose are added to the filter. The filter update, described in the Algorithm 5, consists in use triangulation method presented in 3.4.1 to obtain a measure of the 3D point that is seen by both reference camera pose and current pose, by taking the current detection keypoint projected to the camera frame, $Detection_{cur}^{C}$, which is given by:

$$Detection_{cur}^{C} = K^{-1} \cdot Detection_{cur}^{img} \tag{6.19}$$

Algorithm 4 UpdateDepthEstimator

Input: currentCameraPose, currentDetection, track

```
if Associations ≥ 5 then
    if Candidate not initialized then
        InitializeCandidate(currentDetection, currentCameraPose)
    else
        UpdateCandidateDepthFilter(currentDetection, currentCameraPose)
    end
end
```

and, the keypoint of the reference detection, also projected to the camera frame, $Detection^{C}_{cur}$. After obtaining the 3D point measurement, in global frame, it is computed the distance to it from the current camera position in the global frame, obtaining a depth measurement, $dept_{est}$.

With a new depth measurement, it is necessary to verify its value is greater than the minimum depth, and, if not, the measurement is discarded. If that condition is true, a new validation step is performed, by verifying if the parallax angle is greater than ***threshAngle***, in degrees. The parallax angle is the angle, α, between the vector from current camera pose position to the estimated 3D point, $\overrightarrow{t^{W}_{cur}X}$, and the vector from the reference camera pose position to the same 3D Point, $\overrightarrow{t^{W}_{ref}X}$, in global frame. The value of $\cos(\alpha)$ can be computed by applying the dot product of both vector, and can be obtained by the following equation:

$$\cos(\alpha) = \frac{\overrightarrow{t^{W}_{ref}X} \cdot \overrightarrow{t^{W}_{cur}X}}{||\overrightarrow{t^{W}_{ref}X}||\ ||\overrightarrow{t^{W}_{cur}X}||} \leq \cos\left(\mathbf{threshAngle}\right) \tag{6.20}$$

If the condition, $\cos(\alpha) \leq \cos\left(\mathbf{threshAngle}\right)$ is verified, it is possible to update the filter. The filter update consists in use an inverse-depth parameterization, for describing each estimation, being:

$$\rho = \frac{1}{depth_{est}} \tag{6.21}$$

The benefit of using the inverse-depth parameterization over euclidean coordinates is the possibility of represent points in infinity when $\rho = 0$

If the filter was not initialized yet, the mean value (μ) of the depth filter is set with $\rho_0 = \frac{1}{depth_{est0}}$, using the inverse of first depth provided by triangulation, that comply with the previous conditions. After initialization, the filter will be sequentially updated relying on the method presented in sub-section 3.4.2. In Algorithm 6, the

process to update the depth distribution parameters, called ***DepthFilterUpdateStep***, is presented.

Algorithm 5 UpdateCandidateDepthFilter

Input: $\text{Detection}_{cur}^{img}, Pose_{cur}^{C}$
$\text{Detection}_{cur}^{C} = K^{-1} \cdot Detection_{cur}^{img}$
$depth_{est} = \text{triangulate}(\text{Pose}_{ref}^{C}, Pose_{cur}^{C}, Detection_{ref}^{C}, Detection_{cur}^{C})$
if $depth_{est} \geq minDepth$ **then**
  $P3D_{est}^{W} = Pose_{ref}^{W} \cdot (depth_{est} \cdot Detection_{ref}^{C})$
  $cos(\alpha) = GetParallax(t_{ref}^{W}, t_{cur}^{W}, P3D_{est}^{W})$
  if $cos(\alpha) \leq cos(threshAngle)$ **then**
    if *Filter is not Initialized* **then**
      $\mu = depth_{est}$
      filterInitialized = true
    else
      DepthFilterUpdateStep()
    end
  end
end

The first step of ***DepthFilterUpdateStep*** is to compute the variance of the triangulation process by assuming one pixel of noise when projected to the image frame, whose schematic is presented in Figure 3.18, using the Equations to .

Having the triangulation variance, it is now possible to compute the new parameters μ_{new}, ${\sigma^2}_{new}$, a_{new} and b_{new} from Equations 3.66, 3.67, 3.68 and 3.69.

After obtaining the new filter parameters, it is verified whether the new inverse-depth estimation converged according to the convergence criteria proposed by [50]:

$$l = \frac{4 \cdot \sigma_{new}}{d \cdot \mu_{new}^2} \cdot |\cos\alpha| \tag{6.22}$$

where, α is the parallax angle from the estimated 3D point and the reference and the last observation, and d is the depth obtained from the current camera position.

Using this convergence criteria, when the parallax angle is low, l will be high whereas when the parallax angle increases, l is reduced. It is considered that the probabilistic depth filter converged when l is lower than a threshold.

As soon as there is convergence in the depth estimation, the 3D point of the track is calculated and immediately added to the Global Map of electrical assets found.

Algorithm 6 DepthFilterUpdateStep

Input: $\text{Detection}^{C}_{cur}, Pose^{C}_{cur}, \text{depth}_{est}$
$\text{Pose}^{ref}_{cur} = Pose^{C}_{cur}(Pose^{C}_{ref})^{-1}$
$\tau^2 = \text{ComputeTau}(\text{Pose}^{ref}_{cur}, Detection^{C}_{ref}, \text{depth}_{est})$
$\rho = \frac{1}{depth_{est}}$
$s^2 = \frac{(\tau^2 \times \sigma^2)}{(\tau^2 + \sigma^2)}$
$m = s^2 \times (\frac{\mu}{sigma^2} + \frac{\rho}{\tau^2})$
$C_1 = \frac{a}{(a+b)} \times pdfNormal(\mu, \sqrt{\tau^2 + \sigma^2}, \rho)$
$C_2 = \frac{b}{(a+b)} \cdot \frac{1}{ZRange}$
$norm = C_1 + C_2$
$C_1 = \frac{C_1}{norm}$
$C_2 = \frac{C_2}{norm}$
$d = \frac{C1 \times (a+1)}{(a+b+1)} + \frac{C2 \times a}{(a+b+1)}$
$e = \frac{C1 \times (a+1) \times (a+2)}{((a+b+1) \times (a+b+2))} + \frac{C2 \times a(a+1)}{((a+b+1) \times (a+b+2))}$

// Update parameters
$\mu_{new} = C_1 \times m + C_2 \times \mu$
$\sigma^2{}_{new} = C_1 \times (s^2 + m \times m) + C_2 \times (\sigma^2 + \mu \times \mu) - \mu_{new} \times \mu_{new}$
$\text{a}_{new} = \frac{(e-d)}{(d - \frac{e}{d})}$
$b_{new} = \frac{a*(1-d)}{d}$

if *ConvergenceCriteria()* **then**
  fixed = true
  $P3D_{fixed} = Pose^{W}_{ref} \cdot (\frac{1}{\mu_{new}} \cdot Detection^{C}_{ref})$
end
$\mu = \mu_{new}$
$\sigma^2 = \sigma^2{}_{new}$
$b = b_{new}$
$a = a_{new}$

6.3.2 Candidate Dimensions Estimation

The dimensions estimation consists in the width estimation, in global frame, when the track is a structure, and in the width, height, and hence, the area in image frame, when the track is an insulator. For this, it was assumed that every point of the electrical assets, represented by a surface normal, have the same distance to the UAV camera, being this distance equal to the estimated depth of a converged filter.

By taking the ROI of reference saved when the filter was initialized, the width (W), the height (H), in global frame, are respectively given by:

$$W = \frac{d_w \cdot \frac{1}{\mu}}{f_x} \tag{6.23}$$

$$H = \frac{d_h \cdot \frac{1}{\mu}}{f_y} \tag{6.24}$$

where d_w and d_h are, respectively, the width and height of the detection ROI of reference, $Detection_{ref}^{img}$, bounding box.

The area in the camera image of an insulator, with **W** width and **H** height can be computed as:

$$A = W \cdot H \cdot f_y \cdot f_x \cdot \mu^2 \tag{6.25}$$

Chapter 7

Waypoint Generation and Autonomous Inspection System

In this chapter, the Autonomous Inspection System proposed to perform an autonomous monitoring of the electrical assets using a UAV will be proposed. This system consists in detect and estimate the position and dimensions of a structure on a power transmission line, and for that structure, the insulators that belong to it and appear on the the camera field of view, are detected and mapped. In order to perform an optimized inspection process, a set of inspection states were defined, which consist in a predefined set of paths and/or tasks. The paths are generated by the proposed Waypoint Generator system, which computes a set of waypoints that will compose it, taking a initial and goal position. These waypoints are then sent to the UAV controller system that generates the trajectories that follow these waypoints. The controller doesn't belong to the scope of this work, so a UAV system with a consolidated and stable controller system is required.

7.1 Autonomous Inspection System

The Autonomous Inspection System consists in performing a set of maneuvers that allow the detection and localization of a structure belong to the transmission line and, for that structure, perform the inspection of its insulators. This system is defined by a state machine, which consists of five main states: the ***Init*** state, the ***Prepare Structure Inspection***, the ***Structure Mapping*** state, ***Go to Structure*** state and the ***Insulator Inspection*** state. The ***Insulator Inspection*** state is composed of two sub-states, ***Insulator Monitoring*** and ***End of Insulator Inspection*** Between each state there

is a transition which represent the conditions that allows the change from one state to another. In Figure 7.1, the proposed state machine can be observed and analyzed.

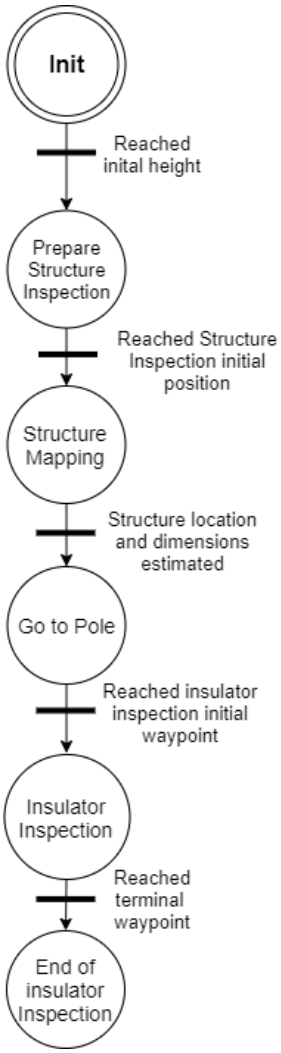


Figure 7.1: State Machine of Autonomous Inspection System

Each state is responsible to trigger a predefined path and/or a task that allow to accomplish the respective objective of that state. The predefined paths are usually defined by a rectilinear segment that connects two points, the initial and target point, or a set of these segments that connect a sequence of points. For each segment, a Waypoint Generator will generate a list of waypoints between the initial and target point, which will be compose by the position and orientation of the UAV. Throughout the process, the UAV will always be pointing to the position of the structure under inspection, whose

position should be known, as it will be referred to later. The set of predefined paths performed by the UAV in the structure inspection is represented in Figure 7.3, and the paths to insulators inspection are represented in Figure 7.4. Each numbered circle in both figures represent the initial and target point of each path segment and their respective order, from point (1) to point (21).

Before starting the entire process, the user must ensure that the following set of preconditions are complied:

- Specify the approximated location of the structures that are planned to be under inspection;
- Positioning the UAV on the ground, with a safety radius of at least 20 meters from the nearest line or structure;
- The user must provide the height that the UAV will have to rise from the ground where it was placed;
- Define the percentage of the height of each structure from where the UAV will have to descend to start the ***Insulator Inspection*** state;
- Define the length of the path segment that the UAV shall perform on each structure in ***Insulator Monitoring*** state;
- Supervise the process, stopping it in a case of failure or when the safety is compromised;

After ensuring that these conditions are met, the system can be started. The system starts in the ***Init state***, which consists in initialize a set of parameters by loading a configuration file provided by the user, where it is stored the information to fulfill the preconditions, the camera parameters, the object detection CNN to be used in the Electrical Assets Detection system, among others. Taking the current UAV's position (point (1) in Figure 7.3), correspondent to location where it was placed, and the ascending height defined by the user, the waypoint generator computes the target waypoint (point (2) in Figure 7.3) and the waypoints betweem them. The generated set of waypoints is then sent to the UAV's controller, which starts the UAV's navigation following each of them. When the target point is reached, the system starts the state ***Prepare Structure Inspection***.

The ***Prepare Structure Inspection*** state consists in move the UAV to a position where the angle described by UAV-Structure vector and the power line is 35°, with

a safety distance of 20 meters from the power line. The path that generated by the Waypoint Generator is represented by the initial point (2) and the target point (3). Reaching the target point, the system changes to ***Structure Mapping*** state.

The ***Structure Mapping*** state, consists of detecting and estimating the structure's dimensions and position of its upper-center point. The movement proposed for this state consists in, initially go from point (3) to (4), and then perform repeatedly lateral translations with the UAV pointing to the structure under inspection, corresponding to points (4) and (5) and vice-versa. For this, it was defined that the angle formed by the vector point (3)-Structure and the path (4) to (5) must be 90 °. In Figure 7.2 it is possible to visualize, with more detail, the representation of movements mentioned. While in the path (4) to (5), both Electrical Assets Detection System and Electrical Assets Monitoring System are running, after configuration for structure dealing, in order to be possible to estimate the structure's dimensions and top-center location. When Monitoring System finishes this estimation, the UAV stops its movement and transition to state ***Go to Structure*** occurs.

The ***Go to Structure*** state aims to move the UAV to the side of the structure where it is at. For such, it added a safety distance of 20 meters to the top-left and top-right points, corresponding of the points (A) or (B) of the Figure 7.4 depending on the side of the power line where the UAV is at. The planned path for this state is computed from point (6), which is the current position of the UAV (anywhere between the points (4) and (5)), to point (7), which can be either (A) or (B). After reaching the waypoint correspondent to point (7), the Inspection System will change to ***Insulator Inspection***.

The ***Insulator Inspection*** state consists in the acquisition of insulators samples

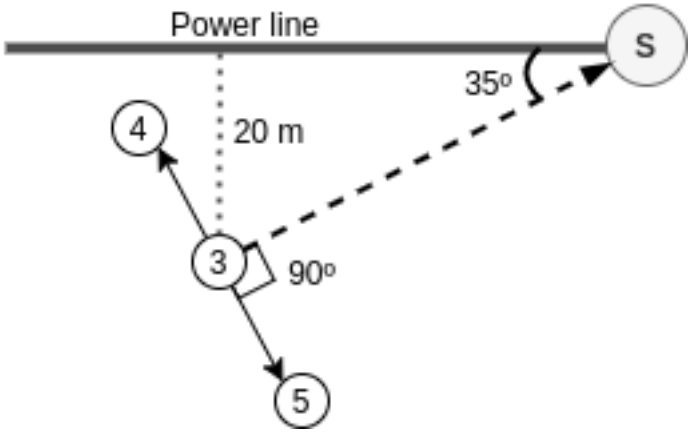


Figure 7.2: Top view of the *Structure Mapping* state movement

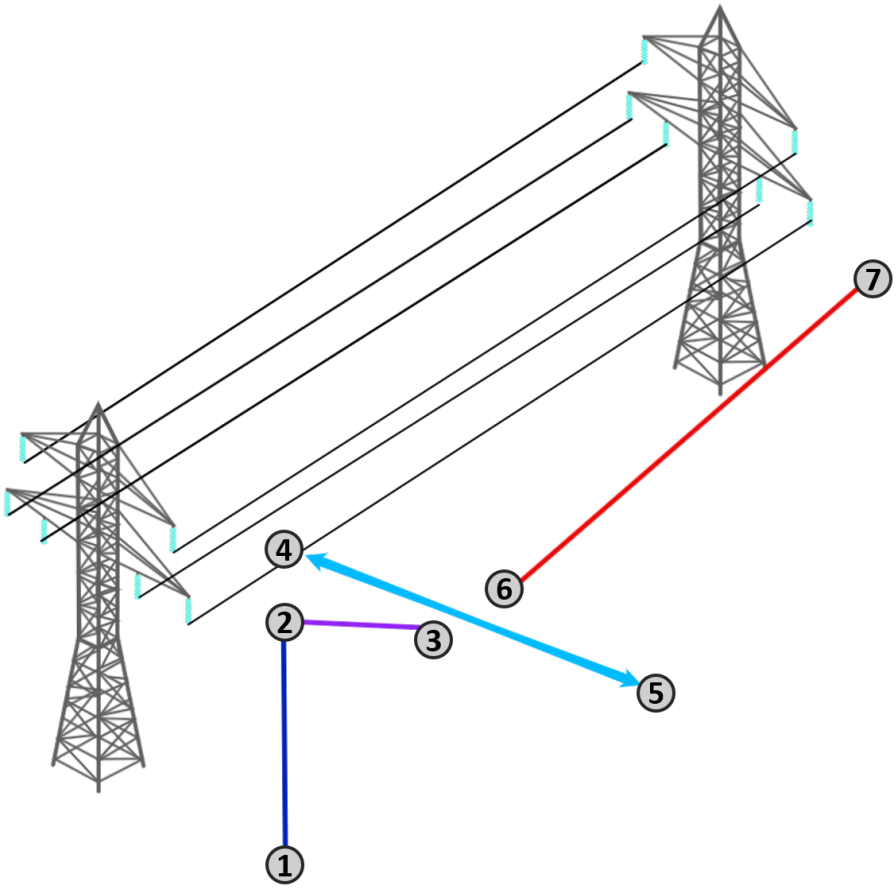


Figure 7.3: Structure inspection movements schematic

present in the structure under inspection through the application of the Electrical Assets Monitoring System system, which during the trajectory shown in Figure 7.4, must locate each of the insulators in the image reference. To this end, a continuous path was designed, which is computed before starting the monitoring process configured to detect insulators, consisting of five segments. The first segment with a path described from point (8) until (17), consists in the inspection path in the first lateral of the structure. This path, parallel to the power line, consists of generating a grid of four rows with a length of 25 meters, whose center is aligned with the center of the structure. The height of the grid is related to the percentage of the height of the structure defined by the user in the preconditions, which corresponds to the height of the lowest line of the grid. The UAV, which is in position (A), thus moves to the starting point of the first line to be performed (8). Reaching point (8), the UAV will perform the first line with direction (8)

to (9), followed by an ascent to the second line ((9) to (10)). When the first point of the second line is reached (10), the UAV makes a translational movement in the opposite direction to that of the first line, from ((10) to (11)), going back up one line when it reaches point (11). From this point on, the process followed from point (8) to (11) is repeated successively until point (17) is reached, which is at a height equal to that of the structure. At this point, the inspection of insulators on the first side of the structure is completed. Thus, it becomes necessary to cross to the other side of the structure, which is made up of segments (17) to (18), (18) to (19) and from (19) to (20). To this end, it was defined that for security reasons the UAV must cross over the structure at 15 meters more than its height. Thus, points (18) and (19), consist of adding 15 meters to the Z coordinate of points (A) and (B), respectively. When the UAV reaches point (19), the UAV descends directly to the starting point of the second side grid, corresponding to point (20). At point (20), the process followed for the path on the previous side will be repeated, going from point (20) to (21). In point (21) the inspection system will change do the ***End of Inspection State***.

The ***End of Inspection State*** state is responsible to process the gathered data from the previous state and waits for an user input in order to give him the control of the UAV. After this point the user will assume the control and can proceed to landing process.

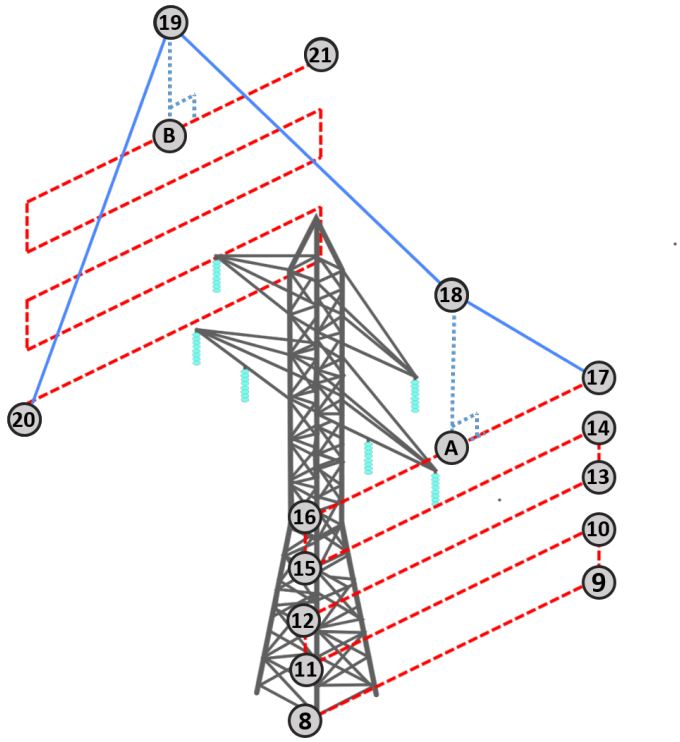


Figure 7.4: Insulator inspection movements schematic

7.2 Waypoint Generator

The Waypoint Generator consists of generating a set of waypoints that constitute different segments of a path that the UAV will have to describe when carrying out the inspection process. The trajectory that the UAV describes when it follows two waypoints is purely straight and depends on the current position and orientation of the UAV. The process of generating paths consists on the location of two reference waypoints, the starting and the destination waypoints. The segment between both point is divided in **N - 2** points, where N is the number of waypoint that describe the path. The Algorithm 7 summarizes this process.

Since for each of these waypoints there is an associated UAV orientation, it is necessary to define a point of reference in the world, so that the UAV is always oriented towards that target or point of interest, as this orientation defines the field of view of the camera, since the UAV only rotates around the Z axis. An example case is the situation where the UAV is inspecting the electrical assets around the structure, where the UAV

Algorithm 7 UpdateDepthEstimator

Input: P_0^W, P_1^W, N

$\mathrm{P}_0^1 = P_1^W - P_0^W$

for $i = 0;\ i \leq N; i++$ **do**
  $curWaypoint.position = P_0 + P_1^W \times P_0^1 \times \frac{i}{N}$
  $curWaypoint.orientation = ComputeYaw(curWaypoint.position)$
  $waypointsQueue.push(curWaypoint)$
end

will always have to be pointing towards this structure. For this, it is necessary to provide the X, Y location of that target in coordinates of the Global reference, in order to obtain the relative orientation between the target and UAV. Since the UAV only rotates around the Z axis, it is only necessary to compute the value of yaw (ϕ), with the rest, pitch (θ) and roll (ψ), being equal to zero. This way, the orientation of the UAV, when in the position (P_x, P_y, P_z) with reference point is (X, Y, Z), is defined by:

$$\begin{bmatrix} \psi \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\arctan\left(\frac{Y-P_y}{X-P_x}\right) \end{bmatrix} \tag{7.1}$$

Chapter 8

Results

This chapter presents the results related to the performance of the proposed inspection process. Initially, the evaluation of the Electric Asset Detection System is carried out, through the realization of a set of experiments that allow to conclude which of the CNNs allows to extract the best performance of this subsystem. Finally, based on the results of the first assessment, the inspection system is evaluated in a simulated scenario.

8.1 Object Detection Evaluation

8.1.1 Experiments

As mentioned in chapter 5.2, the dataset resulting from the data augmentation process was divided into two sub-datasets, where 70% of the images were used for training each network and the remaining 30% were used to test and validate the performance of the networks. Thus, the first step of the performance evaluation is to apply the testing dataset to each network, using the respective framework applied in the training step.

In order to properly evaluate the performance of different networks in different scenarios, a new set of images was defined, different from the one used for training and testing. From this new dataset, six new sets were generated, each of which corresponds to the application of a certain technique used in the data augmentation process. As it is possible to deduce, not all the techniques used in data augmentation allow to simulate the different image conditions that most likely to occur in a real inspection scenario, and therefore only the following techniques were used: used Blur, Salt and Pepper, Fog, Scaling, Rotation, and Gaussian Noise.

Finally, each model obtained after conversion to the Intermediate Representation

using the OpenVino toolkit, is subjected to speed tests. This tests consist of adding the electrical targets detection system, presented in Section 5.1, to a ROS middleware node, which subscribes each image of a rosbag file obtained from one of the missions performed by UAV STORK I. These tests consisted of check the speed of each network on three different platform types, performing the processing using the Movidius NCS or with the native CPU of each platform.

8.1.1.1 Experimental Configuration

To train and test the proposed networks with the previously explained dataset, a workstation powered by an Intel® i7-8700K CPU with 3.70GHz and a NVIDIA® GeForce GTX 1080 Ti GPU with 3584 NVIDIA® CUDA cores was used.

To perform the speed tests were used the presented workstation, a laptop powered by an Intel® i7-4700MQ, Quad-Core, 2.40GHz CPU. All of these platforms had the Ubuntu operating system, the ROS Kinetic distribution and the OpenVino 2019 toolkit.

8.1.2 Results

In the previous sub-section 8.1.1, the experiments performed in order to evaluate the performance of each CNN in the object detection task were exposed. Here it will be presented the results obtained in each proposed experiment.

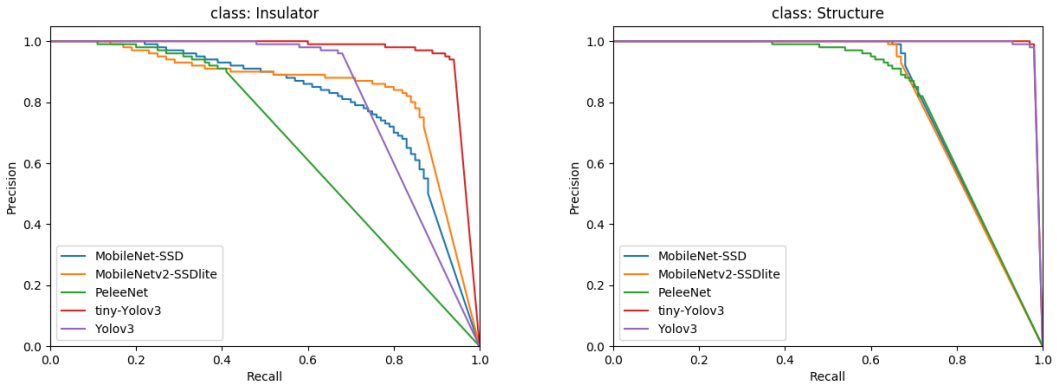
8.1.2.1 Precision Results

The Table 8.1 shows the precision that each network obtained after processing 4738 images that belong to the test dataset. As can be seen from this table, the Yolo based networks out-performed the SSD based ones, being possible to find a difference of 16.7% between the best of each type. Among the Yolo based, tiny-YOLOv3 obtained 6.95% more precision than YOLOv3, reaching 90% on the overall dataset. Among the SSD-based networks, the two mobileNets achieved approximately the same precision performance, as expected. However, the performance of the PeleeNet-SSD network was much lower than the rest, reaching an accuracy of 55%, which is very close to a random classifier (50%).

In Figure 8.1 are represented the precision-recall curves of each network for both classes, insulators, and structures. As we can observe, the curves confirm the results of the obtained precision values (area under the curves). In general, all networks performance was better at detecting structures. This is due to the fact that the aspect ratio of the structures in relation to the size of the images is larger than the aspect ratio of the

Table 8.1: Original dataset precision results

Networks	Precision (%)
MobileNet-SSD	73.2
MobileNetV2-SSD	73.3
PeleeNet-SSD	55.1
tiny-YOLOv3	90.0
YOLOv3	83.05



((a)) Insulators ((b)) Structures

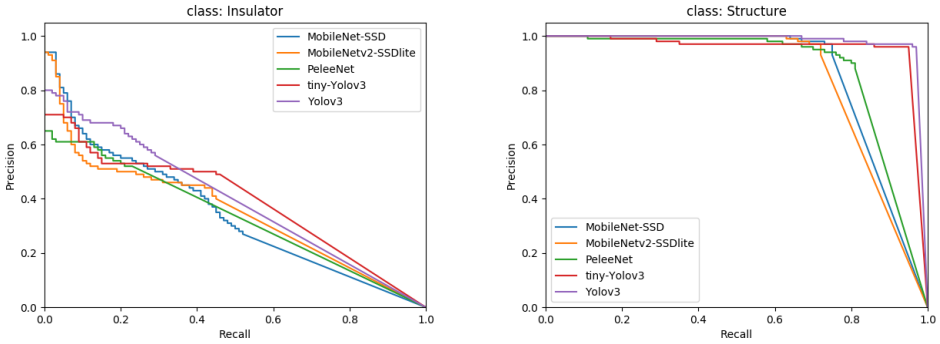
Figure 8.1: Precision-Recall curve for each class

insulators. This factor is more accentuated in the SSD-based networks, since the input size is 300×300, while in the YOLO-based is 416×416.

Regarding the evaluation process with datasets with different conditions, whose results of precision can be analyzed from the table 8.2, it was once again proven that networks based on YOLO perform better than the SSD-based, although it can be seen that the type of conditions under evaluation affect considerably the accuracy of the detections, as shown by the low values obtained in relation to the global values of the previous experiment. From Figures 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, it is possible to observe that the quality of detection of the insulators has decreased considerably, and that in most cases it is below the threshold of the random classifier, while for structures, the quality of detection only showed a slight generalized decrease, although with a greater impact on the rotation scenario.

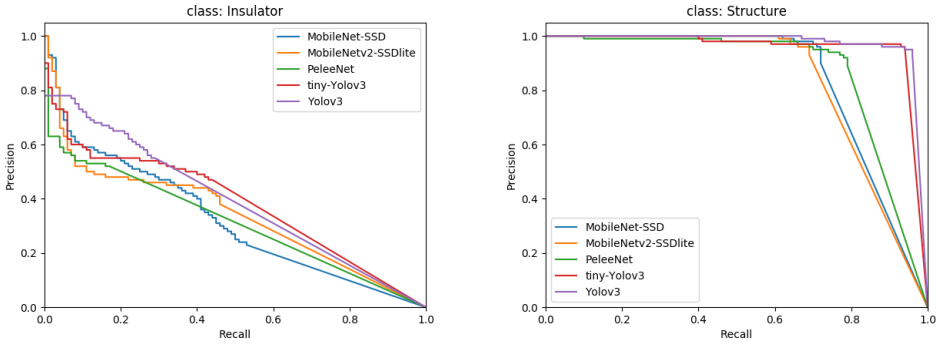
Table 8.2: Data Augmentation precision results

Networks	Blur	Fog	Scaling	Rotations	SaltandPepper	Gaussian
MobileNet-SSD	51.6	49.3	53.6	55.7	51.5	50.7
MobileNetV2-SSD	48.3	46.6	49.82	56.2	49.2	47.8
PeleeNet-SSD	46.8	43.8	48.8	41.4	48.3	41.8
tiny-YOLOv3	**59.4**	**58.5**	**61.1**	56.0	**61.5**	**60.3**
YOLOv3	58.3	57.7	58.6	**57.7**	59.2	58.4



((a)) Insulators ((b)) Structures

Figure 8.2: Precision-Recall curve for each class with blur occurrence



((a)) Insulators ((b)) Structures

Figure 8.3: Precision-Recall curve for each class with fog conditions

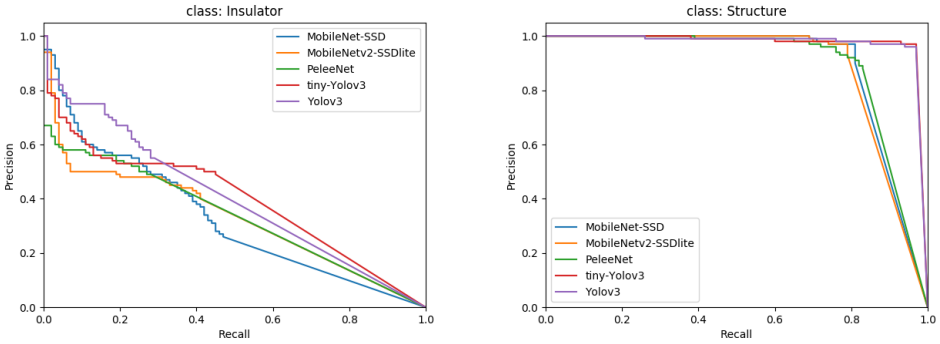
In Figures 8.8 and 8.9, it is possible to observe an example representative of the detection results obtained for tiny-YOLOv3 and MobileNet-SSD, respectively.

8.1.2.2 Speed Results

The inference speed, in frames per second (fps), is presented in the table 8.3 for each network in different platforms, where it is possible to conclude that SSD based networks outperformed the YOLO based ones, in both CPU and NCS. Within YOLO-based, tiny-YOLOv3 is significantly faster than YOLOv3, where when applied to NCS the second does not even reach real-time. Among the SSD-based networks, when applied to CPU, the MobileNet reached the best inference speed on the onboard UAV computer and workstation, but when NCS is used, the MobileNetv2 outperformed the other, reaching the highest mean speed on the onboard UAV computer and on the Odroid. It is also possible to observe that NCS performance slows down on different platforms because it depends on the hardware components of the board.

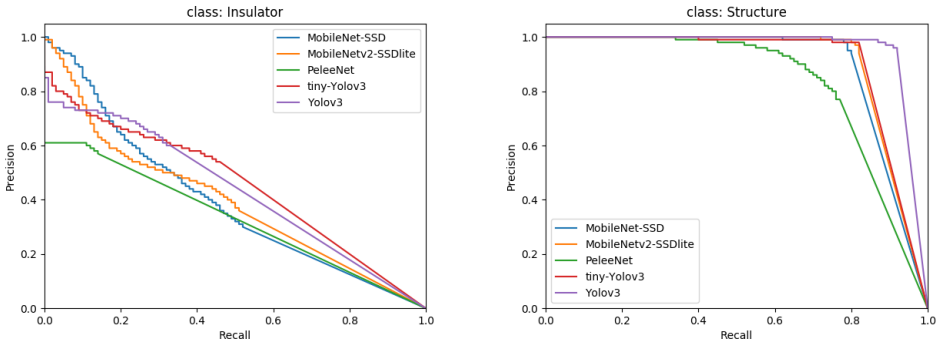
Table 8.3: Inference speed of the networks on different platforms in frames per second

Networks	Onboard UAV PC		Workstation		Odroid XU-3	
	CPU	NCS	CPU	NCS	CPU	NCS
MobileNet	48	12.5	145	NA	NA	10.8
MobileNetv2	26	13.2	60	NA	NA	11.5
Pelee	37	9.2	100	NA	NA	8.7
tiny-Yolov3	22	7.6	79	NA	NA	6.7
Yolov3	2.3	0.74	10	NA	NA	0.73



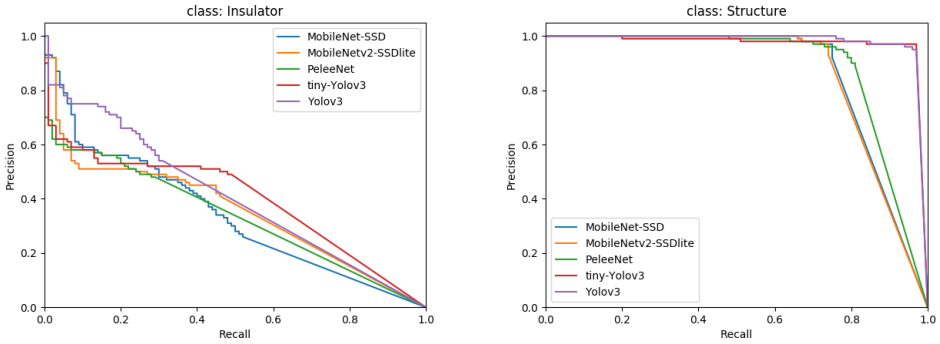
((a)) Insulators ((b)) Structures

Figure 8.4: Precision-Recall curve for each class with scale variance



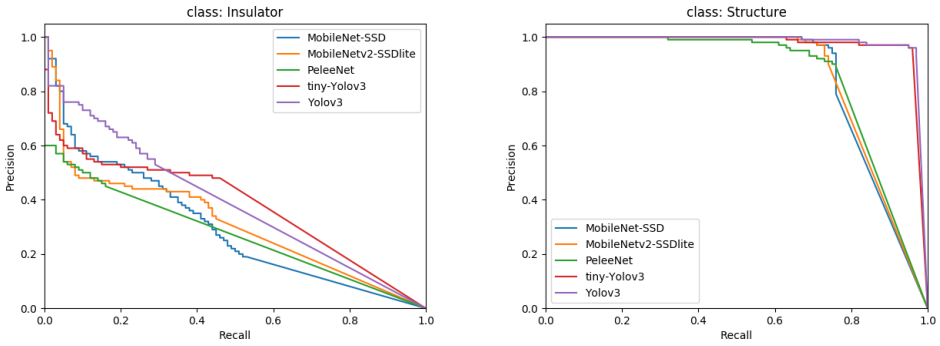
((a)) Insulators ((b)) Structures

Figure 8.5: Precision-Recall curve for each class with rotation variance



((a)) Insulators ((b)) Structures

Figure 8.6: Precision-Recall curve for each class with black and white pixels occurrence



((a)) Insulators ((b)) Structures

Figure 8.7: Precision-Recall curve for each class with Gaussian noise conditions

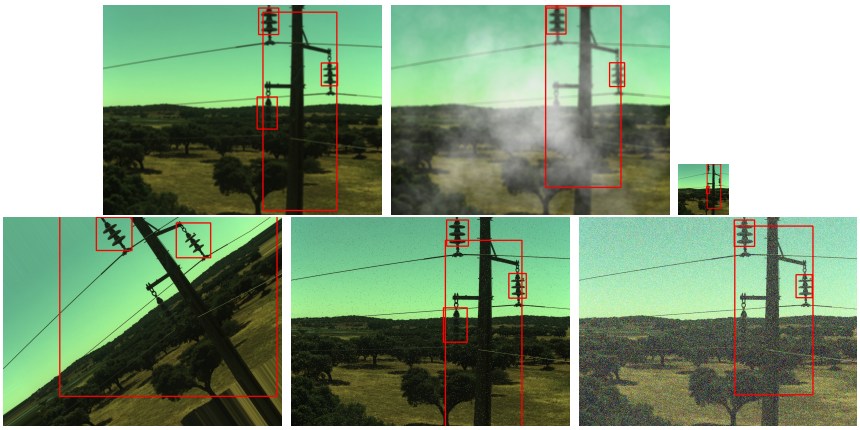


Figure 8.8: Detection examples using MobileNet-SSD

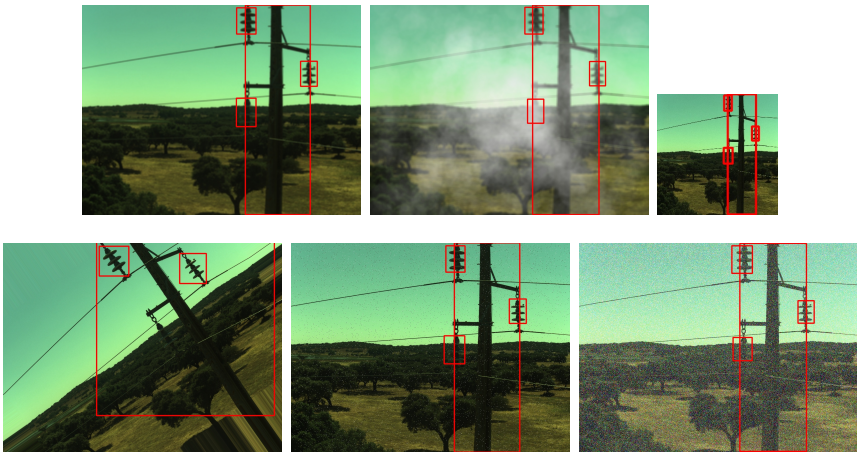


Figure 8.9: Detection examples using tiny-YOLOv3

8.2 Inspection System Evaluation

8.2.1 Experiments

In order to evaluate the performance of proposed inspection system, a simulated scenario was developed using the Modular Open Robots Simulator Engine (MORSE) [51] simulator. MORSE is a powerful simulator that runs on the Blender Game Engine, which contains a variety of communication tools that allows each of the components of MORSE to connect to external applications through middleware used in robotics, such as ROS. The available sensors are fully supported by the middleware and can provide data similar to a real-world sensor, through the addition of modifiers capable to add Gaussian noise to the simulated data.

An outdoor scenario was developed in Blender, which consists of three pylons, each with six insulators, three on each side. Each pylon has 45 meters of height and 11 meters of width, with a distance between consecutive structures equal to 70 meters. In the Figure 8.10 it is possible to see the developed 3D scenario. Regarding the simulation platform, a generic quadcopter, Figure 8.11 was used, with a fixed monocular camera attached to the bottom of the quadcopter and pointing towards its direction axis. The simulator communicates with the system, via ROS middleware, by publishing two topics correspondent to the quadcopter pose measurements and the camera sequence of frames. In addition, it also provides a quadcopter control system that receives a 3D destination point and a desired attitude, moving the platform to that point. This functionality is also accessible from a ROS topic, that requires the publication of each destination point by the system proposed in this dissertation.

From the simulated scenario, it was possible to perform a set of experiments using the proposed system. From the evaluation carried out in Section 8.1, the Electric Asset Detection System will use the tiny-YOLOv3 network as an object detector in each experiment, since it is also capable of carrying out the detections in the simulated environment presented. The first experiment consisted in run the system three times in a row without noise on the quadcopter sensors, in order to be possible to get the error of the estimated position of each electrical asset during the inspection, and the number of samples collected, detailing the number of truly positive samples and the number of samples that were incorrectly assigned. In the second and third experiments, the same type of evaluation was performed, but now with Gaussian noise in measurements of the quadcopter pose. More specifically, in the second experiment, a Gaussian noise with 10 centimeters of standard deviation was added to each component (X,Y,Z) of the UAV position, and in the third experiment, in addition to the noise referred to in the second

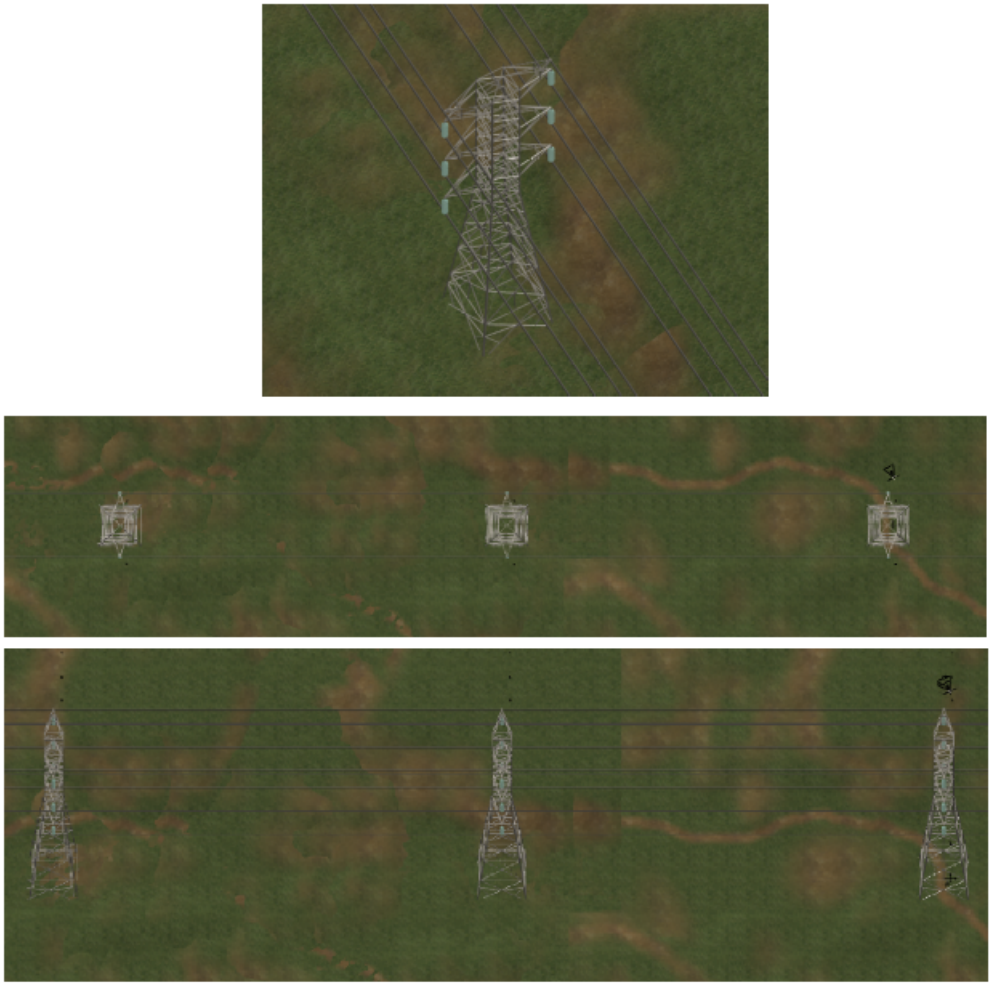


Figure 8.10: 3D scenario proposed for electrical inspection simulation.

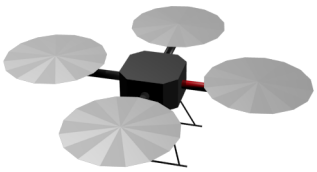


Figure 8.11: Used quadcopter 3D model.

experiment, a Gaussian noise of 1° was added to each of the Euler angles corresponding to the quadcopter's attitude.

8.2.2 Results

Regarding the application of proposed inspection system to the simulated world, it can be seen in Figure 8.12 the trajectory resultant from the application of the Inspection System and Waypoint Generator presented in chapter 7. To achieve this trajectory, it was defined that the quadcopter should descend to 50% of the structure's height and perform movements, on each side, parallel to the power line, with a length equal to 25 meters, 20 meters away from each estimated side of the structure. As can be seen from the figure, the quadcopter performed the expected trajectory as planned, managing to visually cover the entire structure that contains the greatest interest for the inspection process, even when the estimation of the structure's position presents a significant error.

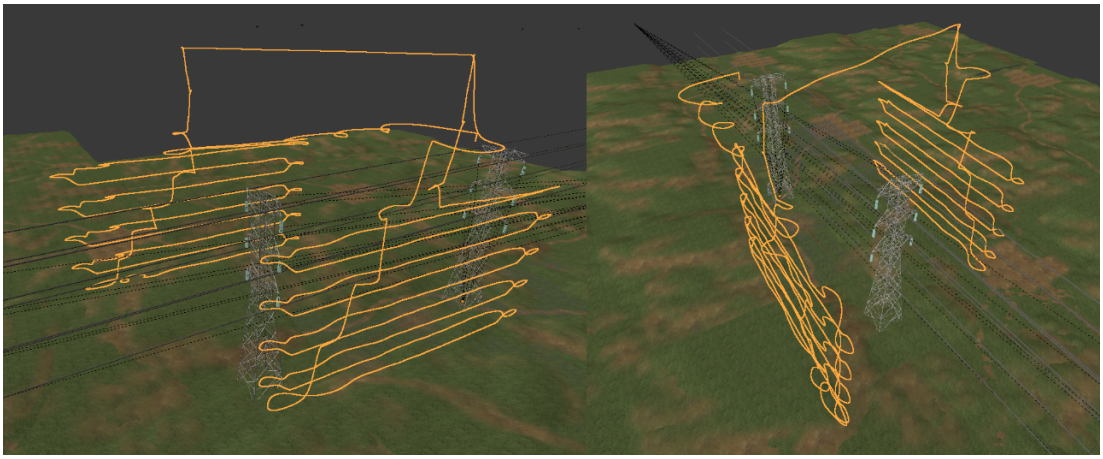


Figure 8.12: Trajectory performed by UAV during a simulation of the inspection process.

The results of the first experiment are presented in Tables 8.4 and 8.5. As can be seen in both tables, the Electrical Assets Monitoring System was able to detect and map all the electrical assets under inspection. On the first trial the system estimated the position of the upper-central point of the structure with an error of 6.81 m and failed by 2.34 m in its width. From this estimation, the Electric Asset Monitoring System was able to detect and map all the insulators present in the structure, without any sample being erroneously collected. The estimation of the location of each insulator showed a maximum error, at the end of the inspection process, of 0.61 m for insulator number 4, while insulator 3 showed the greatest reduction of this error throughout the process, reducing from a maximum error of 1.72 m for the 0.57 m. In the second trial, unlike the first, wrong samples were assigned to some insulators, although in insignificant quantity in relation to the samples correctly collected. In the third trial, it is important to note

that, for insulator 3, only three samples were collected, because it was only estimated at the end of the inspection process.

Table 8.4: Experiment 1: Structure estimation without noise on the UAV pose

	Position error (m)	Width error (m)
Trial 1	6.805	2.340
Trial 2	6.202	2.625
Trial 3	1.986	4.818

Table 8.5: Experiment 1: Insulators estimation without noise on the UAV pose. Where, **FP** are the false positive samples and, **TP**, the true positive samples.

		3D Error (m)				ROI Samples	
	Map Estimations	**Final**	**Mean**	**Min**	**Max**	**TP**	**FP**
Trial #1	Insulator #1	0.330	0.313	0.269	0.466	1658	0
	Insulator #2	0.498	0.826	0.498	1.306	1060	0
	Insulator #3	0.568	0.834	0.568	1.724	485	0
	Insulator #4	0.605	0.872	0.605	1.336	1313	0
	Insulator #5	0.221	0.195	0.129	0.268	1620	0
	Insulator #6	0.302	0.412	0.272	0.902	1193	0
Trial #2	Insulator #1	0.466	0.401	0.304	0.488	960	5
	Insulator #2	0.912	1.401	0.899	1.967	565	3
	Insulator #3	0.672	0.887	0.672	1.219	837	1
	Insulator #4	0.459	0.513	0.459	0.555	1801	7
	Insulator #5	0.397	0.439	0.315	0.685	1462	4
	Insulator #6	0.310	0.388	0.309	0.588	1242	1
Trial #3	Insulator #1	0.263	0.278	0.248	0.326	878	0
	Insulator #2	0.349	0.289	0.266	0.349	58	0
	Insulator #3	0.229	0.229	0.229	0.229	3	0
	Insulator #4	0.821	1.102	0.821	1.352	395	0
	Insulator #5	0.327	0.399	0.327	0.465	844	0
	Insulator #6	0.250	0.354	0.250	0.483	868	0

In Tables 8.6 and 8.7, it is possible to verify that the performance of the inspection system suffered a slight qualitative reduction in the monitoring of the insulators, when added Gaussian noise to the measured position of the quadcopter, having in this experience even estimated four false insulators, one in the second attempt and three third. Despite this, he was able to correctly detect and map all existing insulators in the structure, without presenting errors greater than 0.7 m at the end of the inspection process.

It should also be noted that the number of samples correctly collected has decreased while false positive samples have increased.

Table 8.6: Experiment 2: Structure estimation with noise on the UAV position measurements

	Position error (m)	Width error (m)
Trial 1	1.446	1.446
Trial 2	1.553	0.709
Trial 3	1.677	4.135

Table 8.7: Experiment 2: Insulators estimation with noise on the UAV position measurements. Where, **FP** are the false positive samples and, **TP**, the true positive samples.

		3D Error (m)				ROI Samples	
	Map Estimations	**Final**	**Mean**	**Min**	**Max**	**TP**	**FP**
Trial #1	Insulator #1	0.344	0.325	0.292	0.348	703	5
	Insulator #2	0.705	0.789	0.640	0.878	264	10
	Insulator #3	0.363	0.524	0.362	0.705	829	1
	Insulator #4	0.500	0.722	0.499	1.033	637	7
	Insulator #5	0.092	0.123	0.0823	0.333	459	0
	Insulator #6	0.463	0.563	0.460	1.277	232	8
Trial #2	Insulator #1	0.498	0.529	0.433	0.597	771	15
	Insulator #2	0.591	0.636	0.579	0.824	562	19
	Insulator #3	0.449	0.685	0.449	1.49	399	1
	Insulator #4	0.354	0.499	0.354	0.735	677	3
	Insulator #5	0.422	0.592	0.414	0.863	332	3
	Insulator #6	0.574	0.572	0.535	0.984	655	33
	Insulator #7	NA	NA	NA	NA	NA	2
Trial #3	Insulator #1	0.346	0.321	0.161	0.922	393	6
	Insulator #2	0.527	0.628	0.527	1.218	572	12
	Insulator #3	0.380	0.645	0.366	1.657	507	0
	Insulator #4	0.504	0.758	0.501	1.109	303	4
	Insulator #5	0.380	0.462	0.319	1.081	434	1
	Insulator #6	0.248	0.421	0.245	1.354	228	6
	Insulator #7	NA	NA	NA	NA	NA	49
	Insulator #8	NA	NA	NA	NA	NA	40
	Insulator #9	NA	NA	NA	NA	NA	19

Finally, in Tables 8.8 and 8.9, it is possible to verify that adding a Gaussian noise to the attitude angles measured by the quadcopter, there is a high degradation of the performance of the monitoring of electrical assets. Despite having detected and mapped

Table 8.8: Experiment 3: Structure estimation with noise on the UAV position and orientation measurements

	Position error (m)	Width error (m)
Trial 1	5.586	1.475
Trial 2	12.422	2.735
Trial 3	6.107	3.891

all assets in trials one and three, the number of falsely estimated electrical assets has also increased. In the case of trial 2, only four of the electrical assets were estimated, with a reduced amount of samples collected. In general, the estimation error increased considerably, the number of correctly collected samples decreased and the number of false positives has also increased.

In Figure 8.13, it is possible to see two examples that represent the Electrical Assets Detection System measurements, the ground-truth and the Electrical Assets Monitoring System tracks.

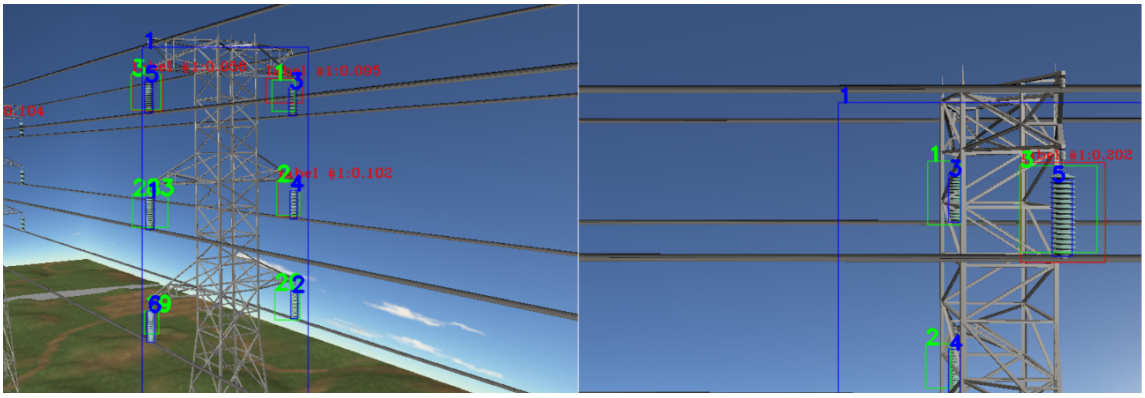


Figure 8.13: Current detections and tracks visualization during the inspection process. The red rectangles are the detections obtained by the Electrical Assets Detection System, in green there are the tracks followed by the Electrical Assets Monitoring System and the blue rectangles represent the ground-truth bounding boxes.

Table 8.9: Experiment 3: Insulators estimation with noise on the UAV position and orientation measurements. Where, **FP** are the false positive samples and, **TP**, the true positive samples.

		3D Error (m)				ROI Samples	
	Map Estimations	**Final**	**Mean**	**Min**	**Max**	**TP**	**FP**
Trial #1	Insulator #1	0.468	0.959	0.348	0.520	102	18
	Insulator #2	0.484	0.653	0.363	3.447	223	22
	Insulator #3	0.867	0.825	0.412	3.094	300	14
	Insulator #4	0.848	0.958	0.348	4.001	93	12
	Insulator #5	2.200	3.46	0.635	6.589	36	41
	Insulator #6	0.567	0.923	0.281	4.220	300	65
	Insulator #7	NA	NA	NA	NA	NA	13
	Insulator #8	NA	NA	NA	NA	NA	7
Trial #2	Insulator #1	1.203	1.144	1.097	1.255	18	16
	Insulator #2	1.607	1.607	1.607	1.607	23	0
	Insulator #4	0.988	0.982	0.976	0.988	14	1
	Insulator #6	3.196	3.477	3.196	3.666	7	3
Trial #3	Insulator #1	2.400	2.774	2.400	7.456	16	7
	Insulator #2	3.755	4.798	3.630	5.820	3	6
	Insulator #3	0.558	0.969	0.528	3.673	14	0
	Insulator #4	0.555	0.555	0.555	0.555	16	0
	Insulator #5	1.265	2.470	1.265	3.994	25	9
	Insulator #6	1.016	0.796	0.539	3.586	45	5
	Insulator #7	NA	NA	NA	NA	NA	9
	Insulator #8	NA	NA	NA	NA	NA	7
	Insulator #9	NA	NA	NA	NA	NA	2

Chapter 9

Conclusion and Future Work

This dissertation was focused on the development of an autonomous inspection system that allows detecting, acquiring and mapping samples of electrical assets, such as insulators and transmission structures, requiring only a sequence of video images from a camera and navigation data with the UAV pose.

In general, it can be concluded that the proposed objectives have been accomplished, since the developed system proved to be capable of carrying out an autonomous inspection sequence in real time, by mapping, tracking and sampling electrical assets, in a low cost and effective way, when integrated in computational systems with low processing capacity, due to the use of Movidius NCS.

From the experiments and results exposed for the Electrical Assets Detection System in Section 8.1, it is possible to conclude that the real-time detection of electrical assets is feasible using lightweight Convolutional Neural Networks on edge devices. These low-cost devices allow great portability and modularity, maintaining the real-time and detection quality requirements on different platforms. The different CNN under testing has shown that they could perform with good precision on different conditions and in the case of tiny-YOLOv3 it was possible to achieve an average precision of 90% on the general dataset at 7 fps.

The evaluation presented in Section 8.2, allows to conclude that the Electrical Assets Monitoring system depends on the quality of the measurements of the UAV pose, where the better these are, the more capable the proposed system is to carry out the mapping and sampling tasks of the set of assets that appear in the camera's field of view, during the inspection process. The quality of the mapping, tracking and sampling of electrical assets is highly influenced by the noise that the UAV pose presents regarding its orientation angles. It can also be assumed that the autonomous Inspection System is capable of

carrying out the set of predefined trajectories in order to be able to take full advantage of the monitoring system.

Part of the work of this dissertation, with emphasis on the electrical assets detection using lightweight Convolutional Neural Networks, was the subject of scientific publication at the Robot2019: Fourth Iberian Robotics Conference [52] in the article Evaluation of Lightweight Convolutional Neural Networks for Real-Time Electrical Assets Detection, available in [53].

In the future, the inspection system would benefit from the application in real scenarios, in order to evaluate the behavior of the monitoring system in terms of ability to estimate depth using real sensors. Another improvement would be the integration of a gimbal and zoom system to be possible to obtain samples with more detail and different perspectives. It would also be interesting to integrate a navigation system guided by the power line in order to add more security to the inspection process and allow to carry out inspection on several structures of the power line in a sequential and continuous way.

Bibliography

[1] Van Nhan Nguyen, Robert Jenssen, and Davide Roverso. Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. *International Journal of Electrical Power Energy Systems*, 99:107 – 120, 2018.

[2] L. F. Luque-Vega, B. Castillo-Toledo, A. Loukianov, and L. E. Gonzalez-Jimenez. Power line inspection via an unmanned aerial system based on the quadrotor helicopter. In *MELECON 2014 - 2014 17th IEEE Mediterranean Electrotechnical Conference*, April 2014.

[3] L. Wang and H. Wang. A survey on insulator inspection robots for power transmission lines. In *2016 4th International Conference on Applied Robotics for the Power Industry (CARPI)*, pages 1–6, 2016.

[4] Liang Zhong, Juan Jia, Rui Guo, Jun Yong, and Jie Ren. Mobile robot for inspection of porcelain insulator strings. In *Proceedings of the 2014 3rd International Conference on Applied Robotics for the Power Industry*, pages 1–4, 2014.

[5] L. Wang, H. Wang, Y. Chang, X. Pan, and H. Zhang. Mechanism design of an insulator cleaning robot for suspension insulator strings. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2217–2222, 2015.

[6] Joon-Young Park, Jae-Kyung Lee, Byung-Hak Cho, and Ki-Yong Oh. An inspection robot for live-line suspension insulator strings in 345kv power lines. *IEEE Transactions on Power Delivery - IEEE TRANS POWER DELIVERY*, 27:632–639, 04 2012.

[7] Carvalho R. Malveiro M., Martins R. Inspection of high voltage overhead power lines with uav's. *Proceedings of the 23rd International Conference on Electricity Distribution*, 2015.

[8] Chuang Deng, Shengwei Wang, Zhi Huang, Zhongfu Tan, and Ji Liu. Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications. *Journal of Communications*, 2014.

[9] Xiaowei Xie, Zhengjun Liu, Caijun Xu, and Yongzhen Zhang. A multiple sensors platform method for power line inspection based on a large unmanned helicopter. *Sensors*, 2017.

[10] Taskeed Jabid and Tanveer Ahsan. Insulator detection and defect classification using rotation invariant local directional pattern. *International Journal of Advanced Computer Science and Applications*, 2018.

[11] Zahid Siddiqui, Unsang Park, Sang-Woong Lee, Nam-Joon Jung, Minhee Choi, Chanuk Lim, and Jang-Hun Seo. Robust powerline equipment inspection system based on a convolutional neural network. *Sensors*, 2018.

[12] Xian Tao, Dapeng Zhang, Zihao Wang, Xilong Liu, Hongyan Zhang, and De Xu. Detection of power line insulator defects using aerial images analyzed with convolutional neural networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[13] Xiaolong Hui, Jiang Bian, Xiaoguang Zhao, and Min Tan. Vision-based autonomous navigation approach for unmanned aerial vehicle transmission-line inspection. *International Journal of Advanced Robotic Systems*, 2018.

[14] X. Hui, J. Bian, X. Zhao, and M. Tan. Deep-learning-based autonomous navigation approach for uav transmission line inspection. In *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, 2018.

[15] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[17] Fei-Fei Li, Andrej Karpathy, and Justin Johnson. *CS231n: Convolutional Neural Networks for Visual Recognition*. University of Standford - Stanford Computer Vision Course, 2019-08-22. `http://cs231n.stanford.edu/`.

[18] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, 2017.

[19] Depth-wise convolution and depth-wise separable convolution. `https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec`. Accessed: 2020-07-29.

[20] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, 2018.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[22] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[23] Robert J. Wang, Xiang Li, Shuang Ao, and Charles X. Ling. Pelee: A real-time object detection system on mobile devices. *CoRR*, 2018.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[25] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[26] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch, 2018.

[27] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[28] R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[31] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, 2018.

[32] K. Granstrom, C. Lundquist, F. Gustafsson, and U. Orguner. Random set methods: Estimation of multiple extended objects. *IEEE Robotics Automation Magazine*, 21(2):73–82, June 2014.

[33] R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 1960.

[34] H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[35] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems, 1972.

[36] K. Hata and S. Savarese. Cs 231 a course notes 1 : Camera models. 2017.

[37] Janne Heikkilä and Olli Silvén. A four-step camera calibration procedure with implicit image correction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1997.

[38] George Vogiatzis and Carlos Hernández. Video-based, real-time multi-view stereo. *Image Vis. Comput.*, 29(7):434–441, 2011.

[39] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014.

[40] M. Pizzoli, C. Forster, and D. Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616, 2014.

[41] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2000.

[42] V. Klema and A. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25(2):164–176, 1980.

[43] Ros.org — powering the world's robots. `https://www.ros.org`. Accessed: 2020-11-10.

[44] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[45] Joseph Redmon. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013–2016.

[46] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*, Sep 2016.

[47] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016.

[48] H. Durrant-Whyte. Introduction to decentralised data fusion. 2006.

[49] Fengwei Yu, Wenbo Li, Quanquan Li, Yu Liu, Xiaohua Shi, and Junjie Yan. POI: multiple object tracking with high performance detection and appearance feature. *CoRR*, abs/1610.06136, 2016.

[50] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.

[51] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan. Modular open robots simulation engine: Morse. In *2011 IEEE International Conference on Robotics and Automation*, pages 46–51, 2011.

[52] Manuel Silva, José Lima, Luís Reis, A. Sanfeliu, and Danilo Tardioli. *Robot 2019: Fourth Iberian Robotics Conference Advances in Robotics, Volume 1*. 11 2019.

[53] Joel Barbosa, André Dias, José Almeida, and Eduardo Silva. Evaluation of lightweight convolutional neural networks for real-time electrical assets detection. In *Robot 2019: Fourth Iberian Robotics Conference*, pages 99–112. Springer International Publishing, 2020.