AALBORG UNIVERSITY
DENMARK

Identification of Cigarette Litter with the use of Outdoor Mobile Robots

MASTER THESIS
ROBOTICS

Author:
Mathiebhan Mahendran

03^{th} June 2021

Attributions

This report was typeset using LaTeX.

AALBORG UNIVERSITY
DENMARK

Department of Electronic Systems
Robotics

Fredrik Bajers Vej 7
9000 Aalborg
http://www.es.aau.dk

Title:

Identification of Cigarette Litter with the use of Outdoor Mobile Robots

Theme:

Master Thesis

Project Period:

Spring 2021

Project Group:

Group 1062b

Participants:

Mathiebhan Mahendran

AAU Supervisor:

Karl Damkjær Hansen

Company Supervisors:

Søren Tranberg Hansen
(Aarhus Municipality)

Niels Jul Jacobsen
(Capra Robotics)

Number of Pages: 88
Date of completion: 03-06-2021

Synopsis:

This thesis presents the current progress in a two-fold project. The thesis is under collaboration with SkodRobot (Cigarette Litter-Robot) project, where several companies are involved. The objective is to split this project into two tasks where the first step is to develop an object detection system for cigarette litter on sidewalks in public areas, and the second step is to make Capra Hircus move autonomously with the use of coverage planning of removal of cigarette litter on sidewalks in public areas. This thesis presents the first step to make an object detection of cigarette litter on sidewalks. The object detection system has to work in real-time, so the deep learning technique Tiny-YOLOV4 is chosen where open-source neural network Darknet is used. The dataset is also looked upon for the cigarette litter in order to make a robust dataset based on data augmentation and several configurations. It is then implemented on Nvidia Jetson Nano that runs inference with the use of Darknet-ROS. Finally, the object detection system is implemented on the Capra Hircus where several tests are done to check the performance of the object detection of cigarette litter.

Preface

This thesis is made as a completion of the Master education in Robotics at Aalborg University, ending Spring 2021. The aim of this thesis is to develop a system for identifying littered cigarettes in public areas using outdoor mobile robots. The project is under collaboration with SkodRobot project where several companies are involved i.e. Aarhus Municipality, Capra Robotics and KK-tech. The outdoor mobile robot is provided by Capra Robotics in which the outdoor mobile robot will be tested with the vision system to identify cigarette litter. The vision system is a developed state of the art method which utilises the deep learning algorithm called YOLO and a new cigarette litter dataset has been made throughout this thesis. The thesis also relies on concepts of machine learning (deep learning), tracking, coverage planning, linear algebra and ROS (Robot Operating System). To understand this thesis, the knowledge within these concepts is required.

Project Structure

The thesis structure is built up from eight chapters, each covering a significant part of the thesis.

Chapter 1 - This introduces background problem of littering and explains how littering cigarettes can ruin the environment where the construction of a cigarette will also be described.
Chapter 2 - The collaboration with SkodProject (Cigarette Litter Robot) Project is described where several companies are involved to make a outdoor mobile robot (Capra Hircus) to clean cigarette litter.
Chapter 3 - This presents the current findings of the state of the art methods, i.e. mobile robots, object detection, tracking and coverage planning.
Chapter 4 - The problem is formulated in this chapter with sections about project hardware and requirement specification.
Chapter 5 - The methods are described on how to detect and localise (distance estimation) the cigarette litter.
Chapter 6 - The implementation of the cigarette litter dataset, object detection and mounting on the Capra Hircus is described.
Chapter 7 - The testing of the object detection is done and whether the tests are met with the requirement specification.
Chapter 8 - This discusses, summarises and concludes the overall results and findings of the thesis based on the problem formulation, and also states the future plans to improve the dataset and object detection system.

Reading directions:

- The Nomenclature contains a list of abbreviations of terms and phrases used throughout this report.
- This thesis uses the IEEE referencing style. Citations are referred to by [1], [2] and correspond to references in the bibliography. The order of the citations is based on their appearance in the report.
- Tables, equations and figures are referenced with numbers related to the order and the chapter in which they appear in.
- The List of Figures/Tables that are not made by the authors of this report, are referenced below them.
- The Appendix contains extra materials that are used in the report, such as test results, large figures, setup/tools, code and video tests.

Nomenclature

Abbreviation	Definition
AAU	Aalborg University
AMR	Autonomous Mobile Robot
FPS	Frames Per Second
GPS	Global Positioning System
ISO	International Organization for Standardization
IMU	Inertial Measurement Unit
ROS	Robot Operating System
SVM	Support Vector Machine
HOG	Histogram of Oriented Gradients
SIFT	Scale Invariant Feature Transform
SURF	Speeded up robust features
FAST	Features from accelerated segment test
BRIEF	Binary Robust Independent Elementary Features
CNN	Convolutional Neural Networks
ANN	Artificial Neural Networks
RNN	Recurrent Neural Networks
YOLO	You Only Look Once
SSD	Single-shot detector
CPU	Central Processing Unit
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
JSON	JavaScript Object Notation
XML	Extensible Markup Language
ROLO	Recurrent YOLO
LSTM	Long short term memory
IBVS	Image Based Servoing
PBVS	Position Based Servoing
CAD	Computer Aided Design
CSI	Camera Serial Interface
R-CNN	Region Based Convolutional Neural Networks
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
SSH	Secure Shell

Contents

1 Introduction

Littering in today's society is an ever-increasing global problem. It was not until the 1950s domestic manufacturing industry began to revolutionise and build new solutions after the Second World War, where manufacturers began to create more litter with high volumes, such as disposable items and plastic materials[1, 2].

The commonly littered item in the world are cigarette litter, plastic wrappers, bottles, caps, bags, lids, straws, takeout containers and styrofoam where the worlds number one most littered item is plastic is cigarette litter. Cigarette litter, in this case, means cigarette butts, cigarette stubs, cigarette filters and even whole cigarettes that are floating on waterways and scattered or blown away along roads[1]. 4.5 trillion cigarettes are littered each year worldwide, which is roughly 38% of all collected litter. Cigarette litter is not biodegradable and is toxic that leads to pollution, and habitat destruction[3, 4].

It is estimated that 75% of purchased cigarettes that are smoked end up on the ground and not thrown into a trash can. Cigarettes also consist of around 4000 chemical substances which produces bad environmental effects such as polluting the environment and oceans[5].

In Denmark, with an estimated population around 5.7 million, 1.3 million adult smokers that consume 6.1 billion cigarettes each year (estimated in the year 2016)[6]. The service employees collected 22,393 cigarette litter at outdoor areas around two institutions (two institutions was a test) in Aarhus Municipality this year. The expense for manually picking up cigarette litter is estimated at 3 DKK for each cigarette litter to be collected. Where in Aarhus Municipality it costs 67,000 DKK for collecting cigarette litter around these two institutions alone[7, 8].

The optimal solution is to avoid it from the beginning, even though most litterers do not consider cigarette butts, stubs and filters as litter. So, information campaigns against litter waste and smoking are essential tools to limiting this problem. However, there is some negotiation happening to stop pollution, when littering is still a reality.

1.1 What is a Cigarette Butt?

The manufacturing of cigarettes contains three simple components. The three components that a cigarette is built up of are tobacco blend, filter and additives such as nicotine. PVA glue is used to wrap the paper for holding the tobacco blend and the filter material. The paper consists of three types of components which are wrapping paper, tipping paper and plug wrap paper. Cigarettes today can vary up to a length of 85mm to 100mm and have a diameter of 8mm. Whereas filters have a size around 20mm to 30mm long, making an actual cigarette with tobacco blend from 50mm to 80mm[9].

A cigarette butt is the remains of a cigarette after smoking. It is around 30% of the original length of a cigarette. ISO 3308 (2012) (Routine analytical cigarette-smoking machine - Definitions and standard conditions) defines a butt length as the "length of unburnt cigarette remaining at the moment when smoking is stopped." The construction of a cigarette butt can be seen in Figure 1.1[10].

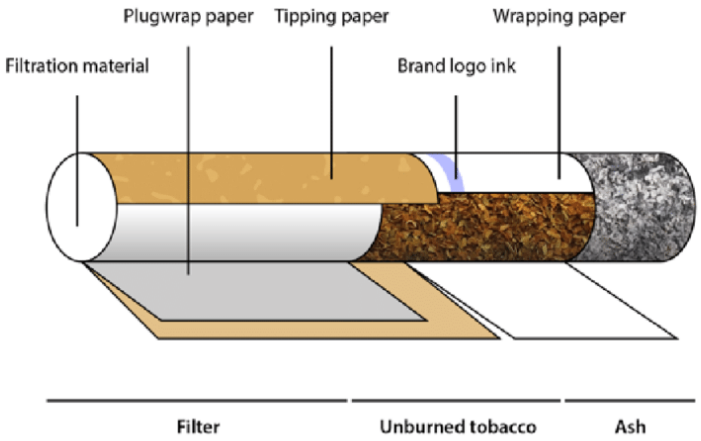

Figure 1.1: Construction of a Cigarette Butt[10]

The three major components that are leftover from a cigarette butt are the filter, unburned tobacco and ash. However, these three leftovers can harm the environment due to various types of chemicals react and be emitted in the environment. One of the effects of this is the aging of the cigarette butt that can affect the environment due to the change in chemical properties[10].

The filter is made of thousands of polymer chains of cellulose acetate, also known as plastic. The filter is toxic, odourless and tasteless. Meanwhile, a filter in a cigarette tries to reduce some substances from smoke but does not make it safer to smoke[11].

Tobacco is made of leaves of flue-cured bright leaf, burley tobacco, and oriental tobacco. In order to initiate attractiveness of the tobacco to smokers, it is implied that some additives are added to the unburned tobacco[12].

The additives can vary in behaviour depending on the molecular weight and boiling point of the additive. Also, the configuration of the cigarette and chemical polarity affects the additive[10].

In this thesis, cigarette butts, cigarette stubs, cigarette filters will be referred to as cigarette litter. In Chapter 2, SkodRobot (Cigarette Litter Robot) project will be explored. The aim of the project is to make an autonomous mobile robot identify and remove cigarette litter in urban/public areas.

2 SkodRobot Project

SkodRobot (Cigarette Litter Robot) is a project that will explore the possibilities of using a mobile robot to automate the collection of cigarette litter. At this present time, manually picking up cigarette litter costs Aarhus Municipality around 3 DKK every time a cigarette litter is collected. The purpose is to establish a more efficient and resource-saving solution for collecting cigarette litter in Aarhus Municipality's urban space. The SkodRobot Project is an innovation project based on a Public-Private Partnership. The project is a collaboration between KK Tech, Capra Robotics and Aarhus Municipality and is financially supported through the municipality's Welfare Technology OPI pool[7, 13]. In order to cooperate SkodRobot project, the current methods will be looked upon. This will mainly focus on Denmark's process of cleaning up litter.

2.1 Technology and the Environment

In Denmark, every municipality has a Technology and Environment Services for maintaining public areas. However, the focus is to look into the SkodRobot Project that is in Aarhus Municipality. The Technology and the Environment in Aarhus Municipality (Teknik og Miljø - Aarhus Kommune) works with urban development, mobility, nature and the environment, public transport, as well as construction and operation of roads. They are in charge of cleaning litter and keeping the city clean in public areas in Aarhus[14]. One of the problems that they are facing with cleaning the public areas are sidewalks. This is a major concern as in larger areas, the sweeper and blower tool can be used but it is more difficult to use them in smaller spaces like sidewalks.

Figure 2.1: Sweeper and a blower is used to clean the litter.

As seen in Figure 2.1 the current techniques today is to use handheld blowers and sweeping machines to clean the roads. The sweeper machine works by a vacuum system where it collects the litter by using water and brushes to make the litter more stable to easily clean up the litter from the floor.

Figure 2.2: The areas that are payed by the technology and the environment in Aarhus Municipality to clean the public areas. (Image taken from Teknik og Miljø - Aarhus Kommune)

As seen in Figure 2.2, it shows the paid areas that Technology and the Environment in Aarhus Municipality are authorised to clean. The different colours represent how often the areas are cleaned. The rest of the areas that are not coloured are cleaned by private companies and sometimes are not cleaned effectively. In Appendix A there are two figures, where Figure A.1 shows where the sweeper machine is used to clean up the roads, and Figure A.2 shows where litter and weed are cleaned up by hand. Techniques such as using garden tools on grass areas and brushes to sweep up dirt from ground surfaces are used by hand. In the next chapter, the state of the art mobile robots will be looked upon, and what approach is needed to detect and localise cigarette litter will be explored in further detail.

3 State of The Art

This chapter introduces the state of the art mobile robots, object detection, object tracking and coverage planning as the aim of this thesis is to look into cigarette littering and how object detection can help. Object tracking will be discussed with the distance estimation approach. The chapter ends with coverage planning that is discussed to know how the mobile robot should move in the urban/public area in Aarhus Municipality based on Chapter 2.

3.1 Mobile Robots

A mobile robot is a machine capable of moving in any given environment[15]. They are able to move around in any ground surface environment. Mobile robots can also be "autonomous" (AMR), allowing them to move, navigate and localise in any uncontrolled environment. Mobile robots can also vary in different types of wheel frames, e.g. differential wheels and navigation approaches.

The different types of robot navigation are:

- Manual remote or tele-op - The mobile robot can be wired or wireless such that it can be controlled with a joystick or other remote control device[16].
- Guarded tele-op - This can sense and avoid obstacles, however having the functionality to remote control the robot[16].
- Line-following - This is based on painted lines on floors or even an electrical wire that is buried underneath the ground. The robot is programmed to follow this line based on the algorithm of the sensors.[16].
- Autonomously randomized - The robot will freely bounce off sensed walls or vice versa with random movements.[16].
- Autonomously guided - The guided robot will know about where the information is, how to reach various goals and direct the way. It uses various sensors, e.g. lasers or cameras, to detect where it is located and based on a path planner, it avoids obstacles and moves from start to end position[16].
- Sliding Autonomy - This combines multiple navigations approaches so it can also be controlled via a joystick and move autonomously[15].

There are also many more different types of robot navigation; however, these are most known for robot navigation.

3.1.1 Indoor Mobile Robots

Indoor Mobile Robots have high demand in industrial settings, which add flexibility to the environment, which means that they can assist workers in the industry and do the tasks at a faster rate leading to a robust system. The known mobile robots today are called MiR (Mobile Industrial Robots)[17] and OMRON[18]. The mobile robots today can also be combined with a collaborative manipulator therefore making it a single system, e.g., Kuka (youBot, iiwa, quantec and flexfellow)[19], Little Helper is a research platform from Aalborg University where they have created 8 generations of the single system plaform[20], Enabled Robotics[21], TIAGo steel (social robot) are also research platforms[22].

Two examples of cleaning robots called Roomba, a vacuum cleaner, and Braava, a floor mopper, were made by iRobot[23]. This was designed to explore further into how effective indoor mobile robots collect trash or litter in households. When looking into an urban/public area robot that collects garbage for collecting trash/garbage, there is a robot called the BARYL[24]. It is an autonomous robot that acts like an active trash bin roaming around and avoiding obstacles; hence this robot does not collect litter efficiently. In this thesis, the mobile robots that will be in focus are outdoor mobile robots since littering of cigarettes mostly takes place in outdoor terrain.

3.1.2 Outdoor Mobile Robots

Outdoor mobile robots are less popular compared to indoor mobile robots. This is due to the uncontrolled environment. There are many factors to be considered when working outdoor such as avoiding and navigating through different types of obstacles and weather conditions. As a result of this, the mobile robots needs to be equipped with safety sensors to work in various environments, and the technology can also vary to enable autonomy. The advantage of outdoor mobile robots is that it is applicable in indoor applications. From January 1^{st} 2021, the danish traffic act has allowed "self-driving devices" in public areas. This will allow Capra Robotics[25], Conpleks[26] and other danish mobile robot companies to work in outdoor terrain in public areas[27].

Most of the current outdoor mobile robots for clearing litter are research projects. These platforms have not yet been released in the public domain. This is due to the fact that robots are not robust compared to human workers at clearing the litter at a certain time frame hence why it is cheaper to use human employees/workers.

3.1.3 Legged Robots

Legged robots are known as a type of mobile robot that adds more degree of freedom in flexibility by using powered articulated limbs. These robots are more versatile when working in different types of terrain and can be used in different types of cases, even for collecting litter in the environment. The current legged robots today are flexible to many situations are the Boston Dynamic Spot[28] and ANYmal[29] these robots are currently

partaking in research. The number of legs can vary, and there are also some hybrid solutions to add wheels on the mobile robot.

3.1.4 Current Research Projects

The current research project for clearing litter with outdoor mobile robots will be explored. Robotech, based in Italy, has designed two robots that can clean and take trash for citizens in the urban/public areas. Both robots are part of a research project called DustBot that is funded by the European Commission in the sixth framework programme (FP6-045299, 2006-2009), and ROBOSWEEP funded by the Tuscany Region (Bando Unico 2012)[30, 31]. The figures below showcase the two autonomous robots involved in DustBot. Figure 3.1b, called Dust Cart, is a robot that takes the trash and Figure 3.1a, called Dust Clean, is equipped with brushes and a container for sweeping and collecting litter from the surface. Both of the robots work with ROS-based autonomous navigation system with laser and GPS for localisation and obstacle avoidance[31, 32].

(a) DustClean[31]

(b) DustCart[32]

Figure 3.1: The two robots that are designed and developed by ROBOTECH.

In the paper presented by Jinqiang Baio et al.[33], the authors made a robot for automatically picking up garbage on the grass. Whereas, in this thesis, the main concern is to look into littering on the ground however, the approach is similar. This paper only focuses on the vision aspects of the robot and how the garbage is detected. The mobile platform uses a robot arm to pick up the garbage from the grass. Thus, the platform is not robust in all weather conditions. It is suitable for clearing the garbage from the grass than the one used by the existing road sweeper truck or a vacuum cleaning robot as this is due to robust navigation it uses[33].

Angosa has developed a mobile robot that can pick up cigarette litter with the use of a vacuum system[34]. They are now willing to look at other types of litter. They now have started up a startup company to remove litter from grass areas where the

mobile platform is updated and will feature anti-theft protection, autonomous mode, so it only navigates in grass areas and using intelligent removal abilities with the use of deep learning. The product is still in its early stages of development and are planning to launch in 2022[35].

3.2 Object Detection and Segmentation

When identifying and detecting cigarette litter, an outdoor mobile robot needs visual perception in order to identify and detect cigarette litter. There are many different approaches to identify and detect cigarette litter with the use of computer vision. In this thesis, the main approach to solve this is by looking into the subsets of artificial intelligence as seen in Figure 3.2.

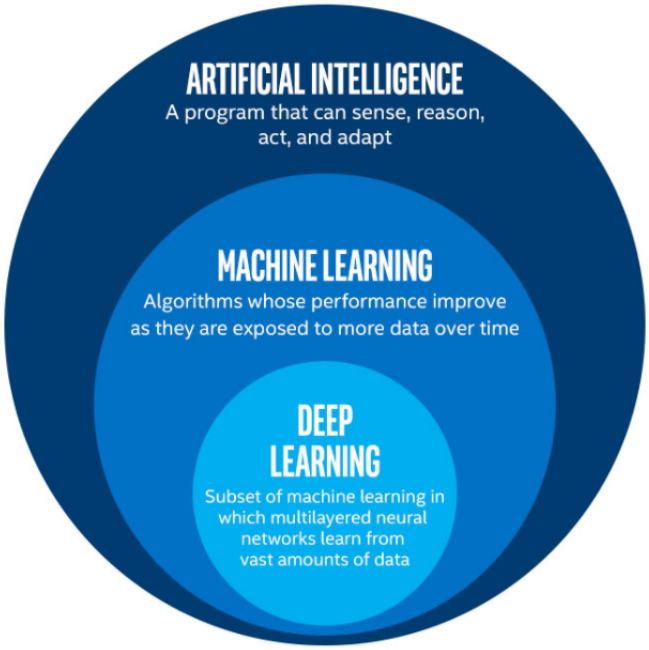

Figure 3.2: The subsets of artificial intelligence[36]

Artificial intelligence is "*the theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages*[37]". Although, in this thesis, the two main systems that need to be outlined and distinguished is machine learning and deep learning from the subset in Figure 3.2.

3.2.1 Machine Learning Overview

Machine learning is the technique in which the computer learns from given data. This is when the computer figures out what the data means without any complex rules. Based on the data and machine learning algorithm, it makes a mathematical model predict what the data is and makes decisions based on the data given[38].

The three main types of machine learning techniques are:

- **Supervised Learning** - This provides the training data with desired annotations/labels (labelled data). This is when the model has to be trained until it predicts the data correctly. An example of supervised learning is used for object detection, image recognition, face & voice recognition[38]. The classic methods for supervised learning are Regression, Decision trees and Support Vector Machines (SVM).
- **Unsupervised Learning** - This provides the training data (unlabelled data) only where the algorithm tries to find clusters of data based on patterns and features[38]. The classic methods for unsupervised learning are clustering: K-means and DBSCAN.
- **Reinforcement Learning (Reward-Based Learning)** - This is based on trial and error method, where the computer (agent) interacts in a certain environment and performs actions. Based on the correct actions, it will either be rewarded or punished. The aim of reinforcement learning is to collect maximum rewards. Reinforcement learning is mostly used in-game development[38].

There are also other types of machine learning techniques such as semi-supervised learning, self-supervised learning, feature learning and etc. However, one of the more advanced learning techniques is to look into deep learning, which is a type of feature learning.

To be able to detect objects, traditional machine learning uses feature detection, and extraction techniques such as HOG, SIFT, SURF, FAST, BRIEF and ORB [39] when the object is detected and extracted from an image it then can be classified with a technique like SVM.

3.2.2 Neural Network & Deep Learning

Deep learning is the sub-field of machine learning and artificial intelligence. At the same time, deep learning uses machine learning and mimics the human brains network of neurons.

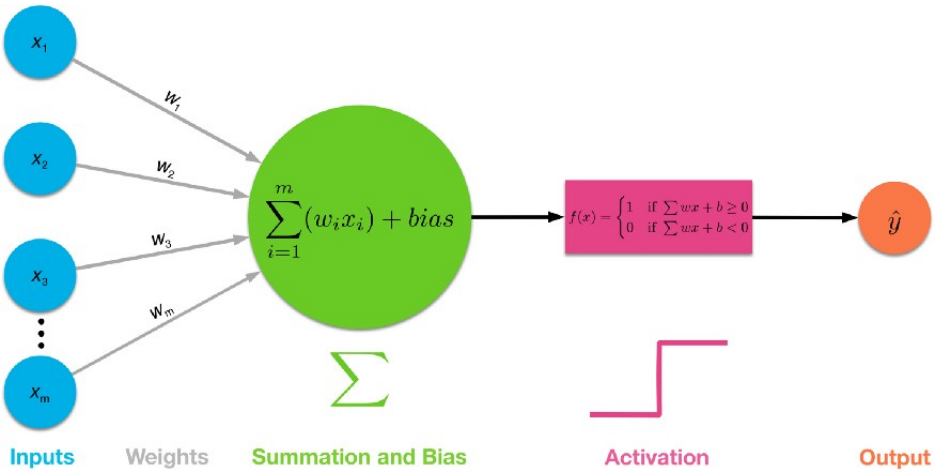

Figure 3.3: A neural network [40]

As seen in Figure 3.3 shows how a neural network works by consisting features of input data, weights, summation and adding bias, activation function and output[40]. A neuron holds a number from 0 to 1. In computer vision, grayscale pixels assume pixel intensities black = 0 (total absence of black) and white is 1 (real presence of white). An example could be an image that has 28 x 28 = 784 neurons (pixels).

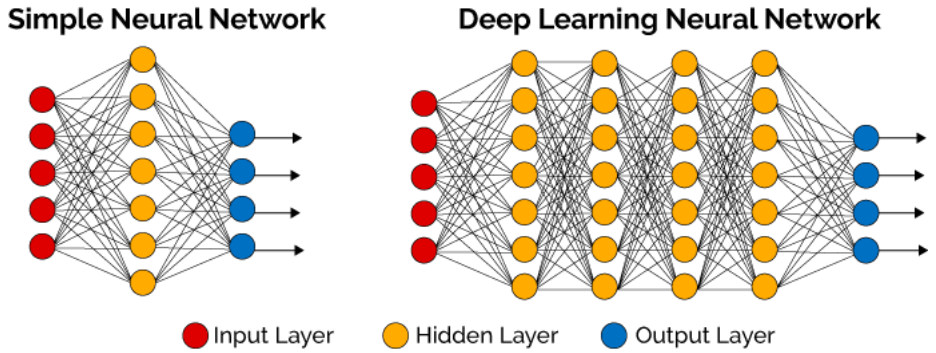

Figure 3.4: A Neural Network and Deep Neural Network [41]

In Figure 3.4 that shows how the layers "Input, Hidden and Output layers" (One circle is a neuron). Layers, in this case, means the series of operations. A simple neural network only consists of one hidden layer, whereas a deep neural network has many hidden layers where there is no limit. The layers will be described and explained how they work below:

- **Input Layer** - The neuron has a feature and it is based on one or many input features, it is transferred to the hidden layer[42].
- **Hidden Layer** - This layer looks into the features and associated weights that is then summed with a bias that is constant. This will then be passed to an activation

function. As mentioned earlier, a neuron holds a number from 0 to 1; this is called activation. Non-linearity is achieved by the activation function. There is no limit to how many hidden layers can be assembled on the user configuration whether the model efficient[42].

- **Output Layer** - Predictions and classifications are based on the information from the hidden layer[42].

Activation Functions

There are several types of activation functions; hence the three most popular ones are Sigmoid, Tanh and ReLU. **Sigmoid** the activation's should be 0 and 1 (binary). That takes the weighted sum and inputs a real number range 0 and 1. The negative number moves to 0, and the positive number moves to 1. The outputs are centred to 0.5. However, Sigmoid is not used nowadays due to the vanishing gradient problem. That is because the information gets squeezed until it vanishes. So no gradients mean no learning[43]. **Tanh** is similar and preferable compared to Sigmoid. The difference is the range from -1 to 1. The outputs are centred to 0. Thus also has the problem with vanishing gradient problem[43]. **ReLU** is one of the most popular activation functions used today. Works with f (x) = max (0, x), which means that the value is 0 when x is less than 0 and linear with a slope of 1 when x is greater than 0. This does not have expensive operations compared to Sigmoid and Tanh. It also learns faster and avoids the vanishing gradient problem[43, 44].

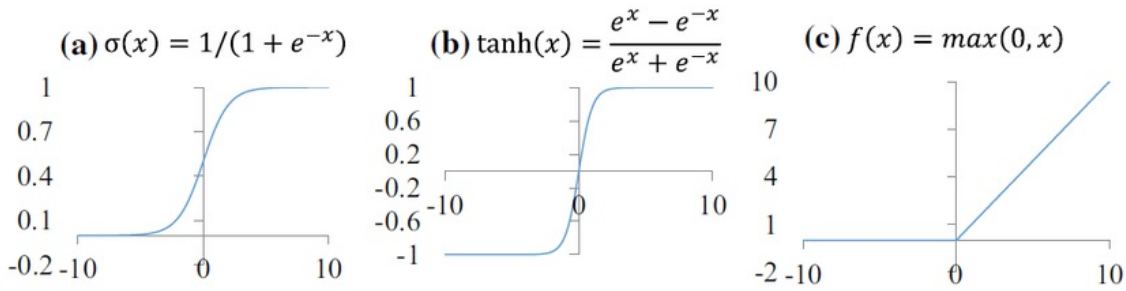

Figure 3.5: (a)Sigmoid function, (b)Tanh function and (c)ReLU function[45]

Figure 3.5 shows the three types of the functions sigmoid, Tanh and ReLU. There are many more activation functions, e.g. Leaky ReLU, Maxout, Softmax (used for the output layer for classification) and ELU, that can be used.

When the hidden layer has found the right features based on the patterns, it is then classified by the output layer.

Neural Networks is so broad that there can be found many different types of them; some examples of neural networks are ANNs (Artificial Neural Network), the simplest form of neural network used for data classification and similar to CNNs (Convolutional Neural Network). CNNs is widely used for image classification, and RNNs (Recurrent

Neural Networks) is widely used for sequence data (audio and text data). Since the aim is to detect cigarette litter, the type of neural network that will be looked upon is CNNs[46].

Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is widely used for object detection and tracking. As mentioned early the difference between ANNs and CNNs is that CNNs works with filters (kernels). Where the filters are used to extract the features from input data and use the convolution operation[46].

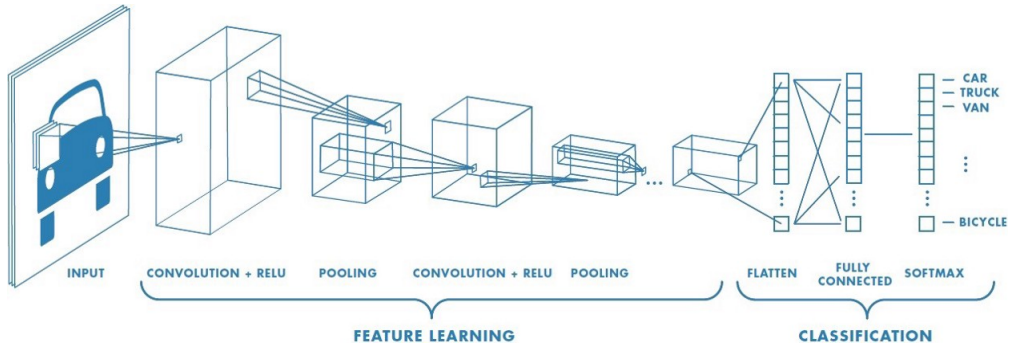

Figure 3.6: Pipeline of Convolutional Neural Network (CNN)[47]

As seen in Figure 3.6 shows the pipeline for how a convolutional neural network works. The general pipeline looks different compared to the general neural network, where all neurons are connected to each layer. In this case, for CNNs, each layer in a collapsed network is not connected to all the other neurons in the next layer. This is due to expensive calculation costs. As can be seen from the image, the car's input is projected with a receptive field. CNN works with three channels, ie. RGB 3D pixels based on image input. The receptive field is part of the image where it is in focus. This is where the collision operation is used (core - series of point products between the weight matrix and the input matrix), and essentially, it slides over the image. Looking at the image, the function learning has three operations (convolution blocks) that run over and over again, leading to an end where folding + ReLU and pooling is a single folding block, convolution means that it produces a function card by applying a fixed size filter to the receptive field. Based on the matrix, it is transferred to the activation function, in this case, ReLU. Pooling layers reduce the parameters to ensure that the information about the image is preserved[47].

Then it leads to the classification part, where it then flattens out to a smaller dimensional vector, and the fully connected layer connects neurons from one layer to the next layer. This leads to a fully connected outer layer where Softmax is then used towards the end when the fully bonded layer compresses and gives the probability of a particular class, e.g., the probability of a cigarette litter seen is 82%. There are several various types

of architecture for CNNs that exist LeNet, DarkNet, AlexNet, VGGNet, GoogLeNet, and ResNet[47][48].

3.2.3 Object detection and segmentation with CNNs

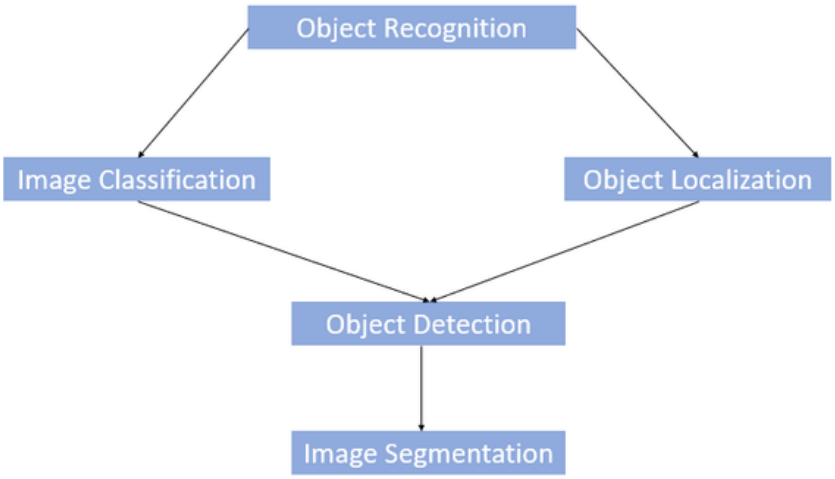

Figure 3.7: Pipeline of Object Detection[49]

Image Classification is based on the class/classes that are labelled on the image. It essentially takes an input image (e.g., an image of a cat and dog) and outputs (e.g., image class cat and image class dog) the class, probability, and other metrics. **Object Localisation** mainly locates the present object that is recognised with a bounding box. The bounding box gives the position (height and width). **Object Detection** merges/-fuses the image classification and object localisation, where the output image is classified and localised with a bounding box. **Image Segmentation** is a further extension for object detection, where it makes it possible to determine the shape of the object being detected. It essential provides more detail of the object detected using segmentation. Under image segmentation, there is instance segmentation and semantic segmentation. Instance segmentation is where the pixel level of segmentation that identifies the object and labels it with different colours. Semantic segmentation is the pixel level of segmentation where everything in the image and background is labelled based on class and label with different colors[49].

Different Strategies to do Object Detection:

The different strategies to do object detection are the sliding windows approach, two-staged detection framework and one-stage detection framework.

Sliding Window Approach is a fixed window size that goes through the image and tries to find the desired object. The disadvantage of this approach is that the

window size has to be determined, and the use of CNNs computationally expensive due to the unnecessary details of locations, scales and background information from an image that is not necessary[50]. **Two-stage detection framework** as seen in Figure 3.8 there is an input image sent for feature extraction using CNNs where the extraction of object proposals is where it extracts several regions of interest called object proposals. Classification and localisation only happen on the object proposals. When looking at the localisation, the coordinates can be extracted by regression, then maps from image to coordinates with a neural network after it can train the network for regression with the ground truth bounding box coordinates with an L2 loss function. It is then passed to the CNN output to the coordinates of the bounding box. It happens with 1-2 fully connected layers. Then it gives the boundary box and classified score. While bounding boxes can be trained with an L2 loss, and class scores can be trained with a Softmax loss. It is said that the regression head gives the delimitation field, while the classification head gives the semantic class. The two-stage detection framework is said to be precise and accurate. The currently known algorithms that use a two-stage detection framework are R-CNN and Overfeat[51]. The disadvantage of two-stage detection is that the existing algorithms don't work real-time[52, 53, 54].

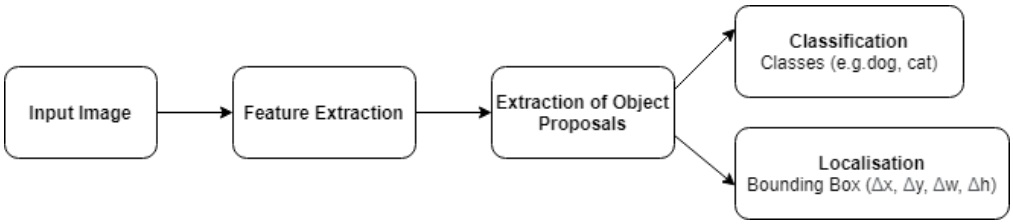

Figure 3.8: Two-stage detection framework

One-stage detection framework as seen in Figure 3.9 the only difference between one-stage and two-stage detection framework is that it does not require extraction of object proposals. One-stage detection is known to work in real-time, and the disadvantage is that it is not as precise and accurate compared to the two-stage detection framework. The currently known algorithms that use one stage approach are YOLO and SSD. The disadvantage of these algorithms is that they have difficulties in detecting small objects[52, 55].

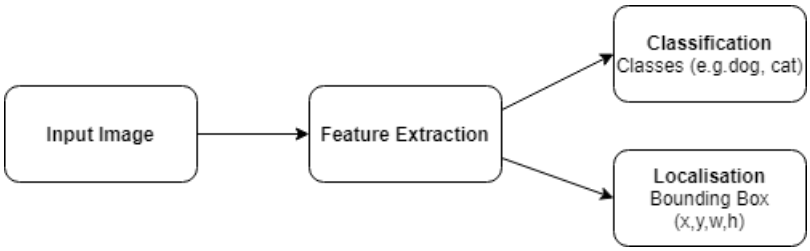

Figure 3.9: One-stage detection framework

Object detection and segmentation with CNNs has been growing fast in recent years. As seen in Figure 3.10 the timeline of the current state of the art object detection and segmentation algorithms. The idea is that deep learning algorithms are rapidly being deployed each year, and the algorithm gets better and better. At the same time, the ones that coloured in red are still used and known and the ones in black are not used.

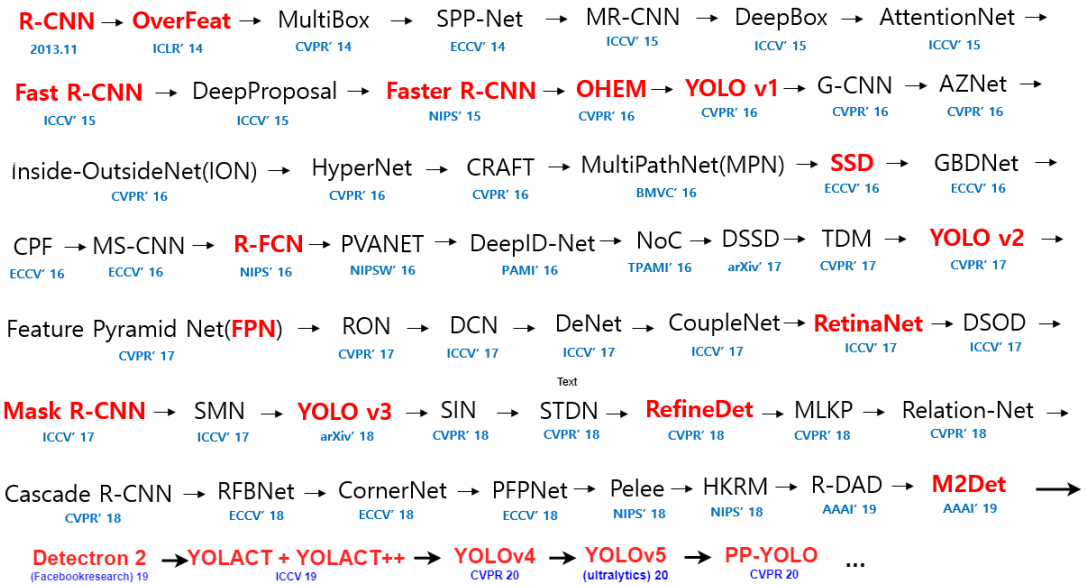

Figure 3.10: Current State of The Art Object Detection and Segmentation Algorithms. The ones in red are still in use and ones in black are not in use[56].

3.2.4 Machine and Deep Learning Tools

There many tools online libraries and tools that are open-source that can be used for machine learning and deep learning. A summary of known existing tools and frameworks will be described.

SciKit Learn is the initial steps to learning machine learning. SciKit is an open-source library for Python programming language that used data mining and data analysis. It is built upon Matplotlib, NumPy and SciPy. This provides a numerous range of supervised and unsupervised learning such as classification, clustering, regression, dimensional reduction and more[57].

Pytorch is created by Facebook in 2016 and is open source. It is a popular tool used by academic research. Pytorch is preferred for deep learning due to its flexibility and speed. It provides tensor computations with high GPUs and building neural networks. It provides pre-trained models, low-level API and custom expressions that can be created[58].

TensorFlow is one of the popular tools used today for machine learning. It was created by the Google brain team in 2015 and is open source. Similar to PyTorch, it is more powerful and gives better visualisation capabilities of models, which the models

can be made for production for mobile platforms and fast and distributed training which works with Python and C++[59].

Theano is similar to TensorFlow but used for evaluating mathematical operations, including multi-dimensional arrays but it is entirely made with Python. It is open-source and was released in 2007[60].

Keras is a neural network library that works with TensorFlow and Theano, which are open-source library for machine learning tasks. It is written in Python. It allows fast prototyping, customisation of layers and can be used on small data[61].

Caffe is mainly used for an academic research project. It has a deep learning framework used for image classification and image segmentation. It is open-source released in 2016[62].

Google Colab is similar to Jupyter notebook (computational notebook) but runs entirely on the cloud released in 2018. There are no hardware configurations needed and do not need to install the packages manually. It also supports other machine learning libraries, e.g., TensorFlow, Keras, PyTorch and more[63].

3.2.5 Comparison between Traditional Machine Learning and Deep Learning

In order to break down and look into the advantages of machine learning and deep learning. Table 3.1 shows the comparison of machine learning and deep learning.

	Traditional Machine Learning	**Deep Learning**
Training Data	Small	Large
Training Time	Short	Long
Choose feature	Yes	No
Nr. of classifiers available	Many	Few
Tuning	Limited	Various of ways

Table 3.1: Differences between Machine Learning and Deep Learning

The advantage of machine learning is that it requires less data, and training is faster. Hardware isn't an issue for the machine learning approach since it can be trained both CPU and GPU. However, deep learning might need a lot of data and takes time to train, which needs GPU. Thus the accuracy of deep learning is more precise and robust. Deep learning can be tuned in various ways, and the number of classes of objects it needs to detect can be many compared to machine learning, e.g. support vector machines (SVM) which can handle two classes, also known as a decision boundary. For machine learning, the features need to be chosen, and the feature detector or extraction needs to be given, whereas deep learning isn't required as Figure 3.11 shows how feature extraction can be automated for deep learning, whereas machine learning feature extraction can be an extremely time-consuming process[46].

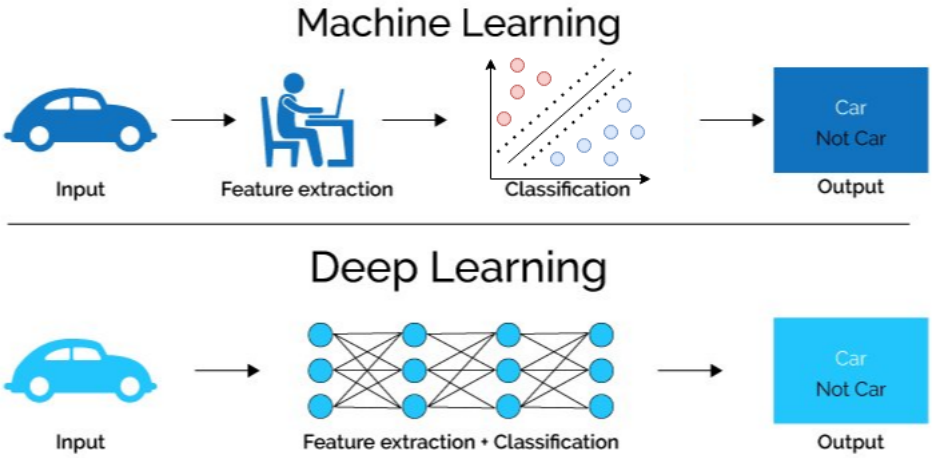

Figure 3.11: Traditional Machine Learning Vs. Deep Learning[46]

3.2.6 Datasets

Dataset is a collection of data. In this case, the collection of data will be images. The current known datasets can be seen in Table 3.2.

Dataset Name	Total Images	Categories	Image Size	Year	Highlights
PASCAL VOC[64]	11,540	20	470 × 380	2005	Covers only 20 categories that are common in everyday life; Large amount of images for training pictures; Close to real applications; Significantly larger intraclass variations; Objects in stage context; Multiple objects in an image; Difficult Samples;
ImageNet[65]	14 millions +	21,841	500 × 400	2009	Large number of object categories; Multiple occurrences and multiple categories of objects in images; More challenging than PASCAL VOC; Images are object oriented Doesn't have object segmentation data
MS COCO[66]	328,000 +	91	640 × 480	2014	Even closer to real-world scenarios; Each image contains multiple instances of objects; richer information on annotating objects; Contains object segmentation notation data.
Places[67]	10 millions +	434	256 × 256	2014	The largest annotations dataset for scene recognition;
Open Images[68]	9 millions +	6000+	Varied	2017	Annotated with image level labels; object instance segmentation, and visual relationship detection

Table 3.2: Popular databases for object recognition[52]

As seen in Figure 3.2 the current known datasets are ImageNet[65], Microsoft Common Objects in Context (COCO)[66], Open Images[68], PASCAL VOC[64] and Places[67] that have many images that are labelled for different object. These images are open source, so they can be used for anything. The datasets contain images of everyday life in natural environments but in this case, there no images of litter items in their dataset. There are other private datasets that can be found online; however, some have licence

agreements that need to be agreed upon to use them. Kaggle, however, is a community that has various datasets from different peers. Kaggle has around 50,000 public datasets used for machine learning purposes[69].

For image annotation, there are several formats. These formats are COCO which is stored in JSON, allows for five different annotations object detection, keypoint detection, stuff segmentation, panoptic segmentation, and image captioning. Then there is Pascal VOC that stores it in an XML file, and the last format tool is YOLO that has .txt format contains object class, object coordinates, height and width[70].

3.2.7 Current Research Projects

The current research projects for detecting litter will be explored. A blog by A.Kelly looks into training an AI system to recognize cigarette litter. In this blog, Kelly uses Mask R-CNN's approach with a synthetic dataset (2000 images). The use of this blog is the synthetic dataset that can be used throughout this thesis[71].

Furthermore, a project by K.Demkova looked into the detection of cigarette litter and made a picker bot with a robot arm. K.Demkova uses Tiny-YOLOV3 and gets an F1 score of 61%. Whereas Tiny-YOLOV3 was able to reach an F1 score of about 82%[72]. As mentioned previously about Jingiang et al. paper that looks into the garbage, whereas the garbage that is looked upon are different to litter; thus, the approach is similar to detecting litter[33]. S.Majchrowska has gathered a list of research papers based on classification, detection and segmentation of various litter, garbage, waste and trash although the concern of this thesis is cigarette litter hence the machine learning technique useful[73].

3.3 Object Tracking

When cigarette litter is detected and localised, the next step is to track the position of the cigarette litter over time; this is called object tracking. Object tracking allows tracking an object over a sequence of frames, in this case with the use of object detection[74]. Object tracking also works by applying a unique ID to each tracked objects, allowing to count the number of objects that are found[75].

3.3.1 Types of Tracking

The problem of object tracking is that it tracks a single object where the object is localised in the first frame, which is mode-free tracking. Mode-free tracking is good for a short time, whereas, for the long term, the performance degrades. A detector cannot be built to detect new images because object appearance is changing over time which is a short-term tracking problem. Only previous frames can be used, not any new images. The challenge to making a visual tracking is that its computational load, which there are many frames to be considered—secondly, the appearances changes over time, e.g., object dynamics, viewpoint, and lighting. There are also occlusions that need to be determined in an image with many objects[76].

Single Target Tracking and Multiple Object Tracking

The first type of single target tracking is by matching or doing a correspondence problem. There is no need for an online appearance model, and an example is GOTURN. The advantage of GOTURN is that there is no online training needed. Tracking is done, by comparison, so no need for retraining the model for new objects. The disadvantage of GOTURN is that if an object moves at high speeds and goes out of the bound, it cannot be recovered. The second type of single target tracking is by appearance learning problem. This is done by quick online fine-tuning of the network as the appearance involves how MDNet works. The advantage of MDNet there's no previous assumption of the location, and the object can move anywhere. The fine-tuning is not costly, but the disadvantage is that it is not fast at GOTURN. The last way is by modelling the temporal prediction problem. An example is ROLO, where there's a CNN appearance model (YOLO detection) and LSTM - Long, short term memory (model the motion). The challenges with multiple object tracking is that there are numerous objects of the same type that leads to heavy occlusions. The appearance is often very similar. The Tracking of Multiple Objects which an example is to use the Hungarian algorithm that works by finding an optimal assignment for a given cost matrix. There's also Global Multi Tracking that works offline. There are also other algorithms for deep learning SiamMask, Deep SORT, TrackR-CNN, Tracktor++, and JDE (Joint Detection and Embedding)[77, 78, 79].

Online and Offline Tracking

Online tracking is when it is useful for real-time applications. It works by processing the current frame and previous frame by tracking an object with this short information. However, the past errors cannot be corrected. There will be some drifting where it is hard to recover from errors or occlusions since the future cannot be determined. An example of online tracking is using the Kalman filter. Offline tracking allows processing a collection of frames so the current frames and all the future and past frames can be recovered. This is not suitable for real-time application. Offline tracking is mostly used for video analysis—the advantage of offline it is more precise and accurate[78].

3.3.2 Visual Tracking and Visual Servoing

The aim is to track the object (visual object tracking) over time, e.g. the cigarette litter. Applying visual servoing allows feedback information extracted from a vision sensor to control the robot's motion, e.g. using object tracking to track the cigarette litter and using servoing to move to the desired location of the cigarette litter[80].

There are two configurations of the camera one is the eye in hand (closed-loop control), and the other one is eye to hand (open-loop control). Eye in hand, the robot is attached to the camera, and eye to hand (stand-along) is fixed in the world. Both ways can observe the relative position and target of an object.

Visual servoing contains two control strategies: position based servoing (PBVS) and

image-based servoing (IBVS). IBVS is the difference between current and desired functions which the control does not involve any pose estimation of the target that works in 2D. The disadvantage of IBVS is working with motions with large rotations. PBVS, however, can do a pose estimation of the object and get the 3D information. For visual tracking and servoing there is a platform called ViSP (Visual Servoing Platform)[80, 81, 82].

3.3.3 Distance estimation vs. Visual Tracking and Servoing

As mentioned earlier in Chapter 2, the Technology and Environment service are having a problem with cleaning litter from sidewalks. However, for a mobile robot to go and pick up the cigarette litter, it needs to localise the object. Currently, it is known that most object detection algorithms have object localisation and image classification, so X and Y are present, where the aim of distance estimation is to configure the Z-axis to get the distance of the cigarette litter with the use of the mobile robot's odometry, visual tracking and servoing; however, the XYZ is not placed, so it can be said where it is in the image the object will track the object and do visual servoing towards the target. One of the disadvantages of visual servoing is it is more computationally heavy than distance estimation and works slower (hardware limitations). However, the disadvantage of distance estimation is the calibration of the camera.

3.4 Coverage Planning

Coverage planning determines how the robot should plan its path that it must take, so it covers the environment and avoids obstacles. Coverage planner determines an exhaustive walkthrough of the proximity graph and calculates explicit robotic motions within each cell[83]. In this thesis, the aim is to cover the map area seen in Chapter 2 Figure 2.2 and collect the cigarette litter with the use of object detection. Potential Fields can also be an approach to do coverage; however, in this case, using potential fields can lead to computational costs since the robot can get stuck in the local minimum of the potential field. Since it should be heading towards the target destination that is the global minimum[84]. To able to cover the entire area, Figure 3.12 shows how a complete coverage path planner works with an exhaustive walk.

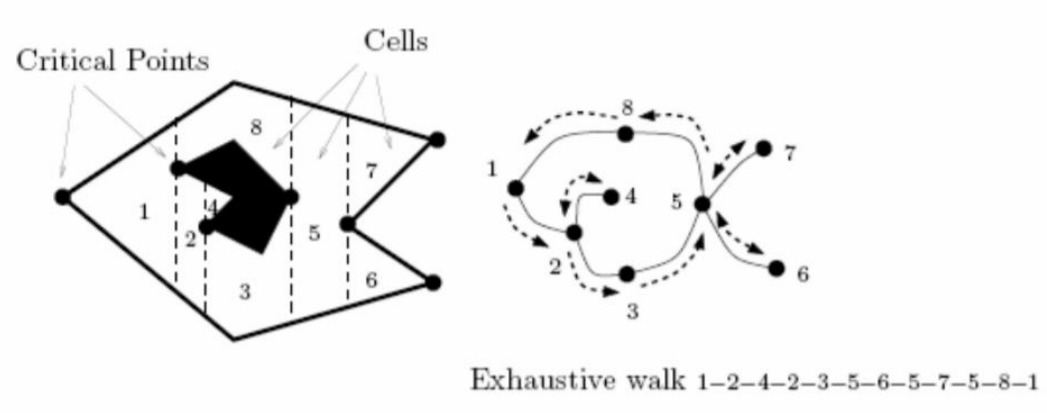

Figure 3.12: Complete Coverage[83][85]

The approaches to do complete coverage planning that will be looked upon are various cell decomposition methods. Cell decomposition consists of adaptive cell decomposition and exact cell decomposition. Adaptive cell decomposition works by defining a configuration space and marks the objects that are blocked. It then finds the remaining cells by using, e.g. A* algorithm. Exact cell decomposition, however, works by fixed direction through the map. Each time a corner occurs, the cells divide, and the cells that are divided and have a border the edges are connected[85].

	Adaptive Cell Decomposition	**Exact Cell Decomposition**
Pros	- Limited Assumptions of Obstacles - Its fast and finds the obvious solution	- Its complete and covers the entire area.
Cons	- Doesn't know/find the optimal path - Completeness & Computation - High Dimensions is difficult to use.	- Computational heavy - Doesn't scale well in High Dimensions

Table 3.3: Adaptive cell decomposition vs. Exact cell decomposition[86]

Adaptive cell decomposition techniques are using regular cell and tree decomposition (Quadtree and Octree). Exact cell decomposition techniques are polygonal cell decomposition and trapezoidal decomposition that is based on path planning algorithms. Morse cell decomposition, boustrophedon decomposition and sensor-based decomposition are coverage based algorithms. The coverage based decomposition algorithms allow exploration and cleaning[85].

3.4.1 Coverage on Sidewalks

The problem that Technology and the Environment service are having a problem with, as seen in Chapter 2 is cleaning litter from sidewalks. Sidewalks can vary in different types of terrain. The aim is to do coverage planning; however, exact cell decomposition on a narrow path is not efficient since the robot will then move in a zigzag pattern

(shoelace pattern) like a robot lawnmower. In order to make a coverage planner that can solve narrow paths, Bug algorithms can be used. Such as Bug 1 algorithm is complete, has memory, goes all the way around obstacles. Bug 2 is also complete, goes around obstacles until it hits them-line (straight line from start to goal), but it does not cover the area compared to Bug 1. The idea is to make a hybrid coverage planner that can do exact cell decomposition, e.g. using boustrophedon cell decomposition and Bug 1 algorithm on narrow sidewalks could be a solution to clearing the cigarette litter. The idea is to start from a start position and end at a goal position where later Dijkstra's Shortest Path algorithm could be deployed to make the costs cheaper and finish the task faster.

A simple approach could also be painting lines on the roads since there is a map, and essentially, the outdoor mobile robot has to just follow the line and clean up the cigarette litter with the use of the detection system.

4 Problem Formulation

The problem at hand is to make a system that can identify and remove cigarette litter. However, this procedure will be broken down, so identification of cigarette litter will be in focus in this thesis. As for removal will be another project and making the system autonomous using coverage planning.

As for detecting cigarette litter, a vision system is needed. As mentioned earlier in Chapter 2, the problem that will be investigated is to help KK Tech, Capra Robotics and Aarhus Municipality with detecting and seek coverage of cigarette litter in outdoor terrain. One of the problems for Technology and the Environment (Teknik og Miljø - Aarhus Kommune) is cleaning sidewalks. As for primary cleaning areas, that is, pedestrian roads, parks and bigger places, the handheld blowers and sweeping machines can be used. As for secondary cleaning for smaller roads is to use outdoor mobile robots. The focus of this thesis is to make the vision system work for the detection of cigarette litter on sidewalks in urban areas. As for the outdoor mobile robot, the hardware is provided by Capra Robotics and KK Tech.

> *"How can an object detection system provide real-time coverage of cigarette litter on sidewalks in urban/public areas with the use of outdoor mobile robots?"*

One of the sub-problem that will be investigated that is analysed in Section 3.2.6. Datasets like MS COCO, Imagenet and other known datasets for machine-learning research do not have classes for cigarette litter. The dataset for cigarette litter needs to be developed. *"How to develop a robust dataset for cigarette litter?"*

4.1 Project Hardware

In this section, the project hardware will be described. The hardware is provided by Capra Robotics. While, KK Tech is making the hardware (vacuum tool) for collecting the cigarette litter, which will help in this thesis to solve the type of method is needed to detect the cigarette litter.

Capra Robotics

For this thesis, the outdoor mobile robot that will be used and tested on is provided by Capra Robotics ApS. Capra Robotics ApS is a company based in Hasselager, Aarhus, Denmark. The company has made an outdoor mobile robot that is based on a patented wheel frame (patent: WO2015197069)[25, 87].

Figure 4.1: The generations of Capra Hircus. Generation P1.0 on the far right and P4.0 on the left[25]

.

The outdoor mobile robot that the company have designed and made based on the unique wheel drive is called Capra Hircus, as seen in Figure 4.1. Hircus is designed to easily carry different types of load. Due to its flexible chassis and large engine power, it can be used for numerous types of tasks in uneven terrain, as well as for driving in urban areas[25]. The company have made several different types of versions of the Hircus. The recent platform is waterproof, which is P4.0 (Proto 4.0), thus for this thesis, the Hircus that will be tested on is P3.3G. The difference between the two systems is the body shape, and P3.3G isn't viable to rainy days, making it not waterproof. The localisation of the robot works by internal odometry and GPS sensor.

The hardware for the vision system for the robot is provided by Capra Robotics ApS.

Figure 4.2: The hardware is provided by Capra Robotics. (The Black Mounts for the hardware 3D printed by the Author of this Thesis)

In Figure 4.2 the provided hardware can be seen. An Nvidia Jetson Nano developer kit with a TP-Link WiFi adapter is provided, and two monocular cameras (CSI camera and USB camera).

Hardware	**Specifications**
Nvidia Jetson Nano (Carrier Board B01) (Fan - NF-A4x10 5V PWM)	Nvidia Maxwell GPU Quad-Core ARM Cortex-A57 Processor 4GB LPDDR4 Memory
Monocular Camera (Surveillance Camera Module for Raspberry Pi)	1080P 30FPS, 720P 60FPS, 5MP Fish Eye (Lens 10° - 200°) (IR Night Vision)
HP Webcam HD 2300 (Y3G74AA) TP-Link WiFi Adapter	USB 2.0 TL-WN722N 150Mbps

Table 4.1: The table shows the specification of the provided hardware for this project.

KK Tech

KK Tech is a company that is located in Odense, Denmark[88]. The company makes hardware products and tools. In this thesis, the gripper that is made to collect the cigarette litter is made from KK Tech. Figure 4.3 shows the CAD model of the Capra Hircus P4.0 with the vacuum system and tool collector for the cigarette litter. As seen

in the figure, the tool collector is positioned on the right-hand side of the robot, and basically, the idea is to collect the cigarette litter on sidewalks.

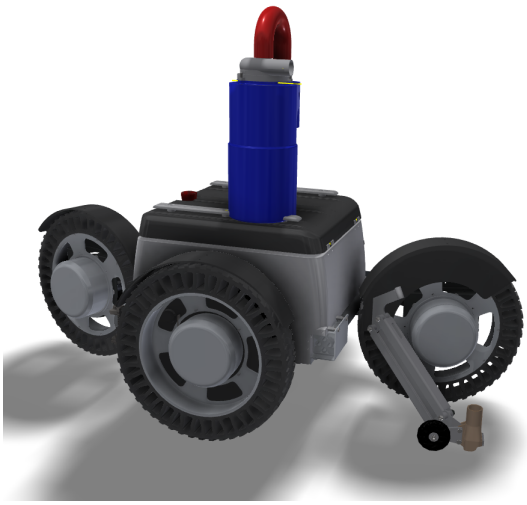

Figure 4.3: Vacuum system to collect the cigarette litter. On top of the robot is the vacuum system and on the bottom is the tool to collector the cigarette litter.

4.2 Requirement Specification

The requirement specifications are made based on making a vision system work for the detection of cigarette litter on sidewalks. The system can detect cigarette litter with the use of Darknet YOLO. The detection algorithm is built on Nvidia Jetson Nano that will then be mounted on the outdoor mobile robot. The outdoor mobile robot currently works remotely with joystick control.

Below are the requirements listed; there are functional and performance requirements. Where the functional requirement is to test if the robot works as intended, and the performance requirement tests whether it performs the various tasks as it supposed to do.

1. Functional requirements
   (a) The system must be able to detect cigarette litter in all types of outdoor terrain in this thesis.
      - The specific type of terrain the system has to work is sidewalks. Snow and night will be excluded from all types of terrain.

   (b) The system must detect all types of cigarette litter.
      - Since there are many types of cigarette litter that can be found, i.e. filters, snus and different colour cigarette butts, the system needs to be able to detect those.

(c) The system must detect cigarette litter on sidewalks.

- The aim is to make the mobile robot clean the sidewalks since in Chapter 2 the Technology and Environment Services have trouble cleaning sidewalks in urban areas.

2. Performance requirements

(a) Correctly classify an object of interest with a precision and recall rate of 95%.

- The recall and precision is set 95%, were 5% can be false classifications.

(b) The system has to detect cigarette litter at human walking speeds (5km)[89].

- The system needs to work in real-time and at a reasonable pace. Whereas in urban areas, humans will be around, and the robot needs to move safely, so it doesn't harm anyone. Also, the detection system needs to detect cigarette litter in real-time.

5 Methods

In this chapter, methods to detect and localise cigarette litter will be explored. Below Figure 5.1 shows the ideal method to identify and remove the cigarette litter. Hence in this project, the focus is the identification of cigarette litter. At the same time, navigation such as obstacle avoidance and removal of the cigarette litter won't be looked upon. The choice that was taken was not to do object tracking and servoing since it will be computationally heavy for the Nvidia Jetson Nano, which Jinqiang Bai et al. [33] does a similar approach on detecting and removing garbage with a mobile base with a robot arm gripper. They conduct some experiments, which takes them an average of 60 minutes to pick up the garbage of 20-50 dropped items (controlled experiment). So the decision made in this thesis is to look at distance estimation with a monocular camera that will be discussed in Section 5.2.

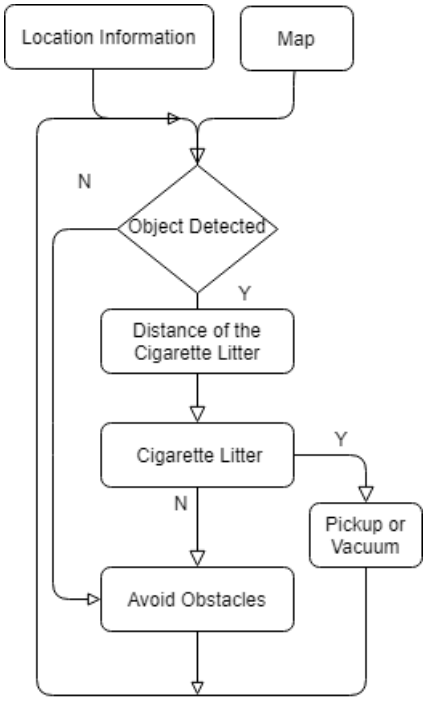

Figure 5.1: Method that will be used to detect and remove the cigarette litter

Figure 5.1 works by having a map (GPS) and location (IMU and Odometry) information from the mobile robot (Capra Hircus). When the object is detected, the distance will be measured by a 2D or 3D camera (In this case monocular 2D camera). Based on the distance, the robot can pick up/vacuum (KK Tech - vacuum system) the cigarette litter. If it didn't find distance or the detection of the cigarette litter, it will avoid and start from the beginning.

5.1 Detection of Cigarette Litter

The approach that will be used to detect cigarette litter is YOLO (You Only Look Once). The reason why this algorithm is chosen will be explored, and how it works will be described.

Darknet YOLO

Darknet is an open-source neural network framework written in C and CUDA developed by Joseph Redmon et al. The framework consists of the You Only Look Once (YOLO) object detection algorithm. Joseph Redmon was the creator of YOLO V1, V2 (also known as YOLO 9000), and V3 (with different version such as the Tiny-YOLO)[48]. Whereas later, he decided not to continue, and he himself has stopped the computer vision research due to avoid potential misuse of the algorithm[90].

Alexey Bochkovskiy et al. introduced YOLOV4[91]. Alexey Bochkovskiy is a former colleague of Joseph Redmon which his repository of Darknet is carried on and is maintained[92]. Darknet also supports ROS, which will help Capra Robotics (Capra Hirucs platform) to implement the object detector[93].

Darknet YOLO - Network Design

The initial YOLOV1 classification has 24 convolutional layers followed by 2 fully connected layers. Also, 1×1 reduction layers followed by 3×3 convolutional layers. It is inspired by GoogleLeNet[94].

It was later improved with YOLOV2 (YOLO 9000) that has a classification model called Darknet-19. It has 19 convolutional layers and 5 max pooling layers. That brings batch normalization, convergence is sped up and regularised model[95].

For YOLOV3 it uses Darknet-53 that has 53 convolutional layers. There was significant amount of upgrades to YOLOV3 such as accuracy, precision, speed, and can handle more classes. With stacked layers there are 106 layers, where the detections are made in layers 82 at stride 32 (32x32 large objects), 94 at stride 16 (26x26 medium objects) and 106 at stride 8 (32x32 small objects). Meanwhile, there are no pooling layers for YOLOV3 and adds extra convolutional layers. This helps from losing low-level features and helps to detect small objects. The batch of images must contain (n (images), 416 (width), 416 (height), 3 (RGB)). The width and height should be divisible by 32[96].

YOLOV4 however uses CSPDarknet53, CSP stands for Cross Stage Partial Connections that means it separates the feature map into two parts[97]. By build up from YOLOV3, YOLOV4 was made by SPP additional module, PANet path-aggregation neck, and YOLOV3 (anchor based) head[91].

Pytorch YOLO (ultralytics) and PP-YOLO

Pytorch YOLO (ultralytics) is implemented by Glenn Jocher[98]. Glenn Jocher introduced YOLOV5 50 days after YOLOV4. The framework is different to the previous

YOLOV1-V4. Whereas YOLOV4 performs slightly better than YOLOV5. The aim of YOLOV5 is to not bring the best mAP (mean average precision); it tries to ease memory requirements such as making training faster and speed of inference[99, 98].

PP-YOLO that is by Xiang Long et al.[100]. It can achieve an mAP of 45.2% COCO dataset which YOLOV4 mAP is 43.5%. The difference is the framework that is used, which doesn't use the Darknet53 and is replaced by ResNet[100].

Meanwhile, since there many types of YOLO algorithms that are made and even today, there are newer versions deployed. YOLOV4 is chosen due to ROS support. There are different variances of YOLOV4, such as Tiny-YOLOV4 that works with a smaller model. A smaller model (RAM memory), meaning it can give fast inference and use less computational power for the Nvidia Jetson Nano.

Nvidia Jetson Nano Benchmark on Deep Learning Algorithms

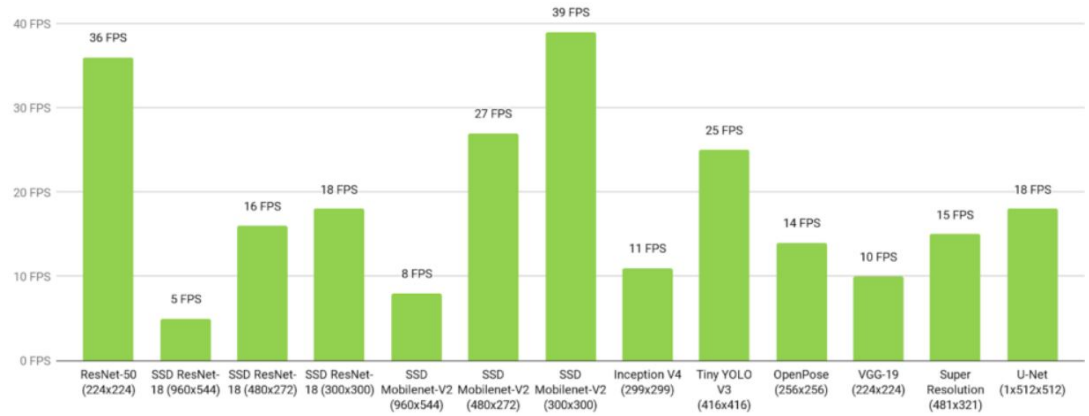

Figure 5.2: Deep Learning Inference (Batch size 1 and FP16 precision) - Nvidia Jetson Nano Benchmark (10W performance mode)[101]

As seen in Figure 5.2 the benchmarks for different Deep learning algorithms for the Nvidia Jetson Nano. SSD-Mobilenet-V2 (size resolution of 300x300) shows the greatest peak for object detection with 39FPS, and with different scale, it gives 27FPS (size resolution 480x272). In comparison, Tiny-YOLOV3 (size resolution of 416x416) gives 25FPS. Hence with this large size resolution of 416x416, it works similar to SSD-Mobilenet-V2. Also, since there are other variations of YOLO that are not benchmarked by Nvidia, e.g. PP-YOLO tiny and Tiny-YOLOV4. Tiny-YOLOV4 is known to work 10% faster than Tiny-YOLOV3[91]. Which YOLO will be chosen due to its resolution and that cigarette litter are quite small. Better resolution and working in real-time will be ideal for detecting cigarette litter. Meanwhile, the figure shows ResNet-50 and SSD-ResNet are used for image classification. OpenPose is used for pose estimation, and Super-resolution used for image processing, so these can be ignored. The rest of the figure is object detection and segmentation algorithms.

5.1.1 YOLO - How it works

YOLO is a single-stage detector that is the state of arts one of the fastest object algorithms. The way YOLO works is it splits the image into SxS grid cells, and for each cell, it predicts object P(Obj). Based on the predicted object, it makes a bounding box presented as a black box behind the human on Figure 5.3. Ground truth is the blue bounding box on Figure 5.3 with a person inside. Based on this, the confidence can be calculated by P(Obj)*IOU (Intersection over Union). The goal for the confidence score is that it should be high, so based on the conditional class probability, P(Person) = P(Person|Obj) = Probability of the object is a person. The position (x, y, w, h) can be found by the centre of the bounding box, and with the grid cell position can be obtained. However, since there will many bounding boxes appearing to reduce the number of bounding boxes, it can be controlled by threshold value and non-max suppression[102]. The loss function of YOLO is based on the sum of squared error which is a regression problem. The components of the loss function are classification loss squared error of the class conditional probabilities, localisation error (x, y, w, h) is a regression problem to find the real value, so errors between the bounding box and ground truth and confidence loss measure the object confidence inside the box[102].

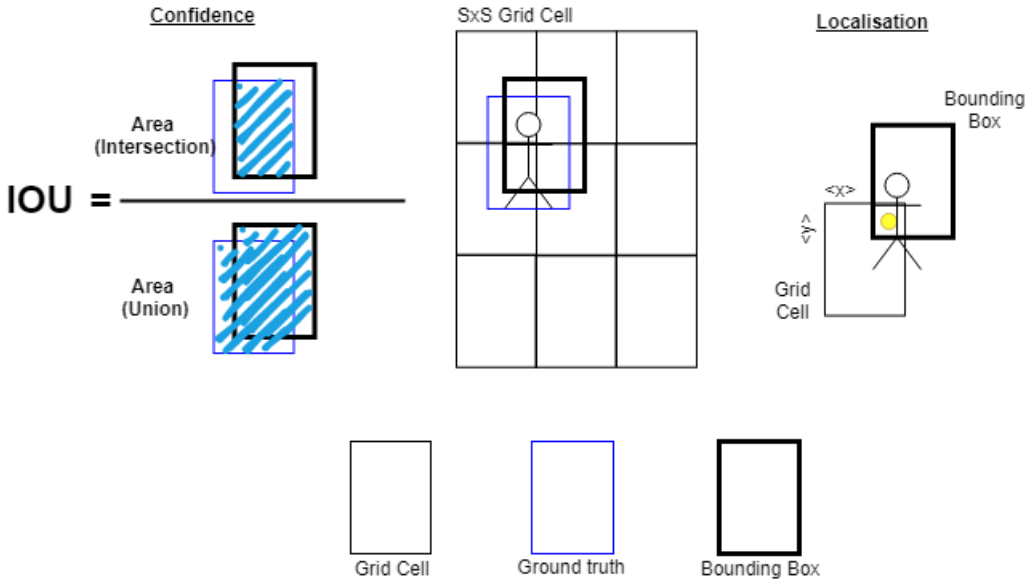

Figure 5.3: YOLO - SxS Grid cell (3x3) with bounding box (black box) and Ground truth (blue) with the person inside.

5.2 Localisation of Cigarette Litter

In this section, several approaches will be discussed to calculate the distance with a monocular camera when the cigarette litter is detected. Since then, with a monocular

camera, X and Y can be obtained by the YOLO object detection algorithm. Where the ideal aim is to measure the distance of the cigarette litter with a monocular camera.

5.2.1 Measuring Size and Distance from Images

Measuring angular size on images from telescopes point of view. This method is how astronomers measure the size and distance of the object. The method works by measuring the angular size of a single object, and that angular distance can be obtained by two objects as seen in Figure 5.4. This works by extending imaginary lines with human perception[103].

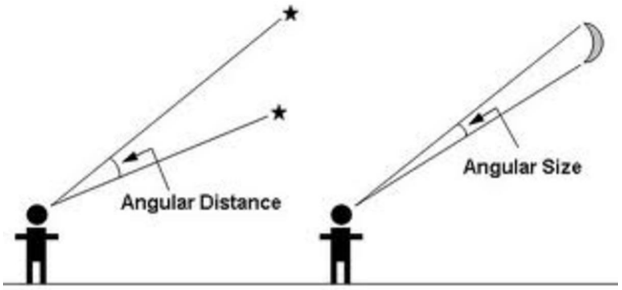

Figure 5.4: Angular Size of a single object (on the right) and Angular Distance of two objects (on the left)[104]

There is a rule that when an object with an angular size of 1 degree is around 57 times its own size away[103]. The angular size (A) of an object (in degrees) to an entire circle that is 360 degrees should correspond to the actual size (B) of the object to the circumference of a circle at that distance from the observer based on both ratios. (Where, A = Angular size, B = Actual Size, D = Distance, DO = Distance of Object, SO = Size of Object and DT = Distance to Object).

$$\frac{A}{360^\circ} = \frac{B}{2\pi D} \tag{5.1}$$

Where, the equation 5.1 can be re-written:

$$\frac{DO}{SO} = \frac{360^\circ}{2(\pi)(A)} \tag{5.2}$$

$$\frac{DO}{SO} = \frac{57}{B} \tag{5.3}$$

An example might be if the cigarette litter is in a range of 1-degree field of view and is known to be 5 feet wide. That then means it's $57 * 5 = 285$ feet away[103]. This allows formulating the following equation to solve the distance from the object.

$$DT = SO * \frac{1^\circ}{A} * 57 \tag{5.4}$$

The advantage of this technique it is not computationally heavy. The disadvantage is that the value that is measured isn't exact since the cigarette litter size needs to be configured manually.

5.2.2 2D to 3D Monocular Distance

In order to localise the cigarette litter, the monocular camera (2D camera) needs to be able to measure distance. In contrast, 2D plane X and Y is known. Whereas to get the Z plane, the camera model needs to be calculated. The calculations can be done by looking into the intrinsic's and extrinsic's parameters of the camera model. The camera model can be seen in Figure 5.5. Intrinsics means the inside model of a camera, and extrinsic means where is the camera in the real world.

The camera model

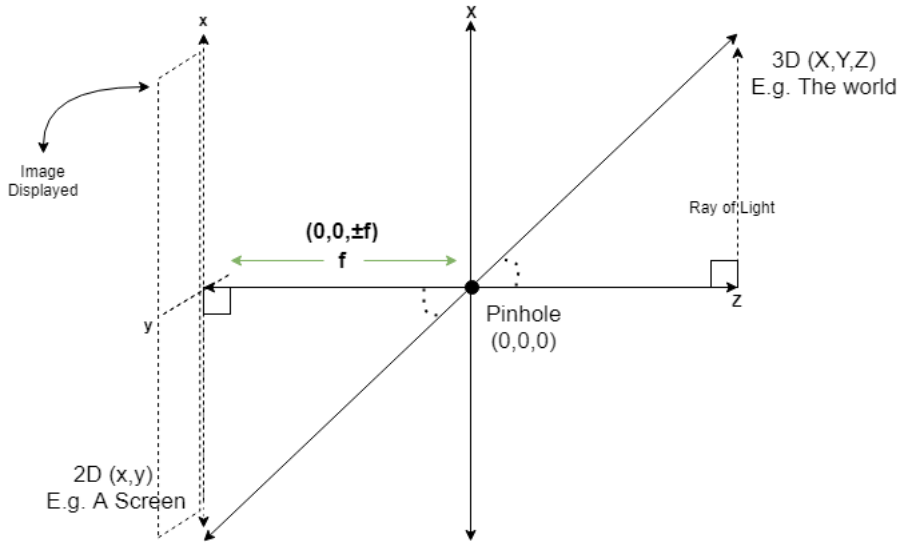

Figure 5.5: Camera model - 3D Coordinate System of the world on the right and on the left is the 2D Coordinate System of the image on the screen.

As seen in Figure 5.5 an object is located at the 3D world, and ray light is sent from the 3D world to the pinhole camera, which is (0, 0, 0), also known as the origin that then is sent to a 2D screen that displays an image. The coordinate system is meant to be 3D, so X is pointing up and down, Y is pointing perpendicular to the plane of the camera model, and Z is right and left. Where f is the focal length (0, 0, f). The aim is to derive the basic 3D world to 2D world for now.

As seen in the figure, there are two triangles that can be formed based on these similarities of triangles; the equations can be formed. As the relation can be seen from the 3D coordinate to the 2D coordinate system.

For x,

$$\frac{x}{f} = \frac{X}{Z} => x = \frac{Xf}{Z}$$

The same can also be done for y,

$$\frac{y}{f} = \frac{Y}{Z} => y = \frac{Yf}{Z} \tag{5.5}$$

Homogeneous coordinates are coordinate for projective geometry. It is essential when working with a camera and understanding how the 3D world is mapped to the 2D image. The advantage is that they provide a variety of transformations as a matrix that allows doing matrix-vector multiplication, and points can be expressed in finite coordinates, e.g. adding a 1 to the euclidean world gives the homogeneous coordinates[105]. So in matrix form, it can be written like this (homogeneous coordinates):

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} => \begin{matrix} x' = Xf \\ y' = Yf \\ w = Z \end{matrix} \tag{5.6}$$

Due to homogeneous coordinates it can be divided by the scale factor w.

$$x = \frac{x^{-1}}{w} = \frac{xf}{Z}$$

$$y = \frac{y^{-1}}{w} = \frac{yf}{Z} \tag{5.7}$$

Intrinsic Camera Matrix

The intrinsic camera matrix is the inside model of a camera, so the input is 3D and output is 2D.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5.8}$$

As seen in Figure 5.6, for practical case, the origin lies within the centre, which isn't the case for computational case for computers. At the same time, a basic image is generated by a matrix where on the computational origin, the origin lies on the top left.

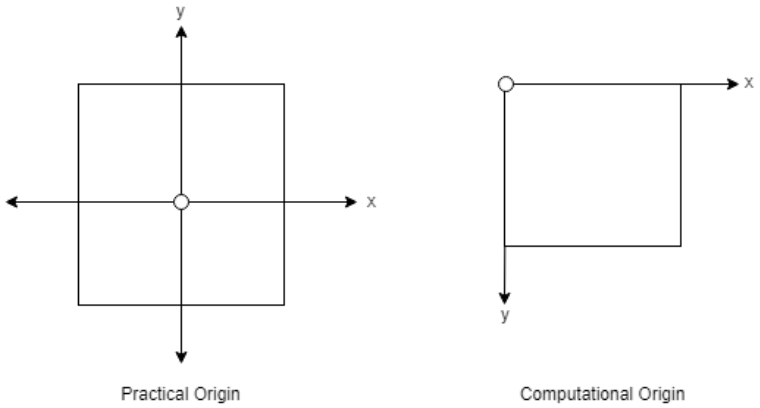

Figure 5.6: Practical Origin and Computational Origin

So to get Px and Py it can be done with simple translation and that they are homogeneous coordinate system it can then be derived like this.

$$x = \frac{fX}{Z} + Px$$

$$y = \frac{fY}{Z} + Py \tag{5.9}$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & Px \\ 0 & f & Py \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5.10}$$

s is the skew factor where the images are skewed, and the skew factor helps it make unskewed.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & s & Px \\ 0 & f & Py \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5.11}$$

So x, y and w are homogeneous from equation 5.11 whereas XYZ is not, which then can be made homogeneous from the following equation. That gives the intrinsic parameters.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & s & Px & 0 \\ 0 & f & Py & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & s & Px \\ 0 & f & Py \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.12}$$

Based on equation 5.12 the intrinsic camera matrix can be formed:

$$Xc_{3x1} = K_{3x3}[I_{3x3}|O_{3x1}]_{3x4}X_w \tag{5.13}$$

(To compute the intrinsic this is usually done by camera calibration.)

Extrinsic Camera Matrix

The aim of the extrinsic camera matrix is to align the camera and world coordinate system (3D -> 3D).

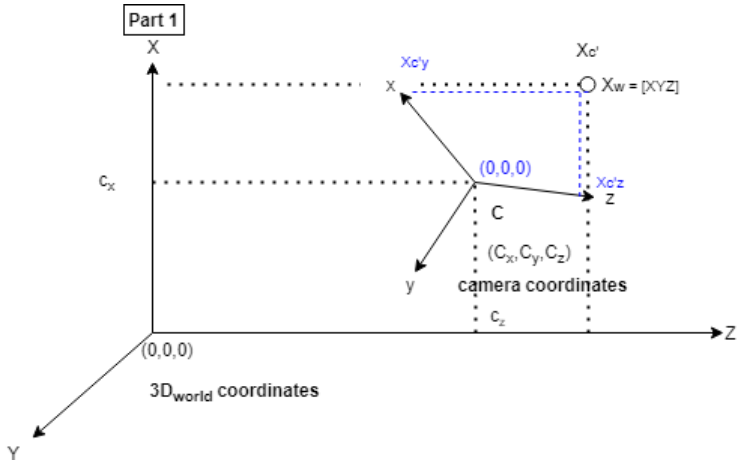

Figure 5.7: Part 1 - 3D world coordinates, Camera coordinates and object point coordinates(Xw).

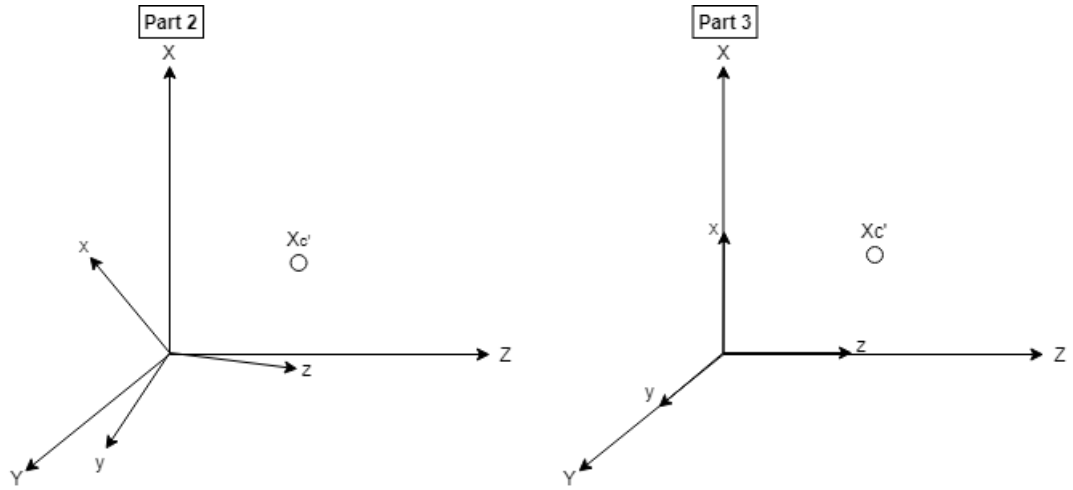

Figure 5.8: Part 2 - Camera coordinates is translated to the 3D world coordinates. Part 3 - Its then rotated with with rotation matrix.

As seen in Figure 5.7 and 5.8 where Part 1 shows the world coordinates related to the camera coordinates. Where the camera coordinate is related to the point coordinate (Xw). Where the origin of the camera related to the object point coordinate. Where later in Part 2 the camera coordinate is translated to the world coordinate with $Xc'_{3x1} = Xw_{3x1} - C_{3x1}$. Since it can be seen in Part 2 that the camera coordinate is not aligned. .It can be aligned using the rotation matrices that map both world and camera coordinates as Part 3 is where in 3D there are three ways to rotate XYZ essentially in the fixed 3D space as seen in equation 5.14.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\theta & -s\theta \\ 0 & s\theta & c\theta \end{bmatrix} R_y(\theta) = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} R_z(\theta) = \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.14}$$

$$X_c = RXc' \tag{5.15}$$

$$X_c = R(X_w - C) \tag{5.16}$$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \right) \tag{5.17}$$

Homogeneous Coordinates can be derived:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} - \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} C_x \\ C_y \\ C_z \\ 1 \end{bmatrix} \tag{5.18}$$

Based on equation 5.18, the extrinsic camera matrix can be formed:

$$\begin{bmatrix} Xc_{3x1} \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3x3} & -RC_{3x4} \\ 0_{1x3} & 1_{1x1} \end{bmatrix} \begin{bmatrix} Xw_{3x1} \\ 1 \end{bmatrix} \tag{5.19}$$

Where,

$$Xc_{3x1} = R_{3x3}Xw_{3x1} - RC_{3x1}$$

$$1 = 0_{1x3}Xw_{3x1} + 1_{1x1} \tag{5.20}$$

(The localisation of the camera is the extrinsics.)

2D to 3D with a Monocular camera

As seen on Figure 5.9 diagonal D is the distance that is unknown. H is the height above the ground where Capra Hircus has the camera mounted. So H is known however whenever the camera is tilted to aim at the cigarette litter. All that is needed is to workout angle A relative to H that can calculate D.

$$D = H * tan(A) \tag{5.21}$$

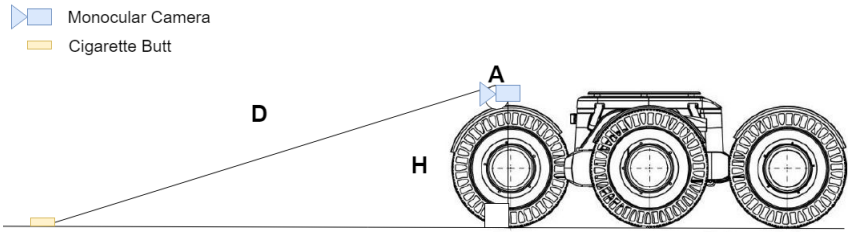

Figure 5.9: Capra Hircus - Camera Projecting at the Cigarette litter. Where D is Distance, A is Angle and H is Height.

The problematic part is how to get this working with an object detector for a specific object, e.g. a cigarette litter. This leads to configuring the 2D camera to 3D so the Z plane can be achieved from the monocular camera. Where the projection matrix is needed, this essentially composed of both intrinsic and extrinsic camera model. The projection matrix P converts 3D world points to image points[106].

$$X_i = PX_w \tag{5.22}$$

$$P = T_c^i T_w^c \tag{5.23}$$

X_i and X_w are pixels coordinates points and 3D world points. T_c^i that converts from world coordinates to camera coordinates and T_w^c that convert from camera coordinates to pixel coordinates and P is Projection matrix[106].

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} P(0,0) & P(0,1) & P(0,2) & P(0,3) \\ P(1,0) & P(1,1) & P(1,2) & P(1,3) \\ P(2,0) & P(2,1) & P(2,2) & P(2,3) \end{bmatrix} \begin{bmatrix} Xw \\ Yw \\ Zw \\ 1 \end{bmatrix} \tag{5.24}$$

Image points (u,v) and scale s that gives different world points (Xw, Yw, Zw). In order to calculate (Xw, Yw, Zw, 1), Projection P needs to be inverted. Hence, P is not invertible since the determinant is not equal to 0. So another constraint can be assumed where the 3D world point lies on another plane in the 3D world, assuming that Zw = 0

in the XY plane. Solving for Zw = 0[106].

$$\begin{bmatrix} u & -P(0,0) & -P(0,1) & -P(0,2) \\ v & -P(1,0) & -P(1,1) & -P(1,2) \\ 1 & -P(2,0) & -P(2,1) & -P(2,2) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ Xw \\ Yw \\ Zw \end{bmatrix} = \begin{bmatrix} P(0,3) \\ P(1,3) \\ P(2,3) \\ 0 \end{bmatrix} \tag{5.25}$$

$$\begin{bmatrix} s \\ Xw \\ Yw \\ Zw \end{bmatrix} = \begin{bmatrix} u & -P(0,0) & -P(0,1) & -P(0,2) \\ v & -P(1,0) & -P(1,1) & -P(1,2) \\ 1 & -P(2,0) & -P(2,1) & -P(2,2) \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} P(0,3) \\ P(1,3) \\ P(2,3) \\ 1 \end{bmatrix} \tag{5.26}$$

Based on this, the world point and 3D distance can be found.

The advantage of this technique is that it is efficient but is not so accurate. Since the Capra Hircus needs to be in level with the ground and the suspension needs to be steady. The camera also needs to be adequately calibrated, which is quite challenging. Even good calibration with a slight variation of angle position of the camera can lead to considerable errors in distance calculations[106].

There is also another method that works by selecting any point on the image and measuring the actual distance of the point, which is called homography. The input image is parallel, whereas, in the output image, it is not parallel in homography. The disadvantage is that the points are fixed, and the points need to lie exact so the distance can be measured[106, 107].

5.2.3 Localising with Convolution Neural Networks

There are two methods that are made by CNN approach for object detection with distance estimation.

Method 1 M A Haseeb et al.[108] are the creators of DisNet. DisNet is a neural network-based object detection distance estimation algorithm for monocular cameras. DisNet uses YOLO and is trained on MS COCO images where it is required to collect dataset including both inputs and outputs of ground truth of the bounding box. DisNet has a feature that can also track objects that are detected and prints the distance of the object. One of the disadvantages of this algorithm is that it is trained with YOLOV3 and not Tiny-YOLO versions and that the dataset does not contain cigarette litter. However, this will also lead to computationally costs for the Nvidia Jetson Nano[108]. Thus, this can be adjusted and trained, so it can work with Tiny-YOLO[109]. The distance estimation works by 2000 input of feature vectors were extracted for each bounding box.

$$v = \begin{bmatrix} 1/B_h & 1/B_w & 1/B_d & C_h & C_w & C_b \end{bmatrix} \tag{5.27}$$

Where B_h is the height of the bounding box in pixel, B_w is the width of the bounding box in pixel, and B_d is the diagonal of the bounding box in pixels. C is the class of an object (height, width and breadth)[108]. The objects are based on different distances,

and the real distance from a laser scanner was used to calculate the distances from 0-60m. Based on this, the model was trained based on laser scanner distance measurements and bounding box size. The drawback seen from this method is that it is artificially made the numbers based on the laser scanner readings that gives actual results, although when training, there is a trade-off of accuracy and precision of the object detected estimated distance[108].

Method 2 M. A. Khan et al.[110] have made an object detection and distance measurement algorithm. It is similar to method 1, however, without tracking. From the bounding box, which is (x0, y0, width, height). The distance measuring works by **do** (distance of object from the lens), **di** (distance of the refracted image from the convex lens) and **f** (focal length or the focal distance) as seen on Figure 5.10.

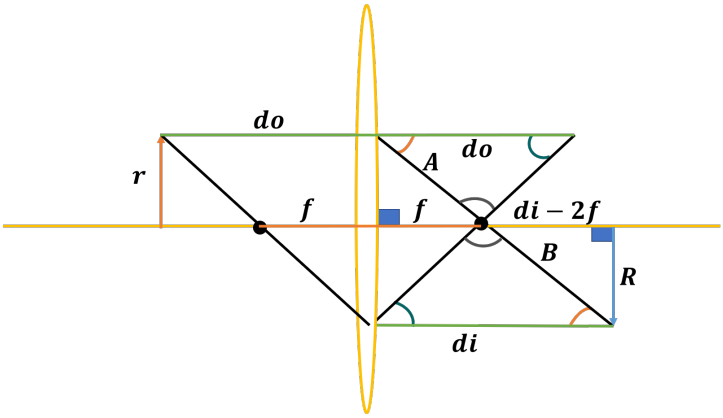

Figure 5.10: M. A. Khan et al.[110] approach for distance estimation. Image shows how the image and corresponding angles look when enters through a lens[111]

The green line is the actual distance of the object from the convex length, which is **do** where **di** is how the real image is seen. As seen, there are two right-angle triangles formed AB where **do** and **di** are parallel. Where it can be assumed that do/di = A/B both have right angle triangles[110, 111]. So the formula can be written as:

$$\frac{do}{di} = \frac{A}{B} = \frac{f}{di - f} \tag{5.28}$$

$$\frac{1}{f} = \frac{1}{d_0} + \frac{1}{d_i} \tag{5.29}$$

$$d = f + \frac{R}{r} \tag{5.30}$$

$$f = \frac{2 * 3.14 * 180}{360} \tag{5.31}$$

Where they use this formula to derive the distance which is in inches (DIS)[110, 111].

$$DIS = \frac{(2 * 3.14 * 180)}{(w + h * 360)} * 1000 + 3 \tag{5.32}$$

Based on both methods, the disadvantage is that both are trained with the MS COCO dataset where there are no cigarette litter classes. Also, both methods have a trade-off with accuracy and precision.

6 Implementation

In this chapter the implementation of the object detection of cigarette litter will be described.

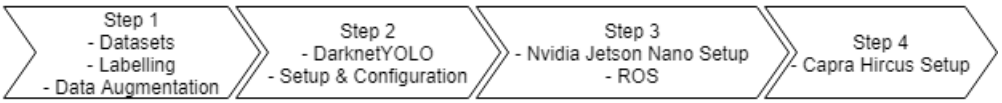

Figure 6.1: Sequence of steps to make the object detection of cigarette litter.

As seen in Figure 6.1, it shows how the pipeline of how the implementation was done to make the cigarette litter detection work. At first, the datasets of cigarette litter are collected, as well as the labelling, and data augmentation is described in Section 6.1. The tool which was used to train YOLO was Darknet as the setup and configuration of DarknetYOLO will be described in Section 6.2. When the weights are made, it can then test the inference on Nvidia Jetson Nano, where ROS is used. This is described in Section 6.3. Lastly, the mounting on the Capra Hircus and how it is controlled will be described in Section 6.4.

6.1 Used Datasets

As seen in Table 6.1 the different datasets can be seen. Two columns says "Raw Images (Original) of Cigarette Litter and Images of (Edited) Cigarette Litter". Essentially, some images were deleted from the "Raw images (Original) of Cigarette Litter" due to background noise causing distorted and blurred images.

	Raw Images (Original) of Cigarette Litter	**Images of (Edited) Cigarette Litter**
A.Kelly Dataset [112] (Synthetic Dataset)	2200	2186
E.Pacanchique Dataset [113] (Kaggle Dataset)	2039	2040
SkodRobot Dataset (Author Dataset)	1335	1294
Total	5576	5522

Table 6.1: Datasets of Cigarette Litter

The Synthetic dataset is artificially made with cigarette litter and random backgrounds that were found online. The dataset, however, contains labels that were labelled for Mask R-CNN (JSON label format), which was re-labelled by the author of this thesis,

so it works with YOLO object detection. The Kaggle dataset also had a labelled dataset which was also modified to YOLO labels that are also found online (Pascal VOC label format). The Author Dataset is mainly the dataset that is created by the author of this thesis. The author's dataset is taken from a mobile phone with 16MP resolution (HUAWEI P20 lite - ANELX1). It also offers images of snus and not many close up images of the cigarette litter due to it being taken from a further distance. Whereas, the Kaggle dataset and Synthetic dataset contains close up images.

6.1.1 Labelling Tool

The labelling tools that is used to annotate the images used in the early stage of this thesis is the BBOX (Bounding Box labelling tool) labeller. Later, it was then changed to use DarkMark, which allows data augmentation and works with Darknet.

BBOX Labeller

The labelling tool (.txt) is a straightforward tool that only labels images for YOLO since it supports any interface such as Windows, Linux and Macintosh. The BBOX labelling interface can be seen in Figure 6.2.

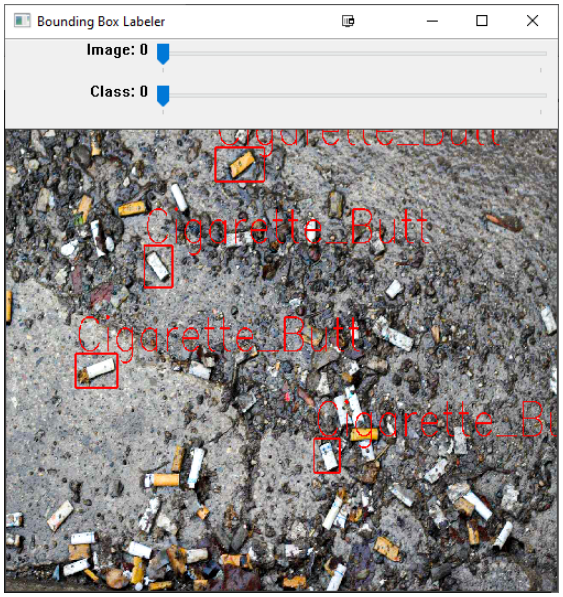

Figure 6.2: BBOX labeller tool[114]

At first, images from the Synthetic dataset and Kaggle dataset were labelled with one class called "Cigarette_Butt". It is labelled with a BBOX tool (txt. label format)[114] since the labelling tool was not efficient in zooming into the image and labelling. It also does not provide data augmentation, therefore, DarkMark was chosen.

DarkMark and DarkHelp

DarkMark is a tool that only works on Ubuntu systems. It provides image labelling (.txt and JSON) and data augmentation. DarkMark was used to augment the data and create new images which includes features rotating the images (90°, 180° and 270°), tiling the images, crop/zoom of images and resize of images[115]. It was also used to create four new classes for better feature recognition for the object detector. As seen in Figure 6.3 the four classes are Yellow (Orange)_Butt, White_Butt, Snus and Cigarette_Butt. These classes were chosen since most of the features of cigarette litter is white and yellow (orange). In some cases, cigarette butts also consist of both colours. Snus is just an extra class to see whether it detects the snus; however, snus does have similar features compared to white_butt.

Figure 6.3: DarkMark labelling tool

DarkHelp is used to analyse the statistics and review of annotated images as seen in Appendix C. The tools help by counting the number of annotated images and averaging the size of images that are annotated. DarkMark checks whether the bounding box is created correctly. It also provides some functions such as DarkHelp::predict(), which can give a prediction of result and show which is the best class and probability. One of the unique functions is DarkHelp::annotate() which can automatically annotate the raw images; however, it needs a trained weight file and cfg (config file) of specific YOLO[116] where Darknet will be described in the next section.

6.2 Darknet YOLO

To be able to use Darknet, some hardware and settings are required. The machine requires Nvidia GPU that works with CUDA. In order to get started with Darknet, the Alexey Bochkovskiy repository was cloned[92]. The repository supports both Windows and Linux systems (Ubuntu). The training and testing setup can be seen in Appendix B. The necessary part is to configure the configuration (cfg file) of YOLO. This helps to train the model with various features by adjusting the settings.

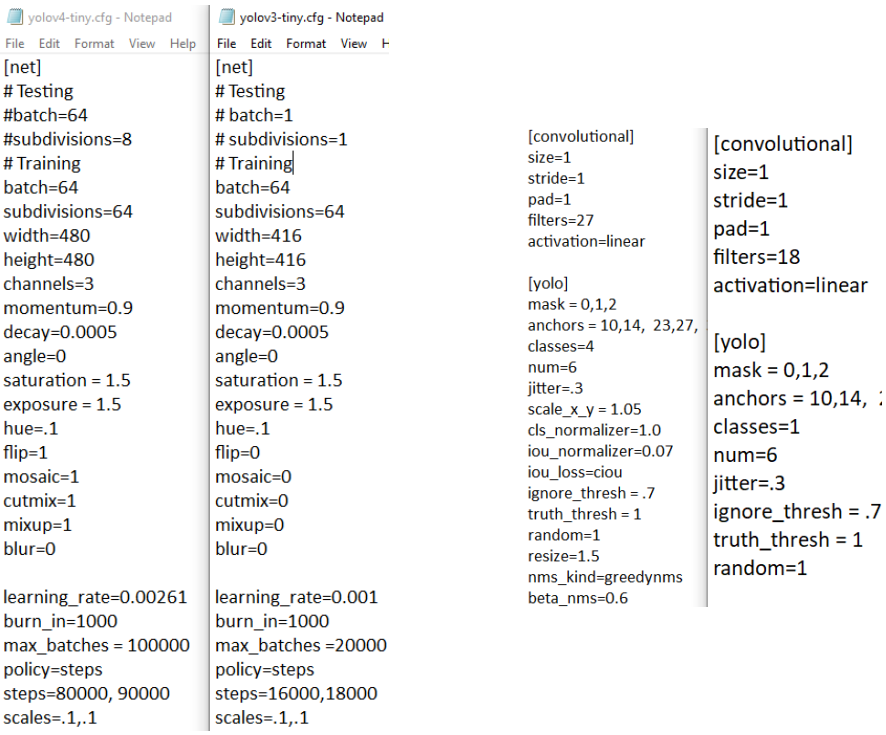

Figure 6.4: Configuration files (Parameter file) for Tiny-YOLOV3 and Tiny-YOLOV4 (not full version)[92]

As seen in Figure 6.4 the cfg file (parameter file) for Tiny-YOLOV3 and Tiny-YOLOV4 is presented. The whole configuration file is not shown in the figure since the user does not need to adjust those values, so only the essential parts of the cfg file are chosen. In the [net] parameter section, the configuration can be set to how the user prefers.

- batch - The number of samples that will be processed in one batch[92].
- subdivision - The number of mini_batches in one batch, mini_batch = batch/subdivisions. e.g., 64/16 = 4 images in one iteration[92].

- width - network size (width), every image will be resized to the network size during Training and Detection. Must be a multiple of 32[92].
- height - network size (height), every image will be resized to the network size during Training and Detection. Must be a multiple of 32[92].
- max_batches - classes * 2000 e.g., train for 4 classes max_batches = 8000. This only the minimum for max_batches, although the user can adjust how long the training should be[92].
- steps - based on the max_batches 80% and 90% is the steps[92].

There is some data augmentation that is provided by Darknet YOLO such as angle, saturation, exposure, hue, flip, mosaic, cutmix, mixup and blur that can be applied throughout training.

For [convolutional] that is above [yolo] must be changed based on the classes that are used.

- filters [convolutional] - is filters = (classes+5)*3. For Tiny-YOLOV2 the filters = (classes+5)*5 [92].
- classes [yolo] - the number of classes that is needed to be trained[92].
- random [yolo] - randomly resizes network size after each 10 iterations, when its set to 1. Otherwise set to 0 it will be faster the training process[92].

The batch and subdivision that was used was 64/64=1 image was processed at a time, and the height and width were set to resize at 416x416. If these numbers increased, the system could be more robust, although due to hardware, the processing cannot be done and will lead to an error stating CUDA ran out of memory. When testing, batch and subdivisions should be 1 and the threshold value can control the amount bounding boxes that should appear based on the confidence the cigarette litter is detected.

When the training has begun, there will be a graph that will be outputted via terminal/command-line.

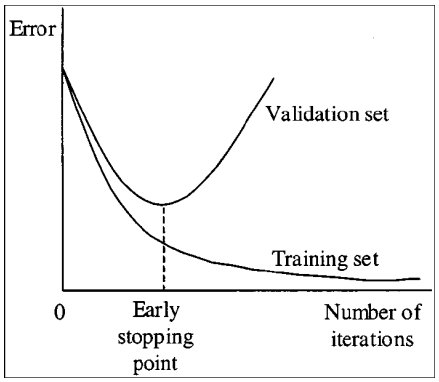

Figure 6.5: An example how the training output graph will show when training (not exactly this graph)[117]

As seen in Figure 6.5, it shows the y-axis, which is the prediction error and the x-axis is the number of iterations where it is assumed that stopping the training at the early stopping point will give better results on the weights since the error scores start to degrade and essentially to avoid memorization[117]. The graph helps the user to see how the training is performing and the user can stop the training at any time.

6.3 Nvidia Jetson Nano Setup

The Nvidia Jetson Nano is used for inference testing of trained YOLO weights. The setup for the Jetson Nano uses Jetpack 4.5, which is Ubuntu 18.04 with all the GPU drivers installed. The procedure of the setup can be found on the official website of Nvidia[118]. To access the Nvidia Jetson Nano headless, it can be done with SSH. Since Capra Robotics uses ROS on their platforms for communication between different software, hardware and more, the basic structure of ROS workspace can be seen in Figure 6.6. The object detection algorithm will also need to work on ROS, where ROS allows nodes to communicate by publishing and subscribing via topic.

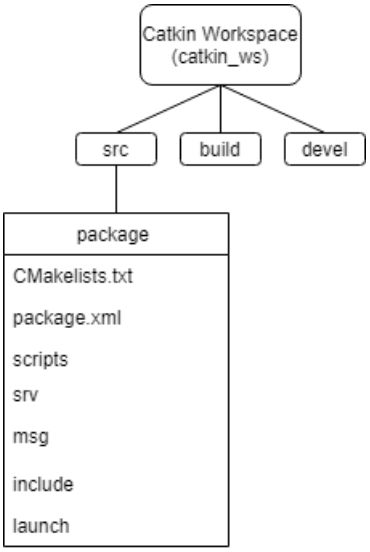

Figure 6.6: The structure of a ROS workspace

ROS Melodic is installed since Jetpack 4.5 has Ubuntu 18.04 that supports ROS Melodic. With the DarknetROS package, Capra Robotics can use or add this package to their system. DarknetROS was modified with Alexey Bochkovskiy repository, where originally, Darknet was built upon using Joseph Redmons repository[93].
Other packages were included as seen in Figure 6.7 such as Jetson camera (CSI Camera) and IP camera (Tested on Mobile Phone Camera). These are driver packages that are needed for DarknetROS to work with these cameras.

Figure 6.7: Packages used on ROS workspace

Currently, what DarknetROS topics publishes that is necessary and is outputted as **object_detector** which publishes the number of detected objects. There is a topic called **bounding_boxes** that publishes an array of bounding boxes where position and size are given in the pixel coordinate system. The last topic is **detection_image** that publishes the image of the detected image within a bounding box. The subscriber is the camera measurements **(camera_reading)**.

6.4 Capra Hircus Setup

On the Capra Hircus, the Nvidia Jetson nano was mounted with the CSI camera. The power cable of 5V that draws 4A was taken from the inside previous setup of the Capra Hircus. The odometry and IMU was not used in this implementation since the aim was to detect cigarette litter and how far they lie. As seen on Figure 6.8, a rubber tool is mounted that is made to mimic the KK Tech vacuum tool which is duct taped. This will help to understand where the camera should be mounted. The CSI camera was also mounted with duct tape. The CSI camera ribbon cable (0.3m) was not long enough thus, it was mounted near the top of the robot that allows to see the rubber tool. The Capra Hircus was tested using joystick (remote) control as seen in Figure 6.8. The joystick controller allows two types of speeds: 'SLOW' and 'NORMAL'. The 'SLOW' speed is roughly human walking speed and 'NORMAL' speed in this case is equivalent to fast speed. However, the Capra Hircus does not have a speedometer so the speed could not be measured accurately. The 'SLOW' speed was chosen since normal human walking speed is efficient enough for the object detection algorithm to detect the cigarette litter.

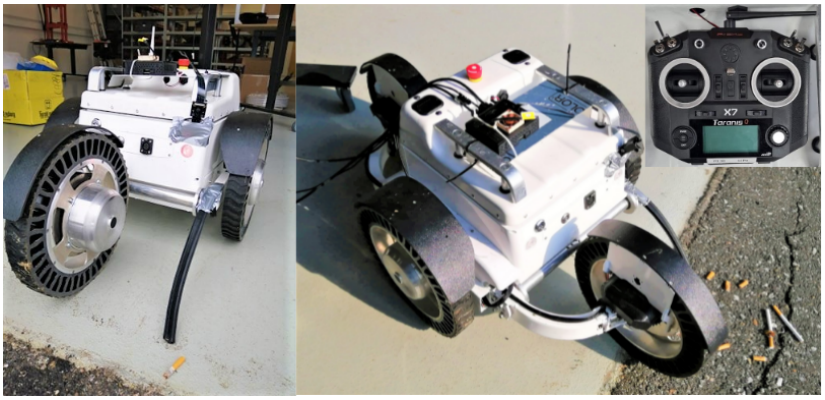

Figure 6.8: Joystick Controller and the Nvidia Jetson Nano mounted on the Capra Hircus

7 Testing & Results

Testing DarknetYOLO weights was done by a sequence of steps with training results outcome. To ensure to pick which YOLO to use and make the system robust by several augmentations, that is described Section 7.1. The Nvidia Jetson Nano was tested based on its performance with DarknetROS described in Section 7.2. The CSI camera lens was also tested to see whether it could detect the cigarette litter at sight and provide coverage. The test is important since the cigarette litter should be within the frame of the camera, and the camera should be able to see all the cigarette litter described in Section 7.3. The test videos are also described since to ensure object detection is performing correctly, visualisation of detection is important where online and offline setups of object detection are done in Section 7.4. Towards the end of the chapter, it would be tested based on the requirements setup in Section 4.2.

7.1 Training Results

In this section shows the results of how to make robust dataset. Where every step taken a new approach was taken to make the object detection algorithm robust.

Step 1 - Training Results with Tiny-YOLOV2, V3 and V4

This training aims to check whether Tiny-YOLO versions have different performance. Since it is known that Tiny-YOLOV2 is lightweight and fast however it struggles to train more classes on top. Tiny-YOLOV3 had many upgrades, which is more accurate and precise. Tiny-YOLOV3 is also said that it can handle more classes. Later Tiny-YOLOV4 has released, which the performance 10% faster and accurate than YOLOV3. So in this training of Synthetic Dataset and Kaggle Dataset datasets, Tiny-YOLO version performs well.

	Tiny-YOLOV2	**Tiny-YOLOV3**	**Tiny-YOLOV4**
A.Kelly Dataset (Synthetic Dataset)	Best mAP=72.93% F1-Score=0.62	Best mAP=93.07% F1-Score=0.85	Best mAP=99.61% F1-Score=0.96
E.Pacanchique Dataset (Kaggle Dataset)	Best mAP=59.29% F1-Score=0.58	Best mAP=80.64% F1-Score=0.71	Best mAP=94.01% F1-Score=0.80

Table 7.1: Comparison of Tiny-YOLOV2, V3 and V4 with (Appendix D)

As seen in Table 7.1 the training was done with 1 class (Cigarette_Butt). This is based on only 80% training and 20% validation data. It can be seen that Tiny-YOLOV4 does outperform all the other Tiny-YOLO versions, where it has better mAP and has a high F1 Score. The mAP stands for all the AP (Average Precision) values averaged over different classes. So without testing, each weight on the testing dataset (Author Dataset) will not show any difference, which Tiny-YOLOV4 is chosen and is known to be accurate, precise and can handle more classes.

Step 2 - Training Results of Tiny-YOLOV4 with 4 classes

In this training, the datasets were merged (Synthetic, Kaggle and Author Datasets). So for training 70% (3864), Validation 15% (828) and Testing 15% (828) as seen on Table 7.2. The four classes are Yellow (Orange) Butt, White Butt, Snus and Cigarette Butt as seen in Appendix C.

	Synthetic Dataset	**Esteban Dataset**	**Author Dataset**	**Total**
Train (70%)	1530	1428	906	3864
Validation (15%)	328	306	194	828
Test (15%)	328	306	194	828
Total	2186	2040	1294	

Table 7.2: The datasets are broken down to 70% training, 15% validation and 15% testing

The training was set at 8000 iteration, which is the minimum iteration needed to train four classes. It was trained with Random = 1 and Random = 0. Since it is said that Random = 1 is more accurate and precise, however, the current average loss takes longer compared to Random = 0; this is due to the fact that Random = 1 there is data augmentation happening by resizing the images to different sizes per batches. Since the average loss becomes 0.0XXXX, it means the weight is trained well or when no longer decreases the training can be stopped[92].

	Tiny-YOLOV4
Random = 0 8000 Iterations	Best mAP=58.27% F1-Score=0.66 current average loss = 0.2136
Random = 1 8000 Iterations	Best mAP=59.27% F1-Score=0.67 current average loss = 0.3304

Table 7.3: Training based on Random = 0 and Random = 1 (Appendix D)

Step 3 - Training Results of Tiny-YOLOV4 with 4 classes with Data augmentation

Later Tiny-YOLOV4 was then further implemented with four classes and augmentation with DarkMark. The type of augmentation are rotations, flip and mosaic. However, the advantage of Tiny-YOLOV4 is that colour and misc augmentation is done automatically at run time while training by configuring the configuration file. However, flip and mosaic are also manually configured on the configuration file to add and make new images. So the focus will be rotating the training and validation images with the use of DarkMark since the YOLO algorithm is rotation invariant.

	Merged Dataset
Train (70%)	15456
Validation (15%)	3312
Test (15%)	828
Total	19565

Table 7.4: Training and validation was augmented 90, 180 and 270 degrees.

For this, the training done will be different since now there are four fixed classes (will not be changed) made with rotation augmentation. The first weight was to set the configuration random = 0 to see whether the current average loss had an effect trained for 50000 iterations. After, another weight was made with random 50000 iterations, whereas the benefit of random=1 is the mAP result. Based on the best mAP result, the more robust the model was where the current average loss of the training data does give an impact not compared to the mAP of the validation data.
Later the two ideas were merged, which allows doing transfer learning. So by merging the configuration settings with random=0 for 50000 iterations trained model and adding random=1 for another 50000 iterations. This is done and tested. They were also an additional data augmentation setup that is mixup.

	Tiny-YOLOV4
Random = 0	Final mAP = 50.76% F1 score = 0.56 Current average loss = 0.0716
Random = 1	Best mAP = 74.17% F1 score = 0.79 Current average loss = 0.4345
Transfer Learning	Best mAP = 77% F1 score = 0.88 Current average loss = 0.3616

Table 7.5: Final Weights (Appendix D)

7.1.1 Testing on Test Images

The weights were tested on the 15% test images of 828 images. Where there were a total of 1410 cigarette litter (classes: cigarette butt, white, yellow (orange) and snus). The four classes are assumed as one since all the litter towards the end needs to be identified and should be removed (cleaned).
Since all the datasets did not have TN (True Negative) images, the confusion matrix is not constructed. Thereby the recognition rate cannot be calculated that gives the accuracy of the model.

	TP	FP	FN
Tiny-YOLOV4 - Random = 0 (50000 iterations)	788	91	531
Tiny-YOLOV4 - Random = 1 (50000 iterations)	985	71	354
Tiny-YOLOV4 - Transfer Learning (100000 iterations)	1325	47	38

Table 7.6: Results after testing on 15% test images

So instead of making the confusion matrix the recall rate, precision and F1 score can be calculated. Hence the F1 score gives the accuracy of the dataset.

- Recall rate = $\frac{TP}{TP+FN}$
- Precision = $\frac{TP}{TP+FP}$
- F_1 score = 2 * $\frac{precision*recall}{precision+recall}$

	Precision	Recall	F1 Score
Tiny-YOLOV4 - Random = 0 (50000iterations)	0.89	0.59	0.71
Tiny-YOLOV4 - Random = 1 (50000iterations)	0.92	0.73	0.82
Tiny-YOLOV4 - Tranfer Learning (100000iterations)	0.97	0.97	0.97

Table 7.7: Precision, Recall and F1 score after testing on 15% test images

As seen Table 7.7 the Tiny-YOLOV4 Random = 0 has the lowest recall, precision and F1 score. As the model that was done by transfer learning shows a robust model that has high end recall, precision and F1 score. The model is also robust to false negatives and false positive as numbers decrease and the true positives increases as seen on Table 7.6.

7.2 Nvidia Jetson Nano Testing

The Nvidia Jetson Nano was bench-marked for using DarknetROS. It was tested with both CSI camera and USB camera. The results were similar and the results were averaged based on the two power modes the Nvidia Jetson Nano can handle 5w and 10w as seen on Figure 7.8.

WxH	Tiny-YOLOV4 5w mode	Tiny-YOLOV4 5w mode (HM)	Tiny-YOLOV4 10w mode	Tiny-YOLOV4 10w mode (HM)
470 x 470	±5.4FPS	±6.4FPS	±7.5FPS	±8.6FPS
448 x 448	±6.0FPS	±7.1FPS	±8.4FPS	±9.7FPS
416 x 416	±6.6FPS	±7.8FPS	±10.3FPS	±11.8FPS
384 x 384	±6.9FPS	±8.1FPS	±11.0FPS	±12.6FPS
352 x 352	±7.5FPS	±9.2FPS	±12.5FPS	±13.7FPS
320 x 320	±8.7FPS	±10.1FPS	±15.0FPS	±17.1FPS
288 x 288	±9.2FPS	±11.0FPS	±19.2FPS	±21.1FPS
256 x 256	±10.3FPS	±12.1FPS	±23.3FPS	±23.3FPS

Table 7.8: Nvidia Jetson Nano - FPS (Frame per second) testing on DarknetROS. (HM - Headless Mode)

As seen in Table 7.8 the Nvidea Jetson Nano testing was done on DarknetROS as the WxH (Width x Height) or the resolution is changed the FPS changes. The sizes are chosen by multiple of 32 since Tiny-YOLOV4 works at that window size. One of the problems with reducing the window size is that the object detection will read many FP (False Positives), although it gives fast FPS readings. Nevertheless, the opposite will happen when the window size is large, the FPS reduces.

7.3 CSI Camera Lens Testing

The CSI camera Lens testing was carried out indoors. It was tested based on the weights from section 7.1 step 1. These weights are not robust to further distances, although the aim is to get an understanding of which lens gives the best detection and distance. The lens and test setup can be seen in Figure E.1 and E.2.

Lens Types	10°	20°	40°	60°	80°	100°	120°	140°	160°	180°	200°
Distance of Object Detected	X	X	X	X	X	5cm-55cm	X	X	X	0-20cm	0-18cm

Table 7.9: CSI camera lens testing

As seen in Table 7.9 the X means the lens does not work, where it can be seen that the 100° lens outperforms 180° and 200° lenses. The 100° can also detect more cigarette litter compared to the rest. It was also compared with the HP webcam that gives a distance range of 10cm-41cm. Although the limitation with the webcam it runs on CPU, and there were many distortions that could be fixed with calibration. It was later decided to use the 100° lens, and further tests were carried in both indoor and outdoor environment. The Final weights, in this case, is Tiny-YOLOV4 - Transfer Learning (100000iterations).

Indoor with Final Weights

As seen in Figure 7.1 done indoors, the 100° lens does a great job at detecting cigarette litter and covering the full field view of the camera. Can measure distance of 5cm-205cm. One of the limitations of the CSI lens is the purple colour of the image. This means there UV (Ultraviolet) light present, which affects the detection.

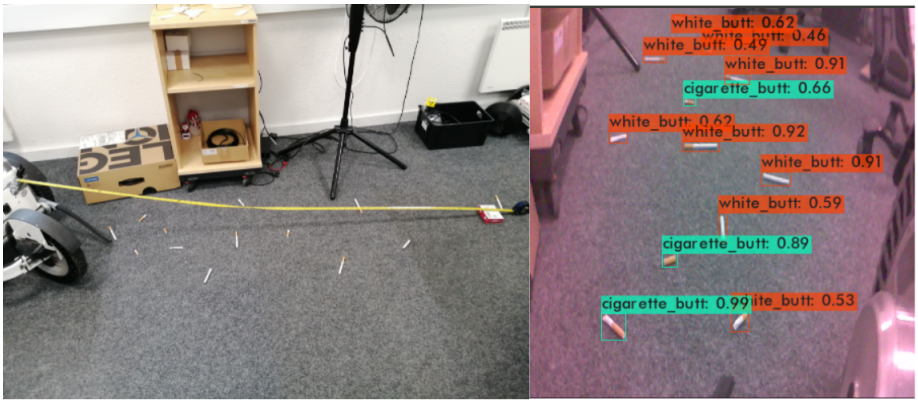

Figure 7.1: Indoor Testing of detection and distance

Outdoor with Final Weights

As seen on Figure 7.2 the same weight was used outside. Where one of the drawbacks was working with sunlight and measure a distance of 5cm-54cm.

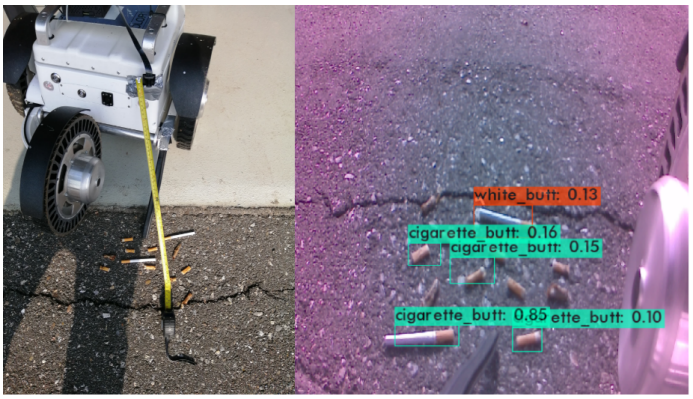

Figure 7.2: Outdoor Testing of detection and distance

7.4 Description of the Video Tests

There were eight tests that was done and recorded on video[1]. Two of the videos show that system works real-time (online) and rest was done offline. The two videos that work real-time (online):

- Tiny-YOLOV4 - RealTime Phone: The test was carried out with mobile phone (HUAWEI P20 lite - ANELX1). The realtime was done by net video cam using phones IP address that is connected to a local router. The testing was done with Darknet (will also work with DarknetROS with IPcamera driver package).
- Tiny-YOLOV4 - Capra Hircus Realtime: The test was done using DarknetROS.

Figure 7.3: The real-time test setup for distance measuring.

The setup could not be done headless since the terminal will only appear with confidence numbers of detected cigarette litter. So the Nvidia Jetson Nano was hooked to a screen and the mobile robot was placed close as possible to the outdoor environment. It was tested with 5w mode since the 10w mode of the Nvidea Jetson Nano started throttling using DarknetROS. The video shows as the resolution size is decreased the FPS increases.

[1]https://youtube.com/playlist?list=PLSG9gXgVHC2OKNinlJHKMO8baoJVxkl3G

The offline testing was done with the Capra Hircus (Nvidia Jetson Nano via SSH connection by recording scenes) and mobile phone (HUAWEI P20 lite - ANELX1 by recording scenes). Where the Capra Hircus testing was controlled setup (wrong environment) of cigarette litter and the mobile phone setup was in the right environment but wrong equipment used. Although both work with detecting cigarette litter. The six videos that were done with Capra Hircus and mobile phone was to test the three weights as seen on Table 7.7. The offline setup of the Capra Hircus is done by SSH connection via laptop to the Nvidia Jetson Nano, and from there, various scenes were recorded outdoor.

7.5 Testing Based on Requirements

The requirements will be evaluated whether the object detection algorithm could perform based on the requirements and whether the requirements were met. Tested based recorded videos[2] and test images (Section 7.1.1).

1. Functional requirements

✓ **(a)** The system must be able to detect cigarette litter in all types of outdoor terrain.

- Based on tests the detection system does work in all outdoor terrain based on public areas and sidewalks. It also works on cloudy, rainy and sunny days as seen on the video tests.

✓ **(b)** The system must detect cigarette litter.

- The system detects all the types of cigarette litter i.e. cigarette butts, filters and even snus as seen on video tests.

✓ **(c)** The system must detect cigarette litter on sidewalks.

- The system detects all the cigarette litter on the sidewalks however with some false positives. Where the false positives is mostly the cylinder of the wheel (in the gaps) of the Capra Hircus which the system thinks its a white butt seen on the video tests.

2. Performance requirements

✓ **(a)** Correctly classify an object of interest with precision and recall rate of 95%

- The requirement was fulfilled since the precision and recall rate was at 97% based on results from Section 7.1.1.

✓ **(b)** The system has to detect cigarette litter at human walking speeds (5km)[89].

[2]https://youtube.com/playlist?list=PLSG9gXgVHC2OKNinlJHKMO8baoJVxkl3G

- The object detection system works at any speeds offline (Since the user can set the FPS of the video). However when it comes to online (real time) the Nvidia Jetson Nano works at max 5-12FPS with DarknetROS at 5w and at max 7.5-23FPS with DarknetROS at 10w depending on the resolution.

One of the requirements that was not added was to see if the Capra Hircus benefits from moving backwards or forwards. The system has delay when it moves forward towards the cigarette litter whereas moving backwards it performs better than moving forwards when detecting the cigarette litter as seen on video in real-time of Capra Hircus video (video name = Tiny-YOLOV4 - Capra Hircus Realtime (Capra Hircus)[3]).

[3]https://youtube.com/playlist?list=PLSG9gXgVHC2OKNinlJHKMO8baoJVxkl3G

8 Conclusion

The purpose of the SkodRobot (Cigarette Litter-Robot) Project is to identify and remove cigarette litter in outdoor urban/public terrain, where one of the problems Technology and the Environment Services in Aarhus Municipality (Teknik og Miljø – Aarhus Kommune) are having problems is with collecting litter on sidewalks. In this thesis, the process was broken down into two parts so that the identification of cigarette litter was looked upon, and the removal of cigarette litter would be another project itself. Based upon this task, two key questions were formulated in Chapter 4. These questions will help to summarise the research findings done in this thesis.
The main question which was explored throughout this thesis is: *"How can an object detection system provide real-time coverage of cigarette litter on sidewalks in urban/public areas with the use of outdoor mobile robots?"* The object detection system gives a recall rate, precision and F1-score of 97% that was achieved as well as all the requirements that were seen in Section 7.5. The system also provides real-time coverage on the Nvidia Jetson Nano at a rate of 5FPS at 5w and 10FPS at 10w (depending on the resolution or size) with YOLO as the tests can be seen on Section 7.2. The system works in real-time with a wired connections setup which a screen was needed for the user to see whether the object detection system was working correctly in real-time. Although with urban/public areas, the real-time was not tested due to working headless mode with the Nvidia Jetson Nano and Capra Hircus was not allowed to be taken to the urban/public areas for testing. Thus, this was tested by offline configuration on urban/public areas on sidewalks since the test was done with a joystick (remote) control from the Capra Hircus. The main system Nvidia Jetson Nano was able to detect cigarette litter. It was also tested via mobile phone and using the training PC to run it in real-time by IP connection.

Another sub-question that was also investigated is *"How to develop a robust dataset for cigarette litter?"* This task was done by getting two online datasets that had 4239 images of cigarette litter: 2200 synthetic images from A.Kelly dataset, and 2039 images of cigarette litter from E.Pacanchique dataset. Based on these datasets, a dataset was created by the author of this thesis called the SkodRobot dataset. This was because the online datasets did not have many variations of cigarette litter and most of the cigarette litter was taken at a close distance. The SkodRobot dataset provides many variations and dense piles of cigarette litter. Altogether there were 5576 images of cigarette litter, the author edited it to get 5522 images of cigarette litter due to noise and blurred images. Later, it was augmented by rotating the images getting to a total of 19565 images. Thus, the weight was trained with more augmentations as that would also increase the images, i.e. flip, mosaic and mix up (estimated images of 78260, which can not be seen). This dataset provides four classes that distinguish cigarette litter as cigarette butt, white butt and yellow (orange) butt, and snus added as an extra feature to the class. Although there were not enough images of snus which sometimes overlaps detection of a white butt (end of the day, every litter needs to found). In Alexey Bochkovskiy[92] repository, it mentions that a minimum of 2000 images of a certain class is needed to make the detection robust, whereas for snus, there was only 242 images and a count of 277 snus

in the images were augmented. The rest of the classes were over the minimum after augmenting them; this can be seen in Appendix C.2. The final weight for Tiny-YOLOV4 which was completed with transfer learning got a precision, recall and F1 score of 97%. If the model was trained with regular YOLOV4, the model would be more robust but not faster than Tiny-YOLOV4 (faster meaning FPS).

One of the methods that was not implemented was the localisation (distance estimations) of the cigarette litter. This was achieved by using the monocular camera seen in Section 5.2. Section 5.2.1 approach was implemented, but it only works on images, and the accuracy of distance measuring was not the greatest, plus the length of the cigarette litter had a fixed value of (e.g. 3cm). An image of this can be seen in Appendix E.2. In Section 5.2.3, the two were tested where method 1 requires training and the model that is being used is heavy to make inference on the Nvidia Jetson Nano. Method 2, however, by M. A. Khan et al.[110] is implemented in Pytorch, where the distance estimation only works on a USB-camera. To make it work on the CSI camera, the Gstreamer did not work with Pytorch (implemented in Python) since YOLO had trouble resizing the CSI camera image. The system was not robust and accurate since it could only measure a maximum distance of 0-12cm. An example of the approach can be seen in Appendix E.2. The goal of the distance estimation is to make Section 5.2.2 work by not getting the depth but getting the Z ray; however, since integrating into ROS, there were some issues, and it required time to implement it.

8.1 Future Works

The SkodRobot project (Cigarette Litter-Robot) is a project that is still under development since, at the moment, cigarette litter identification works, although with some false positives. In this chapter, the key points that were proposed on how to improve the detection system were that:

- Currently, the object detection algorithm gives some false positives as seen on the video tests. This can be improved by having images without cigarette litter and having background images. The system is precise, but to test the model's accuracy (recognition rate), TN (True Negative) images are needed. The threshold value can also be manipulated so the system can be more precise in detecting cigarette litter. To reduce false positives, the camera's projection also needs to be pointing towards the surface since working in an outdoor environment, there would be too much noise that could resemble the same colour of cigarette litter at further distances.
- Another drawback is that the camera lens that is being used to test the detection and the distance of the cigarette litter is affected by the ultraviolet light that blinds the sensor from working outdoors. In order to solve this, an ND filter can be a suggestion to fix this issue.
- The dataset can also be upgraded with more classes of different types of litter, and making more features of the types of litter that could make the model more robust at detecting the specific litter.

- The dataset can be improved by different configurations of the YOLO and doing transfer learning. However, since the images were only augmented at 90°, 180°, and 270°, using more degrees of augmentations will lead to a robust dataset.
- The dataset can be trained on the map that was provided by Technology and the Environment Services in Chapter 2 Figure 2.2. Since the mobile robot would work robustly with known roads or sidewalks, it needs to detect the cigarette litter. It can be done by taking the mobile robot to urban areas and capture videos or images at its height. This allows exactly for the mobile robot to know how the roads or sidewalks are based on the dataset it makes.
- The Nvidia Jetson Nano currently only gives around 10FPS with 10w mode using DarknetROS which has a resolution size of around 416x416. This can be optimised by using a TPU (Google Coral) or by using the TensorRT that can give 25FPS, as seen in Figure 5.2 from the Nvidia Jetson. Whereas, adding more load, e.g. distance estimation or coverage planning on the Nvidia Jetson Nano, could potential over drift the system.
- When looking into training, it can be optimised by better hardware which that could lead to faster training. In this thesis, the training was done by resizing images of 416x416. By increasing the resolution size, it could benefit a more robust system. Although Google Colab has already been looked upon with Darknet, where it provides a suitable cloud server hardware, the problem with Google Colab is the run-time that stops after 12 hours and importing many images to drive sometimes breaks the server. When testing the system, the sizing can also be modified, although Nvidia Jetson Nano 416x416 is ideal; otherwise, the system will give an error saying CUDA ran of memory (Not enough RAM).
- Using IP Camera and a dedicated system could be a possibility to improve the system since it will work faster in real-time. Hence, instead of using Tiny-YOLOV4, the normal YOLOV4 version could be used since it is known to be robust compared to Tiny-YOLOV4.
- The aim is to make Section 5.2.2 distance estimation approach work instead of getting depth from a monocular camera but to get the ray of Z. Although, Stereo cameras can also be utilised, DarknetROS has a package and provides distance estimation and tracking of objects[119].

Since the identification of the cigarette litter is reasonably working, the removal will be the next step by using coverage planning and making the platform autonomous.

Bibliography

[1] A. A, "Solving the global litter problem," 2016. [Online]. Available: https://news.janegoodall.org/2016/06/30/solving-global-litter-problem/

[2] bristolwastecompany, "The history of litter," N/A. [Online]. Available: https://bristolwastecompany.co.uk/history-of-litter/

[3] P. Rubbish, "How litter affects the environment," 2020. [Online]. Available: https://www.paulsrubbish.com.au/how-litter-affects-the-environment/

[4] truthinitiative, "5 ways cigarette litter impacts the environment," 2017. [Online]. Available: https://truthinitiative.org/research-resources/harmful-effects-tobacco/5-ways-cigarette-litter-impacts-environment

[5] M. DUNDAS, "Cigarette butts: The world's most littered item," 2020. [Online]. Available: https://www.france24.com/en/20200306-down-earth-cigarette-butts-the-world-s-most-littered-item

[6] "Folketal den 1. i kvartalet efter civilstand, alder, køn, område og tid," Danmarks Statistik - (FOLK1A). [Online]. Available: https://www.statistikbanken.dk/folk1a

[7] velfaerdsteknologi, "Skodrobot," 2021. [Online]. Available: https://velfaerdsteknologi.aarhus.dk/opi/velfaerdsteknologisk-opi-pulje/2021-projekter-stoettet-af-velfaerdsteknologisk-opi-pulje/skodrobot/

[8] A. K. Jensen, C. Jørgensen, J. Thygesen, K. Hansen, and M. Tvilling, "Udvidet producentansvar på cigaretter analyse af mulige organiserings-modeller i danmark," Miljøstyrelsen, Odense, Denmark, 2020. [Online]. Available: https://www2.mst.dk/Udgiv/publikationer/2020/04/978-87-7038-181-9.pdf

[9] C. V. Waterways, "Cigarette filters," 2021. [Online]. Available: http://www.longwood.edu/cleanva/cigbuttfilters.htm

[10] S. E. Dustin Poppendieck, Shahana Khurshid, "Measuring airborne emissions from cigarette butts: Literature review and experimental plan," 2016. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8147.pdf

[11] B. Harris, "The intractable cigarette 'filter problem'," 2011. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3088411/

[12] P. Morris, "How cigarettes are made: A presentation by david e. merrill [parts 1-2]," 2006. [Online]. Available: https://archive.org/details/haq23e00

[13] S. AARHUS, "Skodrobotten. måske ikke så ringe endda?" 2021. [Online]. Available: https://smart.aarhus.dk/aktuelt/2021/skodrobotten-maaske-ikke-saa-ringe-endda/

[14] A. Kommune, "Teknik og miljø," 2021. [Online]. Available: https://www.aarhus.dk/om-kommunen/teknik-og-miljoe/#2

[15] A. L. André Potenza, Andrey Kiselev and A. Saffiotti, "Towards sliding autonomy in mobile robotic telepresence – a position paper," 2021. [Online]. Available: https://oru.diva-portal.org/smash/get/diva2:1160505/FULLTEXT01.pdf

[16] M. E. College, "Mobile robots module 6," 2021. [Online]. Available: https://www.marian.ac.in/public/images/uploads/pdf/online-class/MOBILE%20ROBOTS.pdf

[17] M. M. I. Robots, "Mir a better way," 2021. [Online]. Available: https://www.mobile-industrial-robots.com/en/

[18] OMRON, "Mød omrons serie af autonome mobile robotter," 2020. [Online]. Available: https://industrial.omron.dk/da/products/mobile-robot

[19] Kuka, "mobility," 2021. [Online]. Available: https://www.kuka.com/en-au/products/mobility

[20] Robotics, A. G. . D. of Materials, and P. . A. University, "Robotics and automation group," 2013-2018. [Online]. Available: http://robotics-automation.aau.dk/

[21] enabled robotics, "enabled-robotics," 2021. [Online]. Available: enabled-roboticshttps://www.enabled-robotics.com/

[22] P. ROBOTICS, "Tiago," 2021. [Online]. Available: https://pal-robotics.com/robots/tiago/

[23] iRobot, "irobot," 2021. [Online]. Available: https://www.irobot.dk/roomba

[24] immersive robotics, "Trash can robot," 2016. [Online]. Available: http://immersive-robotics.com/

[25] C. ROBOTICS, "Capra robotics," 2021. [Online]. Available: https://capra.ooo/

[26] C. I. ApS, "Conpleks innovation aps," N/A. [Online]. Available: https://conpleks.com/

[27] D. T. Institute, "The danish traffic act has changed: Now robots can be found on the streets," 2016. [Online]. Available: https://www.dti.dk/specialists/now-robots-can-be-found-on-the/43031?fbclid=IwAR3-4wjY3RTfGCWuv7rFC0uWP3UcbQWs4ywkD6zsrEJcY22BP0UJ3EVLptU

[28] BostonDynamics, "Spot®," 2010. [Online]. Available: https://www.bostondynamics.com/spot

[29] anybotics, "Autonomous robots at work. anywhere." 2020. [Online]. Available: https://www.anybotics.com/anymal-legged-robot/

[30] DustBot, "Dustbot," 2020. [Online]. Available: http://dustbot.org/

[31] Robotech, "Dustclean," 2021. [Online]. Available: https://www.robotechsrl.com/dustclean-en-robot-sweeper/

[32] DustBot, "Dustbot," 2021. [Online]. Available: https://www.robotechsrl.com/dustcart-en-urban-robot/

[33] J. B. et al., "Deep learning based robot for automatically picking up garbage on the grass," 2018. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1904/1904.13034.pdf

[34] B. M. Aqid, "Building a smart cigarette cleaning robot in 2 weeks," 2020. [Online]. Available: https://bukhorimuhammad.medium.com/building-a-smart-cigarette-cleaning-robot-in-2-weeks-936db857f310

[35] A. Robotics, "Technologies combined in one robot." 2020. [Online]. Available: https://angsa-robotics.com/en-de/roboter/

[36] Laptrinhx, "Artificial intelligence, machine learning, and deep learning. what's the real difference?" 2019-2021. [Online]. Available: https://laptrinhx.com/artificial-intelligence-machine-learning-and-deep-learning-what-s-the-real-difference-1974627010/

[37] O. Languages, "Oxford languages and google," 2021. [Online]. Available: https://languages.oup.com/google-dictionary-en/

[38] B. Kavindra, "What is machine learning - a brief introduction," 2015-2021. [Online]. Available: https://www.buddhilive.com/2020/06/what-is-machine-learning-brief.html

[39] MATLAB, "Feature detection and extraction," 2021. [Online]. Available: https://www.mathworks.com/help/vision/feature-detection-and-extraction.html?s_tid=CRUX_lftnav

[40] N. Kang, "Multi-layer neural networks with sigmoid function— deep learning for rookies (2)," 2017. [Online]. Available: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

[41] F. Vázquez, "Deep learning made easy with deep cognition," 2017. [Online]. Available: https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351

[42] J. Brownlee, "How to configure the number of layers and nodes in a neural network," 2019. [Online]. Available: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/

[43] K. Goyal, "6 types of activation function in neural networks you need to know," 2015-2021. [Online]. Available: https://www.upgrad.com/blog/types-of-activation-function-in-neural-networks/

[44] S. Raval, "Which activation function should i use?" 2017. [Online]. Available: https://www.youtube.com/watch?v=-7scQpJT7uo

[45] Z. T. et al., "Structural damage detection based on real-time vibration signal and convolutional neural network," 2020. [Online]. Available: https://www.researchgate.net/figure/Activation-function-a-Sigmoid-b-tanh-c-ReLU_fig6_342831065

[46] A. PAI, "Cnn vs. rnn vs. ann – analyzing 3 types of neural networks in deep learning," 2013-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/

[47] S. Saha, "A comprehensive guide to convolutional neural networks," 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[48] J. Redmon, "Darknet: Open source neural networks in c," 2013-2016. [Online]. Available: http://pjreddie.com/darknet/

[49] Pawangfg, "Object detection vs object recognition vs image segmentation," 2020. [Online]. Available: https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/

[50] A. Rosebrock, "Sliding windows for object detection with python and opencv," 2021. [Online]. Available: https://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/

[51] P. S. et al., "Overfeat: Integrated recognition, localization and detection using convolutional networks," 2013. [Online]. Available: https://arxiv.org/pdf/1312.6229.pdf

[52] L. L. et al., "Deep learning for generic object detection: A survey," 2018. [Online]. Available: https://arxiv.org/abs/1809.02165

[53] R. Gandhi, "R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms," 2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[54] L. LealTaixe, "Cv3dst - two-stage object detectors," 2020. [Online]. Available: https://www.youtube.com/watch?v=6I3m0SsLPo4

[55] ——, "Cv3dst - one-stage object detectors," 2020. [Online]. Available: https://www.youtube.com/watch?v=J9LSeOGoNW0

[56] ProgrammerSought, "Yolo summary," 2018-2021. [Online]. Available: https://www.programmersought.com/article/34194183439/

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[58] A. P. et al., "pytorch," 2021. [Online]. Available: https://pytorch.org/

[59] G. B. team, "Tensorflow," 2021. [Online]. Available: https://www.tensorflow.org/

[60] M. I. for Learning Algorithms (MILA) at the Université de Montréal, "Theano (software)," 2021. [Online]. Available: https://en.wikipedia.org/wiki/Theano_(software)

[61] F. Chollet, "Keras," 2021. [Online]. Available: https://keras.io/

[62] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[63] Google, "What is colaboratory?" 2021. [Online]. Available: https://colab.research.google.com/notebooks/intro.ipynb

[64] M. E. et al., "The pascal visual object classes challenge: A retrospective," 2014. [Online]. Available: https://link.springer.com/article/10.1007/s11263-014-0733-5

[65] O. R. et al., "Imagenet large scale visual recognition challenge," 2014. [Online]. Available: https://arxiv.org/pdf/1409.0575.pdf

[66] T.-Y. L. et al., "Microsoft coco: Common objects in context," 2014. [Online]. Available: Tsung-YiLin

[67] B. Z. et al., "Places: A 10 million image database for scene recognition," 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7968387

[68] A. K. et al., "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," 2018. [Online]. Available: https://arxiv.org/pdf/1811.00982.pdf

[69] Kaggle, "Kaggle," 2019. [Online]. Available: https://www.kaggle.com/

[70] S. Pokhrel, "Image data labelling and annotation — everything you need to know," 2020. [Online]. Available: https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1

[71] A. Kelly, "Training an ai to recognize cigarette butts," 2018. [Online]. Available: https://medium.com/@aktwelve/training-an-ai-to-recognize-cigarette-butts-5cff9e11c0a7

[72] K. Demkova, "Emily - the cigarette butt picker bot," 2018. [Online]. Available: https://medium.com/@aktwelve/training-an-ai-to-recognize-cigarette-butts-5cff9e11c0a7

[73] S. Majchrowska, "Trash detection - review of useful resources," N/A. [Online]. Available: https://github.com/majsylw/litter-detection-review

[74] R. GADE, "Advanced robotic perception 4: Tracking," 2019. [Online]. Available: https://www.moodle.aau.dk/pluginfile.php/1649761/mod_resource/content/0/ARP4-Tracking.pdf

[75] A. Rosebrock, "Simple object tracking with opencv," 2021. [Online]. Available: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

[76] A. Konushin, "Visual object tracking," 2021. [Online]. Available: https://www.coursera.org/lecture/deep-learning-in-computer-vision/visual-object-tracking-KcPB8

[77] B. Georgievski, "Object detection and tracking in 2020," 2020. [Online]. Available: https://blog.netcetera.com/object-detection-and-tracking-in-2020-f10fb6ff9af3

[78] L. LealTaixe, "Cv3dst - object tracking," 2020. [Online]. Available: https://www.youtube.com/watch?v=QtAYgtBnhws

[79] A. e. a. Milan, "Improving global multi-target tracking with local updates," 2015. [Online]. Available: https://vbn.aau.dk/ws/files/212438069/eccvws2014.pdf

[80] H. I. Christensen, "Visual servoing," N/A. [Online]. Available: http://www.hichristensen.com/CSE291-D-18-Intro-Robotics/pdf/12-visual-servoing.pdf

[81] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, December 2005.

[82] P. C. et al., "A new partitioned approach to image-based visual servo control," 2001. [Online]. Available: https://ieeexplore.ieee.org/document/954764

[83] H. Choset., "Robotic motion planning: Cell decompositions," N/A. [Online]. Available: https://www.cs.cmu.edu/~motionplanning/lecture/Chap6-CellDecomp_howie.pdf

[84] Stanford, "Motion planning in robotics," N/A. [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/basicmotion.html

[85] H. C. et al., "Principles of robot motion," 2005. [Online]. Available: https://mitpress.mit.edu/books/principles-robot-motion

[86] K. Choset, "Cell decomposition methods," N/A. [Online]. Available: http://www.ccs.neu.edu/home/rplatt/cs5335_fall2017/slides/cell_decomposition.pdf

[87] M. BENDT, "Wo2015197069 - chassis for vehicle," 2015. [Online]. Available: https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2015197069

[88] kk tech, "kk-tech," 2021. [Online]. Available: http://kk-tech.dk/?page_id=136&lang=en

[89] R. C. B. et al., "Effects of obesity and sex on the energetic cost and preferred speed of walking," 2006. [Online]. Available: https://journals.physiology.org/doi/full/10.1152/japplphysiol.00767.2005

[90] Y. Yuan, "Yolo creator joseph redmon stopped cv research due to ethical concerns," 2020. [Online]. Available: https://syncedreview.com/2020/02/24/yolo-creator-says-he-stopped-cv-research-due-to-ethical-concerns/

[91] A. B. et al, "Yolov4: Optimal speed and accuracy of object detection," 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[92] A. Bochkovskiy, "Yolo v4, v3 and v2 for windows and linux," 2021. [Online]. Available: https://github.com/AlexeyAB/darknet

[93] M. Bjelonic, "YOLO ROS: Real-time object detection for ROS," https://github.com/leggedrobotics/darknet_ros, 2016–2018.

[94] J. R. et al., "You only look once: Unified, real-time object detection," 2016. [Online]. Available: https://arxiv.org/pdf/1506.02640.pdf

[95] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016. [Online]. Available: https://arxiv.org/pdf/1612.08242.pdf

[96] J. R. et al., "Yolov3: An incremental improvement," 2018. [Online]. Available: https://arxiv.org/pdf/1804.02767.pdf

[97] C.-Y. W. et al., "Cspnet: A new backbone that can enhance learning capability of cnn," 2019. [Online]. Available: https://ieeexplore.ieee.org/document/954764

[98] G. Jocher, "Yolov5," 2020. [Online]. Available: https://ultralytics.com/yolov5

[99] J. S. Joseph Nelson, "Responding to the controversy about yolov5," 2020. [Online]. Available: https://blog.roboflow.com/yolov4-versus-yolov5/

[100] X. L. et al, "Pp-yolo: An effective and efficient implementation of object detector," 2020. [Online]. Available: https://arxiv.org/pdf/2007.12099.pdf

[101] Nvidia, "Jetson nano: Deep learning inference benchmarks," 2021. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks

[102] J. Hui, "Real-time object detection with yolo, yolov2 and now yolov3," 2018. [Online]. Available: https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

[103] F. T. G. UP!, "Measuring size from images: A wrangle with angles and image scale," N/A. [Online]. Available: https://lweb.cfa.harvard.edu/webscope/activities/pdfs/measureSize.PDF?fbclid=IwAR3WMf4U1G4JpCmGGRSFByKkLS11_pMj006_RC1ThUnWUFn7rrtovxjlzec

[104] J. J. GUTIERREZ, "How to measure the angular size of the big dipper," 2021. [Online]. Available: https://owlcation.com/stem/Angular-distances?fbclid=IwAR0thZG8iXpYD7-pfHnS_e3zuGUp5SfBw7QFLh-bumSA3zY9sAu7nwchcmA

[105] S. Birchfield, "Homogeneous coordinates," 1998. [Online]. Available: http://robotics.stanford.edu/~birch/projective/node4.html

[106] S. Agarwal, "How do we calculate distances of objects using monocular cameras?" 2020. [Online]. Available: https://medium.com/all-things-about-robotics-and-computer-vision/how-do-we-calculate-distances-of-objects-using-monocular-cameras-67c4822c538e

[107] OpenCV, "Basic concepts of the homography explained with code," N/A. [Online]. Available: https://docs.opencv.org/master/d9/dab/tutorial_homography.html

[108] M. A. H. et al., "Disnet: A novel method for distance estimation from monocular camera," 2018. [Online]. Available: https://project.inria.fr/ppniv18/files/2018/10/paper22.pdf

[109] G. Jianyu, "Disnet," 2017. [Online]. Available: https://github.com/guanjianyu/DisNet

[110] M. A. K. et al., "An ai-based visual aid with integrated reading assistant for the completely blind," 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9234074

[111] P. Paul, "Object-detection-and-distance-measurement," N/A. [Online]. Available: https://github.com/paul-pias/Object-Detection-and-Distance-Measurement

[112] Immersivelimit, "Cigarette butt dataset trained weights," 2015–2021. [Online]. Available: https://www.immersivelimit.com/datasets/cigarette-butts

[113] E. Pacanchique, "Kaggle - cigarette butt," 2019. [Online]. Available: https://www.kaggle.com/estebanpacanchique/cigarette-butt

[114] J. Cartucho, R. Ventura, and M. Veloso, "Robust object recognition through symbiotic deep learning in mobile robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2336–2341.

[115] S. Charette, "Darkmark," 2019-2021. [Online]. Available: https://www.ccoderun.ca/darkmark/

[116] ——, "Darkhelp," 2019-2021. [Online]. Available: https://www.ccoderun.ca/DarkHelp/api/index.html

[117] P. V. V, "Part-2: Error analysis — the wild west. algorithms to improve neuralnetwork accuracy." 2016. [Online]. Available: https://medium.com/autonomous-agents/part-2-error-analysis-the-wild-west-algorithms-to-improve-neuralnetwork-accuracy-6121569e66a5

[118] NVIDIA, "Getting started with jetson nano developer kit," 2021. [Online]. Available: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit

[119] F. G. Ramos, "darknet_ros_3d," N/A. [Online]. Available: https://github.com/IntelligentRoboticsLabs/gb_visual_detection_3d

A Appendix 1 - Teknik og Miljø

Technology and the environment in Aarhus Municipality (Teknik og Miljø - Aarhus Kommune)

Figure A.1: A Map that shows where the sweeper machine is used to clean up the roads in Aarhus Municipality (Image taken from Teknik og Miljø - Aarhus Kommune)

Figure A.2: A map that shows where litter is cleaned up by hand in Aarhus Municipality (Image taken from Teknik og Miljø - Aarhus Kommune)

B Appendix 2 - Darknet Setup

At first AlexeyAB Darknet Repository needs to cloned or forked[92]. The operating system that was used for training was a Windows 10 machine.

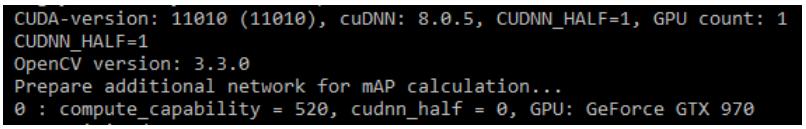

Figure B.1: Windows 10 setup (16.0GB ram and Intel Core i7)

Figure B.1 shows the Windows 10 machine setup for training the Darknet YOLO.

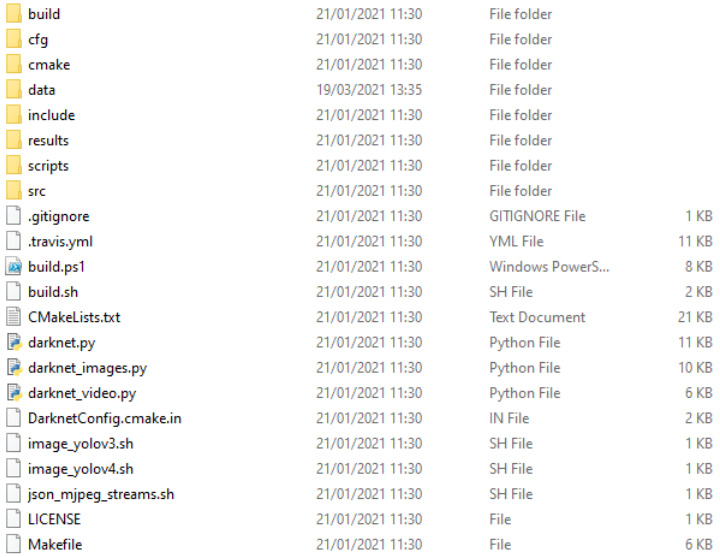

Figure B.2: The files from AlexeyAB Darknet

The necessary files/folder that shall be looked upon as seen on Figure B.2 are data, cfg, build and Makefile.

- **data & cfg folder**: is for Ubuntu/Google Colab configurations. data folder contains the training parts which will be discussed later in this thesis and cfg folder is the configurations files YOLO (and other).
- **build folder**: is for Windows configurations. Files are quite similar as seen above. Well the most important files are 'cfg and data'. Backup is where the weights are outputted.
- **Makefile**: is also for Ubuntu/Google Colab configurations. Where e.g., the GPU=1 means its on (GPU=0 its off and maybe not available).

When the images are labelled as seen on Section 6.1.1 (how to label the images), there needs to be two files that needs to be created obj.names and obj.data (essentially two .txt files) as seen on Figure B.3. Inside the data folder in this case for Windows setup will be in C:\(path_of_user)\darknet\build\darknet\x64\data. For Ubuntu/Google Colab it will be in /(pathofuser)/home/darknet/data.

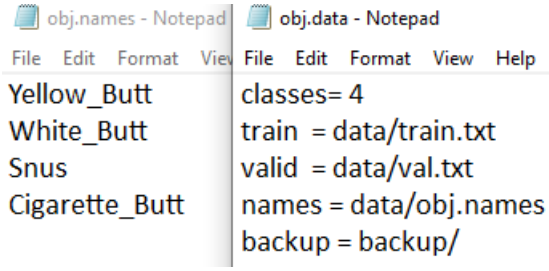

Figure B.3: obj.names contains the class names and obj.data contains the classes, train, val and backup paths to start training.

There also needs to be a training .txt file and validation .txt file this contains all the path image files location e.g., 80% for training and 20% for validation can be chosen for training.

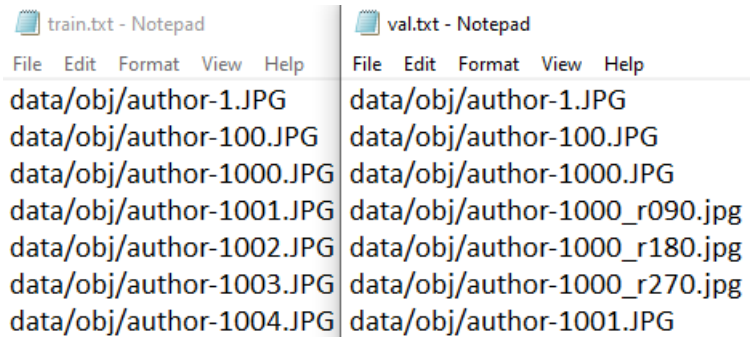

Figure B.4: Example of train and val images paths.

To train darknet YOLOV2, V3 and V4 the pre-trained weights for the convolutional layers are needed darknet19 for YOLOV2, darknet53 for YOLOV3 and CSPdarknet53 for YOLOV4.

- for `yolov4.cfg`, `yolov4-custom.cfg` (162 MB): yolov4.conv.137 (Google drive mirror yolov4.conv.137)
- for `yolov4-tiny.cfg`, `yolov4-tiny-3l.cfg`, `yolov4-tiny-custom.cfg` (19 MB): yolov4-tiny.conv.29
- for `csresnext50-panet-spp.cfg` (133 MB): csresnext50-panet-spp.conv.112
- for `yolov3.cfg, yolov3-spp.cfg` (154 MB): darknet53.conv.74
- for `yolov3-tiny-prn.cfg , yolov3-tiny.cfg` (6 MB): yolov3-tiny.conv.11
- for `enet-coco.cfg (EfficientNetB0-Yolov3)` (14 MB): enetb0-coco.conv.132

Figure B.5: Pre-trained weights for convolutional layers[92]

When everything is set it then needs to be built on an IDE e.g., visual studio where the 'release and x64' needs to be set.

To start the training. The following command needs to be typed on the command line from the path x64: **darknet.exe detector train data/obj.data data/yolov3-tiny.cfg darknet53.conv.74 -map** as seen in Figure B.6.
Hence this trains only for Tiny-YOLOV3. To train for other YOLOS the cfg path and pre-trained weights for the convolutional layers needed to be modified.

```
C:\Users\...\Downloads\darknet-master\build\darknet\x64>darknet.exe detector train data/obj.data data/yolov3-tiny.cfg
darknet53.conv.74 -map
 CUDA-version: 11010 (11010), cuDNN: 8.0.5, CUDNN_HALF=1, GPU count: 1
 CUDNN_HALF=1
 OpenCV version: 3.3.0
 Prepare additional network for mAP calculation...
 0 : compute_capability = 520, cudnn_half = 0, GPU: GeForce GTX 970
net.optimized_memory = 0
mini_batch = 1, batch = 64, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 conv     16       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  16 0.150 BF
   1 max                2x 2/ 2    416 x 416 x  16 ->  208 x 208 x  16 0.003 BF
   2 conv     32       3 x 3/ 1    208 x 208 x  16 ->  208 x 208 x  32 0.399 BF
   3 max                2x 2/ 2    208 x 208 x  32 ->  104 x 104 x  32 0.001 BF
   4 conv     64       3 x 3/ 1    104 x 104 x  32 ->  104 x 104 x  64 0.399 BF
   5 max                2x 2/ 2    104 x 104 x  64 ->   52 x  52 x  64 0.001 BF
   6 conv    128       3 x 3/ 1     52 x  52 x  64 ->   52 x  52 x 128 0.399 BF
   7 max                2x 2/ 2     52 x  52 x 128 ->   26 x  26 x 128 0.000 BF
   8 conv    256       3 x 3/ 1     26 x  26 x 128 ->   26 x  26 x 256 0.399 BF
   9 max                2x 2/ 2     26 x  26 x 256 ->   13 x  13 x 256 0.000 BF
  10 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
  11 max                2x 2/ 1     13 x  13 x 512 ->   13 x  13 x 512 0.000 BF
  12 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  13 conv    256       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 256 0.089 BF
  14 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
  15 conv     18       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x  18 0.003 BF
  16 yolo
```

Figure B.6: Example command initiated for training.

The **-map** is the mAP (mean average precision) flag to helps to visualise training vs validation. There are several commands for testing on images, videos, and live video feed (webcam) testing. When testing, the threshold value can also be adjusted so that bounding-boxes do not appear at certain confidence. That can be found on Alexey Bochkovskiy repository[92].

C Appendix 3 - DarkMark & DarkHelp

class id	class name	count	images	min size	avg size	max size	SD width	SD height	min marks	max marks
0	Cigarette_Butt	5198	4242	24 x 25	96.26 x 103.30	311 x 512	40.57	43.20	1	22

#	mark	size	rectangle	overlap	filename	type	message
5147		82 x 87	340, 23, 422, 110		/media/mathie/Seagate Back...	image/jpeg	
5148		71 x 138	78, 290, 149, 428		/media/mathie/Seagate Back...	image/jpeg	
5149		66 x 98	205, 78, 271, 176		/media/mathie/Seagate Back...	image/jpeg	
5150		81 x 46	371, 252, 452, 298		/media/mathie/Seagate Back...	image/jpeg	
5151		64 x 164	293, 24, 357, 188		/media/mathie/Seagate Back...	image/jpeg	

Figure C.1: DarkMark Review/Statistics on Kaggle and Synthetic Dataset

C.1 DarkMark Review

Yellow_Butt | White_Butt | Snus | Cigarette_Butt

#	mark	size	rectangle	overlap	filename	type	message
159		87 x 80	462, 116, 549, 196		/media/mathie/Seagate Back...	image/jpeg	
160		69 x 33	490, 780, 559, 813		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 24x9 .
161		76 x 30	797, 547, 873, 577	17%	/media/mathie/Seagate Back...	image/jpeg	overlap (intersection over uni..
162		15 x 76	0, 291, 15, 367		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 5x22 .
163		51 x 36	152, 124, 203, 160		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 17x1..
164		74 x 39	253, 127, 327, 166		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 25x1..

Figure C.2: Yellow (Orange) Butts

Yellow_Butt | White_Butt | Snus | Cigarette_Butt

#	mark	size	rectangle	overlap	filename	type	message
4		83 x 93	2595, 2221, 2678, 2314		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 10x5 ...
5		171 x 109	3849, 3250, 4020, 3359		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 16x8 ...
6		148 x 218	1662, 1374, 1810, 1592		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 14x1...
7		222 x 163	1761, 1767, 1983, 1930		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 21x1...
8		93 x 222	1444, 1777, 1537, 1999		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 9x16 ...
9		210 x 167	1697, 2111, 1907, 2278		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 20x1...

Figure C.3: White Butts

Yellow_Butt | White_Butt | Snus | Cigarette_Butt

#	mark	size	rectangle	overlap	filename	type	message
1		317 x 384	1715, 2283, 2032, 2667		/media/mathie/Seagate Back...	image/jpeg	
2		300 x 214	1991, 2244, 2291, 2458		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 29x1...
3		175 x 253	2656, 2329, 2831, 2582		/media/mathie/Seagate Back...	image/jpeg	
4		202 x 218	2164, 1452, 2366, 1670		/media/mathie/Seagate Back...	image/jpeg	
5		249 x 222	2502, 1858, 2751, 2080		/media/mathie/Seagate Back...	image/jpeg	
6		350 x 288	2179, 1802, 2529, 2090		/media/mathie/Seagate Back...	image/jpeg	

Figure C.4: Snus

Yellow_Butt | White_Butt | Snus | Cigarette_Butt

#	mark	size	rectangle	overlap	filename	type	message
15		471 x 315	1640, 796, 2111, 1111		/media/mathie/Seagate Back...	image/jpeg	
16		171 x 163	1386, 1074, 1557, 1237		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 16x1...
17		175 x 257	1566, 1950, 1741, 2207		/media/mathie/Seagate Back...	image/jpeg	
18		218 x 187	1825, 1919, 2043, 2106		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 21x1...
19		234 x 195	2355, 1670, 2589, 1865		/media/mathie/Seagate Back...	image/jpeg	scaled mark measuring 22x1...

Figure C.5: Cigarette Butts

C.2 DarkMark Statistics

class id	class name	count	images	min size	avg size	max size	SD width	SD height	min marks	max marks
0	Yellow_Butt	1032	807	36 x 22	106.71 x 107.84	307 x 501	70.12	71.22	1	17
1	White_Butt	2115	1664	52 x 23	109.13 x 109.02	470 x 408	69.02	68.81	1	30
2	Snus	88	80	83 x 145	231.48 x 230.62	441 x 417	83.37	83.38	1	2
3	Cigarette_Butt	1813	1396	18 x 32	140.66 x 140.58	389 x 1287	86.38	86.35	1	18

Figure C.6: Validation 15%

class id	class name	count	images	min size	avg size	max size	SD width	SD height	min marks	max marks
0	Yellow_Butt	1175	784	25 x 28	104.82 x 99.14	376 x 375	54.16	51.03	1	11
1	White_Butt	2860	1971	4 x 3	106.28 x 103.32	313 x 697	58.46	54.62	1	67
2	Snus	67	59	56 x 117	205.90 x 200.87	627 x 400	91.08	77.12	1	3
3	Cigarette_Butt	2640	1856	6 x 4	130.99 x 124.49	946 x 487	71.93	62.76	1	29

Figure C.7: Train 70%

class id	class name	count	images	min size	avg size	max size	SD width	SD height	min marks	max marks
0	Yellow_Butt	258	198	18 x 25	108.07 x 110.26	307 x 501	75.74	66.44	1	17
1	White_Butt	525	410	52 x 23	110.55 x 107.13	470 x 408	67.72	68.71	1	30
2	Snus	24	21	83 x 145	227.96 x 228.42	441 x 417	92.35	70.90	1	3
3	Cigarette_Butt	463	342	18 x 32	143.17 x 141.93	389 x 1287	76.45	95.18	1	28

Figure C.8: Validation with Rotations 15% (90, 180 and 270 degrees)

class id	class name	count	images	min size	avg size	max size	SD width	SD height	min marks	max marks
0	Yellow_Butt	4741	3114	15 x 11	102.27 x 102.37	376 x 375	53.02	53.12	1	16
1	White_Butt	11517	7958	4 x 3	104.64 x 104.65	313 x 697	56.41	56.41	1	67
2	Snus	253	221	56 x 117	206.40 x 206.61	627 x 400	84.86	84.68	1	3
3	Cigarette_Butt	10452	7367	6 x 4	127.88 x 127.81	946 x 487	67.70	67.61	1	31

Figure C.9: Train with Rotations 70% (90, 180 and 270 degrees)

D Appendix 4 - Darknet Training Results

This shows the training outcomes of different weights that was produced during this thesis. The outcomes show the results that gets printed after a weight has finished doing its last weight/final weight or iteration (duration). It uses only the training data and the validation data to get the statics of precision, recall, F1-score and average IOU (printed on terminal and shows a graph (training vs. validation)).

The training of Tiny-YOLOV2, V3 and V4 was carried out with random = 1 and with an iteration (duration) of 20000 iterations. Shown Table D.1.

	Tiny-YOLOV2	**Tiny-YOLOV3**	**Tiny-YOLOV4**
Kaggle Dataset	Last Accuracy mAP = 62.53% Best mAP = 72.93% class_id = 0 name = Cigarette_Butt precision = 0.92 recall = 0.47 F1-score = 0.62 TP=93, FP=8 and FN=107 average IOU=68.35%	Last Accuracy mAP = 87.73% Best mAP = 93.07% class_id = 0 name = Cigarette_Butt precision = 0.99 recall = 0.75 F1-score = 0.85 TP =150, FP=1 and FN=50 average IOU=78.12%	Last Accuracy mAP = 97.88% Best mAP = 99.61% class_id = 0 name = Cigarette_Butt precision = 1.00 recall = 0.92 F1-score = 0.96 TP=184, FP=0 and FN=16 average IOU=80.91%
Synthetic Dataset	Last Accuracy mAP = 53.18% Best mAP = 59.29% class_id = 0 name = Cigarette_Butt precision = 0.96 recall = 0.41 F1-score = 0.58 TP=235, FP=9 and FN=335 average IOU=70.93%	Last Accuracy mAP = 72.28% Best mAP = 80.64% class_id = 0 name = Cigarette_Butt precision = 0.99 recall = 0.56 F1-score = 0.71 TP=317, FP=4 and FN =253 average IOU=76.99%	Last Accuracy mAP = 82.60% Best mAP = 94.01% class_id = 0 name = Cigarette_Butt precision = 0.83 recall = 0.78 F1-score = 0.80 TP=443, FP=89 and FN=127 average IOU = 65.42%

Table D.1: Training Tiny-YOLOV2, V3 and V4

This training was based on whether Random = 1 and Random = 0 affects the training. The duration that was taken was 8000 iteration which the minimum for 4 classes. The current average loss also tells how much the dataset has been trained. Shown Table D.2.

	Tiny-YOLOV4
Random = 0 **8000 iterations**	Last Accuracy mAP = 53.69% Best mAP = 58.67% class_id = 0 name = Yellow_Butt class_id = 1 name = White_Butt class_id = 2 name = Snus class_id = 3 name = Cigarette_Butt precision = 0.77 recall = 0.58 F1-score = 0.66 TP=784, FP=220 and FN=529 average IOU=62.62% current average loss = 0.2136
Random = 1 **8000 iterations**	Last Accuracy mAP = 54.82% Best mAP = 59.27% class_id = 0 name = Yellow_Butt class_id = 1 name = White_Butt class_id = 2 name = Snus class_id = 3 name = Cigarette_Butt precision = 0.86 recall = 0.55 F1-score = 0.67 TP=693, FP=113 and FN=577 average IOU = 75.20% current average loss = 0.3304

Table D.2: Training of Tiny-YOLOV4

This training was based data augmentation. The duration that was taken was 50000 iteration for random = 1 and 0. It was later decided to continue from the weight trained with random = 0 to train with random = 1 changing the configuration file for another 50000 iteration. That led to 100000 iterations also known as transfer learning. Shown Table D.3.

	Tiny-YOLOV4
Random = 0 **50000 iterations**	Last Accuracy mAP = 40.76% Best mAP = 50.76% class_id = 0 name = Yellow_Butt class_id = 1 name = White_Butt class_id = 2 name = Snus class_id = 3 name = Cigarette_Butt precision = 0.74 recall = 0.45 F1-score = 0.56 TP=12031, FP=4183 and FN=14932 average IOU=60.04% current average loss = 0.0716
Random = 1 **50000 iterations**	Last Accuracy mAP = 73.99% Best mAP = 74.17% class_id = 0 name = Yellow_Butt class_id = 1 name = White_Butt class_id = 2 name = Snus class_id = 3 name = Cigarette_Butt precision = 0.81 recall = 0.78 F1-score = 0.79 TP=21024, FP=4993 and FN=5939 average IOU = 66.48% current average loss = 0.4345
Transfer Learning **100000 iterations**	Last Accuracy mAP = 74.87% Best mAP = 78.21% class_id = 0 name = Yellow_Butt class_id = 1 name = White_Butt class_id = 2 name = Snus class_id = 3 name = Cigarette_Butt precision = 0.89 recall = 0.46 F1-score = 0.60 TP = 12314, FP =1520 and FN =14649 average IOU = 71.54% current average loss = 0.3616

Table D.3: Training of Tiny-YOLOV4 with Data Augmentation

E Appendix 5 - Code, Dataset & Tests

Everything to get started with Darknet, Labelling, Datasets (and weights) and DarknetROS is explained in this repository.

Code: https://github.com/Mathiebhan/darknet_ros

Test Video: https://youtube.com/playlist?list=PLSG9gXgVHC2OKNinlJHKMO8baoJVxkl3G

E.1 CSI camera Lens Testing

The setup of the CSI camera lens testing. The lens that CSI camera uses is by ArduCam.

Figure E.1: CSI Lenses from Arducam

The test setup of the various lens of the CSI camera. The tests were done based on weights from section 7.1 step 1. This weight only had one class.

Figure E.2: CSI Lens Test Setup

E.2 Distance estimation

The two methods that was tested and implemented to estimate the distance of the cigarette litter (implementation was not fully done).

Figure E.3: Based on Section 5.2.1 - Measuring Size and Distance from Images

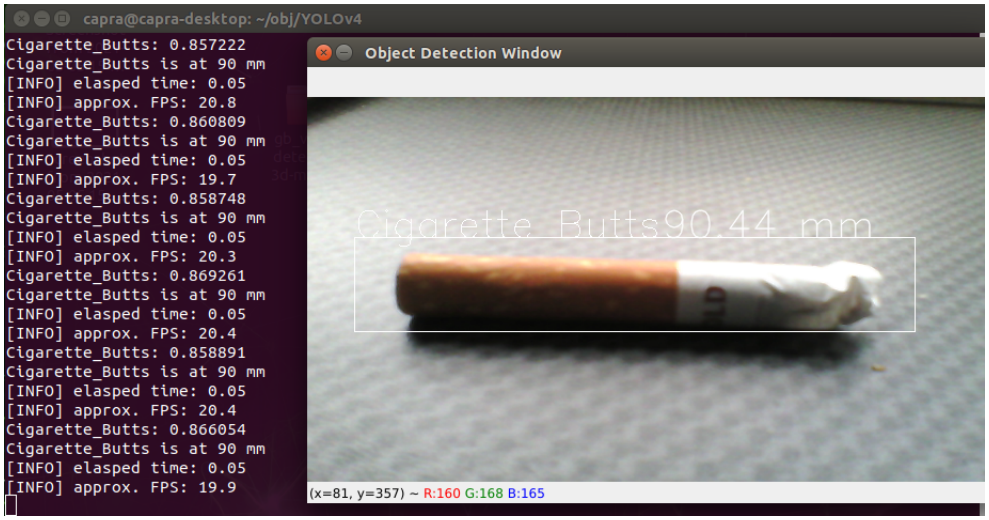

Figure E.4: Based on Section 5.2.3 - Method 2 by M. A. Khan et al.[110]

List of Figures

List of Tables