Recognition and Navigation of a mobile robot by fusing laser and camera information

A Thesis

submitted to the designated

by the Assembly

of the Department of Computer Science and Engineering

Examination Committee

by

Spyridon Syntakas

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION

IN DATA SCIENCE AND ENGINEERING

University of Ioannina

School of Engineering

Ioannina 2021

Examining Committee:

- **Aristidis Likas**, Professor, Department of Computer Science and Engineering, University of Ioannina (Advisor)
- **Kostas Vlachos**, Assist. Professor, Department of Computer Science and Engineering, University of Ioannina
- **Konstantinos D. Blekas**, Professor, Department of Computer Science and Engineering, University of Ioannina

Dedication

To my family
and in loving memory of my Grandmother.

Acknowledgements

I would like to thank both my supervisor Prof. Aristidis Likas and co-supervisor Asst. Prof. Kostas Vlachos for their noble guidance and support with full encouragement and motivation over the course of this thesis and their generous sharing of knowledge during their graduate courses. Their levels of patience, multidisciplinary knowledge and ingenuity is something I am grateful for and a constant inspiration for future scientific research and accomplishment.

I would also like to express my sincere gratitude to all the members of the Dept. of Computer Science and Engineering that through teaching and knowledge sharing over the course of my graduate studies have contributed to my development as an engineer.

Above all else, I am heartfelt grateful to my parents and my grandmother for their inexhaustible support, love and motivation, not only over the course of this thesis but for every step taken towards it. Without their care and engagement, non of this could have been possible.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Spyridon Syntakas, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, 2021.
Recognition and Navigation of a mobile robot by fusing laser and camera information.
Advisor: Aristidis Likas, Professor.

Robotic navigation, motion planning and robotic perceprion are fields of great importance in robotics that are nowadays greatly enhanced by the advances in Artificial Intelligence. Through the usage of sensory modalities and Machine Learning techniques, the robots have a sensory experience of the surrounding environment. Of great importance in robotic perception is the field of robotic vision and especially its applications in solving the task of object detection, which gives mobile robots the ability to interact in various ways with objects of interest in the surrounding environment.

The objective of this thesis is the study, design, and implementation of a method that solves the problem of object detection by fusing laser and camera information, in comparison to most approaches that solve the problem using a single sensory modality. This is accomplished by solving the sub problems of object detection, i.e., object localization and object recognition, in different spaces using different sensory modalities . In the proposed method, object localization takes place in the 3D world surrounding the robot, by segmenting the point cloud given by a 2D laser scanner attached to the robot. By clustering the point cloud using DBSCAN, the position and the 2D layout of objects in the detection range of the 2D laser scanner are obtained as areas of high point density. By projecting the clusters on the image plane of the camera sensor via a Direct Linear Transformation and having the sensors calibrated, the localization of the object is transferred from the 3D world to the 2D plane of the digital image and proposed bounding boxes are obtained. Given the proposed bounding boxes recognition is achieved with the usage of a Convolutional Neural

Network, i.e., a pretrained ResNet, that focuses recognition on the image location of the object and labels are assigned to the corresponding box. The above detection system is used combined with a navigation schema that uses the Potential Fields method in such a way that the robot interacts with detected objects of interest, while is autonomously navigates.

The proposed method that solves the problem of object detection as well as the combinatoric application of both the detection and the navigation system, has been implemented in ROS (Robotic Operating System) and applied and tested both in simulation, using Gazebo, as well as in real case scenarios using the mobile robot Pioneer 3-DX.

Εκτεταμενη Περιληψη

Σπυρίδων Συντάκας, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, 2021.
Αναγνώριση και Πλοήγηση ρομπότ κινητής βάσης συνδυάζοντας πληροφορία από laser και κάμερα.
Επιβλέπων: Αριστείδης Λύκας, Καθηγητής.

Η πλοήγη ρομπότ, η σχεδίαση κίνησης και η ρομποτική αντίληψη αποτελούν από τα πιό σημαντικά πεδία έρευνας και εφαρμογών της Ρομποτικής, όπου σήμερα βελτιώνονται κατα κόρον από την ραγδαία εξέλιξη και έντονη ερευνητική δραστηριότητα στο πεδίο της Τεχνητής Νοημοσύνης. Μέσω της χρήσης αισθητήρων και εφαρμόζοντας τεχνικές Μηχανικής Μάθησης, τα ρομπότ έχουν μια αισθητηριακή εμπειρία του περιβάλλοντος. Αντικείμενο κύριας σημασίας στη ρομποτική αντίληψη, αποτελεί αυτό της ρομποτικής όρασης και κυρίως η εφαρμογή του στην επίλυση του προβλήματος της ανίχνευσης αντικειμένων, που δίνει σε κινητά ρομπότ τη δυνατότητα να αλληλεπιδρούν με διάφορους τρόπους με αντικείμενα ενδιαφέροντος στον περιβάλλοντα χώρο. Η επίλυση του προβλήματος της ανίχνευση αντικειμένων μεταβιβάζεται στην επίλυση δύο υποπροβλημάτων, αυτό του εντοπισμού αντικειμένων και αυτό της αναγνώρισης αντικειμένων.

Σκοπός αυτής της διπλωματικής εργασίας είναι η μελέτη, ο σχεδιασμός και η εφαρμογή μιας μεθόδου που λύνει το πρόβλημα της ανίχνευσης αντικειμένων συνδυάζοντας ετερογενείς πληροφορίες προερχόμενες από λέιζερ και κάμερα, σε αντίθεση με τις περισσότερες προσεγγίσεις που χρησιμοποιούν ως μοναδικό αισθητήριο αυτό της κάμερας. Η προτεινόμενη προσέγγιση επιλύει τα δυο υποπροβλήματα της ανίχνευσης αντικειμένων σε διαφορετικούς χώρους χρησιμοποιώντας δύο διαφορετικά ετερογενή αισθητήρια. Η επίλυση του υποπροβλήματος του εντοπισμού αντικειμένων λαμβάνει χώρα στον πραγματικό περιβάλλοντα κόσμο του ρομπότ

μέσω laser. Το ρομπότ μέσω του laser αντιλαμβλανεται και τις τρείς διαστάσεις του περιβάλλοντος εργασίας του και μέσω ομαδοποίησης του point cloud που προέρχεται από τον 2D laser αισθητήρα με χρήση του αλγορίθμου DBSCAN αποκτώνται ομάδες (clusters) σημείων του point cloud ως περιοχές υψηλής πυκνότητας σημείων που αντιστοιχούν τόσο στην θεση όσο και σε 2D αναπαραστάσεις εντοπισθέντων αντικειμένων. Προβάλλοντας τα clusters στην εικόνα της κάμερας, αφού έχει προηγηθεί βαθμονόμηση των δυο αισθητήρων, ο εντοπισμός του αντικειμένου μεταφέρεται από τον πραγματικό κόσμο στην δισδιάστατη εικόνα, όπου γνωρίζοντας πλεον την σχετική θέση των αντικειμένων προτείνονται bounding boxes. Δεδομένων των bounding boxes το πρόβλημα της αναγνώρισης επιλύεται με την χρήση συνελικτικών νευρωνικών δικτύων, συγκεκριμένα ενός προεκπαιδευμένου δικτύου τύπου ResNet, που εστιάζει την αναγνώριση στις περιοχές ενδιαφέροντος της εικόνας εντός των προτεινόμενων bounding boxes και αναθέτοντας ταμπέλες στα αντίστοιχα κουτιά. Το προτεινόμενο σύστημα εντοπισμού συνδυάζεται με ένα σύστημα πλοήγησης που αναπτύχθηκε βασιμένο στην μεθοδο των Τεχνητών Δυναμικών Πεδίων, και ο συνδυασμός των οποίων επιτρέπει στο ρομπότ να αλληλεπιδρά με εντοπισθέντα αντικείμενα ενδιαφέροντος με την μορφη αισθητηριακών μετρήσεων, καθώς πλοηγείται αυτόνομα στο άγνωστο δυναμικό περιβάλλον εργασίας.

Η προτεινόμενη μέθοδος για την επίλυση του προβλήματος του εντοπισμού αντικειμένων από το ρομπότ καθώς και η συνδυαστική εφαρμογή της με το σύστημα πλοήγησης που αναπτύχθηκε, υλοποιήθηκε με χρήση του ROS (Robotic Operating System) και εφαρμόστηκε και ελέγθηκε τόσο στο περιβάλλον προσομοίωσης Gazebo όσο και σε πραγματικές συνθήκες στο ρομπότ Pioneer 3-DX.

Chapter 1

Introduction

1.1 Motivation

Robotic navigation and motion planning is a field of great importance in robotics that is nowadays greatly enhanced by the advances in Artificial Intelligence and Intelligent Perception. Through the usage of sensory modalities and Machine Learning techniques, the robots have a sensory experience of the surrounding environment. Like humans, robots rely heavily on vision to perceive the environment and the advances in sensory technology together with Deep Learning enable the robot's ability to visually perceive and sense the workspace. Robotic Vision is a core concept in robotic navigation. Analysis of visual data is tied to a lot of mobile robotic applications, alongside point clouds and laser data, which are traditionally used in motion planning. Of great importance, in the field of robotic vision, is the task of object detection which gives mobile robots the ability to interact in various ways with points of interest in the surrounding environment. The advances in object detection by the usage of Deep Learning gave birth to state-of-the-art models like YOLO, that provide real time object detection with high accuracy capable of reinforcing reliability in navigation while requiring only visual data in the form of images, captured by camera sensors.

Although the state-of-the-art models have shaped and revolutionized the task of object detection, the difficulty in training these models to new and large datasets, the difficulty to create new datasets for specific applications and the existence of various other sensors attached to robots give space to new approaches.

1.2 Approach and Contribution

The objective of this thesis is the study, design, and implementation of such an approach, that initially solves the problem of object detection by breaking it to its two core sub problems, that of object localization and object recognition via fusion of laser and camera information in a serial manner. A navigation system is also implemented that achieves autonomous navigation, while the robot perceives the surrounding environment and interacts with it using the proposed detection system.

In the proposed method the two sub problems of object detection are solved in different spaces using different sensory modalities. Object localization takes place in the 3D world surrounding the robot, instead of the image plane as in state-of-the art models, by segmenting the point cloud given by a 2D laser scanner attached to the robot. By clustering the point cloud using DBSCAN areas of high point density, i.e., the clusters, represent individual objects. Thus, information about the location and width of surrounding objects in the 3D world is perceived by the mobile robot and by projecting the clusters on the image plane of the camera sensor, the localization of the object is transferred from the 3D world that took place, to that of the digital image. With knowledge of the location of the object on the image plane, essentially via the laser scanner, the second core component of detection, i.e., recognition, can be achieved with the usage of pretrained CNNs that are focusing recognition on the image location of the object, i.e., recognizing the object that is represented by the 2d cluster on the image plane. CNNs such as ResNet are trained on huge datasets, like the ImageNet, and are capable of recognizing hundreds of objects with high accuracy and fast speed of inference. In addition to the already huge number of classes they have been trained on, new classes can be easily added via transfer learning and fine-tuning.

The number of classes as well as the convenience of flexible fine tuning and transfer learning are the advantages of the proposed method compared to state-of-the-art detection techniques, like YOLO. In addition, the classification of the detected

object is enhanced with distance information provided by the laser scanner, making the control of the robot a lot more convenient in the case of Visual servoing.

The proposed object detection system, that combines laser and camera information, is applied, and tested to a real robot, the Pioneer 3-DX, as well as in simulation. With software developed in the ROS ecosystem, the robot perceives and navigates through dynamic workspace. As proof of concept, using the method of Potential Fields, the robot dynamically moves towards a given goal, by incrementally defining a free path, and interacts with objects of interest that are detected by the proposed system.

1.3 Thesis Structure

This thesis consists of 5 chapters. Chapter 2 is dedicated to a detailed analysis of all the theoretical background that is related to the proposed method and implementation. The concepts discussed have to do with Sensory Data and Sensor Fusion, Machine and Deep Learning for Sensory Data Analysis and Autonomous Navigation of mobile robots.

In Chapter 3 the proposed Perception and Navigation System is presented. The Chapter focuses on both a explanatory analysis of the proposed method as well as on the details of the implementation.

Chapter 4 contains the experimental results obtained mainly from real world scenarios but also from simulation environments. The Chapter starts with a real world detection case for the mobile robot while it autonomously navigates and ends with a summary of real time object detection cases from several workspaces.

The last Chapter, 5 presents the conclusion of this thesis as well as future work proposals.

Chapter 2

Preliminary Concepts

Perception (from the Latin perceptio, meaning gathering or receiving) refers to the act of organization, identification, and interpretation of sensory information that leads to sensory understanding of the environment [1]. Perception in living organisms is achieved via signals that travel through the nervous system, which result from the sensory system of the brain reacting to sensory modalities. The advances in hardware and sensory technology in the recent years gave computers the ability to mimic the way the human perception in achieved [2].

Machine perception is the capability of data organization, identification and interpretation achieved by computer systems and machines in a way that mimics the sensory experience of the environment that a human brain can receive. In most cases a computer system or a machine receive and respond to their environment through attached hardware. The signals and sensory data, that a machine perceives through its hardware is processed and comprehended nowadays mainly through the usage of Artificial Intelligence algorithms and approaches. Machine perception is limited in

comparison to General AI in a way that it does not aim to full consciousness and intentional behavior of the machine but restricts the goal of perception to that of simple sentience. Thus, the aim of machine perception is that of subjective sensing and not that of self-awareness and subjective thought[3], [4].

In robotics, perception is the system that enables the robot to perceive, understand and make inferences about the surrounding environment. The most important component of robotic perception as a system is sensory data processing and modeling, which is essentially modeling of the surrounding environment. Through the interpretation of stimulus modalities, a robot can understand the surrounding environment and have a sensory experience of it. Commonly, Machine and Deep Learning technics are used to achieve the environmental modeling and infer based on the received data. That gives access to the required information that becomes the input to complex control systems which gives the robot the ability to interact with its environment [5], [6].

Given the above, its easily understood that the main difference between perception as a concept in AI in comparison to robotic systems is that a robot makes actions in the real world.The key component of robotic perception, the one that enables the inference based on sensory data and signals is no other than Machine Learning. The advances in Machine and specifically in Deep Learning are these that make robotic perception an applicable and reliable system and a mandatory aspect of modern robotics. Before the analysis focuses on machine learning algorithms it's important to understand where the data come from, the sensory modalities, their representation, and the corresponding sensors. So, the first topic that is analyzed in this thesis and one of the key components of the application is that of sensory data aggregation and fusion.

2.1 Data Fusion and Sensory Data

Data Fusion is the process of combining and integrating data from multiple sources to enhance information and accuracy of measurements taken by using each data source specific asset. When the data come from sensors, then the term Sensor Fusion is used. Thus, sensor fusion, a subset of information fusion, is the process of combining data derived from similar or heterogeneous sensors to obtain information that is more

accurate than the information derived from sensors as individuals [7]. In the case that the data are derived from heterogeneous sensory modalities the term Multi-modal sensor fusion is used. Multi-modal data are a key concept in perception of the environment for both humans and robots. In the case of robots, the two most common sensors are laser scanners, more specifically LiDAR, and cameras. These are also the two sensors that are used in the presented application. By combining the information received by the laser scanner and the camera, the proposed detection system is achieved.

2.1.1 Laser Scanner – LiDAR

Laser Scanners – using LiDAR technology – measure distance with respect to the frame of the sensor and are a key component in robotic perception and navigation, obstacle avoidance, feature extraction, and SLAM. In the context of the proposed application laser scanners can be also used to enhance object detection.

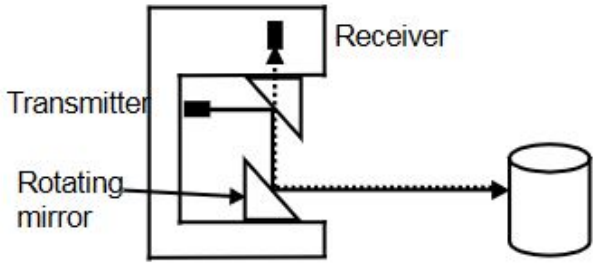


Figure 2.1: Laser Scanner working principle.

There are different types of laser scanners but typically all of them emit a laser beam which is reflected from an object surface back to the sensor and, depending on the method used, the distance of the reflecting surface is calculated [8].

Time-Of-Flight

In Time-Of-Flight (TOF) distance measurement technique a laser pulse is send by the sensor that is reflected by an object surface with a frequency of few GHz. The emitted time and received time from the reflection is measured and the distance between the sensor frame and the object detected is computed as:

$$d = \frac{c}{2\Delta t} \tag{2.1}$$

where d denotes the distance between the sensor frame and the object detected, c is the speed of light (the laser pulse is most of times infrared) and Δt denotes the time between laser pulses emission and receival.

Phase – Shift Principle

The second technique that is used to calculate the distance of a reflective surface from the laser sensors frame is that of phase shift distance measurement. A modulation frequency and a phase shift between a reference and a return signal is used and the distance of the reflective surface is given as:

$$d = \frac{c}{2f}\frac{\phi}{2\pi} \tag{2.2}$$

where f denotes the modulation frequency and ϕ the phase difference between the reference and return signal. The laser pulse is often steered using moving mirrors. The laser beam is deflected by the usage of rotating or non-rotating mirrors but in a lot of cases other deflecting elements, such as prisms, can be used.

The robot used in this thesis has attached a laser scanner with rotating mirrors. To control the motion of scanning, a motor with a rotary encoder and a control driver is used to rotate the deflecting mirror in the desired frequency so that acceptable measurements are taken with respect to the application [9].

Using the above configuration, the laser beams basically correspond to lines in the span of two bases vectors of the sensor frame. In the case of this thesis a mobile robot with a 2D laser scanner with rotting mirrors intuitively emits beams in a hyper plane parallel to the ground and depending on the angle and range of the reflected beams, the distance of all detected objects is measured via TOF.

Point Cloud

Each point in the 3-dimensional space can be described by Cartesian coordinates $[X, Y, Z]$ with respect to the sensor frame or any other frame via corresponding transformations.

The set of points of the 3D space, $\mathbf{P_i} = \mathbf{P_1}, \mathbf{P_2}, \mathbf{P_3}, \ldots, \mathbf{P_n}$, with $n \in \mathbb{N}$ and $\mathbf{P_i} = [X_i,\ Y_i, Z_i]$ that correspond to the points on the objects where the laser beam got reflected, i.e., the points where the laser hit the object, are called a point cloud.This $< \mathbf{X}, \mathbf{Y}, \mathbf{Z} >$ data set represents the shape of the surrounding environment.

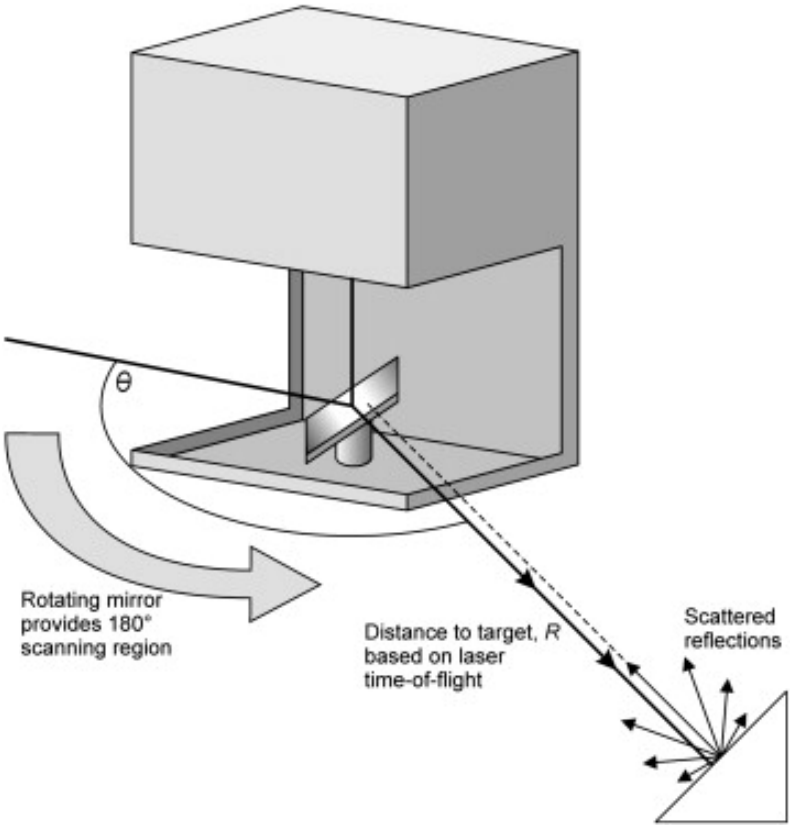


Figure 2.2: A rotating mirror case

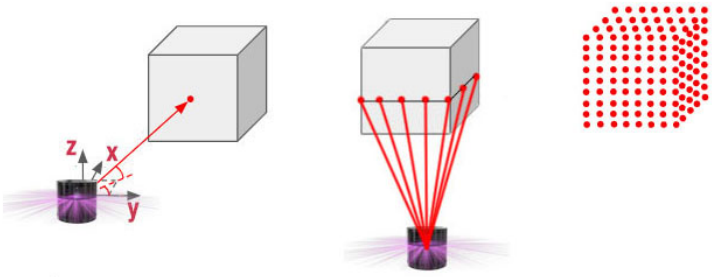


Figure 2.3: 2D and 3D point cloud of a cube

In the general case, point clouds are data sets or collections of points that represent 3-dimensional shapes. They are the simplest form of a 3D model in a way that the plotted pictorial representation of the point-cloud in $\mathbb{R}^3$ is the 3D model of the corresponding object. Sometimes other data is stored alongside the $[X_i, Y_i, Z_i]$ coordinates of each point such as color or intensity, making the point clouds more informative.

In the case of this thesis the point clouds that are created by the laser scanner are 2-dimensional as result of the degrees of freedom of the sensor joint. The point

clouds created via the measurements of the laser scanner determine the exact Cartesian position of the points on the reflective objects detected, giving a 2-dimensional representation of the robots surrounding environment.

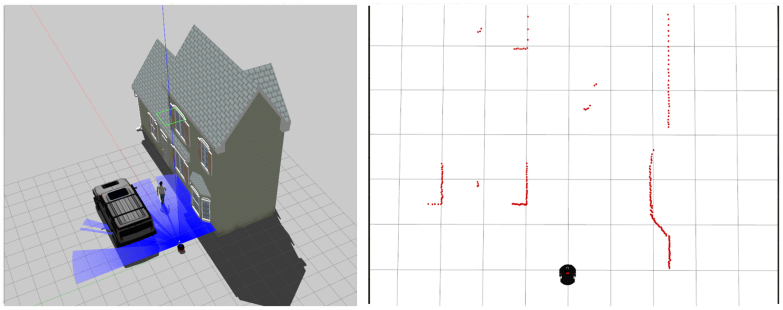


Figure 2.4: 2D Point cloud represented in Gazebo

Note, that by giving one more degree of freedom to the 2D scanner construction of 3D point clouds is possible, although not used in the context of this thesis.

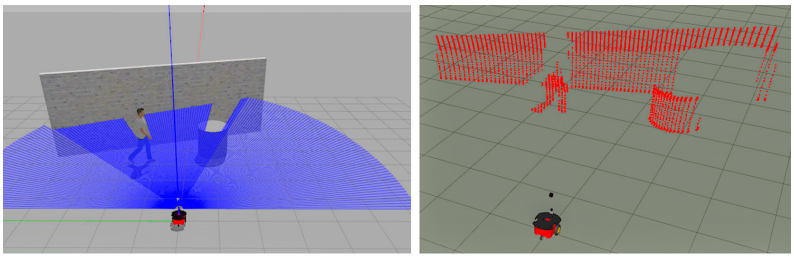


Figure 2.5: 3D Point cloud represented in Gazebo

Point clouds created by the laser scanner of a robot can be used to extract information about the surrounding environment and have many applications in robotic perception and navigation, some of which are used in the proposed application.

2.1.2 Vision Sensor - Camera

Cameras are optical instruments that measure the intensity of light reflected to them by an object. As complex visual systems, they are composed of many parts such as the lens, the shutter, the sensor, processing digital circuits and many more. Two types of sampling is done by a camera, a spatial sampling of light intensity by the sensor and a temporal sampling using the shutter.

The light that is reflected by an object is focused by the camera lens onto a surface of the sensor, which is called the image plain. That is where photosensitive elements

transform the light into a digital image. Most of the sensors used by cameras are based in the photoelectric effect of semiconductors with the most used and known sensors being CCD and CMOS. No further analysis will be done to the differences between the two technologies with respect to the objective of this thesis. The one thing they have in common thought, is that of an element called pixel, that is the photosensitive element mentioned before, that transforms the EM energy of the light to electrical signals, which are later digitized into an image [10].

The model of a camera and the image formation can be approximated by the pinhole camera model with central projection [11].

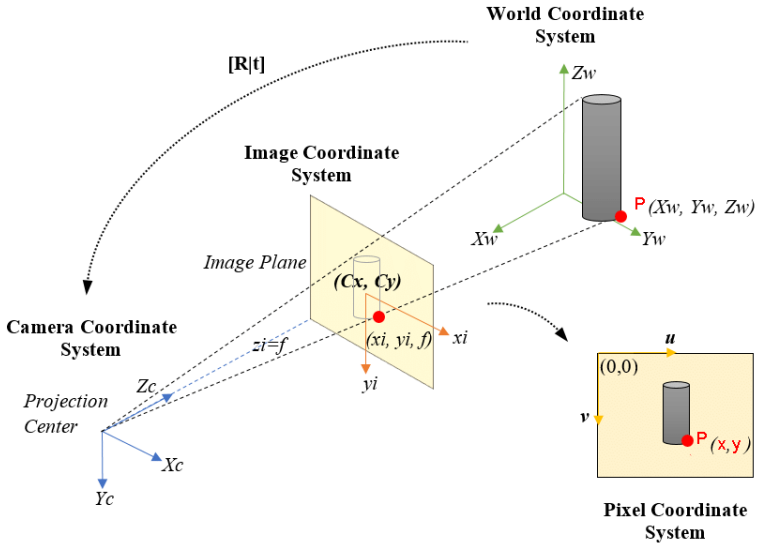


Figure 2.6: Pinhole Camera Model

If **x** describes the 2-dimensional pixel coordinates in the image plane, i.e., $[x, y]$, and **X** describes the $[X, Y, Z]$ Cartesian coordinates of a point in the world frame, then a transformation exists between **x** and **X** that maps the 3D point to the 2D pixel as:

$$\mathbf{x} = P\mathbf{X} \tag{2.3}$$

with P being the projection matrix. In the most general case, there are 4 coordinate systems that transformations take place. These are:

- World coordinate system S_w where $\mathbf{X_w} = [X, Y, Z]^T$
- Camera coordinate system S_k where ${}^{\mathbf{k}}\mathbf{X} = [{}^kX, {}^kY, {}^kZ]^T$
- Image plain coordinate system S_c where ${}^{\mathbf{c}}\mathbf{p} = [{}^cx, {}^cy]^T$

- Sensor plain coordinate system S_s where $^{s}\mathbf{p} = [^{s}x, ^{c}y]^T$

The projection matrix P describes a chain of transformations between the 4 coordinate systems [12], [13], [14].

The first transformation is the one that describes the position and orientation of the camera with respect to the world frame. For the given point it's a 3D to 3D ($\mathbb{R}^3 \rightarrow \mathbb{R}^3$) transformation from the world frame to the camera frame. It can be described as a translation of the center of projection point, i.e., the cameras coordinate system origin, and a rotation giving the orientation of the camera. By denoting the rotation as R and the center of projection as $\mathbf{X_o}$ the transformation matrix is written in homogeneous coordinates as:

$$^{k}H_w = \begin{bmatrix} R & -\mathrm{R}\mathbf{X}_o \\ 0^T & 1 \end{bmatrix} \tag{2.4}$$

thus, $^{k}X_w = {}^{k}H_w X_w$.

The application of this transformation requires the knowledge of the 6 parameters that are used to describe the position and the orientation of the camera frame with respect to the world frame. These parameters are called the extrinsic parameters of the camera. The above transformation is invertible.

The parameters that are used to compute the coordinates of the pixel, where the point from the camera frame will be projected on the 2D image plane, are called the intrinsic parameters. One particular parameter is called focal length. Focal length is a distance parameter, referring to distance from the camera origin to the image plain, describing the magnification properties of the lens and is used in the perspective transformation which maps the 3D point on to the image plane. If the focal length is denoted f then in homogeneous coordinates the $^{c}\mathbf{p}$ pixel's location on the image plane is described by the transformation:

$$^{c}O_k = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.5}$$

thus, $^{c}p = {}^{c}\ O_k\ {}^{k}X_w$. The above transformation is expressed in distance metric units and the minus sign in front of the f parameter is a mathematical representation of the upside-down appearance of an object on the image plane due to the pinhole model. To ease computations a virtual image plane is used positioned before the lens

thus the minus sign can be avoided. ${}^{c}O_k = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ is called frontal perspective transformation and ${}^{c}K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is called the ideal camera matrix.

The units used in the digital image is that of pixels, corresponding to the spatial sampling done by the photosensitive elements and they represent an intensity of light at each discrete point. The unit in the image plane represents physical measurements of distance thus a change in the per unit system must be made. Parameters a_x and b_y correspond to the change of units in each of the two axes of the image plane. Considering square pixels a_x = b_y, but to retain generality we denote $f_x = a_x f$ and $f_y = b_y f$.

Thus, now ${}^{c}O_k = \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ and the camera matrix is ${}^{c}K' = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$
The above transformation applies to the ideal camera. In reality the image plain and the sensor plain differ by a translation described by the translation vector $[c_x, c_y]^T$, because the principal point and the origin of the 2D sensor coordinate frame are not the same. This translation is described through the transformation: ${}^{s}H_c = \begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix}$
The perspective transformation matrix is combined with the translation with respect to the principle point and form the Calibration Matrix also called a Camera matrix: $K = {}^{s}H_c \, {}^{c}K' = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ which is the camera or calibration matrix.

Any point $\mathbf{P} = [X, Y, Z]^T$ in the world frame can be transformed to the corresponding pixel ${}^{c}\mathbf{p} = [x, y]^T$ using the formula in homogeneous coordinates:

$$ {}^{c}\mathbf{p} = K \, {}^{k}H_w \, \mathbf{P} \tag{2.6} $$

where K contains the intrinsic and ${}^{k}H_w$ the extrinsic parameters. Note that this transformation is not reversible, as the depth information is lost. Another intrinsic parameter is called sheer compensation. For digital cameras sheer compensation is typically zero so it is not used with respect to the transformation. The mapping from 3D world

frame coordinates to pixels in the image plane is called a Direct Linear Transformation (DLT), and has eleven degrees of freedom. The six are given from the extrinsic matrix, i.e., the rotation and translation of the camera frame with respect to the world frame. The other 5 are given by the intrinsic matrix. They depend on the camera's manufacturing and are unique for each sensor and camera.

Camera Calibration

Finding the location of a pixel in the image that corresponds to a 3D point in the real world requires knowledge of the intrinsic and the extrinsic parameters of the camera. These parameters are not always available, especially the intrinsic parameters, which depend on the camera manufacturing and are in a way unique for each camera. Given the fact that there is access to the images the camera captures, there is a way to infer these parameters from the taken images. This problem of estimating the intrinsic and extrinsic parameters of a camera is called camera calibration. There are 11 parameters to be estimated, as the projection from world coordinate to image plane is a transformation with 11 degrees of freedom.

Camera calibration uses images with simple patterns and known dimensions, with most common being the checkerboard with known square dimensions. By defining the word coordinate origin on the upper left corner of the image, i.e., the printed checkerboard, and knowing the height and width of the squares, the points $\mathbf{P} = [X, Y, Z]^T$ with respect to the word frame are known. We can also find the corresponding pixels on the image, thus we have pairs of $(\mathbf{p}, \mathbf{P})$. The most common technique for image calibration is named Zhang's Method which uses planar checkerboards [15].

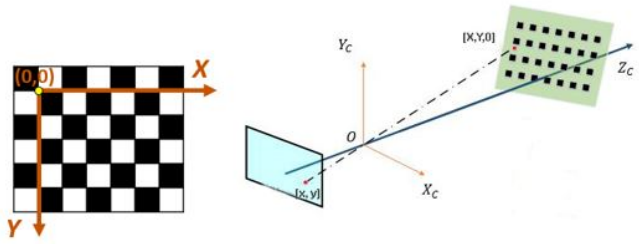


Figure 2.7: Checkerboard - Zhang's Method

The corresponding projections of these points in the image plain, thus the pixel

values $\mathbf{p} = [x, y]^T$ can be described by the transformation explained above. For the i^th pixel $\mathbf{p_i}$ a transformation that describes the homography can be written as:

$$\mathbf{p_i} = \begin{bmatrix} x \\ y \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K\,{}^{k}H_w\,\mathbf{P} = K[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{r_3}\ \mathbf{t}] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{t}] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} =$$

$$H\,\mathbf{P} = [\mathbf{h_1}^T\ \mathbf{h_2}^T\ \mathbf{h_3}^T]\mathbf{P}$$

where M is the 3 x 4 camera matrix, containing the 11 parameters. This homography reduces the degrees of freedom of the intrinsic matrix K by two as the rotation matrix is orthonormal so:

$$\left.\begin{aligned} \mathbf{r_1}^T\mathbf{r_2} &= 0 \\ \mathbf{r_1}^T\mathbf{r_1} &= \mathbf{r_2}^T\mathbf{r_2} \end{aligned}\right\} \quad \Longrightarrow \quad \begin{cases} \mathbf{h_1}^T(K^{-1})^TK^{-1}\mathbf{h_2} = 0 \\ \mathbf{h_1}^T(K^{-1})^TK^{-1}\mathbf{h_1} = \mathbf{h_2}^T(K^{-1})^TK^{-1}\mathbf{h_2} \end{cases}$$

From the above equations is inferred that matrix $(K^{-1})^TK^{-1}$ reduces its degrees of freedom by two with each $(\mathbf{p}, \mathbf{P})$ projection.

$B = (K^{-1})^TK^{-1}$ is a symmetric matrix, thus it can be represented by the parameter vector $\mathbf{b} = [b_{11}\ b_{12}\ b_{22}\ b_{13}\ b_{23}\ b_{33}]^T$

By denoting: $\mathbf{v_{ij}} = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix} \Longrightarrow \mathbf{h_i}^TB\mathbf{h_j} = \mathbf{v_{ij}}^T\mathbf{b}$ and we can infer:

$$\begin{bmatrix} \mathbf{v_{12}}^T \\ \mathbf{v_{11}}^T - \mathbf{v_{22}}^T \end{bmatrix} \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.7}$$

Thus the matric H that contains the 11 parameters is calculated. The above is Zhang's Method for camera calibration which gives estimations of the intrinsic and extrinsic parameters of a camera. The actual application of the Zhang's Method will be further analyzed in the last chapter as it is used to calibrate the camera of the robot.

The sensory data that are used in this thesis and their corresponding sensors have been described in the prior paragraphs. What follows is the approach used to model

the environment using those data in a way that object detection will be applicable and robot navigation can be achieved. In the following sub chapter the theory of the algorithms used to model and analyze the data is presented with respect to the application. Besides the mathematical aspect an effort is made to describe the intuition behind the presented methods.

2.2 Machine Learning

Machine Learning is a subfield of computer science and an application of Artificial Intelligence which Arthur Samuel, a pioneer in machine learning formally defines as a *field of study that gives computers the ability to learn without being explicitly programmed*.

Machine Learning is linked and rooted to domains such as Pattern Recognition, Computational Statistics while also having strong ties with Numerical Optimization. The distinguishing component of machine learning in comparison to traditional data analysis and modeling techniques, that discover hidden knowledge or structure from data and identify patterns within given data sets, is that of learning. So, in a deeper sense machine learning can be seen as the study of algorithms that improve themselves automatically and learn by the usage of data making them capable of predictions and inference based on the given data as well as allowing generalization on previously unknown information. By learning based on sensory data from the robots perception system, algorithms can precisely model the robots surrounding environment and allow the robot to navigate autonomously by enhancing its control [16]. Machine learning algorithms can be categorized as follows based on intuition and results [17], [18]:

- Supervised Learning: Given a set of data in the form of $< X = \{\mathbf{x^i}, t^i\},\ i = 1, 2, ..N >$ where $\mathbf{x^i}, t^i$ are respectively input/features and desired output/label, a function is learned inductively and thus a model is constructed that can make predictions and generalize on new data

- Unsupervised Learning: Given an unlabeled data set $< X = \{\mathbf{x^i}\}, i = 1, 2, ..N >$ the algorithms learn the hidden patterns in the unlabeled data and using that learned hidden structure they categorize the data into clusters, i.e., groups of similar data

- Reinforcement Learning: Algorithms, also called intelligent agents, learn with respect to the dynamic environment they interact with, in order to maximize a cumulative reward.

Due to the algorithms that are used in this thesis, only Unsupervised Learning and Supervised Learning will be analyzed further. More specifically, the Unsupervised Learning algorithm that is used belongs to the category of Density based clustering methods. The Supervised Learning algorithms that are used belong to a subset of machine learning that grew, in performance and applicability, in recent years and is that of Deep Learning.

Deep Learning is based on artificial neural networks and representation learning. In the field of visual perception, which is a key component of this thesis, a specific type of Neural Network is used – the Convolutional Neural Network.

2.2.1 Density based Clustering - DBSCAN

Clustering is the action of dividing data points to groups based on similarity, in a manner that data points that end up in the same group are more similar than data points that end up in other groups. Based on the approach, clustering can be model-free (partitioning), hierarchical, similarity-based and model-based. Model based clustering algorithms can be categorized further into three distinct groups of algorithms based on the clustering model:

- Connectivity based clustering
- Centroid-based clustering
- Density-Based clustering

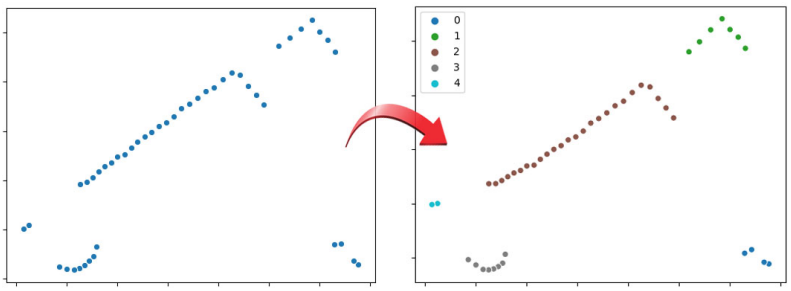


Figure 2.8: Clustering

Further analysis will focus on Density based clustering, because that approach is used in the application described in this thesis.

Density based Clustering

According to density-based clustering, clusters are defined as areas of higher density separated by lower density areas. In other words, cluster is a region of high density that surrounds similar points and is separated from other clusters with regions of minimum variance. Comparing this type of clustering to the other categories, it excels in detecting clusters with arbitrary shapes, does not require prior knowledge on the number of clusters and has a notion of noise. These advantages of Density based clustering are used in extent in the proposed detection system and the versatility on applicability is one of the reasons this type of clustering was chosen for the application.

DBSCAN

DBSCAN is a popular clustering algorithm which requires only two parameters – ε and minPts. DBSCAN model is based on minimum density level estimation, using a threshold for the number of neighbors, minPts, within the radius ε of a point. Based on these two hyperpameters DBSCAN classifies all points in a data set as core points, density reachable points or noise [19], [20].

- A point p that has more than minPts neighbors with respect to the radius ϵ is considered as a core point.
- If a point p belongs to the neighborhood $N_\epsilon(q)$ of point q and q is a core point, then point p is called directly density-reachable from q. That's a non-symmetric relation.
- If q is a core point and p is a non-core point that is directly density reachable from q, p is called a boarder point.
- If a path $p_1, ..., p_n$ exists, with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly density reachable from p_i, then point q is density-reachable from p.
- Two points that are density-reachable from point p are called density – connected. That's a symmetric relation.

- A point o that is neither a core-point nor a boarder point is called noise or outlier.

Given a dataset and hyperparameters ϵ and minPts, a cluster S must satisfy two conditions:

1. Given a point $p \in S$, if a point q is density-reachable from p, then $q \in S$.
2. All points within a cluster must be density – connected.

Intuitively, if p is a core-point, then all reachable points from p form a cluster. This can be clearly understood from the figure below: The circles around the points denote

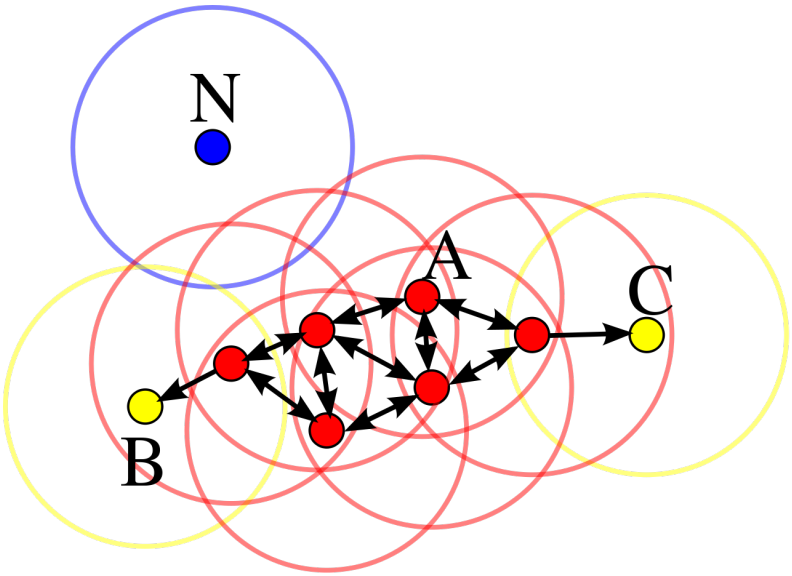


Figure 2.9: DBSCAN pictorial intuition

the radius ϵ. minPts is set to 4 and the arrows correspond to density reachability from point A. Point A is a core point and points B, C are boarder points both satisfying density reachable definition. Point N is considered an outlier, or a noise point, because it is not density reachable with respect to point A.

The pseudocode for the Original Sequential DBSCAN Algorithm is given in Algorithm 2.1. DBSCAN run time complexity is $O(n^2)$ and it depends on how many times RANGEQUERY is called. In the most simple and naive implementation RANGEQUERY is called once for each point and it checks the distance of one point from all other n-1 points in the dataset, thus the $O(n^2)$ complexity. By using R*-three in the RANGEQUERY, average time complexity can be reduced to $O(nlogn)$ since by having a small ϵ-neighborhood range, on average querying for neighbors has to traverse a

Algorithm 2.1 Pseudocode for the Original Sequential DBSCAN Algorithm

Require: $DS : dataset, \epsilon : radius, minPts : Density\,threshold, dist : Distance\,function$

1: **for each** $p \in DS$ **do**
2: **if** $label(p) \neq undefined$ **then**
3: $continue$
4: **end if**
5: $Neighbors\ N \leftarrow$ **RANGEQUERY**$(DS, dist, p, \epsilon)$
6: **if** $|N| < minPts$ **then**
7: $label(p) \leftarrow Noise$
8: $continue$
9: **end if**
10: $c \leftarrow next\ cluster\ label$
11: $label(p) \leftarrow c$
12: $Seed\ set\ S \leftarrow N \backslash \{p\}$
13: **for each** $q \in S$ **do**
14: **if** $label(q) = Noise$ **then**
15: $label(q) \leftarrow c$
16: **end if**
17: **if** $label(q) \neq undefined$ **then**
18: **continue**
19: **end if**
20: $Neighbors\ N \leftarrow$ **RANGEQUERY**$(DS, dist, q, \epsilon)$
21: $label(q) \leftarrow c$
22: **if** $|N| < minPts$ **then**
23: **continue**
24: **end if**
25: **end for**
26: $S \leftarrow S \cup N$
27: **end for**

small amount of paths in the tree, but there is no guaranty that this will always be the case.

2.2.2 Supervised Learning - Artificial Neural Networks

In machine learning, Supervised learning is the action of learning a function $f(.)$ that maps an input to an output, based on prior knowledge of input-output pairs, called a data set [21]. The data set consists of examples each having a set of features/attributes, with one of them being the label/class which characterizes the example. An example is typically represented as a vector and the corresponding label. So a data set can be denoted as $< X = \{(\mathbf{x_n}, y_n), n = 1, 2, 3, ..N\} >$ with $\mathbf{x_n} \in \mathbb{R}^d$ and y_n being the corresponding label. The labeled data set, the one that based on its examples the function in learned, is called the training set. The inferred function $f(.)$ acts as a model or a procedure that given an input vector, consisting of the attributes, predicts the class attribute so $\forall \mathbf{x_n} \in X \exists f(\mathbf{x_n}) = y_n$.

The objective of supervised learning, is inferring such a function that can classify and predict previously unseen examples with high accuracy, also called as achieving generalization. In comparison to unsupervised learning, which objective is to find hidden structure and patterns in an unlabeled data set, supervised learning intuitively can be seen as the task that best approximates association between input and output with respect to the prior knowledge of the labeled features.

There are many supervised learning algorithms and even more applications. With respect to the objective of this thesis, further analysis will focus on Deep Learning and more specifically on Convolutional Neural Networks (CNNs on ConvNets) for image recognition and detection. Before that though, a summary analysis will be done on Artificial Neural Networks, starting from the single node perceptron, so that an intuition of the methods and technics used can be made before further analysis is done to CNNs.

2.2.3 Perceptron

Perceptron is an algorithm for supervised learning of binary classifiers. The algorithm was invented by Frank Rosenblat and is considered to be the first generation of Artificial Neural Networks. It is actually a single neuron. As a binary classifier, it is a function that given an input vector of features assigns either the label +1 or -1 on the

given vector. More specifically perceptron is a linear model, so it bases the prediction on a linear prediction function and a set of weights corresponding to each input vector feature. So, the problem that the perceptron algorithm solves can be written as follows.

Given a training dataset $< X = \{\mathbf{x_n}, y_n\} \forall \mathbf{x_n} \in \mathbb{R}^d, y_n \in \{-1, +1\}, n = 1, 2, 3, ..., N >$ estimate the weights - parameters $w; w_0$ of the linear model such that

$$\forall n \in N \rightarrow y_n(w^T \mathbf{x_n} + w_0) \geq 0$$

The approach of the perceptron algorithm for solving the above problem is the following and is called the Perceptron criterion:

- If the $\mathbf{x_n}$ is correctly classified, which corresponds to $y_n(w^T \mathbf{x_n} + w_0) \geq 0$, zero action are taken
- If $\mathbf{x_n}$ is incorrectly classified, which corresponds to $y_n(w^T \mathbf{x_n} + w_0) < 0$, then $-y_n(w^T \mathbf{x_n} + w_0)$ is the contribution of the $\mathbf{x_n}$ example to the cumulative error.

The weighted sum, a linear combination of the features, is passed through an activation function, which in the simplest case can be a step function, i.e., $sign(.)$. A step function has a discontinuity thus is non-differentiable, but still fits the solution of the simple problem.

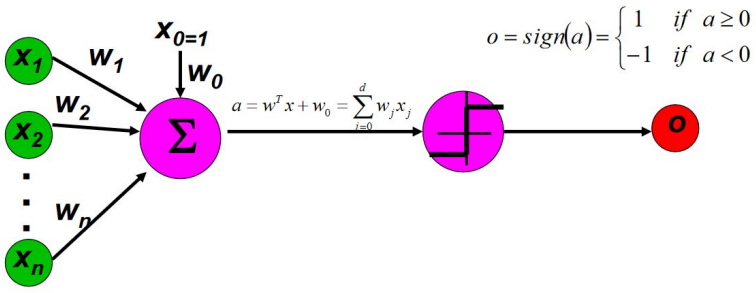


Figure 2.10: Perceptron pictorial representation

The Perceptron Criterion Function is constructed as

$$E(w) = -\sum_{\mathbf{x_n} \in M} y_n(w^T \mathbf{x_n} + w_0) \tag{2.8}$$

where $M = \{\mathbf{x_n} : y_n(w^T\mathbf{x_n} + w_0) < 0\}$ and is minimized if $\{w, w_0\}$ is a solution vector.

If no examples are misclassified, then the criterion function evaluates zero. The weights are updated as:

$$w^{new} = w^{old} + ny_n\mathbf{x_n} \tag{2.9}$$

$$w_0^{new} = w_0^{old} + ny_n \tag{2.10}$$

repeatedly until convergence or until a threshold of maximum updates is reached, where n is the learning rate.

Note that a different activation function than the step function, for example a sigmoid can be applied after the linear transformation. The sigmoid being a continuous nonlinearity, in comparison to the step activation function, makes the overall function of the perceptron differentiable and that is enabling new approaches to training and learning the weights w.

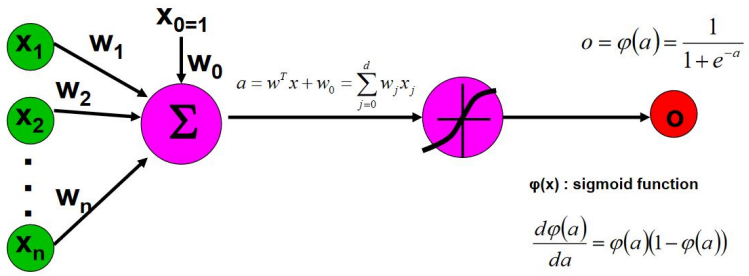


Figure 2.11: Non linear activation

So the perceptron unit computation is now written as

$$f(\mathbf{x_n}) = \phi(w_0 + w^T\mathbf{x_n}) \tag{2.11}$$

where $\phi(\alpha) = \frac{1}{1+\exp^{-\lambda\alpha}}$ Using as an error function, now called also a loss function, the sum-of-least squares,

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}(f(\mathbf{x_n}) - y_n)^2 \tag{2.12}$$

the vector w that minimizes $E(w)$ is the solution to the classification problem.

So actually, the classification problem is reduced to finding $w : min\{E(w)\}$ which is solved via gradient descent and the weights are repeatedly updated until convergence as:

$$w^{new} = w^{old} - n\frac{\partial E}{\partial w} \tag{2.13}$$

Due to a biological analogy, a perceptron unit is commonly called a neuron, or in the context of a neural networks as it will be explained below, is also called a node. Biologically, even though the analogy is a simplification which adds a lot of restrictions to the ways of Neural Network intuition, the output of a neuron on the synapse is a signal that is the result of the input signals of its dendrites.

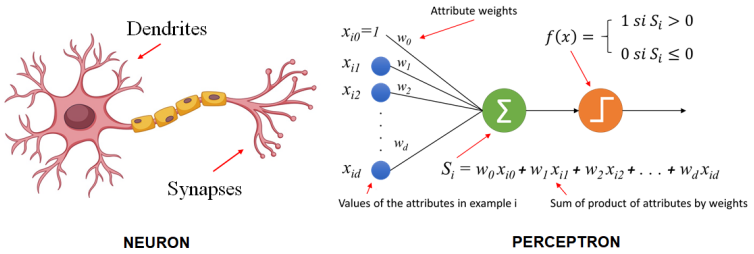


Figure 2.12: Biological analogy of neuron to perceptron

From a geometrical standpoint, perceptron tries to find a hyperplane that separates the input data, thus classifying them.

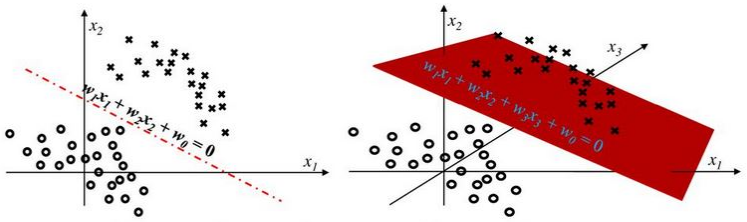


Figure 2.13: Data separation

A single perceptron, being a linear model unit, cannot separate non-linear separable data, for example cannot solve the XOR problem. For non-linearly separable data a combination of perceptron units solves the problem.

2.2.4 Artificial Neural Networks

By stacking perceptron units we can create linear layers, which are actually linear functions that apply linear transformations on the input vectors $x_n \in \mathbb{R}^d$ to an output vector $x_{out} \in \mathbb{R}^o$ where o and d spaces can be of different dimension, i.e., $dim(d) \neq dim(o)$. The biological analogy is that, of a set of neurons having their dendrites connected to the same input signals. The firing of a biological neuron is mimicked by the addition of a non-linearity after the linear transformation, such as the sigmoid that the sum of products of the vectors with the corresponding weights is passed. Mathematically, by having nonlinear activation functions after the linear transformation is what enables a combination of linear layers to infer any kind of function, even non-convex ones. Note that non-linearities in most cases are continuous, thus differentiable, that also makes the network function differentiable with interesting training methods applicable. An Artificial Neural Network is a network consisting of layers of connected perceptron units. Layers are characterized as input, output and hidden layer.

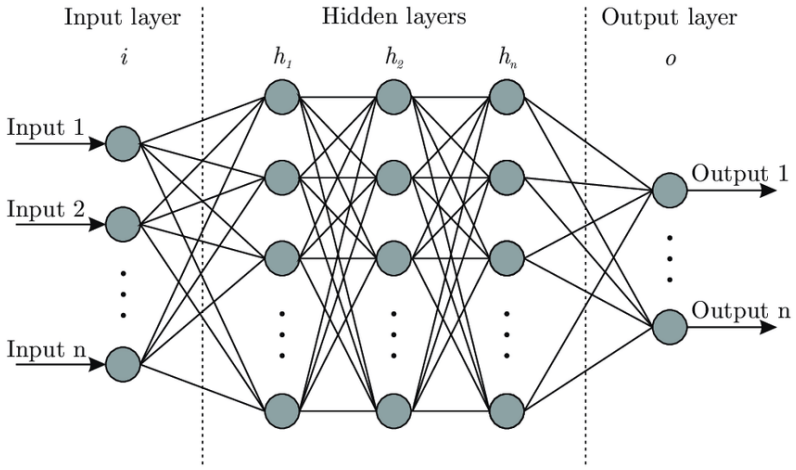


Figure 2.14: Graph representation of an Artificial Neural Network

- Input layer: The first layer of the network. Its nodes take the values of the input vector.
- Output Layer: Last layer of the network. It has one node for each value of the network output. In the case of binary classification or regression there is only one unit and K-units in the case of K-classification problem.

- Hidden layer: Layers between the Input and Output layer. The values of these nodes are learned via training using an algorithm called backpropagation.

The number of hidden layers in a network is called the depth of the network – thus the term deep learning. The number of nodes in a layer is called the width of the layer.

Except of the input layer, all other neurons use a nonlinear activation function. Between the input and output layer there should be at least one hidden layer but there can be many more hidden layers, depending on the complexity of problem to be solved.

A Multi Layer Perceptron (MLP) is a feed-forward neural network, which means that nodes are connected in an acyclic graph with no connections between neurons in the same layer, fully connected neurons between consecutive layers and the computations are done sequentially.

Mathematically, there is a little difference between the single node perceptron explained in the previous section and the MLP case, in terms of how the models are trained and how they infer.

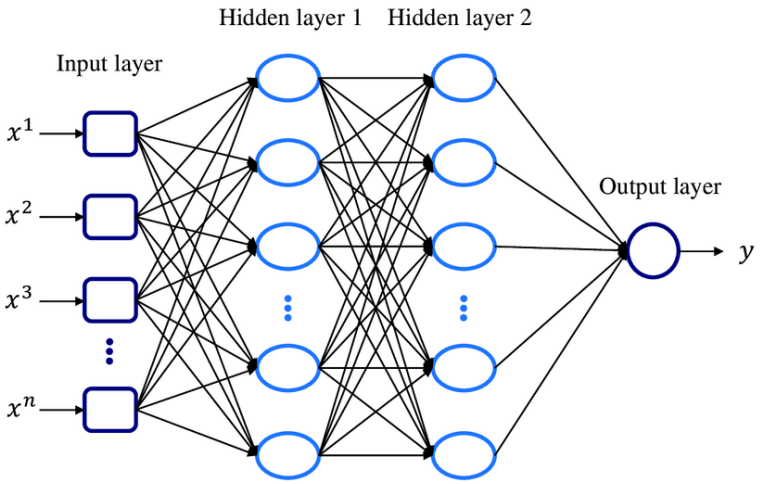


Figure 2.15: Graph representation of an Artificial Neural Network

- i^l as the i^th neuron in the layer l
- $u_i^{(l)}$ as the total input of the i^{th} neuron in the layer l
- $y_i^{(l)}$ as the corresponding output
- $w_{ij}^{(l)}$ as the weight parameter of connection between neuron j^{l-1} to neuron i^l

- $w_{i0}^{(l)}$ as the bias
- $\varphi^l(.)$ as the activation function of layer l
- d^l as the number of neurons in layer l
- H as the depth of the network

then a forward pass, i.e., the calculation of the output layer with respect to input data, of a network with vectorial input data with d components and p outputs is described as follows:

- For the input layer $y_i^{(0)} = x_i$ and $y_0^{(0)} = x_0 = 1$.
- For the hidden layers the total input of the i^{th} neuron of the layer h is calculated as:

$$u_i^{(h)} = \sum_{j=1}^{d_{h-1}} w_{ij}^{(h)} y_j^{(h-1)} + w_{i0}^{(h)}, i = 1, 2, 3, ..., d_h$$

  and the output of the i-th neuron in the layer h is:

$$y_j^{(h)} = \varphi_h(u_i^{(h)}), i = 1, 2, 3, ..., d_h$$

  and

$$y_0^{(h)} = 1$$

- As for the total output, the inferred value is $o_i = y^{(H+1)}, i = 1, ..p$.

The network acts a function that performs a mapping from $\mathbb{R}^d$ to $\mathbb{R}^p$, with respect to the training data set, though which the weight parameters $\{w_{ij}, w_{i0}\}$ of the function are learned.

By applying linear transformations and by passing the weighed linear combinations to nonlinearities sequentially from layer to layer, the inferred function of the network can solve complex problems with high dimensionality. The algorithm that enables the training of such neural networks so the weights are obtained is called backpropagation.

Given a training data set $< D = \{(\mathbf{x_n}, l_n)\}, n = 1, 2, 3, ..., N >$, where $\mathbf{x_n} = (x_{n1}, x_{n2}, x_{n3}, ..., x_{nd})$ is the vector of features of each example and l_n the assigned pre-known continuous value, which is a regression problem, and given a specific MPL architecture, as far as the depth d and width of each layer, $o(\mathbf{x_n}; w)$ denotes the output vector of the MLP after the forward pass of the input vector $\mathbf{x_n}$, with $w = \{w_{ij}; w_{i0}\}$.

Training

As done in the simple single node perceptron, a loss function is defined and via optimization, the learned weights are obtained [18]. The loss function that is used is again that of sum-of-least squares, and by denoting the target values vector as $\mathbf{t} = (t_1, t_2, ..., t_n) \in \mathbb{R}^N$, the loss function is written as:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (o(\mathbf{x_n}, w) - t_n)^2 \tag{2.14}$$

Once again by finding the weights $w = \{w_{ij}; w_{i0}\}$ that solve the $min\{E(w)\}$ optimization problem gives the learned parameters of the MLP.

Gradient descent, or a more efficient version called stochastic gradient decent, can be used so that the weights can be obtained through the already known iterative application using one example from training set at a time ,or a batch of them in the case of stochastic gradient descent:

$$w^{new} = w^{old} - n \nabla E(w^{old}) \tag{2.15}$$

The gradients, that must be computed to apply the iterative formula, are obtained using an algorithm called backpropagation. In detail description of the backpropagation algorithm falls out the topic of this thesis. Intuitively thought, after a forward pass is done for an input vector x, an error is obtained based in the output, i.e., the inferred predicted value - and the actual value of the input x. That error flows backwards to the network - a backward pass- and by applying the chain rule backwards, layer by layer, starting from the last layer, the gradients of the error function with respect to each layers weights and bias are computed. So, using the iterative update formula, until convergence or a max threshold, the total learned parameters of the network are computed. There are many more details about MLP and Neural Nets but with respect to the topic of this thesis, no further analysis will be done In the context of this thesis classification is the main task of the network. A different loss function than the sum-of-squared-errors is used, and a specific activation function is applied most of the times in the last hidden layer. So, SoftMax activation and Categorical cross entropy loss function will be briefly presented below.

SoftMax activation

The SoftMax function takes as input a vector $\mathbf{x} = [x_1, x_2, ..., x_n] \in \mathbb{R}^N$ and applies a normalization so that the output is a vector consisting of N probabilities proportional to the exponentials of the input vector components. By normalizing the input vector components into a probability distribution over N possible outcomes, the output describes a categorical distribution, also known as the Boltzmann distribution [18], [22]. After the application of SoftMax to a vector all its components are bounded between $[0, 1]$. By denoting SoftMax as σ:

$$\sigma(\mathbf{x}) = \frac{\exp^{x_i}}{\sum_{j=1}^{N} \exp^{x_j}}, \;\; i = 1, 2, 3, ..., N \;, \; \forall \mathbf{x} = [x_1, x_2, ..., x_n] \in \mathbb{R}^N \tag{2.16}$$

SoftMax is used as the activation function of the last linear layer for K – class classification problems. It takes as input the networks last layer output in the form of the vector x, consisting of K linear transformations with respect to the weights, and outputs the probability of the i^{th} class prediction as:

$$P(y = i|\mathbf{x}) = \frac{\exp^{\mathbf{x}^T w_i}}{\sum_{j=1}^{K} \exp^{\mathbf{x}^T w_K}} \tag{2.17}$$

Thus, by applying the SoftMax activation function we get a probability distribution of the predictions for the K-classes in the training dataset.

Categorical Cross Entropy

Categorical cross entropy is used as a loss function in multinomial classification problems. The function calculates the difference between two discrete probability distributions and specifically between the probability distribution of predictions and the one-hot encoded vector representation of the actual class [18], [22]. Formally, this is written, for each example of the dataset, as:

$$loss = H(y_i, \hat{y}_i) = -\sum_{i=1}^{K} y_i \log \hat{y}_i \tag{2.18}$$

where y_i corresponds to the i^{th} index value of the one hot encoded vector, $\hat{y}_i$ corresponds to the i^{th} index value of the networks output.

Categorical cross entropy is well combined with SoftMax activation function. SoftMax activation function makes the output of the network a categorical distribution of the K-classes. One hot encoded representation of an example is a discrete probability distribution thus the difference between the two can be computed with cross entropy.

2.3 Image Classification

Image classification is the task of categorizing and labeling groups of pixels within an image or the entire image itself based on its content. Simply put, it is the problem of assigning a class label to an image. As a problem of Computer Vision it falls in the general category of problems known as object recognition, 2.16. The two other main problems of object recognition is that of object localization and that of object detection. Object localization involves putting a bounding box around an object on a digital image, whereas object detection combines image classification and localization and puts a label on the bounding box, classifying its contents [23].

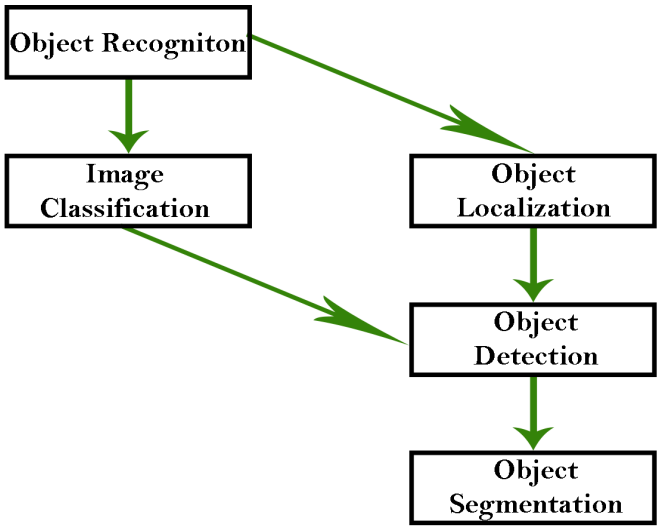


Figure 2.16: Computer Vision tasks block diagram

In this thesis an approach is used such as that image detection is achieved by fusing data from laser scanner, i.e., object localization, and assigning a label thought image classification of the bounded area.

So further analysis will focus on image classification, and specifically achieving classification with Convolutional Neural Networks (CNN). Before that, in order that the importance of CNNs in image classification accuracy is made clear, a small introduction is made by using Linear classifier and MLPs for the task [24]. Where they fail is where the ConvNets shine.

2.3.1 Linear Classifier and ANN for Image Classification

An image could be used as input to an MLP by flattening the 2d tensor that the image is represented through (assuming a greyscale image) and by training an MLP on a labeled image data set, an MLP classifier of images would seem possible.

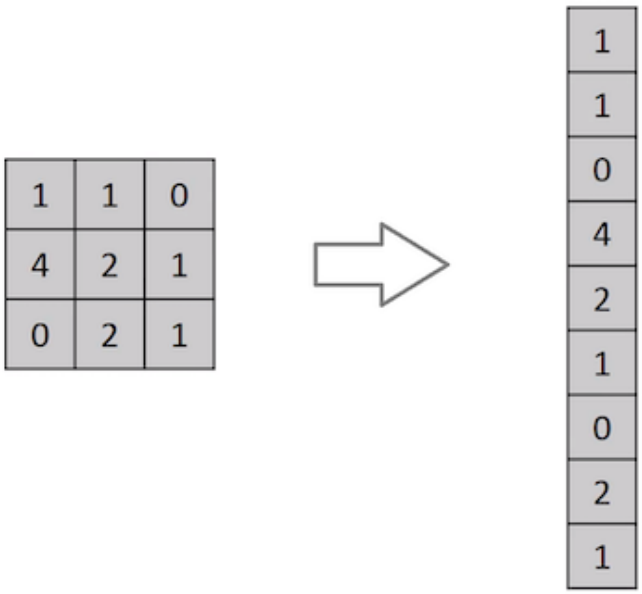


Figure 2.17: Grey scale Image flattened to vector

There are many issues with respect to the above - specifically due to spatial structure information on images and the importance of it in classification, which in this case is lost and not taken into account - and that can be easily seen, by training a simple linear classifier like the perceptron on an image data set.The Figure 2.18 below pictorially describes such a classifier.

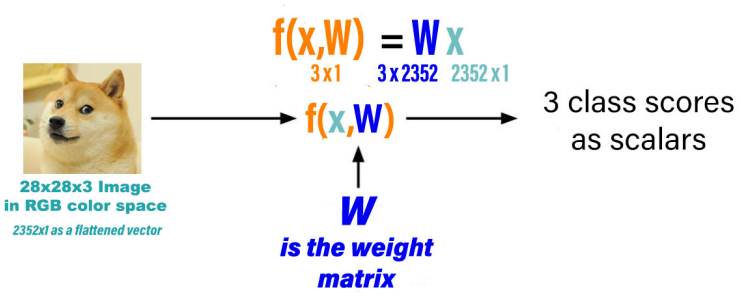


Figure 2.18: Linear Classifier for Image Recognition

For simplicity assuming there are only 4 pixels in the image, a flattened tensor

of the image passed to $f(\mathbf{x}, W)$ that is a 3-class linear classifier, would look like in figure 2.19.

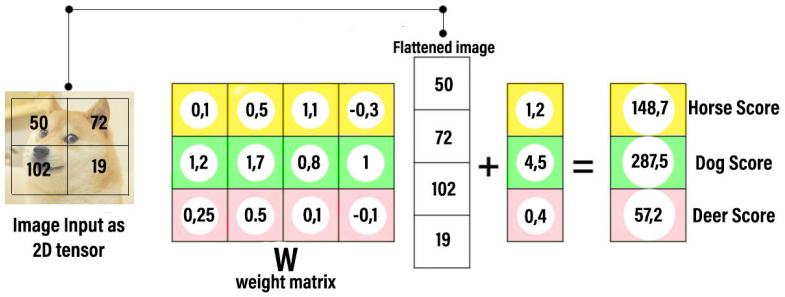


Figure 2.19: Image content inferred from a Linear Classifier

The weight matrix W is $4x3$ matrix due to 4 pixels and 3 classes in this simplified example. Each row is intuitively a learned template for each one of the three classes. Thus, classification of images using linear classifiers is similar to template matching. The dot product of each row with the flattened pixels, give a similarity score of the template with the given image. The 3x1 bias, acts as an offset that correspond to class independence.

What the linear classifier is really doing can be seen, by visualizing the learned templates – weights - as images, by reshaping the rows of the W matrix to tensors of size equal to the images fed to the linear classifier. That pictorially can be seen on the Figure 2.20.

Figure 2.20: Learned weights visualization

As it is now clear, the linear classifier only learns one template per class. That

template tries to average all different variations of the image of the same class on the training dataset and that results to bad classification accuracy on complex images.

By using MLP - thus stacking linear layers acting as linear classifiers followed by non-linearities in the form of activation functions - we can have the hidden layer weights be intermediate templates trying to match the details that the single linear classifier template averages and fails to take advantage of.

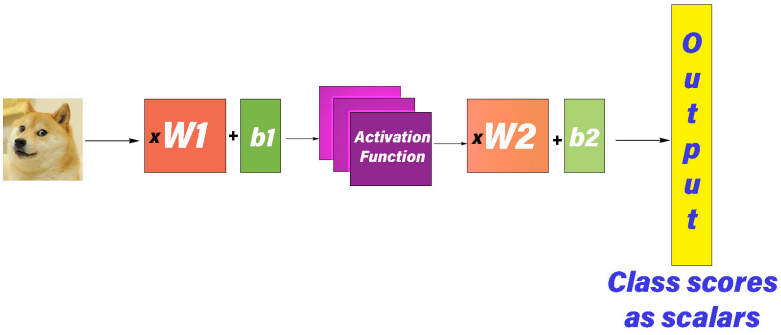


Figure 2.21: Image passed to a two layer ANN.

Intuitively the output of the dot product of the templates – row vectors- of W_1 with the flattened image vector, which corresponds to a similarity of the image with the template, is further fed to the next layer which tries to enhance template matching further, computing the similarity with the row vector templates of W_2. An approach like this, although it gives insight of how an image classifier would work, does not make use of the spatial structure of an image and the correlations between adjacent pixels. That is the where the idea of the convolution and Convolutional Neural Networks (CNN) comes into play and spatial and temporal dependencies of images are taken into account. The next chapter will make use of the ideas used in MLP such like backpropagation and optimization of the loss function, and with the addition of convolutional layers, image classifiers that use images spatial information will be explained.

2.4 Convolutional Neural Networks

Convolutional Neural Network (CNN) [25] is a type of Artificial Neural Network specializing in processing and classifying grid-like topology data, making it perfectly fitted for image classification. CNNs revolutionized fields like Computer Vision, where

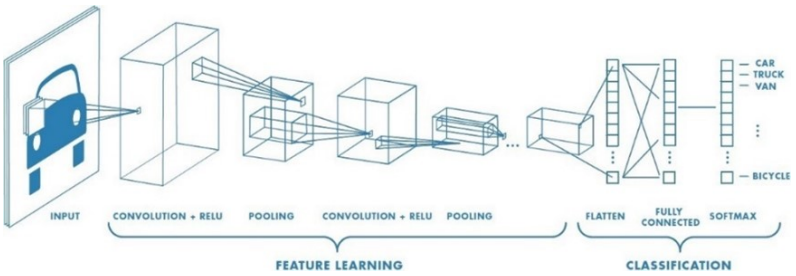


Figure 2.22: Convolutional Neural Network Architecture.

prior to deep learning approaches, the preprocessing of imagery data was hand-made and mandatory before passing the images as input to the image recognition algorithms, while CNNs require minimum preprocessing as the network acts as feature extractor, using filters that are learned via training. These filters, also called convolution kernels, using the mathematical operation of convolution, extract features from the images, by sliding in a way along these features and providing responses named feature maps. The use of convolution kernels also reduces dramatically the number of computations needed to be done in order to successfully do the task of image classification especially in huge data sets. A CNN actually uses a system like the Multilayer Perceptron which has been altered with the addition of the convolution operation and redesigned for spatial data reservation and reduced processing requirements. Key role in the CNN algorithm is that of Convolution [24], [22], [26].

2.4.1 Convolution

Convolution is a mathematical operation between two functions f(t) and g(t), commonly signals, denoted by the * symbol, that derives a third function as [27]:

$$f * g(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau \tag{2.19}$$

That is the definition of continuous 1-dimensional convolution which is calculated as the integral of the multiplication between the first signal and the reversed and translated in time second signal.

Convolution is a really important operation in signal processing and one of the most important concepts of electrical engineering. That is because the output signal of a system that is linear time invariant is the convolution of the input to the system with the impulse response, which is a function of time that describes the dynamic system[28].

The discrete version of the convolution, which is an operation on two discrete time signals is defined by the integral,

$$f * g[n] = \sum_{k=-\infty}^{\infty} f[k]g[n-k] \tag{2.20}$$

So, convolution of two discrete signals is done by multiplying and accumulating the values of the overlapping samples of the fist signal with the flipped version of the second signal, which is the impulsive response.

Moving from one dimension to the n-dimensional space, the concept of convolution, both continuous and discrete can be applied to n signals spanning among n mutually perpendicular dimensions, or simply put having n-dimensional signals, thus multidimensional convolution is defined.

One of the most used cases of multi-dimensional convolution is that performed on 2-dimensional discrete spatial signals. Such a signal is the digital image, that is represented as a 2d tensor and in the field of both image processing and deep learning, 2D discrete convolution plays a really important role. The way the 2-dimensional discrete convolution is performed is the same as in the case of the one dimension, with the difference that the impulsive response signal is flipped twice. The impulsive response signal is called convolution kernel or filter in the case of 2D convolutions. Mathematically, 2D convolution on images can be defined as:

$$y[i,j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m,n]x[i-m, j-n] \tag{2.21}$$

where x is the input image that the convolution is performed, h is the filter and y is the output image. The kernel, usually having a lot smaller dimensions that the input image, is placed in every spatial possible location over the input image and multiplication and accumulation of the overlapping values is performed in every position. A pictorial example is given in Figure 2.23.

Images are commonly defined in the RGB Color space, thus are defined as 3d tensors. Convolution is performed is each one of the 2D tensor of the corresponding color channels with a 2D filter, outputting three 2-dimensional tensors. The 2D tensors are accumulated into a single value.

2D convolutions in a key concept in image classification and computer vision in general. It's the changing factor in neural networks that makes them have amazing

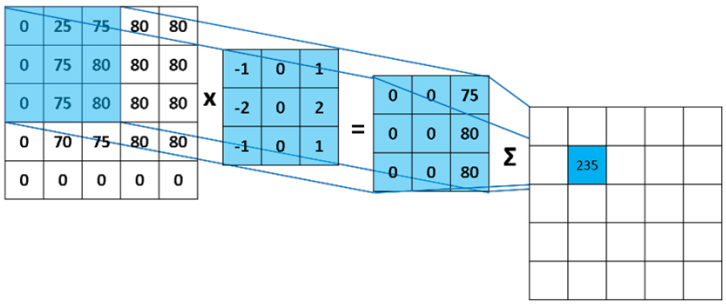


Figure 2.23: 2 dimensional discrete convolution.

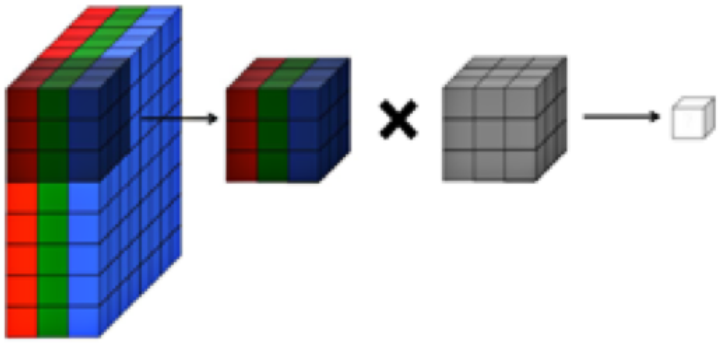


Figure 2.24: Convolution operation on a 3 dimensional tensor .

performance on data where spatial or temporal information exists as convolution retains that information intact.

2.4.2 Convolutional Neural Networks for Image Classification

CNNS [22], [29] use the concept of 2-dimentional discrete convolution and create Convolution layers. The structure of a CNN layer is that of a 3-dimentional grid structure, which has height, width, and depth. The spatial dimensions of one layer are dependent on the spatial dimensions of the previous layer and this dependency relationship is important to be preserved among grid structures because the convolution operation depends on these relationships. The depth-dimension corresponds to the number of channels in each layer and is a dimensionality of the 3D tensor that represents the transformed data in each layer. For the input layer and assuming RGB color space, the depth is equal to 3, as many as the channels of the input image data. For the hidden layers the depth is equal to the number of feature maps.

A CNN acts like an MLP in the way the data flows through the network and the

way it is trained, i.e., via backpropagation, but the big difference is that of spatial connection between consecutive layers. The common types of layers in a CNN are the convolution layers, the pooling layers and the non-linear activation layer, commonly ReLU. After the stacks of convolutional and pooling layers which as a total are also called the feature extractor, a fully connected or dense network is attached which maps the input images to classes as in an feed forward architecture, using commonly SoftMax and categorical cross entropy at the output.

The input data, which in this case are images, are represented as 3-dimetional tensors. The two dimensions are containing the spatial structure and properties between pixels and the third dimension is distinguishing these relationships for each one of the 3 channels. For the input layer the properties represented in the 3-d tensor (the RGB image) are the intensities of each color. For the hidden layers, which have depth dimension > 3, the values of the pixels in the 3-d tensors represent other properties such as edges or shapes that were extracted from specific spatial parts of the image. By denoting H, W, d as height Width and depth of layer, the l^{th} layer has an input of

$$H_l \times W_l \times d_l$$

All convolution layers have this 3-dimensional structure, with the input layer having a predefined dimensionality with respect to the image data and always d=3. All hidden layers maintain the 3D structure but the values of the individual d_l 2-dimensional tensors are no longer pixel intensities but represent specific features extracted through the convolution operation. For the hidden layers, the individual 2D tensors are called feature maps or activation maps.

The learned parameters of the Convent's are called kernels or filters. They are 3-dimensional tensors with height and width a lot smaller than of the corresponding layer and depth equal to that of the corresponding layer. So by denoting the height and width of the kernel as K the 3D structure of a kernel of the l^{th} layer can be described by

$$K_l \times K_l \times d_l$$

The discrete convolution operation that takes place, slides the 3D kernel through the layer placing it at each possible position and calculates the dot product between the $K_l \times K_l \times d_l$ values of the kernel with the $K_l \times K_l \times d_l$ proportion of the image at each position. Each position the filter takes in the current layer corresponds to a feature in the next layer. Thus the number of possible position of a filter in the

current layer defines two of the dimensions of the 3D tensor of the next layer. That is the next layers height and width. These dimensions can be calculated as:

$$H_{l+1} = H_l - K_l + 1 \tag{2.22}$$

$$W_{l+1} = W_l - K_l + 1 \tag{2.23}$$

The depth of the next layer depends on the number of filters used in the current layer. Every kernel gives through the convolution operation each own feature map, of which the next layer consists of. The number of kernels used in each layer is something to be taken under consideration as the kernels are the learnable parameters of the network. Each kernel, though the convolution operation, tries to identify and extract a spatial pattern in the part of the image that the dot product is calculated. Thus, by using many kernels the convolutional layer can recognize many spatial patterns and extract many features, but the complexity of the model increases as the learnable parameters increase.

If we denote:

- Tensor $W^{(p,q)} = [w_{ijk}^{(p,q)}]$ as the p^{th} filter in the q^{th} layer, where i, j, k are the positions of the height width and depth of the filter
- Tensor $H^{(q)} = [h_{ijk}^{(q)}]$ the feature map of the q^{th} layer

the convolution operation performed that maps layer q to layer $q+1$ is written as:

$$h_{ijp}^{(l+1)} = \sum_{r=1}^{K_l} \sum_{n=1}^{K_l} \sum_{k=1}^{d_l} w_{rnk}^{(p,l)} h_{i+r-1,j+n-1,k} \tag{2.24}$$
$$\forall i \in 1, 2, 3, ..., H_l - K_l + 1, \forall j \in 1, 2, 3, .., W_l - K_l + 1, \forall p \in 1, 2, 3, ..., d_l$$

The convolutions performed by the first layers of the network are recognizing simple shapes and patterns and as the data is passed through the network and spatially reshaped respecting the spatial dimensionality connections of the convolutions, the later layers of the network detect more complex features. Thus, the convolutional layers act as a feature extractor really similar to that of human vision, where sets of neurons are activated when certain shapes are in the visual field. Recognition and construction of complex shapes and patterns is in a way hierarchical. That hierarchal engineering is one of the most important aspects of neural networks.

Three important properties are present in the convolutional layers due to the operation of convolution, and these are sparse interactions, parameters sharing and equivariant representations.

Sparse interaction is achieved by using filters with dimensions of height and width much smaller than that of the corresponding layer. In addition to the intuitive use of small kernel size in order to extract smalls patterns in digital images, the usage of such kernels results in reducing the number of connections between the input and output layer, resulting in reducing the overall number of parameters. That lowers the models number of computations to infer which makes the model more memory efficient and increases the generalization.

Parameter sharing, also known as tied weights, refers to the fact that a specific weight in the filter is used in all the possible positions that can spatially be placed with respect to the layer its applied. So the weights are shared by all the neurons in a feature map. During training, tied weights, result to a filter updating its weight via backpropagation with respect to the hole 3d-tensor that is applied. That means that even though a kernel is extracting features in a small spatial region of the image (or 3D tensor for hidden layers) by taking every possible position and computing the dot product with the same weight over all the image, the filter is not learned for every specific position it was placed but with respect to the images as whole.

Equivariant representation means that Convnets are equivariant with respect to linear translation, though they are not equivariant with respect to scaling and distortion. If $f(g(x)) = g(f(x))$ then we say that $f(.)$ is equivariant with respect to g(.). If g(.) is a linear translation, then the function that is produced through convolution is equivariant to $g(.)$. Intuitively that means that by shifting spatially the input in a convolutional layer, the output is equally shifted.

Detector Stage - Non Linearity

After the convolution layer, as in the case of MLP, a non-linearity is applied. The operation of convolution creates a weighted linear combination of the spatial inputs, so convolution layers are basically performing linear transformations. In order to enhance networks expressiveness so that it can represent any function, just as in the MLP case, non-linearities are applied. That way the mappings between layers achieve the desired complexity to model the given data.

The most common non-linear activation function used in CNNs is ReLU. Rectified

Linear Unit is the ramp function:

$$f(x) = x^+ = max(0, x) \tag{2.25}$$

with x being the product of the convolution operation.

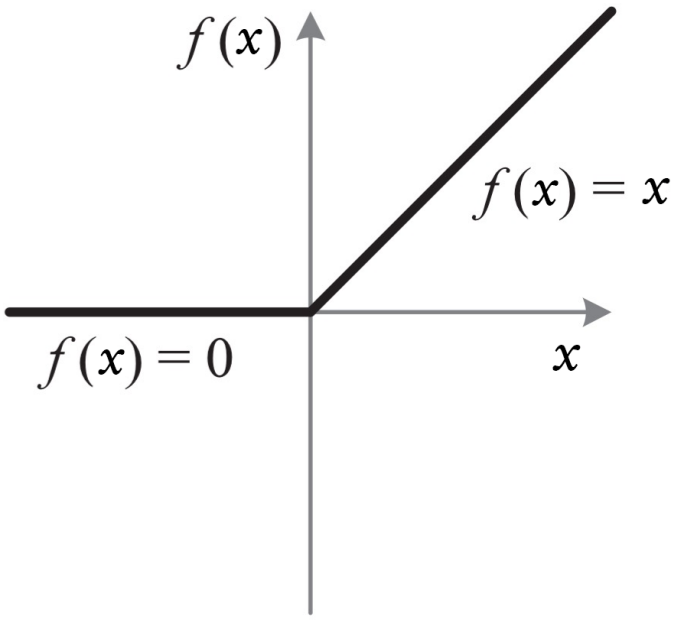


Figure 2.25: ReLU activation function.

In the context of CNNs ReLU applies a threshold to the $H_l \times W_l \times d_l$ values in the layer, which are then fed forward to the next layer. ReLU is a big improvement, in terms of speed and accuracy of the network, compared to activation functions like the sigmoid or tanh. The speed of ReLU also enhances the speed of training making it effective as the networks becomes deeper and the number of parameters increases.

Pooling Layer

Pooling is an operation that takes place in small regions of the image, or the feature map if it refers to a hidden layer , denoted as $P_l \times P_l$. The Pooling operation is performed in each 2d-tensor corresponding to the d_l activations maps. For every $P_l \times P_l$ value only the maximum is returned. That approach is called max pooling.

So pulling lowers the height and width dimensions of the activation maps and leave the depth unchanged. The intuition of pooling is described by what is called translation invariance. Two images that contain similar shapes and patterns may have these same patterns occur in different locations of the image. Translation invariance

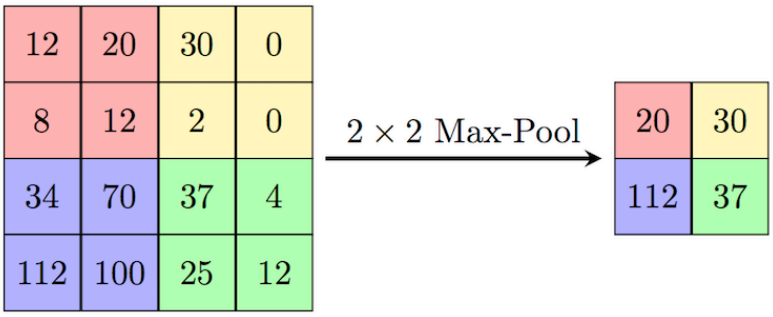


Figure 2.26: Max Pooling operation.

is what gives the network the ability to classify such images with the same label. So basically, by the use of pooling, even if an images feature or shape pattern is translated in a small manner the activation maps remain relatively the same.

Pooling layers are not used after every convolution + ReLU combination. In fact only few pooling layers are enough, because the reduction in spatial dimensions of the 3d tensors that they do with respect to height and width are such, that the tensors would reduce to tiny dimensions fast.

CNN Training

By combining Convolutional Layers followed by ReLU activations and adding max-pooling layers every once in a while, the feature extractor part of the CNN is complete. There are many more things one could study with CNN such Local response Normalization, data augmentation to avoid overfitting, but they fall out of the scope of this thesis. After the Feature Extractor part in a CNN always follows a fully connected neural network (Dense), that is similar to an MLP. The last layer of the feature extractor flattens the feature maps to single vector, which represents the input image after all the transformations, and is fed through the fully connected linear layer. The Dense network using a SoftMax in most cases, outputs a probability distribution over all the classes, and the label is assigned to the image that the network is more confident.

Both the weights of the fully connected network and the kernels that are used to extract features via convolution are the learnable parameters of the CNN and are obtained via backpropagation. Both the convolution operation, that is a linear transformation over a small region of the tensor, as well as the ReLU are continuous and differentiable, so backpropagation is applicable.

In general, training of CNNs uses the same general methods used by MLP [18],

with the basic algorithm being that of backpropagation. A loss function is calculated by feeding the network with data from a training dataset. By solving the minimization problem of finding the kernels and the weights that minimize the loss function over the hole training dataset, the model parameters are given. The minimization is done with gradient descent, or other optimization methods such stochastic gradient descent or Adam, which use an iterative update of the filters and weights using gradients at each layer until convergence. The gradients of the kernels with respect to the loss function are calculated though backpropagation as in the case of MLPs. One problem though that comes with training (one that occurs when training CNNs or networks with many layers in general) is that of the gradient flow. Two of the most common problems, regarding gradient flow, are that of vanishing gradients and exploding gradients.

The weights of a network are updated iteratively and proportionally to the partial derivative of the error function with respect to the weight. When the gradient is reduced to a really small value, the weight update is close to zero, so basically the training process stop. The application of the chain rule makes the computation of the gradients of the fist layers of the network to be a multiplication of really small values. That makes the weight updates on early layers impossible, so the training is either really slow or it stops as a whole. This seems like the gradients vanish in the first layers, thus the name of vanishing gradients.

An inversely proportional problem is that's of the exploding gradients. Large value of the error function cause gradients to take unacceptable large values and as the error propagates backwards the training becomes unstable and unreliable due to large changes in the weights via the iterative updates.

Although these problems can be solved in many cases via batch normalization, still the training of deep neural network architectures may fail due to problems in convergence during the optimization process. One way to counter this is by using ResNets, an architecture that is used in this thesis application for inference, so a small analysis follows.

Residual Neural Network

An specific CNN architecture that is used in the application described in this thesis is the ResNet [22]. ResNet is the 2015 Imagenet winner with top-error 3.6%. More specifically to achieve that performance, an ensemble of ResNets was used, each with

152 layers. Networks with such depth are generally difficult to train, and in the time of the network's creation the tools available where not many. The problems of vanishing gradients and exploding gradients that appear in backpropagation while gradients flow between layers, are solved sufficiently with batch normalization, weight regularization, gradient clipping etc.

The main problem of training deep architectures is that of convergence during the optimization process. Networks tend to have low accuracy and the creation of the Residual Neural Network solved that by adding skip connections as a concept. Although hierarchy is very important concept in neural networks, the addition of skipped connections overcomes the problem of convergence during optimization, by giving alternative paths to the data flowing through the network. The idea is given pictorially in the Figure 2.27.

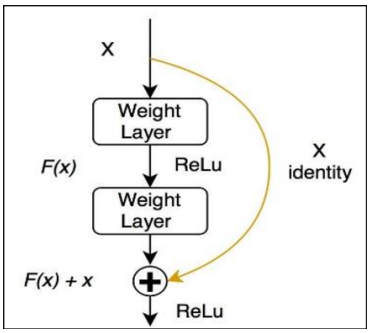


Figure 2.27: ResNet Block.

As its obvious, instead of the standard flow between layer l and $l+1$, the input of layer l is also fed to the layer $l+r$, where $r > 1$. That way the input of the layer $l+r+1$ is not $F(\mathbf{x})$, but $H(\mathbf{x}) = F(\mathbf{x}) + \mathbf{x}$. By the usage of padding that keeps the spatial dimensionality intact, now backpropagation has alternative paths to flow the error backwards and compute the gradients, which solves the problem of convergence. Also, these alternate paths are used in forward feed, so the network decides the feature maps used for inference depending on the input.

The above block is called the residual block and is the distinguishing characteristic of Residual Neural Nets, whose general abstract architecture of the feature extractor can be seen in the Figure 2.28.

In the context of this thesis a pretrained Residual Neural Network is used to achieve recognition of small areas of an input image, where the laser proposes that an object of interest is located.

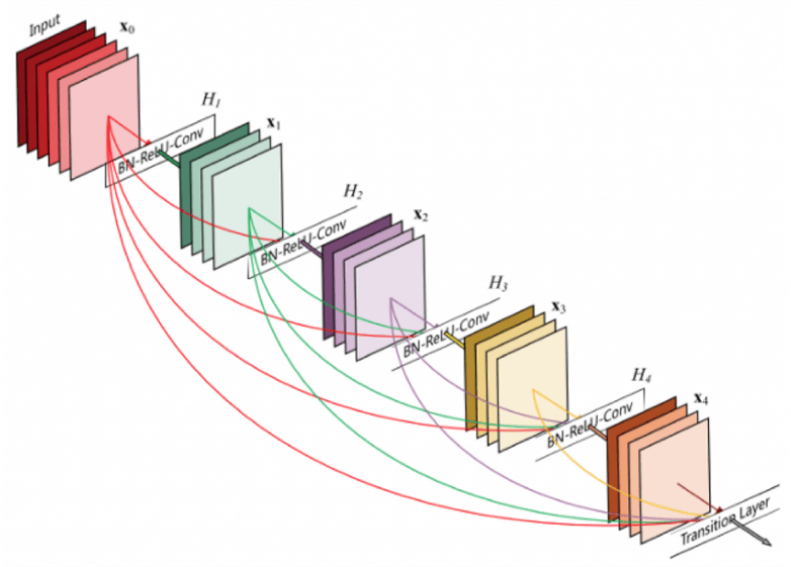


Figure 2.28: Residual Neural Network architecture.

YOLOv3

The YOLO (You Only Look Once) [30], [31] approach for object detection in images is an algorithm that is applied only once to the full image. Object detection in YOLO is solved as a regression task that provides class probabilities for detected images. In comparison to previous approaches that apply an image detection model several times at various regions of the image, the predictions of YOLO model are fast since only a single forward propagation through a neural network, i.e., a CNN, is needed for each image. The CNN infers the bounding boxes as well as the labels simultaneously. Thus, for the training of the CNN, a loss function is used that is defined using contributions of both the bounding boxes accuracy and the classification accuracy. The current version is YOLOv5 but in this thesis the YOLOv3 model is used and further described.

The YOLO model divides the image into an $S \times S$ grid of cells and simultaneously predicts bounding boxes and class probabilities for each cell. The cells detect objects that are included in them. If the center of an object falls into a receptive field of the cell, this cell is responsible for detecting that object. For each grid cell, the parameters of B bounding boxes are predicted along with confidence scores indicating the confidence of object inclusion in the corresponding box. Moreover for each cell, K class conditional probabilities are computed that are used to assign a class label to the cell object. Since four parameters are used to specify a bounding box location, the total number of outputs that YOLO provides for an input image is equal to $S*S*(B*5+K)$.

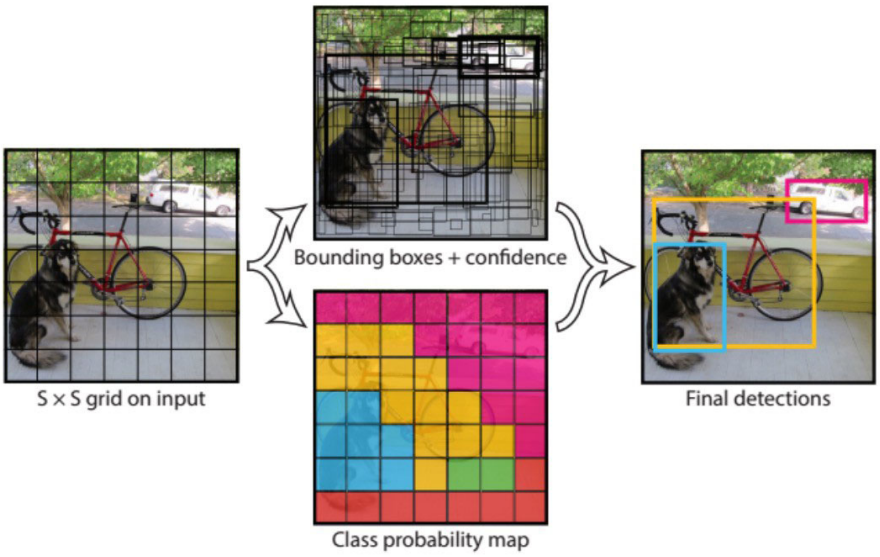


Figure 2.29: Yolov3 model localization.

The YOLOv3 model uses anchor boxes as prior knowledge of the bounding boxes dimensions. Anchor boxes are representative bounding boxes obtained by clustering the bounding boxes of the training set into B clusters using k-means. The centroid box of each cluster is considered as an anchor. Anchors are actually log-space transforms which can be intuitively seen as offsets to prior defined default bounding boxes, i.e, the anchors. YOLO v3 has three anchors, which results in the prediction of three bounding boxes per cell.

Bounding box parameters are predicted using regression . If a cell is offset from the top left corner of the image by (c_x, c_y) then each one of the B bounding boxes for this cell is defined by predicting four coordinates t_x, t_y, t_w, t_h. If the bounding box anchor has width p_w and height p_h, then the coordinates of the predicted bounding box will be,

$$b_x = \sigma(t_x) + c_x \tag{2.26}$$

$$b_y = \sigma(t_y) + c_y \tag{2.27}$$

$$b_w = p_w \, e^{t_w} \tag{2.28}$$

$$b_h = p_h \, e^{t_h} \tag{2.29}$$

The probability of an object belonging to a specific bounding box is described by the objectness score. As a convolutional backbone, YOLOv3 uses Darknet-53.

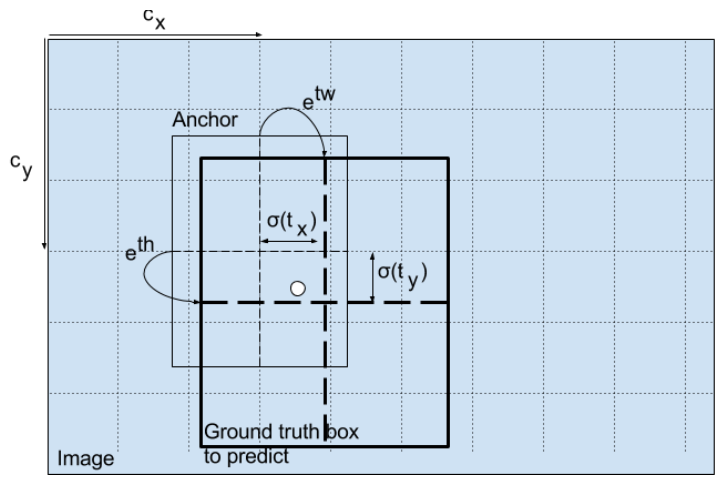


Figure 2.30: Detector Output Log-space Transform.

YOLOv3 uses logistic regression to predict the objectness score of a bounding box. In additon, YOLOv3 uses the sum-of-squared-error loss method as the error for the bounding box predictions. For object class predictions, the network uses the Binary Cross-Entropy error function. YOLOv3 makes use of the Non-Maximum Suppression technique in order to get rid of the unnecessary bounding boxes that are predicted, so that we eventually have only one bounding box for each object. A threshold is also defined, based on which boxes with objectness score lower than the threshold are ignored. YOLOv3 also uses prediction among different scales by downsampling the input image. At each scale, each cell predicts 3 bounding boxes using 3 anchors, making the total number of anchors used a total of 9.

A pictorial application of the YOLO model is described in the Figure 2.31.

2.5 Autonomous Navigation

As mentioned before, the difference between robot perception and machine perception is than in the case of the robots the goal is interaction with the surrounding environment though their sensory experience. By feeding the inferred information of the perception system to control systems, the robots can gain the autonomy to interact with the surrounding world. One specific control system that can use the inferred information is the motion control system. Of great importance in the field of robotics is that of navigation of mobile robots, that in modern applications utilizes the perception systems inference in order to achieve autonomy in navigation through motion

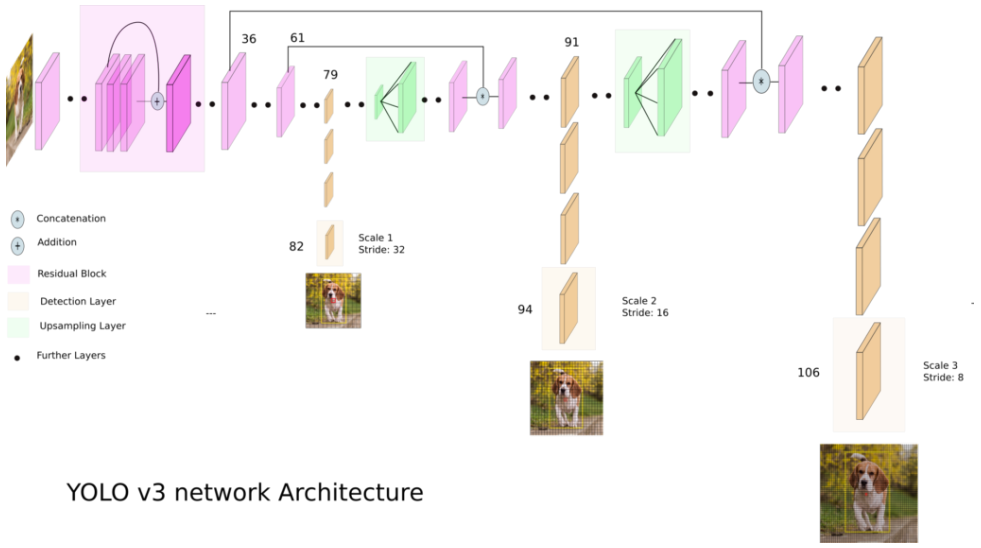


Figure 2.31: YOLOv3 model Architecture.

planning. The space that the robots is assumed to operate is called the workspace. In the case of this thesis application workspace is the $\mathbb{R}^3$ and denoted as W. The workspace contains obstacles, that need to be avoided. By denoting the space of the i^{th} obstacle as W_{O_i}, the free workspace is given as $W_{free} = W - \bigcup W_{O_i}$

To plan the motion of a mobile robot[32], [11], [33], its pose must be explicitly specified in most of the cases. The space of all possible poses of the robot is called the configuration space C. Configuration spaces topologically are k-dimensional differentiable manifolds enhanced with a local property. Every configuration of the robot, i.e., the position and orientation, is denoted as q and is defined by n-parameters, where n is the dimensions of the configuration space. The most common type of such spaces are Euclidean spaces that are locally diffeomorphic. So, configuration spaces are k-dimensional manifolds that are locally diffeomorphic.

The subset of C that the robot can move and be positioned freely is called Free Configuration Space C_{free}, and the space that is resulting from configurations q of the robot that collide with obstacles in the working space is called the Obstacle configuration space denoted as $C_{obstacle}$, i.e., $C_{free} = C - C_{obstacle}$.

Motion planning as a problem, takes place in the configuration space and refers to obtaining a trajectory $c(t)$ between the robots current configuration $\mathbf{q_{start}}$ and a given goal configuration denoted as $\mathbf{q_{goal}}$. By calculating the derivatives of $c(t)$ with respect to time, velocities and accelerations can be calculated thus actions can be taken so that the robot follows the trajectory from $\mathbf{q_{start}}$ to $\mathbf{q_{goal}}$ following a curve in the C

space called a path.

Given the above, for a robot to move from $\mathbf{q_{start}}$ to $\mathbf{q_{goal}}$ following a free path, the C space must be expressed and represented explicitly. Although applicable in many cases, in multi-dimensional or even in non – Euclidean spaces the representation of the configuration space can by quite challenging. Thus, there are methods that define the free path incrementally as the robot operates in an online manner. One such method that is used in this thesis is called Potential Functions. Another name that gives more intuition is that of Artificial Potential Fields.

2.5.1 Artificial Potential Fields

A potential function is a differentiable function $U : \mathbb{R}^n \rightarrow \mathbb{R}$ that describes an Artificial Potential Field U [33]. The returning value of the function is scalar that represents the potential energy $U(\mathbf{q})$ at each $\mathbf{q}$ configuration of the robot. The gradient of the potential is a force, the negative of which points at a direction of locally minimal potential energy. By calculating the gradient of the potential function at each point in the configuration space a vector field is generated.

$$\forall \mathbf{q} \in C_{space} \; \exists \; \nabla U(\mathbf{q}) = \left[\frac{\partial U(\mathbf{q})}{\partial \mathbf{q_1}} \quad , \frac{\partial U(\mathbf{q})}{\partial \mathbf{q_2}}, ..., \frac{\partial U(\mathbf{q})}{\partial \mathbf{q_n}} \right] \tag{2.30}$$

The robot, that is in reality a neutral charged rigid body, can be viewed as a positive charged particle in a magnetic field defined by the potential function, where the goal position is negative charged. This way the vectors given by the gradients in each point in the manifold are magnetic forces, that pull the robot towards the goal i.e., the positive robot is attracted by the oppositely charged goal. Given there are obstacles in the workspace of the robot, they are represented as positive charged entities that repel the identically charged robot.

$$U(\mathbf{q}) = U_{attractive}(\mathbf{q}) + U_{repulsive}(\mathbf{q}) \tag{2.31}$$

$$F(\mathbf{q}) = -\nabla U(\mathbf{q}) = -\nabla U_{attractive}(\mathbf{q}) - \nabla U_{repulsive}(\mathbf{q}) = \left[\frac{dU}{dx} \quad , \frac{dU}{dy} \right]^T \tag{2.32}$$

Where $[x, y]$ define the robots coordinates with respect to a world frame. By ignoring the dynamics of such a system, the gradients can be seen as velocity vectors and by defining the potential function in a manner that the goal position has minimum

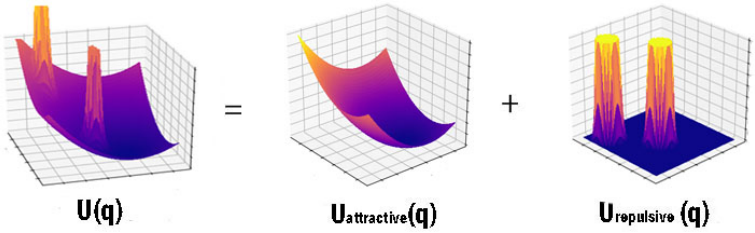


Figure 2.32: Pictorial Representation of Total field given two obstacles.

potential, i.e. $\nabla \mathbf{q_{goal}} = 0$, following the negative velocity at each point the robot will reach its goal. The path is obtained through the simple but effective optimization method of gradient descent.

The potential function is defined as a non-flat function with non-singular Hessian at $\mathbf{q_{goal}}$ so the minimum at goal is isolated. From any given $\mathbf{q_{start}}$ the robot, following the negative gradient, will head towards a path of decreasing potential and will stop at a local minima where $\nabla \mathbf{q} = 0$. If the only minima of the function is the one at the goal, then the robot successfully stops at the desired goal location, where the gradient vanishes. In many situations there are local minima different than the critical point at the q_{goal}, so one fault of the above intuition is the existence of local minima different.

Below are the definitions of the potential functions and the corresponding derivatives describing the vector field used in this thesis.

Attractive potential field and Attractive Force

$$U_{attractive}(\mathbf{q}) = \frac{1}{2} K_{attractive} \left| \mathbf{q} - \mathbf{q_{goal}} \right|^2 \tag{2.33}$$

$$F_{attractive} = -\nabla U_{attractive} = -K_{attractive} \left| \mathbf{q} - \mathbf{q_{goal}} \right| \tag{2.34}$$

where $\mathbf{q}, \mathbf{q_{goal}}$ are the configurations of the robot at the current position and at goal position in the form of vectors, thus $\left| \mathbf{q} - \mathbf{q_{goal}} \right|$ is the distance of the robot measured from the goal. $K_{attractive}$ is the gain of the field which can be seen as the gain of the controller.

$U_{attractive}$ is defined in a way that is continuously differentiable and monotonically increasing as the distance from goal increases. Thus, as the value of the gradient decreases as the goal is reached by the mobile robot.

Repulsive potential field and Repulsive Force

For the k^{th} obstacle the potential is given as

$$U^k_{repulsive}(\mathbf{q}) = \begin{cases} \frac{1}{2}K_{repulsive}\left(\frac{1}{|\mathbf{q}-\mathbf{q_k}|} - \frac{1}{\rho}\right)^2, if|\mathbf{q}-\mathbf{q_k}| < \rho \\ 0, if\ |\mathbf{q}-\mathbf{q_k}| \geq \rho \end{cases} \tag{2.35}$$

where q, q_k correspond to the configurations of the robot at the current position and the configuration of the k^{th} obstacle in the form of vectors, thus $|\mathbf{q}-\mathbf{q_k}|$ is the distance of the robot from the k^{th} obstacle. $K_{repulsive}$ is the gain of the repulsive potential and ρ is a threshold distance that limits and nullifies the effect of the repulsive potential.

The total repulsive potential that corresponds to all the obstacles detected by the robot is given as,

$$U_{repulsive}(\mathbf{q}) = \sum_{k=1}^{N} U^k_{repulsive}(\mathbf{q}) \tag{2.36}$$

The repulsive force that corresponds to the k^{th} obstacle is,

$$F^k_{repulsive}(\mathbf{q}) = -\nabla U^k_{repulsive}(\mathbf{q}) = K_{repulsive}\left(\frac{1}{|\mathbf{q}-\mathbf{q_k}|} - \frac{1}{\rho}\right)\frac{1}{|\mathbf{q}-\mathbf{q_k}|^2}\frac{\mathbf{q}-\mathbf{q_k}}{|\mathbf{q}-\mathbf{q_k}|}, if|\mathbf{q}-\mathbf{q_k}| \leq \rho \tag{2.37}$$

and the resultant repulsive force applied to the robot is given as

$$F_{repulsive}(\mathbf{q}) = \sum_{k=1}^{N} F^k_{repulsive}(\mathbf{q}) \tag{2.38}$$

Thus the resultant force of the total potential filed is given by the resultant of $F_{attractive}(\mathbf{q})$ and $F_{repulsive}(\mathbf{q})$ as:

$$F(\mathbf{q}) = F_{attractive}(\mathbf{q}) + F_{repulsive}(\mathbf{q}) \tag{2.39}$$

By using the resultant force $F(\mathbf{q})$ of the total field as a control law of the motion of the robot, given a goal configuration the robot will follow the gradient descending to the $\mathbf{q_{goal}}$.

The calculation of repulsive potential and its gradient require knowledge of the distance of all obstacles surrounding the robot that can affect it, i.e., $|\mathbf{q}-\mathbf{q_k}| \leq \rho$, via repulsive forces. The required distances can be obtained by using sensory information of range sensors, such as a laser scanner. For each obstacle the laser scanner

detects its minimum distance from the robot, that is the required information for the computation of forces.

Thus, for k obstacles detected, the k local minima of the sensitized distances are the required information for repulsive force computation.

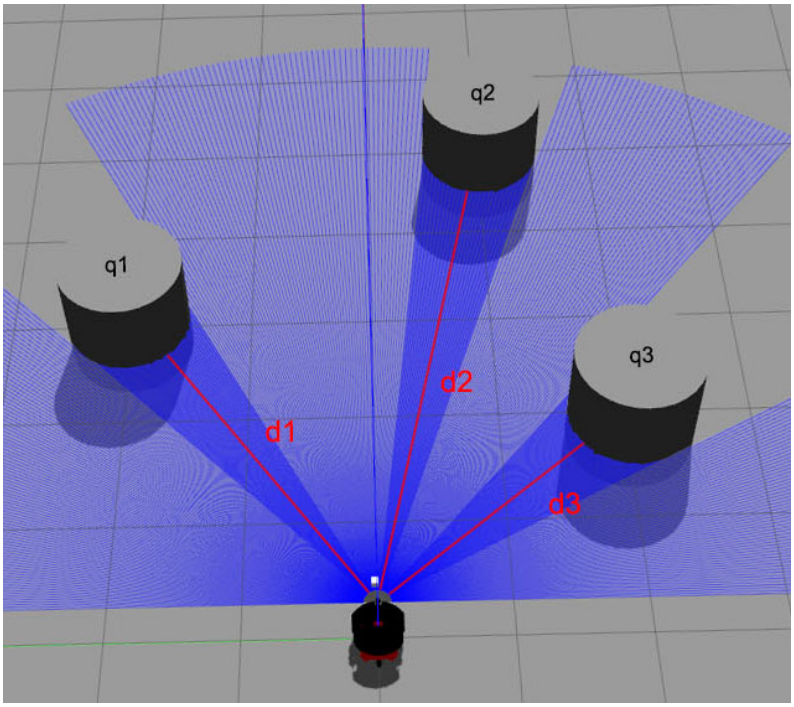


Figure 2.33: Local Minima as minimum distances detected by the laser scanner.

The biggest advantage of this method, as a navigation technique, is that no prior knowledge is required about the robots workspace. Thus, no map of the workspace is required.

Chapter 3

The proposed approach

Object detection traditionally uses images as the data, based on which, the task is performed. Both the localization of the object on the image, that is concluded with the placement of a bounding box around the objects extent, as well as the recognition of the object, that is concluded by labeling the contents of the bounding box, the combination of which follows through the task of object detection, are achieved using only the image of a camera sensor as input.

Instead of using a single sensory modality, the proposed method fuses two sensors, a range sensor, i.e., a 2D laser scanner and a camera, which not only follows thought the task of object detection but enhances the label with range information.

An important note is that every joint of a robot as well as the attached sensors, all have a coordinate system attached to them, referred also as the frame of the joint or the corresponding sensor. Every point of interest in the 3D world can be referred to any of the coordinate systems. Coordinate systems of the robot joints are always in 3D, and are right-handed, with X forward, Y left, and Z up. Between the different coordinate systems exist homogeneous transformations, so that the coordinates of a point with known coordinates referred to one frame can be obtained for every other frame.

3.1 Method Explanatory Analysis

The method developed in this thesis performs the task of object detection by solving the two embedded sub problems in the detection task, i.e., object localization and object recognition, using two distinctly heterogeneous sensors, in two different spaces by fusing the sensory data in serial manner.

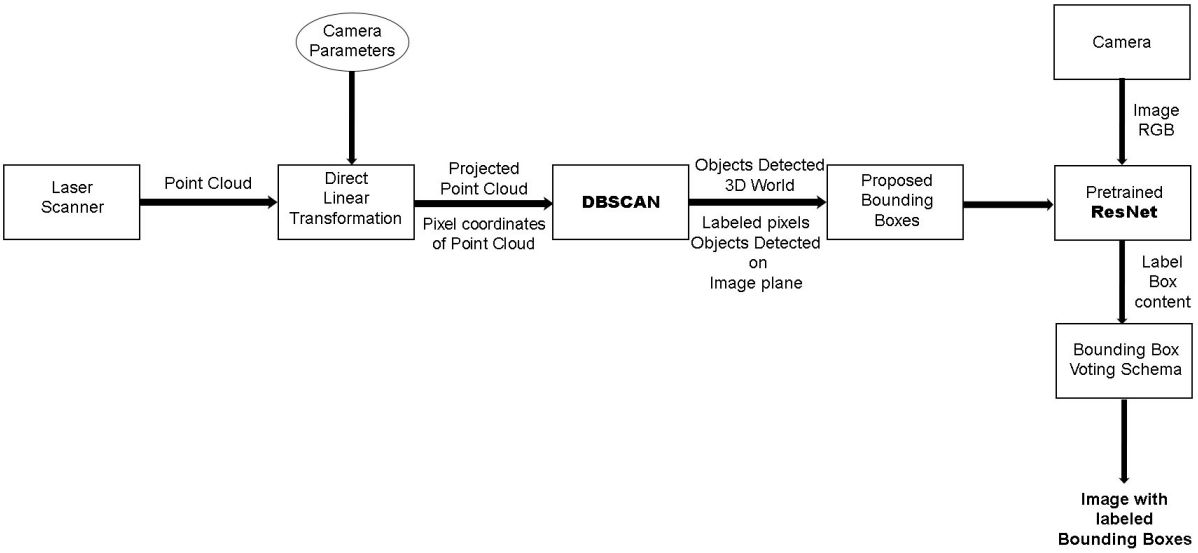
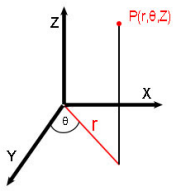


Figure 3.1: Block Diagram of the Proposed Detection Method

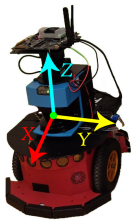
The laser scanner is the sensor that achieves the object localization task, initially in the 3D world surrounding the robot. The located object is projected using the camera parameters to the 2D image plane of the camera sensor. With knowledge of the location of the object on the image plane, bounding boxes are proposed, so recognition is achieved for the image area bounded by the box and corresponding labels are obtained using a ConvNet and a voting schema.

3.1.1 Object Localization

The sensor that is mainly responsible for the localization task is the 2D laser scanner, that gives the location of an object in the 3D world and with auxiliary usage of the camera parameters the location is projected on the camera image.



(a) Cylindrical Coordinates of point P.



(b) Laser frame.

Figure 3.2: Laser Coordinate System.

The data that the laser scanner initially provides is given as corresponding Euclidean distances and angles of the vectors between the laser frame and the points on the object surfaces that the laser beams are reflected from, i.e., (r, θ). That data is transformed into the point cloud data set $[X, Y, Z]$ that corresponds to a 2D representation of the environment surrounding the robot. That is a Cylindrical to Cartesian coordinate system transformation, that for a point $\mathbf{P} = (r, \theta)$ is given as:

$$X = r\cos\theta \tag{3.1}$$

$$Y = r\sin\theta \tag{3.2}$$

$$Z = Z_c \tag{3.3}$$

The coordinate Z_c in the Z-axis is a constant scalar equal to the height of the laser sensor measured from the ground, in the case where the hyperplane that the X,Y base vectors span is parallel to the ground, as in the case of this thesis. Thus, the point cloud is represented as a data set denoted as PCL, where $PCL =< \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\} >$, and it is made more informative with the addition of the Euclidean distance parameter of each point referenced from the laser frame. So depending on the sampling frequency of the sensor, a data set is formed as $PCL =< \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{r}\} >$ containing all the points detected by the laser scanner.

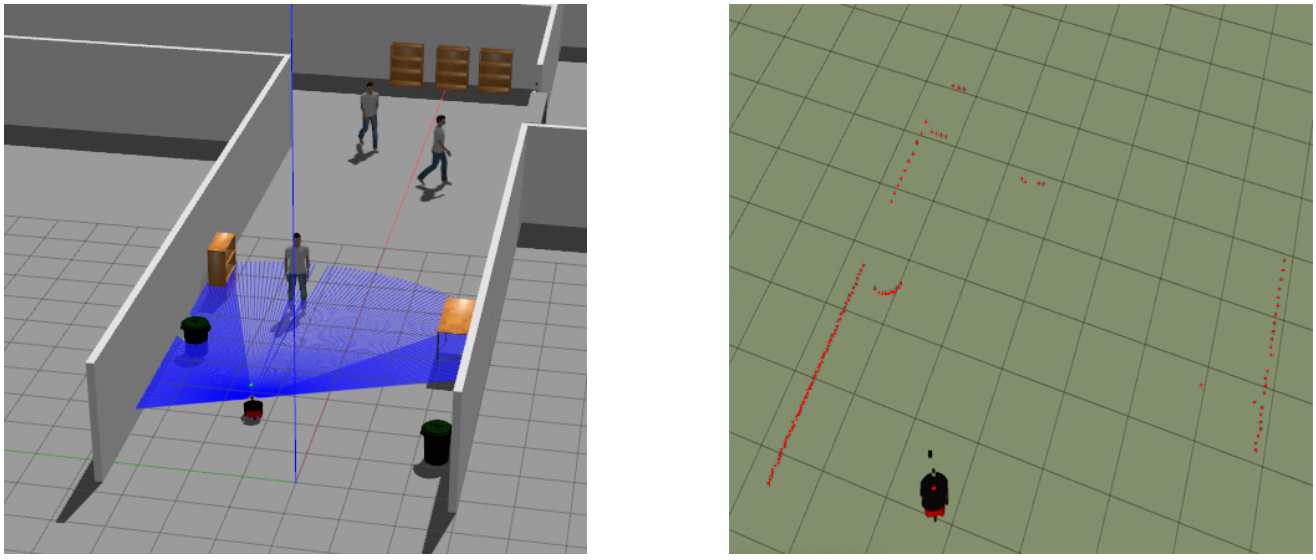
By plotting the distance r of each point against its corresponding Y-coordinate with respect to the laser frame, an approximation of the layout of the surrounding detected environment is obtained.

The state of the perceived environment given in Figure 3.3 is depicted in Figure 3.4.

With the knowledge of the intrinsic and extrinsic parameters of the camera that is attached to the robot, each point of the point cloud can by projected on the digital

Table 3.1: Enhanced Point cloud data set with point distance information.

X	**Y**	**Z**	**Distance**
X_1	Y_1	Z_1	r_1
X_2	Y_2	Z_2	r_2
X_3	Y_3	Z_3	r_3
...	...	...	...
X_N	Y_N	Z_N	r_N



(a) Gazebo Corridor Simulation

(b) RViz Point cloud representation

Figure 3.3: Point Cloud angular perspective in simulation.

image of the camera. That projection transfers the point cloud from the 3D world to the 2D image plane.

The extrinsic parameters are obtained from the pose of the camera with respect to the laser frame. It is generally given as a translation and a rotation as described in 2.1.2 but in the case of this thesis only a translation is performed given the relative poses of the laser and camera sensors. A transformation is performed between the camera and laser data frame, so that the $[X, Y, Z]$ of each point now refers to the camera frame, denoted as $[{}^cX, {}^cY, {}^cZ]$.

$$\begin{bmatrix} {}^cX \\ {}^cY \\ {}^cZ \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \mathbf{T}^T$$

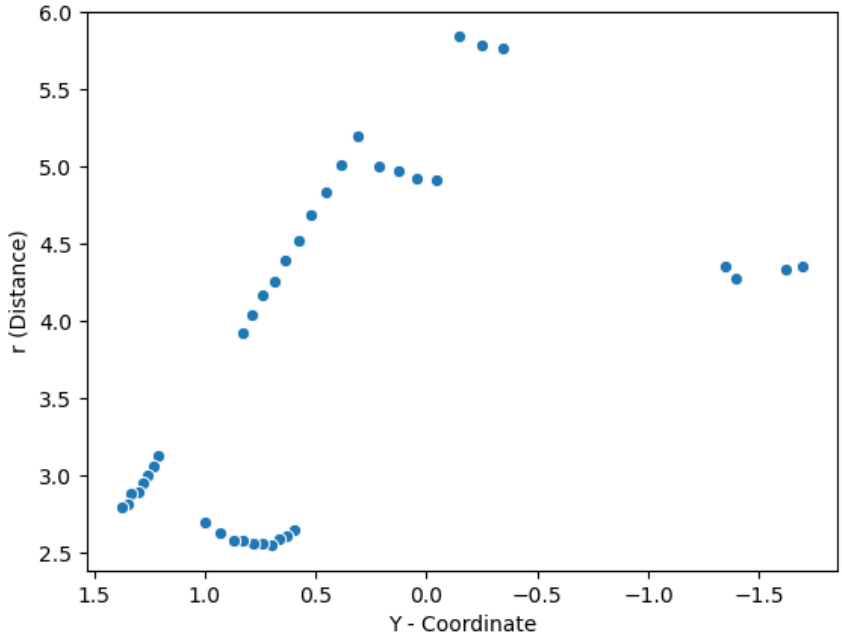


Figure 3.4: Point cloud layout

with denoting the translation vector.

The intrinsic parameters of the camera are obtained via calibration of the camera using the Zhang's method explained in 2.1.2. By use of the computations described in Chapter 2 or simply using the computations described in equations (3.4),(3.5) for each translated point, the corresponding pixel coordinate ${}^{p}x,{}^{p}y$ on the 2D image plane is obtained for the point cloud as:

$$ {}^{p}x = width_{image} - (\frac{{}^{c}Y * f_x}{X} + c_x) \tag{3.4} $$

$$ {}^{p}y = height_{image} - (\frac{{}^{c}Z * f_y}{X} + c_y) \tag{3.5} $$

By performing the above, the PCL data set is now in the form of the Table 3.2:

The $< \{{}^{c}\mathbf{Y}, {}^{c}\mathbf{r}\} >$, i.e, the data corresponding to Y-coordinate and distance of the $PLC =< \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{r}\} >$ data set with respect to the camera frame, are passed to the density based clustering algorithm called DBSCAN. Note that the rate that the data are passed to the DBSCAN algorithm is defined by the sampling rate of the laser scanner. DBSCAN also serves as a filter of the laser sensor by labeling noisy measurement points as outliers. Pictorially, the application of DBSCAN on the point cloud that corresponds to the word state represented in Figure 3.3, when passed the $< \{{}^{c}\mathbf{Y}, {}^{c}\mathbf{r}\} >$ data, is given in the Figure 3.5 the follows.

Table 3.2: Data set of projected Point Cloud - Point, pixel correspondence.

${}^{c}X$	${}^{c}Y$	${}^{c}Z$	**Distance**	${}^{p}x$	${}^{p}y$
${}^{c}X_1$	${}^{c}Y_1$	${}^{c}Z_1$	${}^{c}r_1$	${}^{p}x_1$	${}^{p}y_1$
${}^{c}X_2$	${}^{c}Y_2$	${}^{c}Z_2$	${}^{c}r_2$	${}^{p}x_2$	${}^{p}y_2$
${}^{c}X_3$	${}^{c}Y_3$	${}^{c}Z_3$	${}^{c}r_3$	${}^{p}x_3$	${}^{p}y_3$
...	...	...	...	...	...
${}^{c}X_N$	${}^{c}Y_N$	${}^{c}Z_N$	${}^{c}r_N$	${}^{p}x_N$	${}^{p}y_N$

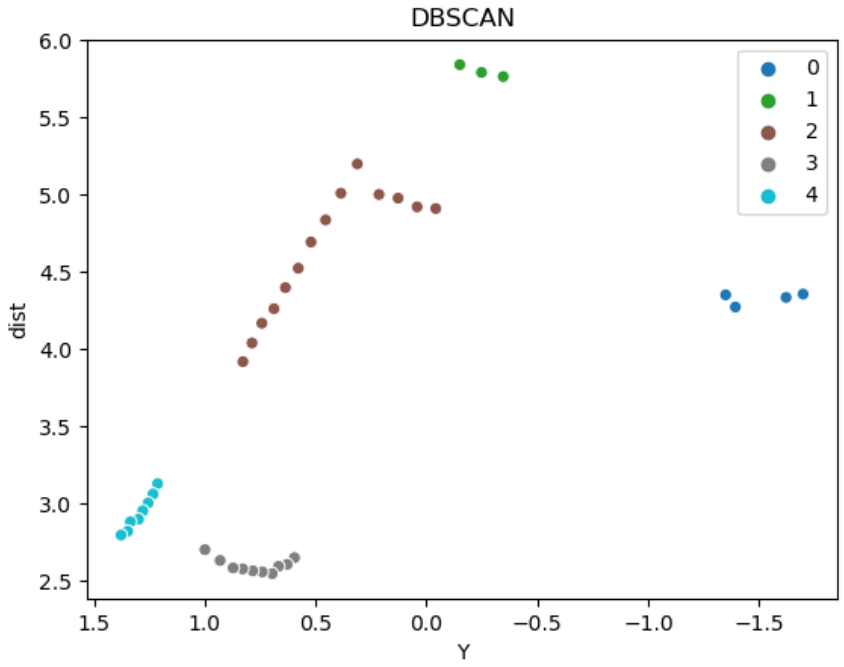


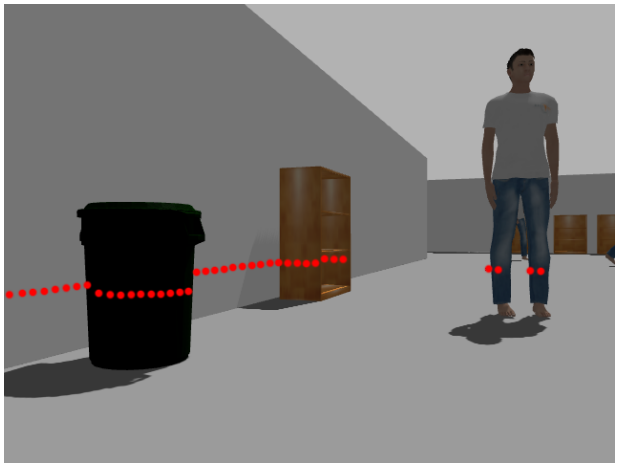
Figure 3.5: Clustered Point cloud - Object Cluster Correspondence

Each of the clusters represents an object, thus by clustering the $< \{{}^{c}\mathbf{Y}, {}^{c}\mathbf{r}\} >$ data, each point of the data set is assigned a label that corresponds to a distinct object in the robots environment. Points that belong to the same cluster provide a 2D model layout of an object in the robot's detected workspace. As seen in the table 3.3 the pixel coordinates are now labeled according to the objects that they belong.

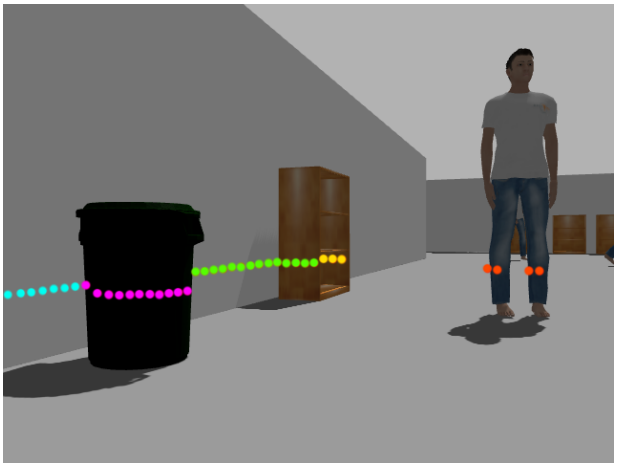
By clustering the point cloud using $< \{{}^{c}\mathbf{Y}, {}^{c}\mathbf{r}\} >$ data , object localization is performed in the 3D world surrounding the robot. With the projection of objects on the 2D image plane, the corresponding pixel coordinates of all objects detected in the 3D world are known. Essentially what happens is that by performing DBSCAN in the initial $[X, Y, Z]$ point cloud information, labeled pixels corresponding to objects are

Table 3.3: Clustered Point cloud data set for K objects surrounding the robot.

${}^{c}X$	${}^{c}Y$	${}^{c}Z$	**Distance**	${}^{p}x$	${}^{p}y$	**Label/Object**
${}^{c}X_1$	${}^{c}Y_1$	${}^{c}Z_1$	${}^{c}r_1$	${}^{p}x_1$	${}^{p}y_1$	0
${}^{c}X_2$	${}^{c}Y_2$	${}^{c}Z_2$	${}^{c}r_2$	${}^{p}x_2$	${}^{p}y_2$	0
${}^{c}X_3$	${}^{c}Y_3$	${}^{c}Z_3$	${}^{c}r_3$	${}^{p}x_3$	${}^{p}y_3$	0
${}^{c}X_4$	${}^{c}Y_4$	${}^{c}Z_4$	${}^{c}r_4$	${}^{p}x_4$	${}^{p}y_4$	1
${}^{c}X_5$	${}^{c}Y_5$	${}^{c}Z_5$	${}^{c}r_5$	${}^{p}x_5$	${}^{p}y_5$	1
...	...	...	...	...	...	...
${}^{c}X_N$	${}^{c}Y_N$	${}^{c}Z_N$	${}^{c}r_N$	${}^{p}x_N$	${}^{p}y_N$	K



(a) Projection of point cloud on Image



(b) Clustering of the projected point cloud - Labeled Pixels

Figure 3.6: Object Localization from 3D world to 2D Image plane.

obtained. This way the object localization in the image plain is performed in the 3D world surrounding the robot.Note that distance information is also available.

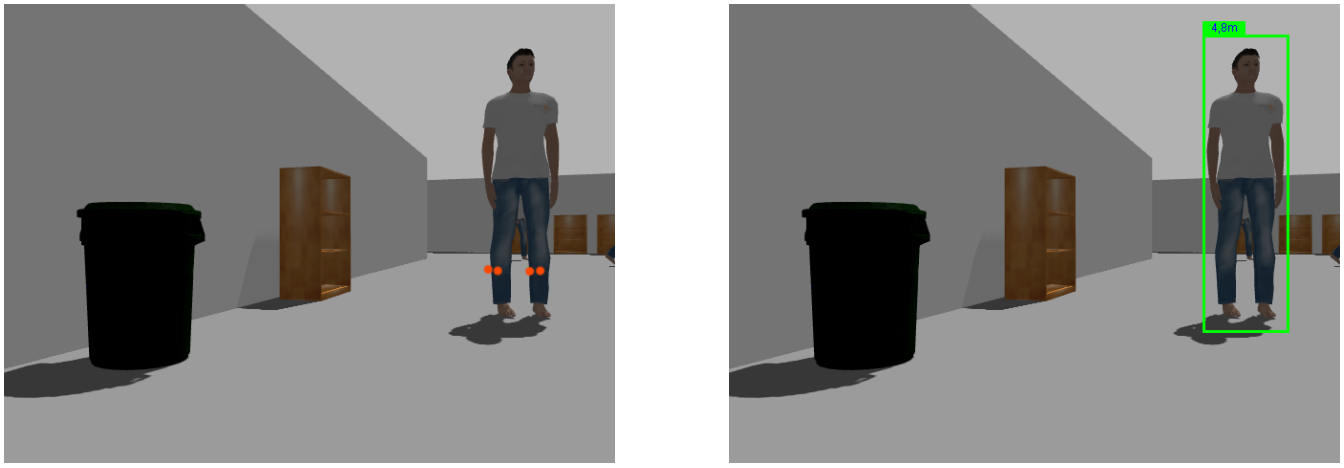
With knowledge of the location and the width of the objects in pixels on the image space a bounding box can be placed with a rough approximation of the height, which due to the 2D laser scanner is not an available dimension. A good case of box placement can be seen on the Figure 3.7

A choice made for the proposed implementation is to focus only on the object that is closest to the robot. So from now on the analysis will be with respect to the cluster with the closest distance.

The absence of the objects height information, thus the exact location of the bound-

Table 3.4: Clustered Point cloud data set for K objects surrounding the robot.

${}^{p}x$	${}^{p}y$	**Label/Object**	**Distance**
${}^{p}x_1$	${}^{p}y_1$	0	r_1
${}^{p}x_2$	${}^{p}y_2$	0	r_2
${}^{p}x_3$	${}^{p}y_3$	0	r_3
${}^{p}x_4$	${}^{p}y_4$	1	r_4
${}^{p}x_5$	${}^{p}y_5$	1	r_5
...	...	...	...
${}^{p}x_N$	${}^{p}y_N$	K	r_N



(a) Isolated cluster corresponding to detected object

(b) Bounding Box with distance information

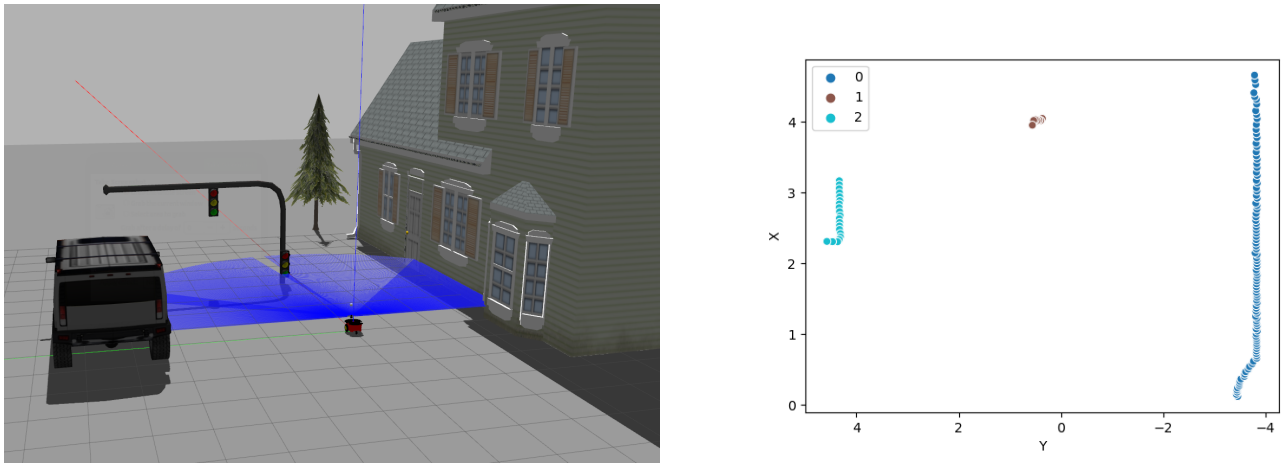
Figure 3.7: Object Localization.

ing box is solved using an approximation with information coming from the CNN that performs the task of recognition. The intuition of the above as well as the object recognition solution is further discussed in the subsection that follows.

3.1.2 Object Recognition

By clustering the point cloud and projecting it to the image plane using the Direct Liner Transformation the location of the objects on the image plane is obtained. The known dimension of the object is that of the width as a result of the 2D laser scanner. In order to make an assumption about the objects height, a Convolutional Neural Network is used that in addition to the recognition task, approximates intuitively the

missing height dimension.



(a) (b)

Figure 3.8: Simulation world - Point cloud clustering.

More specifically two voting methods have been used, the details each one of which follows. Both methods start with the same approach. By focusing on the closest object ,i.e., cluster, to the robot and using the width in pixels known from the localization task, three bounding boxes are drawn around the objects extent, with increasing height dimension as seen in the Figure 3.9.

The imagery content of each box is isolated from the rest of the image by cropping the image in the boxes perimeter. Thus, three images are created, one for each corresponding box.

The three images are resized in the specific size, that the CNN used, is trained on. The CNN that is chosen is the ResNet18, trained on the ImageNet data set, for reasons that will be explained in the Section 3.2.

Cumulative Box voting schema

The three resized images, that depict the boxes imagery content, are forward passed to the pretrained ResNet18. For each one of the three, 1000 class probabilities are inferred. The top 5 class probabilities of each image, i.e., the 5 that the network is more confident about, are the ones that the proposed method uses via the following voting schema. In the following the three boxes and three images are used interchangeably. Essentially, each one of the three boxes proposes 5 labels. A map data structure is used with the labels as keys, as in Tables 3.5, 3.6. As value for each label/key, a score

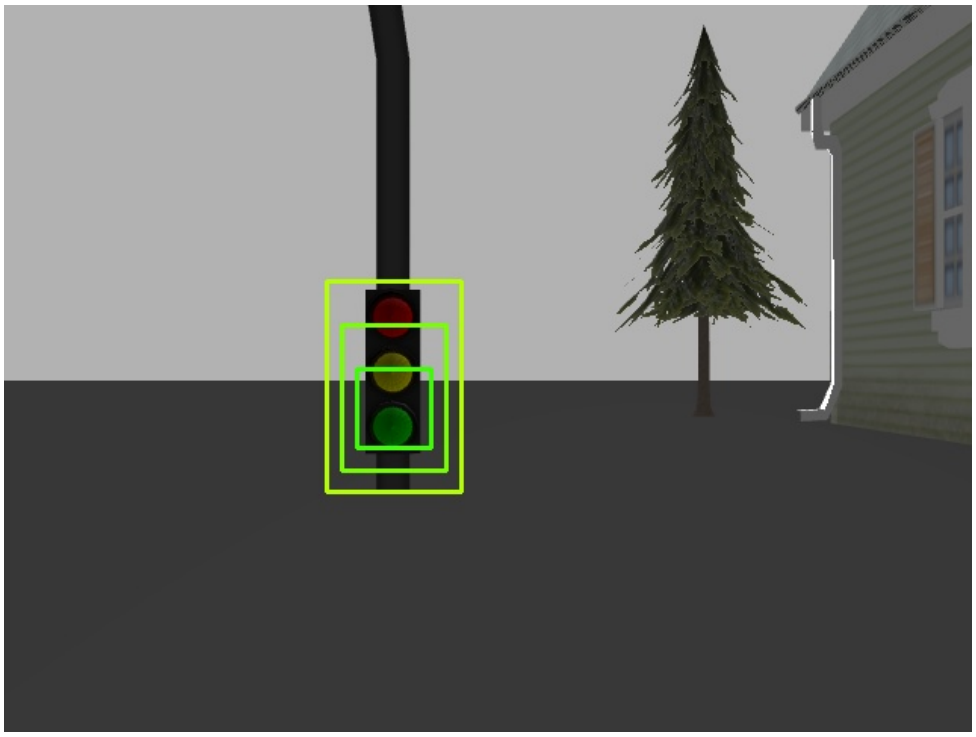
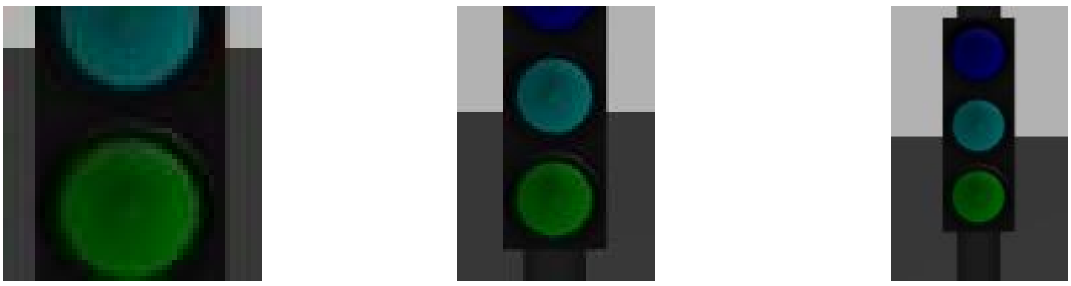


Figure 3.9: Image with 3 bounding boxes



(a) (b) (c)

Figure 3.10: The three cropped images used for inference.

is used in the form of a scalar, computed as the total sum of the corresponding label's probabilities among the three boxes. The label with the highest value/score is chosen as the label for the detected object. The box that proposed the chosen label with the highest probability is the chosen box, that is drawn around the objects extent. That way, the missing height information is approximated through voting of feasible boxes.

Simple voting schema

A simpler voting schema is used with similar effectiveness that avoids the usage of the map data structure making the final inference a bit faster. In this schema, once

Table 3.5: Example of Cumulative Voting schema

(a) 1^{st} Box Top5 Inferred Labels

Label	% Probability
traffic light	64.87
loudspeaker	23.44
oscilloscope	1.94
digital clock	0.96
cassette player	0.74

(b) 2^{nd} Box Top5 Inferred Labels

Label	% Probability
loudspeaker	72.32
traffic light	5.04
switch	4.05
face powder	2.13
lipstick	1.99

(c) 3^{rd} Box Top5 Inferred Labels

Label	% Probability
traffic light	71.67
loudspeaker	13.30
spotlight	1.57
switch	1.14
knee pad	0.77

Table 3.6: Cumulative scores of proposed labels.

Label	**Score**
traffic light	141.58
loudspeaker	109.06
switch	5.19
face powder	2.13
lipstick	1.99
oscilloscope	1.94
spotlight	1.57
digital clock	0.96
knee pad	0.77
cassette player	0.74

again the three resized images, cropped on the boxes perimeter, are forward passed to the pretrained ResNet18. Each one of the three, proposes as label the one with the highest probability among the 1000. Of the three labels, the one with the highest

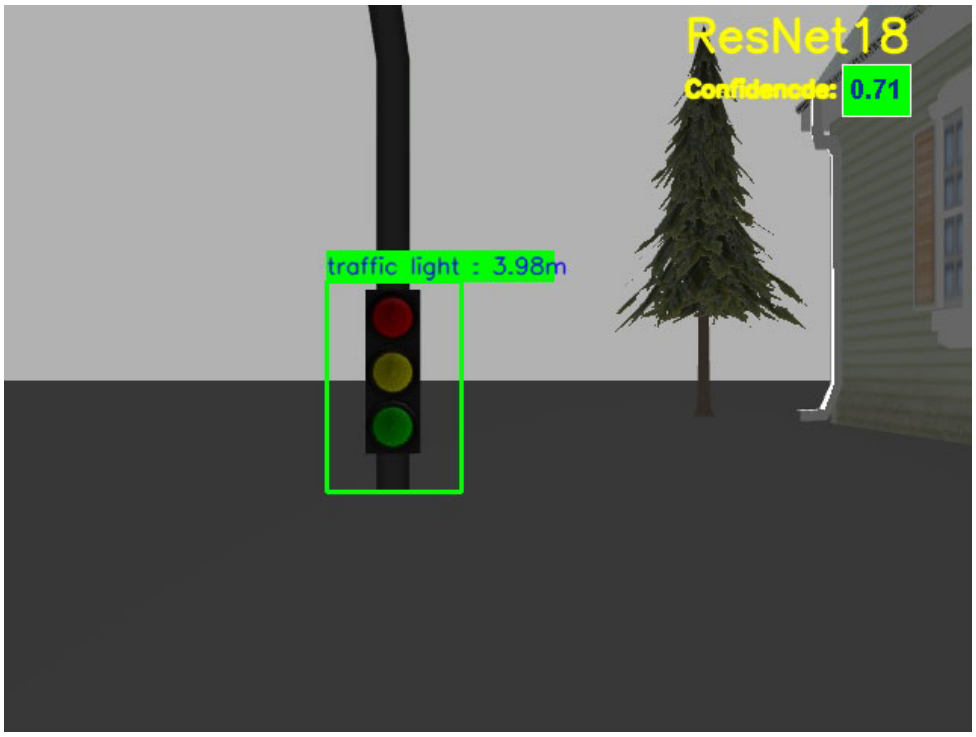


Figure 3.11: Proposed Bounding Box and Label - Cumulative Voting Schema

probability is chosen as the correct label together with the corresponding box, that is in the end drawn around the objects extend.

3.1.3 Autonomous Navigation

As a proof of concept for the proposed detection system, motion planning is achieved using odometry data, i.e., data from motion sensors that are used to estimate the robot's pose over time. The proposed approach to motion planning achieves autonomous navigation without prior knowledge of the robot's workspace and is based mainly on the Artificial Potential Fields method. The robot is given a goal position and orientation and navigates through the unknown environment by updating a free path incrementally until the goal pose is reached, by the use of velocity commands as a control law. While the robot navigates, if the proposed perception system detects an object of interest that is predefined by the application, the robot stops and interacts with the object in the simplest form by capturing a photo.

Together with the Potential Fields, a simple P-controller is designed, the velocity commands of which are used when a certain distance threshold is exceeded.The distance threshold is a hyperparamater that represents the closed Euclidean distance

an obstacle can be detected from the robot, so that a swapping between the two controllers,i.e, P-controler and Potential Fields can occur. By defining it as $D_{threshold}$ the swapping between Potential Fields and P-control is described as below:

$$\textbf{Velocity Controller} = \begin{cases} Potential\ Functions,\ \ if\ D_{threshold} \leq d \\ P - Controler\ \ if\ D_{threshold} > d \end{cases}$$

where d denotes the robot's distance from the closest obstacle.

The swapping between the Potential Functions and the P-Controller is a choice made so that the navigation becomes more adaptive and responsive depending on the obstacles configuration. The gains of both the Potential Functions and the P-Controller are obtained through trail and chosen fitting the application.

As derived from the Potential functions equations, the attractive potential acts as a P-Controller in the case when no obstacle is in the near distance. The existence of a single gain, i.e., that of the attractive field, so that the robot behaves smoothly both in the case of only an attractive force and also in the case of a resultant from attractive and repulsive forces, creates some inconvenience to the velocity commands, that affect the smooth and continuous movement of the robot.

More specifically, the gain of the attractive potential when acting by itself, tends to be such, that the velocity commands reduce to zero as the robot approaches the goal configuration prior to reaching the goal position. To solve that, a swap to the simple P-Controller with a higher gain than the attractive field's, is done, so that the robot has a faster response. The use of the higher gain P-Controller also makes the robot's motion control more responsive throughout the navigation.

3.1.4 Potential Functions

The approach used, follows exactly the functions and intuition described in ??. The difference in the developed controller is that the aperture angle of the laser scanner that is responsible for detecting the closest distance from any obstacle, is divided in to three discrete sectors as seen in Figure 3.12 . For each sector, the closest detected distance is used as the parameter in the repulsive force equation ??.

Thus, at maximum, three repulsive forces are acting on the robot originating from the three obstacles detected, one from each sector. The potential fields method is activated as the control law that gives the velocity commands, when the configuration of at least one obstacle falls within the red denoted circular area with the green

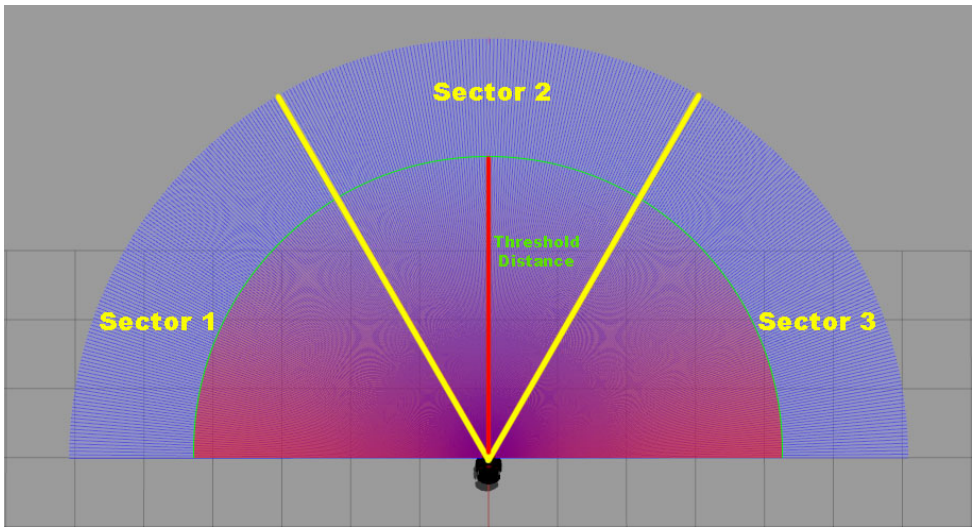


Figure 3.12: Point cloud layout

circumference in Figure 3.12. The velocity commands are given with respect to the resultant force that acts on the robot from the potential field over time. The magnitude of the robot's linear velocity is proportional to the magnitude of the resultant force and the angular velocity tries to minimize the error between the current yaw,given by the odometry, and the desired direction given by the resultant force vector angle at each point. The robots velocity $\mathbf{u} = [u_x, u_y]^T$ is given as:

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \mathbf{u} = -k\nabla U(\mathbf{q}) = -k \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \\ \frac{\partial U}{\partial \theta} \end{bmatrix} \tag{3.6}$$

3.1.5 P-Controller

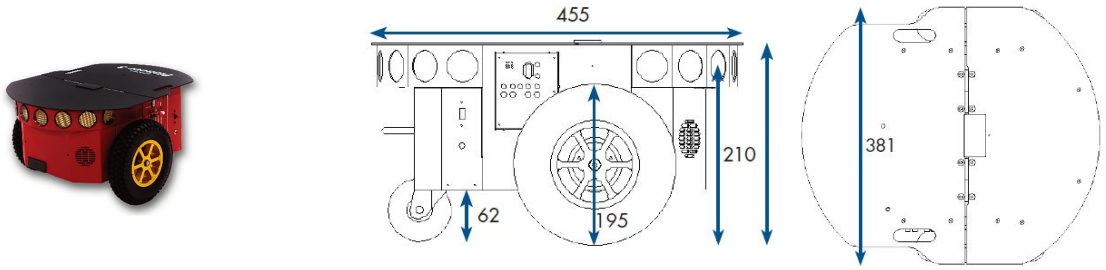
When no obstacles fall in the threshold area, the robot moves toward the goal configuration using a P-controller. Specifically, the velocity commands are given by the controller in a way resulting to a linear velocity with a magnitude proportional to the distance of the robot from the given goal and a angular velocity that is proportional to the difference of the current and desired orientation, i.e., the yaw. Both the distance from the goal configuration as well as the yaw of the robot is given by the odometry.

3.2 Implementation Details

The proposed detection system was implemented using ROS and tested in simulation using the Gazebo simulator, as well as using the Pionner 3-DX robot in the laboratory and other dynamic environments.

3.2.1 Pioneer 3-DX

The Pioneer 3-DX is a two-wheel two-motor differential drive robot created by Adept Technology Inc. The vanilla version of the robot, prior to any modifications or attachments can be seen in the Figure 3.13 below.



(a) (b) (c)

Figure 3.13: The Pioneer 3-DX vanilla version.

As a differential drive robot it has three degrees of freedom (DOF) and with the usage of actuators on the two wheels only the two DOF are controllable making the robot non-holonomic. Assuming knowledge of the left and right wheel velocities V_l, V_r with respect to the ground and the Instantaneous Center of Curvature (ICC) the kinematics of the 3-DX robot is described from the kinematics of differential drive robots.

The linear velocity v and the angular velocity ω of the robot is calculated as:

$$v = \frac{V_l + V_r}{2} \tag{3.7}$$

$$\omega = \frac{V_l - V_r}{W} \tag{3.8}$$

where W denotes the distance between the robot's wheels.

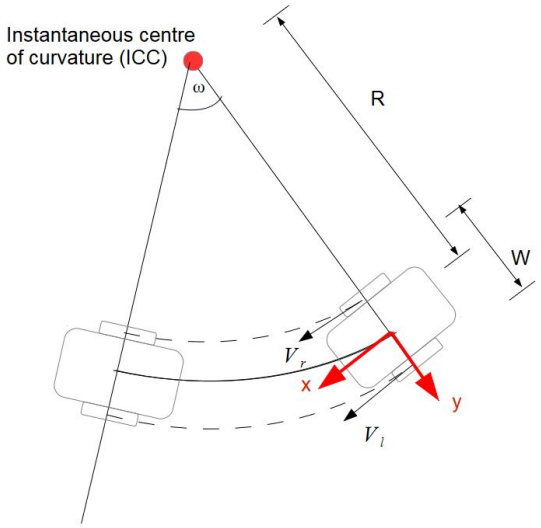


Figure 3.14: Differential Drive Kinematics.

The distance between the robot's base frame and the ICC, denote as R, is calculated as:

$$R = \frac{W}{2}\frac{V_l + V_r}{V_l - V_r} \tag{3.9}$$

Having knowledge of R and the robot's current pose $\mathbf{q} = [x, y, \theta]^T$ the ICC is calculated as:

$$ICC = \begin{bmatrix} x - R\sin\theta \\ y + R\cos\theta \end{bmatrix} \tag{3.10}$$

Assuming that the robot has moved with constant wheel velocities for Δt the new pose is given as $\mathbf{q}' = [x', y', \theta']^T$:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos\omega\Delta t & -\sin\omega\Delta t & 0 \\ \sin\omega\Delta t & \cos\omega\Delta t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\Delta t \end{bmatrix} \tag{3.11}$$

Figure 3.15: NVIDIA Jetson TX2 kit

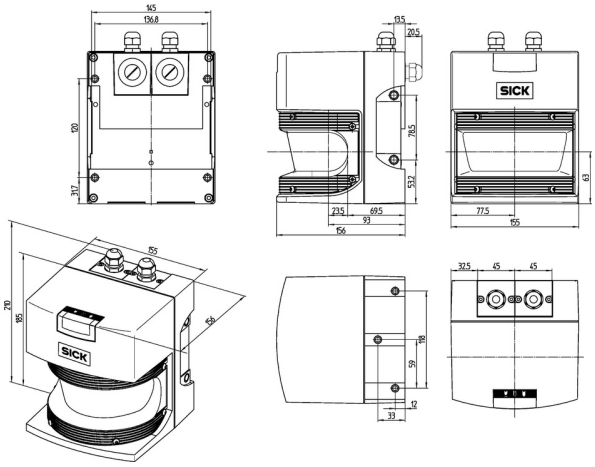
The 3-DX robot used in this thesis has been modified with the integration of the NVIDIA Jetson TX2 Module that acts as the computational unit of the robot, see

Figure 3.15. TX2 development kit is built around an NVIDIA Pascal-family GPU with 256 NVIDIA CUDA cores and is loaded with 8 GB of memory and 59.7 GB/s of memory bandwidth.

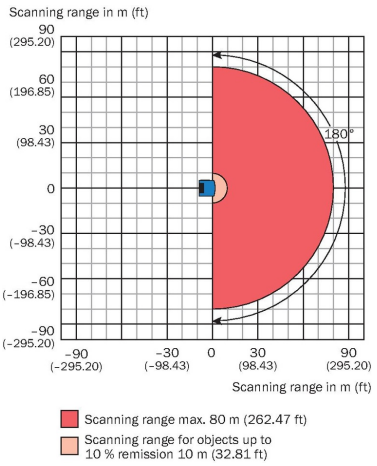
The module has an CMOS camera attached to it that is used as the main camera of the application. More specifically, it is a CMOS 5 megapixel (2592 x 1944) image sensor with OmniBSI-2TM technology that supports images sizes of 5 Mpixel, EIS1080p, 1080p, 720p, VGA, QVGA and has a maximum transfer rate of 30 fps for 5 Mpixel, EIS1080p, 1080p sizes. The scan mode is progressive with lens size of 1/4" and non-linear lens chief ray angle of 29.7°. The image area is 3673.6 µm x 2738.4 µm with pixel size of 1.4 µm x 1.4 µm.

Figure 3.16: The LMS200 laser range scanner

The robot is also integrated with a 2D laser scanner sensor, specifically the LMS 200 manufactured by SICK AG , see Figure 3.16. It is a short range, i.e., Max. range with 10 % reflectivity at 10 m, infrared(905 nm), 2D laser scanner with 180° aperture angle and adjustable angular resolution of 0.25°, 0.5° and 1° at 75 Hz. The angular resolution used in this thesis is that of 0.5°.



(a) Dimensional Drawing



(b) Operating Range

Figure 3.17: LMS200 Details.

3.2.2 Sensor Frames Transformations and Synchronization of Sensory Data

The Pioneer 3-DX after the modifications and attachments of the TX2 module and the LMS200 range scanner can be seen in the Figure 3.18 bellow. The laser frame and camera frame relative position is also denoted.

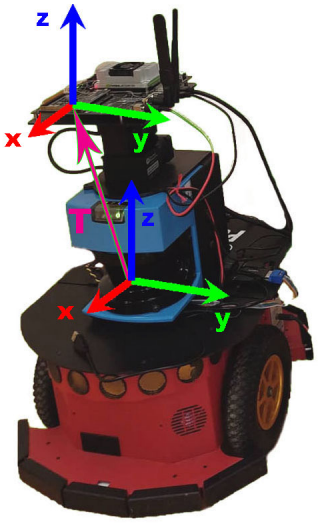


Figure 3.18: Pioneer 3-DX with Jetson TX2 and LMS200

A transformation between the two frames is applicable via a translation that is described by the vector $\mathbf{T} = [0, 0.015, 0.591]^T$. The value zero in the first index corresponds to a really small translation in the x axis that is omitted.

Using that translation every point P of the point cloud can be transformed to the coordinated system of the camera.

On key factor for precise detection using fusion of the camera and laser data is that of data synchronization. The data that are fused must represent the exact same state of the surrounding world. Given the fact that the two sensors have different sampling frequencies and the starting time of their operation differs, the synchronization is achieved with the use of timestamps provided by the ROS message representation of the sensory data. That way it is ensured that the fused data correspond to the same state.

3.2.3 Camera Calibration

The calibration of the camera was done using the camera_calibration ROS package of the image_pipeline. The package uses the OpenCV camera calibration API and a

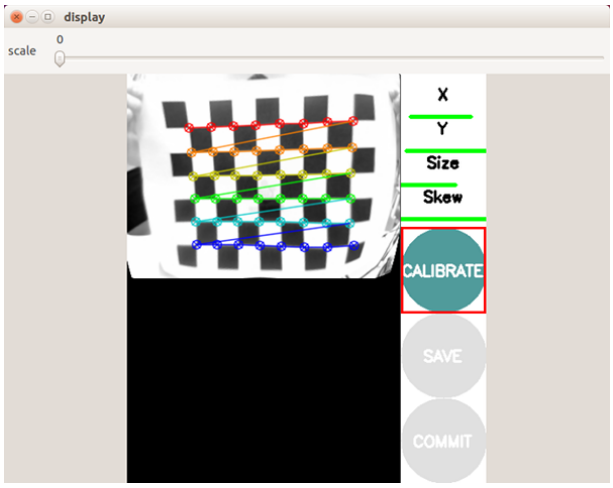


Figure 3.19: Intrinsic Parameters - Camera Calibration

planar checkerboard.

The checkerboard used was a A4 with 25mm squares, 8x6 vertices, 9x7 squares. By positioning the checkerboard in some specific configurations captured by the camera sensor, the camera_calibration package, produces the intrinsic parameters of the camera that are used in the projection of the point cloud on the image plain. The extrinsic parameters are known from the vector **T** that describes the translation between the laser and the camera frame.

3.2.4 DBSCAN

The pointcloud data is published with a frequency of 30Hz using a ROS service that transforms the laser sensor data from the (r, θ) Cylindrical coordinates to that of $[X, Y, Z]$ Cartesian coordinates. The pointcloud is further transformed so that it corresponds to the camera frame.

The DBSCAN algorithm that is used to cluster the Point Cloud in the one implemented by the sci-kit learn toolkit. Compered to the original DBSCAN implementation that is discussed in sector 2.2.1, the sci-kit learn provided implementation is using a variation but still in the concept of the same theoretical cluster model. It first materializes all neighborhoods (which yields worst-case quadratic memory), then performs the cluster expansion in a "vectorized" way on the core points only.

The choice of ϵ and minPTS, corresponding to eps and $min\_samples$ hyperparameters is dove via trail and error for the specific application. These two hyperparameters are of great importance to the overall reliability of the detection system and

are chosen appropriately for the specific application, considering that the point cloud shape depends greatly on the robot's environment. So based on the data of the point cloud and the distance metric that is used, which is the Euclidean, the ϵ and minPTS hyperparameters were chosen, fitting the application.

One of the main reasons DBSCAN algorithm was chosen over other clustering algorithms for this specific application is that it does not require prior knowledge of the number of clusters in the point cloud. In addition to that, DBSCAN has good clustering capabilities on data sets with arbitrary cluster shapes and its speed of inference matches the desired speed of the detection system.

3.2.5 ResNet18 and Recognition

The ResNet18 is the chosen CNN for the task of object detection. It is pretrained on the ImageNet dataset, thus capable of recognizing 1000 classes. The choice of the ResNet18, instead of other architectures and pretrained CNNs on the same data set, was based a balance between speed of inference and accuracy in recognizing specific classes of interest, such as doors. Deeper networks have better accuracy but the speed of inference does not match the applications requirements.

The chosen framework that is used for the task of recognition is the PyTroch open source machine learning framework. Torchvision, a library that is part of the PyTorch project provides model architectures, including the ResNet18 and common image transformations for computer vision. Such image transformation are performed on the cropped images that correspond to the bounding boxes imagery content.

After the projection of the closest cluster on the image plane, the known dimension of the object is the width as discussed in section 3.1.4. The height is proposed with arbitrary dimensionality and chosen as a side product of the ResNet18 inference. The cropped image is transformed the same way the ImageNet data set images are transformed prior to the ResNet training. More specifically the three images are resized as 256 x 256 images, transformed to torch tensors and normalized with $mean = [0.485, 0.456, 0.406]$ and $std = [0.229, 0.224, 0.225]$. After the transformation the three images are stacked into a batch which is transferred to the GPU. The pretrained ResNet model is also transferred to the GPU, thus the inference can be done using CUDA operations further enhancing the speed and making use of the Jetson TX2 computational power.

Chapter 4

Experimental Results

The proposed perception system was tested thoroughly both in simulation and in real dynamic environments. The behavior of the robot was almost identical in the simulation and the real world scenarios making the simulation a great debugging and controlled testing environment with applicable optimization in the real workspace. The implementation of the proposed system is done using ROS and is tested in Gazebo simulation scenarios as well as in the laboratory environment and the faculty's corridors and parking place.

The main difference between the simulation and the real world application comes from the fact that the interface for the Pioneer 3-DX robot, in the case of the real world application, comes from the RosAria node which is implemented on the open source ARIA library. Information from the robot base, main joint transformations, and velocity and acceleration control, is implemented via the RosAria node. The node publishes ROS topics providing data received from the robot's embedded controller by ARIA and sets desired commands originating from other nodes [34]. The ROS interface for the laser and camera sensors also comes from corresponding nodes that act as drivers and provide the topics that the sensory data is available and updated based on the sampling frequency of each sensor.

In the Gazebo simulation environment, the Pioneer 3-DX robot is modeled using URDF files that describe the exact robot used in this thesis. The URDF files used, were initially created by Rafael Berkvens and modified by Mario Serna Hernández. The URDF files where further modified in order to be adapted to the specific robot used in this thesis which has been upgraded with the addition of the Jetson TX2. Further more a simulated enviroment has been build specificly for the prosposed application using Gazebo in order for testing to applicable.

Thus, in the case of the real world application the joint transformations are described internally by the RosAria node while in the Gazebo simulation case, the transformations are explicitly described using the URDF files, resulting to a more complex frame tree, although it is referring to the exact same robot with the same behavior. The comparison of the two can be seen in the Figures 4.1, 4.2 below. Note that in the real world application, the camera frame transform with respect to the laser is taken care in the implemented nodes related to detection, thus it can not be depicted by the ROS tf debugging tools.

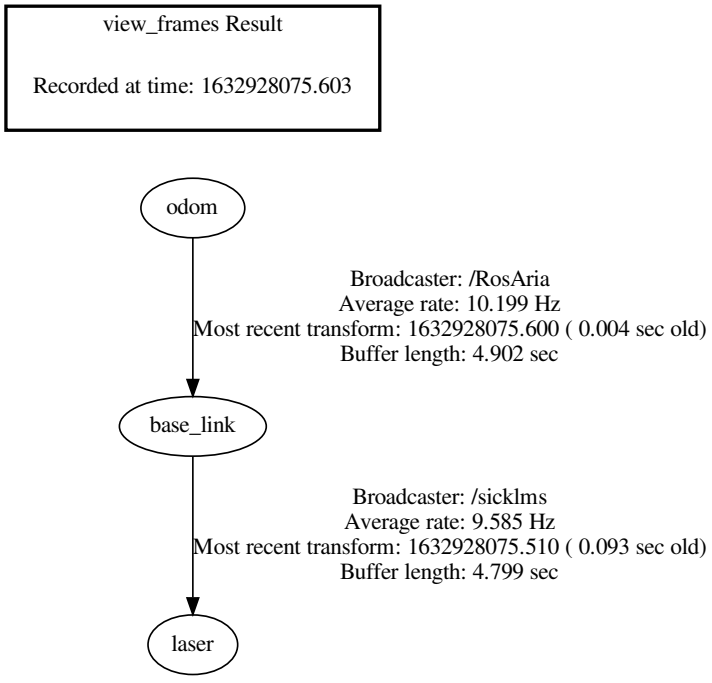


Figure 4.1: Frames of the 3-DX Robot - Real World Application.

Figure 4.2: Frames of the 3-DX Robot - Simulation.

The ROS computation graph corresponding to the combination of nodes used synergetically from both the navigation and perception systems, is obtained by the `rqt_common_plugins` ROS package and is depicted in the Figure 4.3. The graph describes in detail the sequence of computations and the dependencies between nodes within the same system as well as between nodes of the two distinct systems. The computation graph of Figure 4.3 is the one corresponding to the real world application and is thus corresponding also to the experiment that is described in the followings sections .

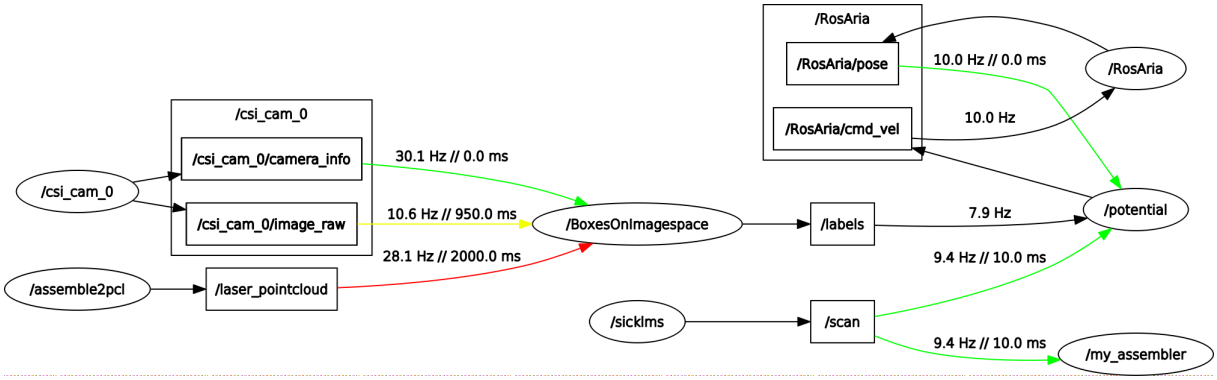


Figure 4.3: Real world experiment scenarios - ROS Computation Graph

Note that a similar Computation Graph can be obtained for the Gazebo simulation environment, which can be seen in the Figure 4.4. It is describing exactly the same computation sequence and dependencies with respect to the navigation and perception system as the one in the Figure 4.3. It is referenced here as a proof that the simulation scenarios that were implemented for this thesis were really close to the real world application.

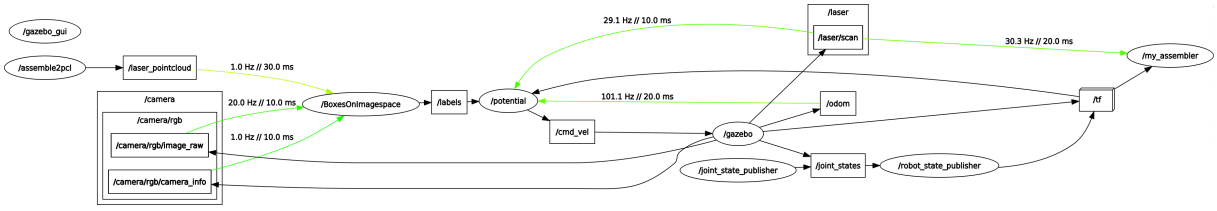


Figure 4.4: Gazebo - ROS Computation Graph

4.1 Proof of concept - Perception and Navigation System Application

Using the navigation system described in subsection 3.1.3, as a proof of concept scenario, the detection of an object of interest using the proposed detection system was accomplished in the laboratory environment. The robot has no information about its workspace and it must navigate dynamically adapting its path to the environment using the laser range scanner data. The robot has to navigate autonomously to a given known goal pose, avoiding obstacles and if a given object of interest is detected by the perception system, the robot stops and interacts with it in the from of capturing a image with a labeled bounding box enclosing the object of interest. Thus, the interaction of the robot with the environment falls in the general case of interaction through environment sensor measurements, thus not changing the state of the environment, in comparison to control actions that change the state of the world [35]. The captured image is a proof of concept, that the proposed perception system behaves as described, achieving real time object detection while the robot navigates autonomously and dynamically in an unknown workspace.

The object of interest in this experiment is an oscilloscope randomly placed in the laboratory's floor, with the robot having no information of the oscilloscope's pose. As obstacles, carton boxes are placed on the laboratory's floor with poses and configurations unknown to the robot. The robot is given a goal pose in the form of $[X, Y, \theta]$ with X, Y denoting the Cartesian Coordinates of the goals position and θ denoting the yaw with respect to a coordinate system coinciding with the robot's coordinate system at the initial position prior to any movement, i.e. a world coordinate system also coinciding with the odometry coordinate frame. The state of the environment can be seen in the Figure 4.5.

The robot is given a goal pose at $[X, Y, \theta] = [5, 5.2, 15^{\circ}]$ as seen in the Figure 4.5 and using the odometry data, i.e., data from motion sensors that determine the robot's change in position relative to the world frame, the robot navigates using as a control law, velocity commands derived by the proposed navigation system until it reaches the goal pose. If the proposed detection system perceives an oscilloscope while navigating, then the robot stops for 10 seconds, captures an image and then continues navigating dynamically until the goal pose is reached.

The path that the robot follows in this scenario, can be seen from a perspective in

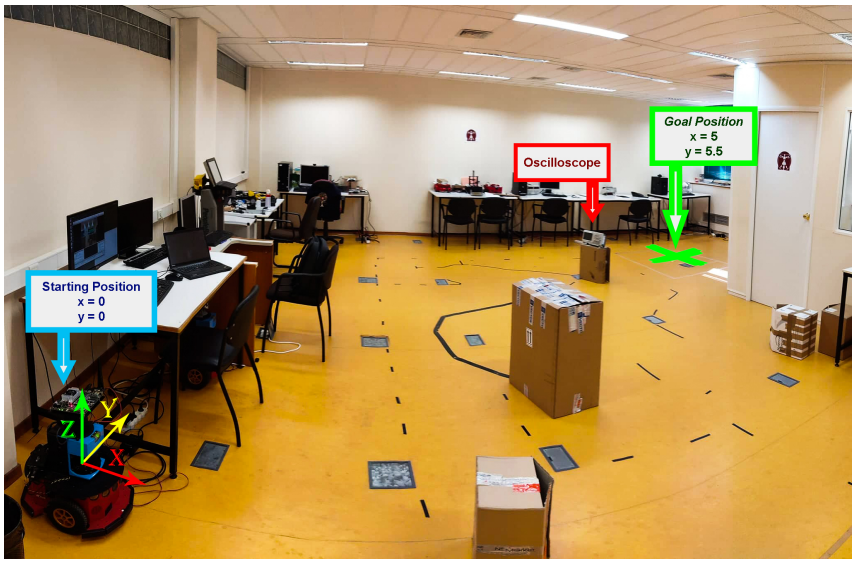


Figure 4.5: World State.

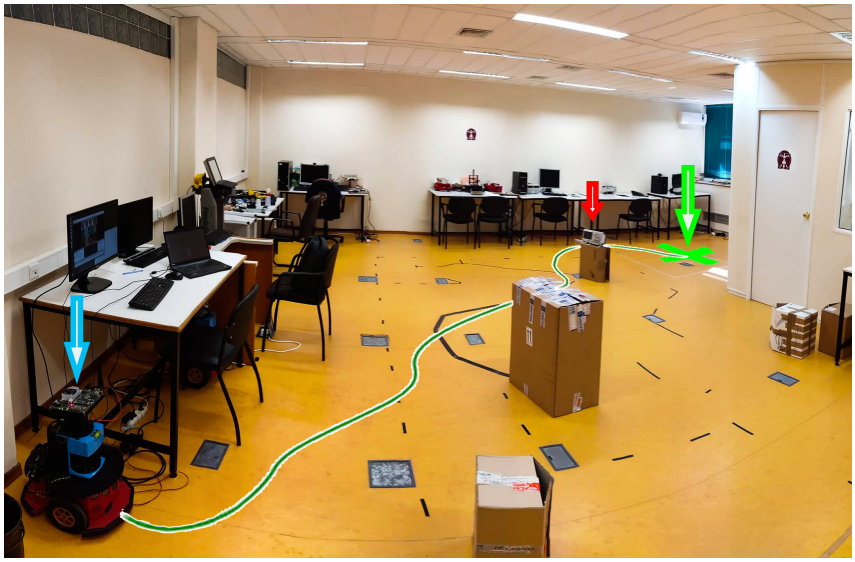


Figure 4.6: Path followed until goal pose.

the Figures 4.6 represented by the green and white line.The odometry data from the robot's motion sensory system give the actual path by plotting Y coordinate versus the corresponding X coordinate with respect to the world frame and is given in figure 4.7.

The robot, while dynamically navigating , detects the oscilloscope by the use of the proposed perception system, when the point cloud cluster representing the oscilloscope is the one closest to the robot, at a distance of exactly 1.19 meters. The captured image is depicted in Figure 4.8 and is a proof of correct functionality.

The time interval of 10 seconds that the robot is motionless, as expected by the

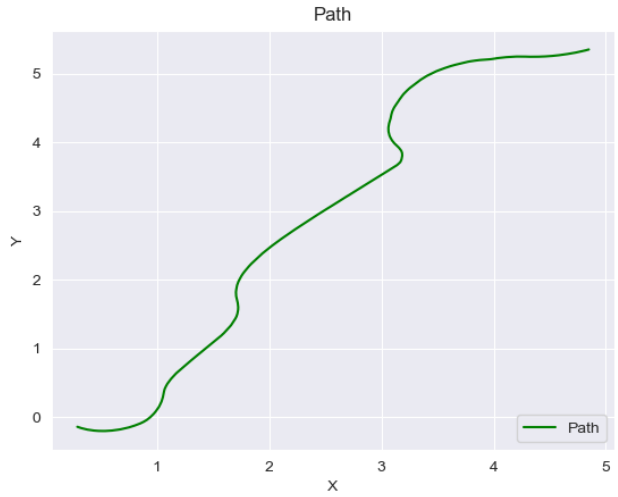
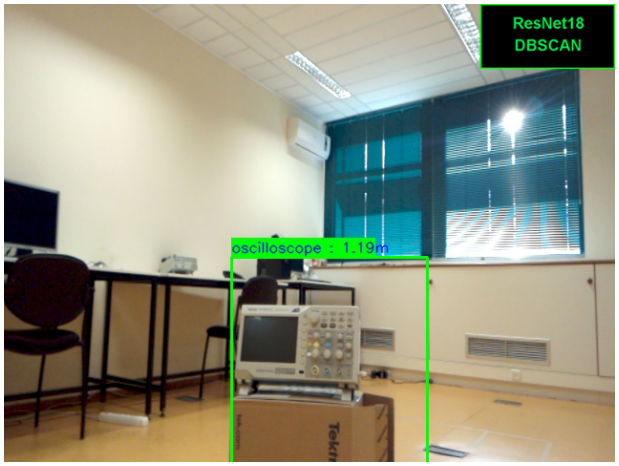
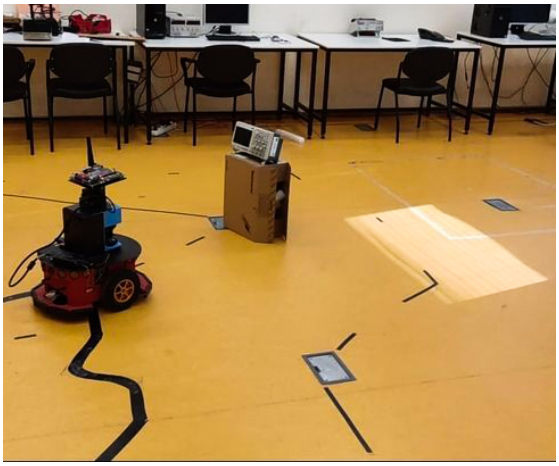


Figure 4.7: Actual path using Odometry Position Data



(a) Image Captured-Object Detection.



(b) Object Perceived-Motionless State

Figure 4.8: Environmental Sensor Measurement.

experiments design, when the object of interest is perceived, as well as all the overall motion of the robot can be seen in the diagrams of position and speed trajectories, plotted in Figures 4.9, 4.10,4.11.

The time interval between $22.758 - 32.500$ seconds, i.e., 9.742 seconds, that both the linear and the angular velocity of the differential drive robot is zero, is the approximate 10 second interval that the robot is motionless in order for the environmental sensor measurement to take place, as required.

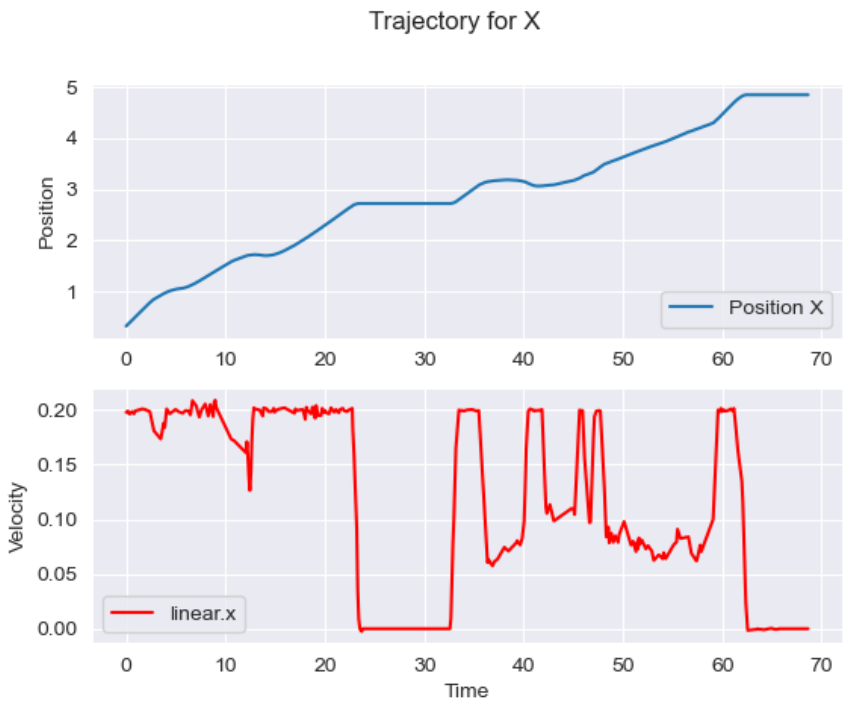


Figure 4.9: Trajectory for X.

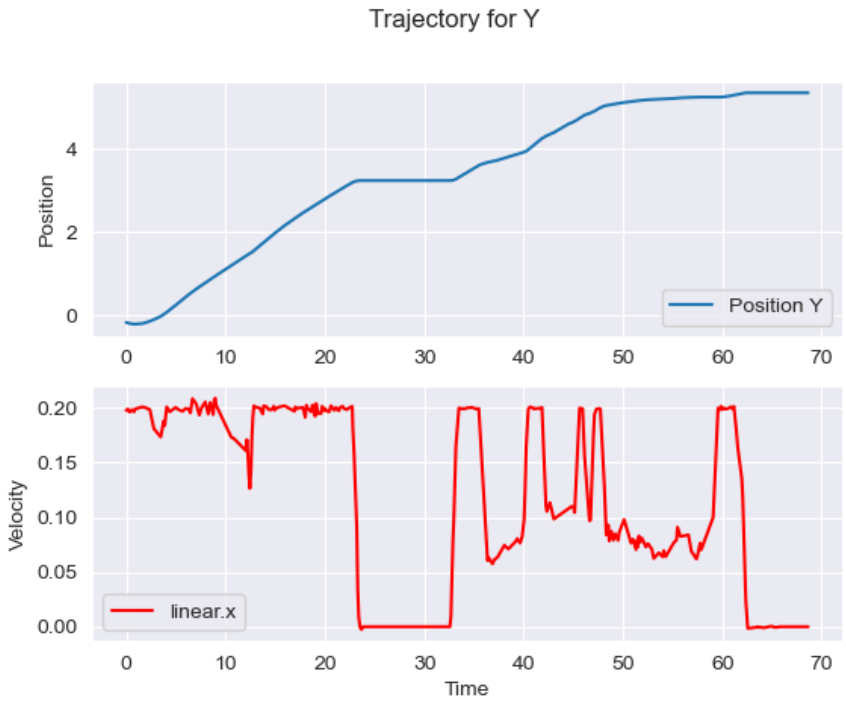


Figure 4.10: Trajectory for Y.

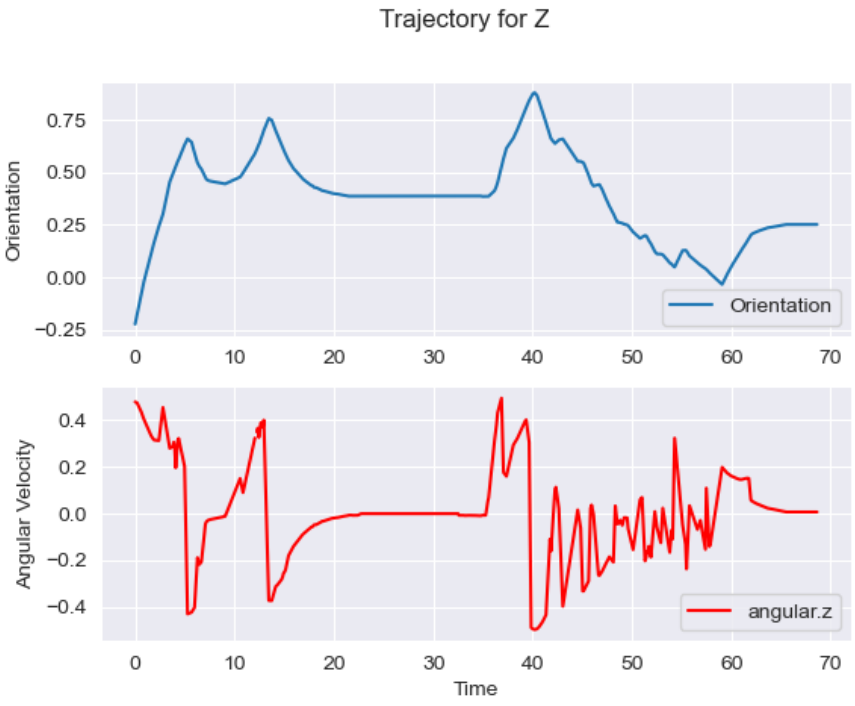


Figure 4.11: Trajectory for Z.

4.2 Perception System Real Time Object Detection Cases

The results of the application of the proposed object detection system can be seen in the rest of the section. The photos are taken in real time while the robot navigates in the corresponding work spaces, some using the autonomous navigation system discussed in 3.1.3 and some via remote control of the robot motion system for testing purposes.

Note that for the below detected objects, no changes have been done in the class label names of the ImageNet data set in order to fit the specific objects that can be found in a University faculty, thus the labels have a lot of generality.

The Figures 4.12 - 4.21, give a layout representation of the clustered point cloud along with the detected object.

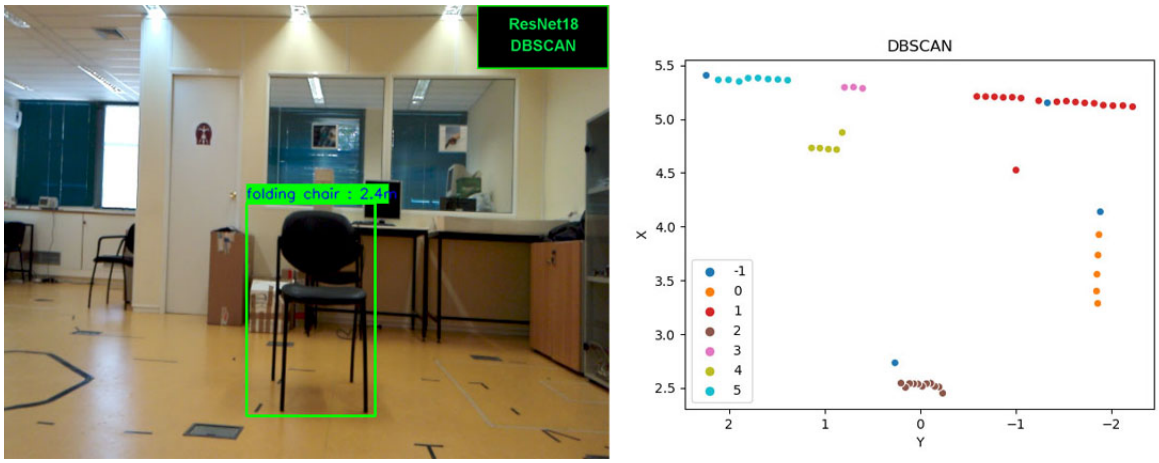


Figure 4.12: Detection of a chair in the laboratory environment.

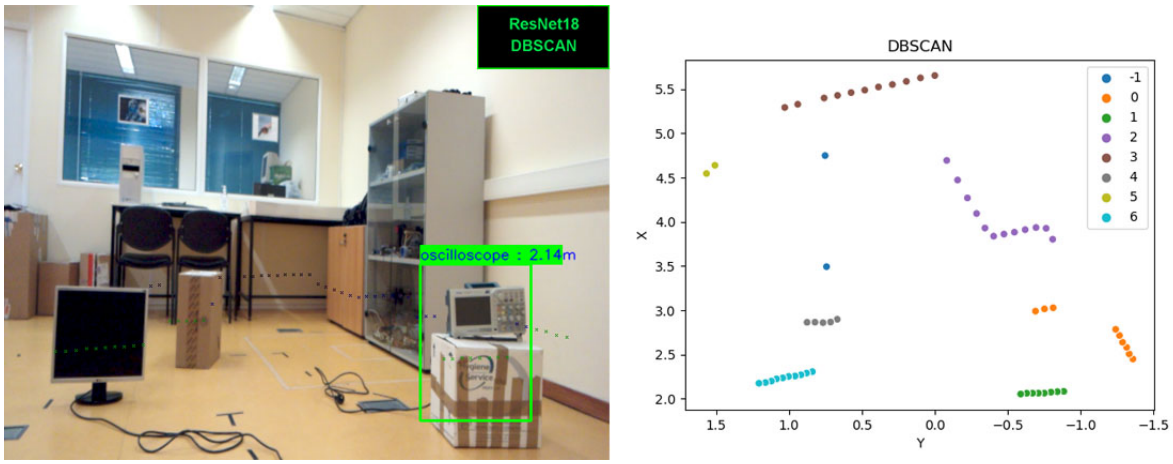


Figure 4.13: Detection of oscilloscope in the laboratory.

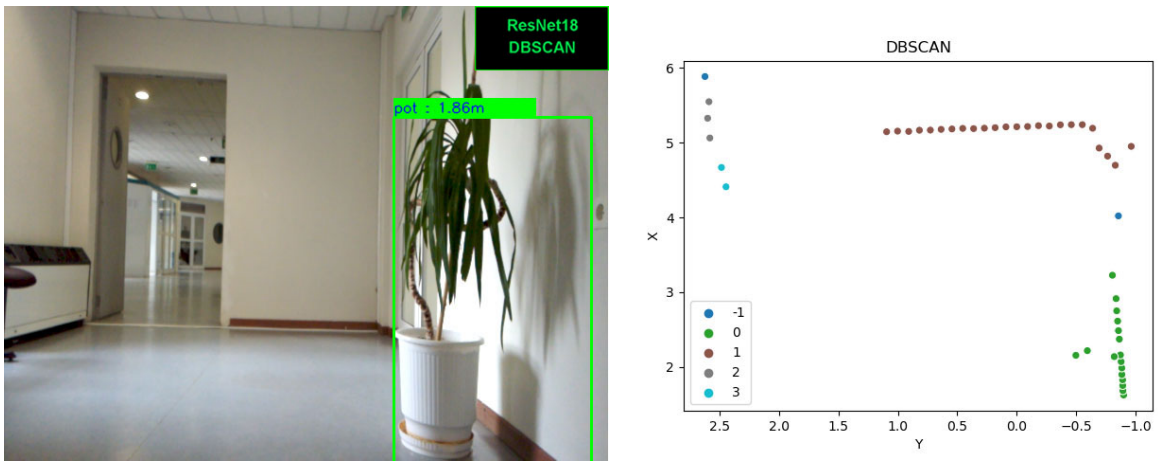


Figure 4.14: Detection of a potted plant in the faculty's corridor.

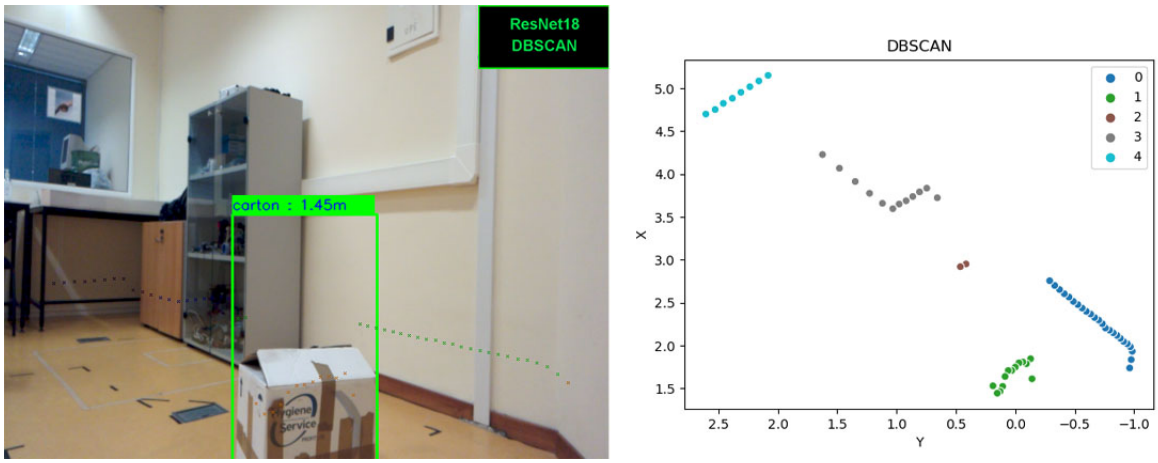


Figure 4.15: Detection of a carton in the laboratory environment.

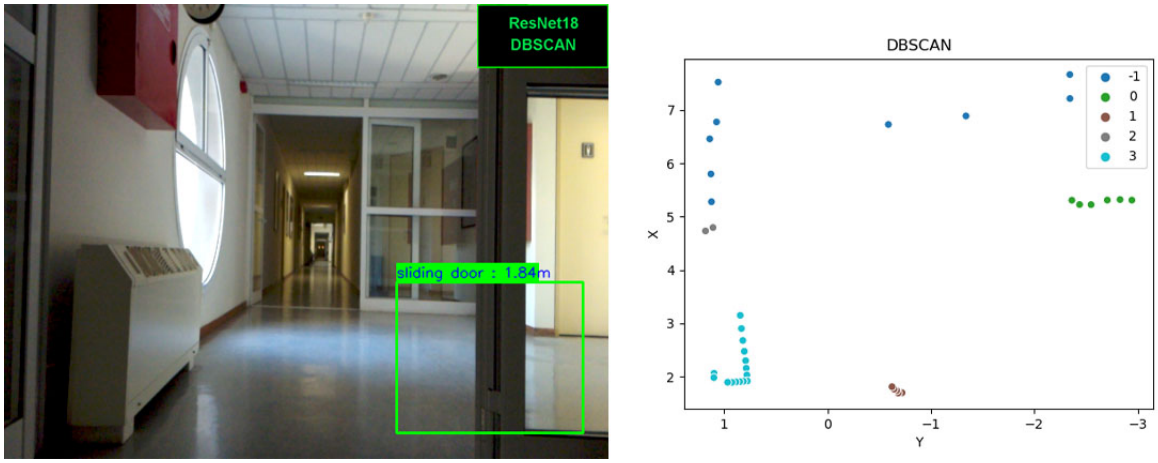


Figure 4.16: Detection of a door in the faculty's corridor.

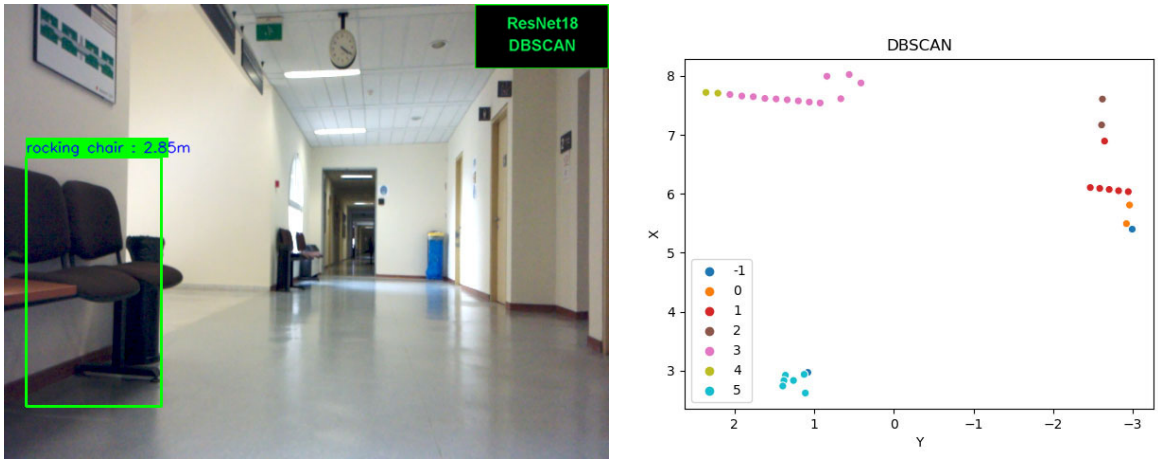


Figure 4.17: Detection of chair in the faculty's corridor.

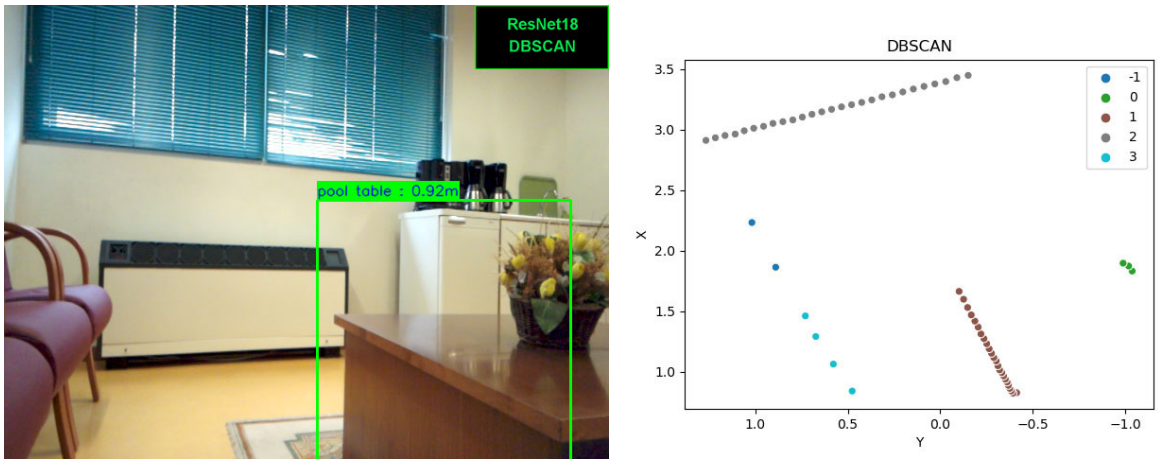


Figure 4.18: Detection of table in the faculty's coffee room.

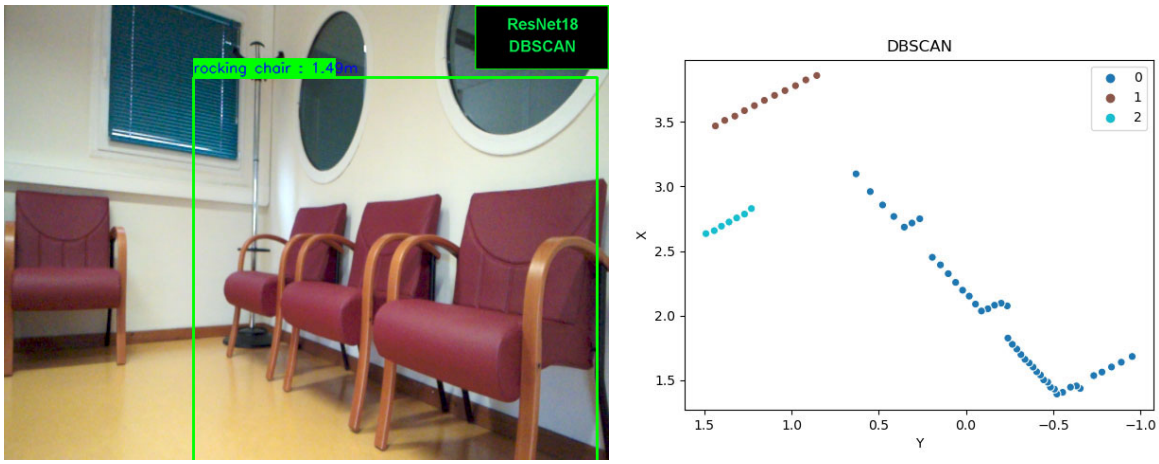


Figure 4.19: Detection of chairs in the laboratory environment.

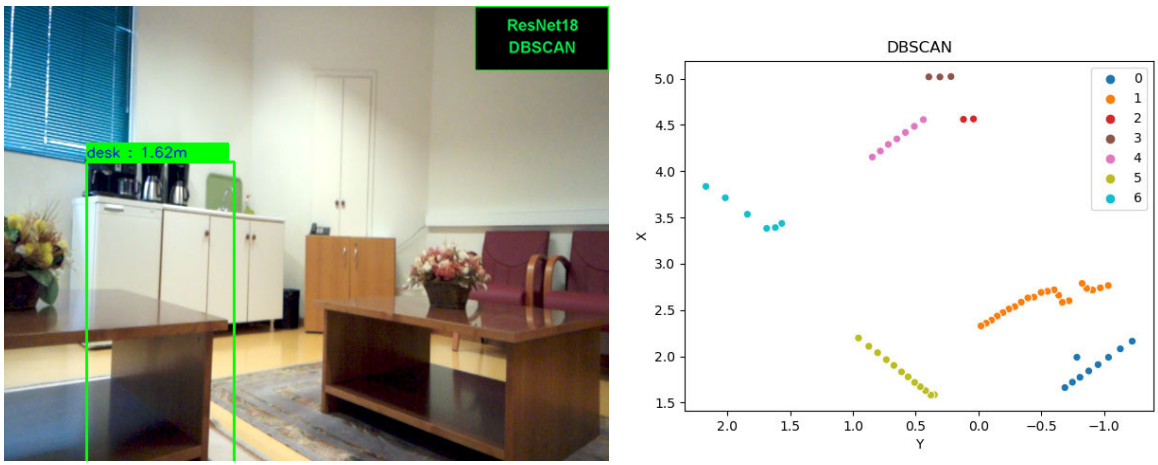


Figure 4.20: Detection of desk in the faculty's coffee room.

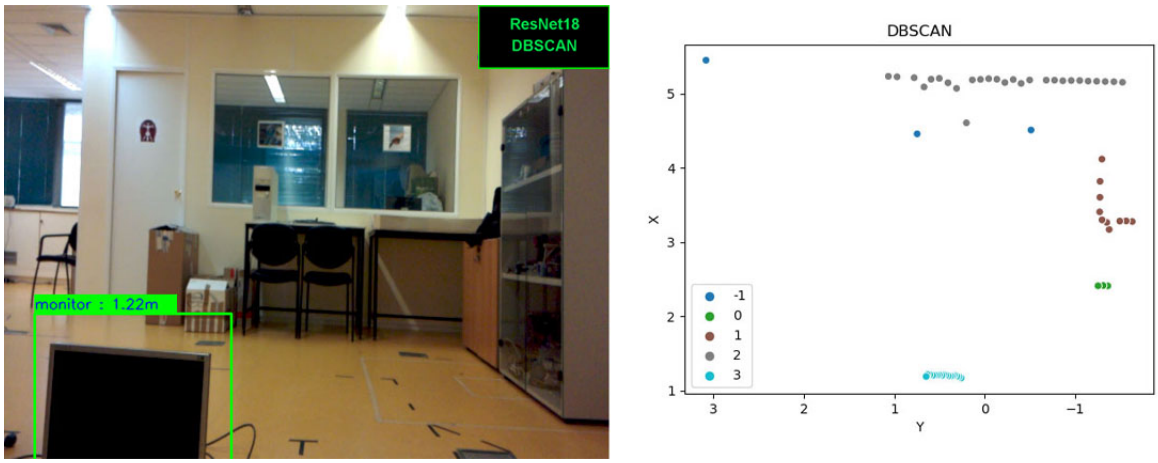


Figure 4.21: Detection of monitor in the laboratory.

In all cases detection is focused on the closest object to the robot and the DBSCAN algorithm successfully segments the point cloud so that the closest object can be successfully labeled. The Figures 4.22, 4.23, 4.24 give a more abstract pictorial representation of the proposed system, giving a perspective view of the full point cloud representing the detected environment.

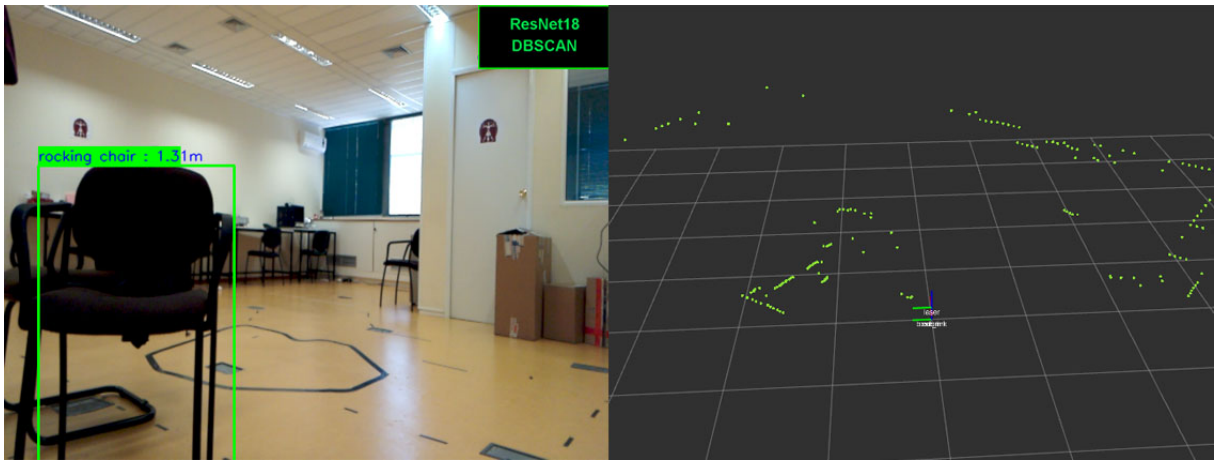


Figure 4.22: Case of a chair.

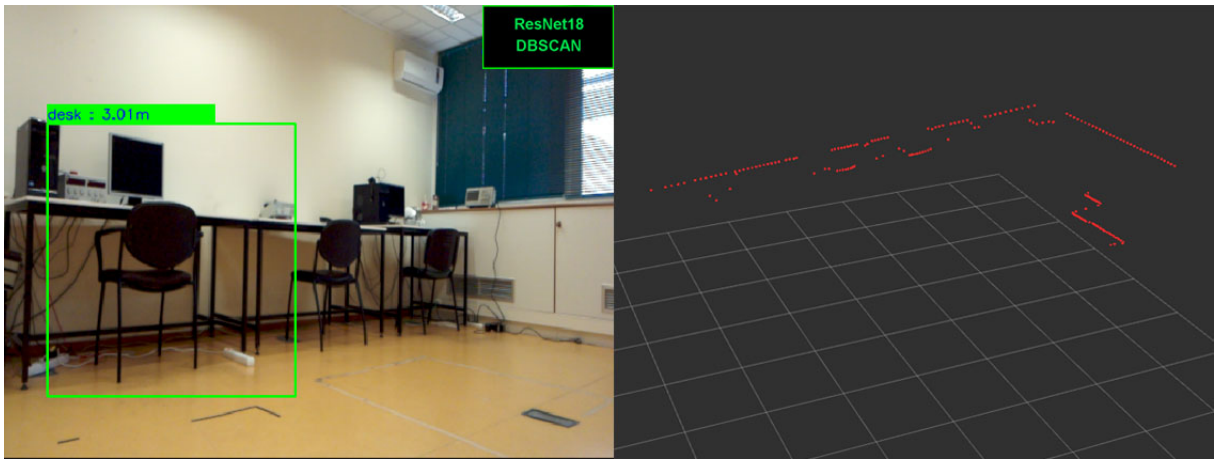


Figure 4.23: Case of a desk.

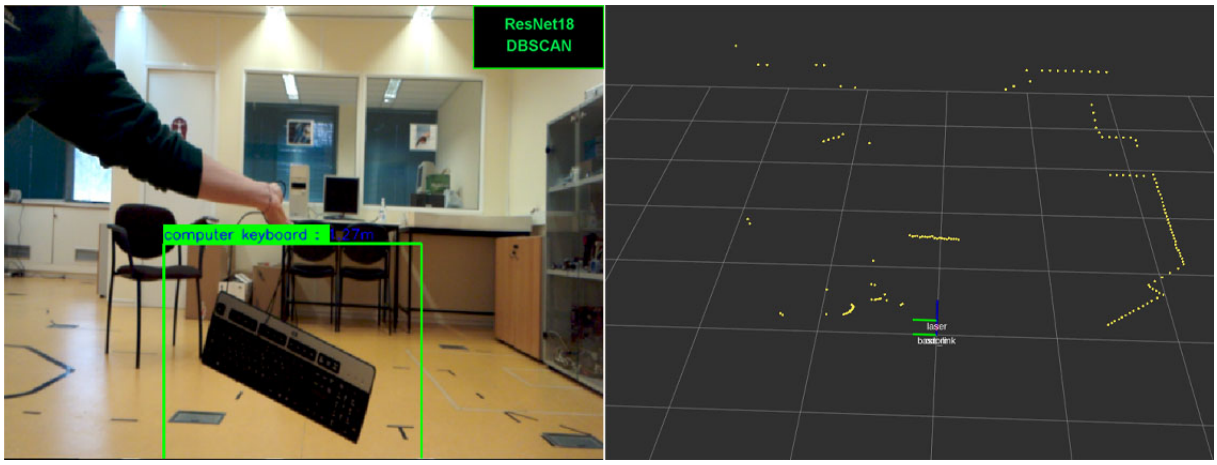


Figure 4.24: Case of a Keyboard.

Some cases of detected object of interest captured can be seen in Figures 4.25, 4.26,4.27, 4.28 and 4.29.

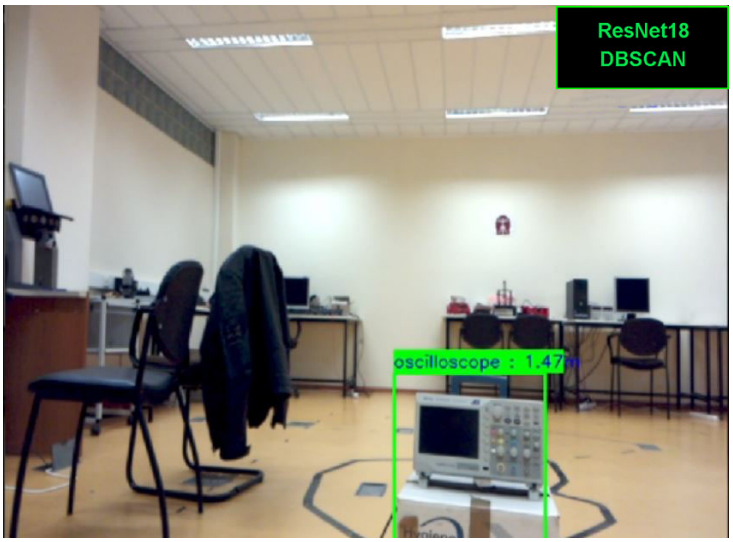
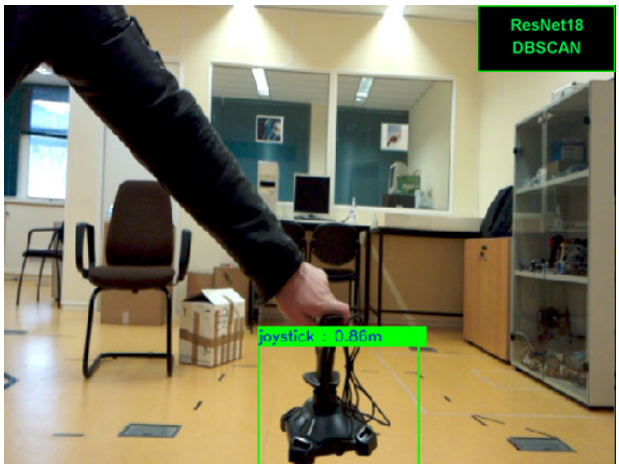


Figure 4.25: Oscilloscope case.

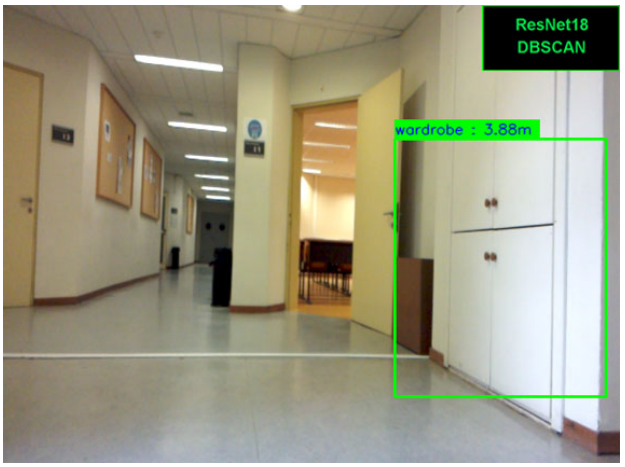


(a) Joystick case

(b) Car case

Figure 4.26: Joystick & Car

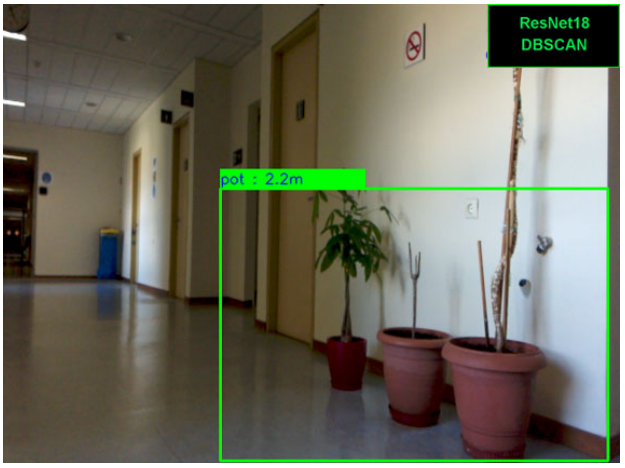
(a) Space heater case.



(b) Wardrobe case.

Figure 4.27: Space heater & Wardrobe

(a) Single pot.



(b) Many pots.

Figure 4.28: A single pot & many pots case

(a) Door case.

(b) An other door case.

Figure 4.29: Two door cases

Chapter 5

Conclusion

The application of sensor fusion to achieve accuracy in object detection is a challenging and multidisciplinary task. Given the nature of a robot that navigates autonomously in a dynamic environment and acts on it, the complexity of the task is further enhanced.

Given the analysis discussed in the previous chapters and the experimental results, it is concluded that in many cases and different scenarios the proposed perception system achieves its goal and the robot navigates autonomously in a dynamic environment while it interacts with objects of interest. The fusion of 2D-laser and camera information successfully completes in most cases the task of real time object detection and distance evaluation. The proposed voting schema gives an intuition of the objects height dimension , even if the height dimension of an object is not directly inferrable from the point cloud. The proposed perception system has also some limitations as described in the following sub sectors.

5.1 Comparison to state of the art YOLO approach

The first clear advantage of the proposed detection system compared to YOLOv3 is the ability to detect 1000 objects in comparison to YOLO that in most cases is trained

to detect 80 objects. That is a direct advantage coming from the fact that the CNNs that can be used as the backbone for the recognition in the detection system, i.e., the ResNet18 in the specific implementation, are trained on the ImageNet data set, in comparison to YOLO that is trained mainly on the COCO data set.

A second advantage of the proposed detection system is the easiness and convenience in adding more classes and retraining the CNN models, adapting the detection system to many different and specialized use cases. Transfer learning and fine tuning of pretrained models, available in many framworks, is an easy task compared to retraining or fine tuning the YOLOv3 approach in order to detect new objects and make the YOLO approach applicable in specialized applications.

The third advantage of the proposed approach comes from the fusion of the laser with the camera sensor, that gives accurate information of the detected objects distance from the robot. That particular advantage over YOLO enables Visual Servoing directly from the fused information, which enhances the interaction with the perceived surrounding environment and increases the reliability.

A disadvantage, though, of the specific perception system stems from the limitations of the 2-D laser scanner. The point cloud of the closest object that is produced lacks of height dimension. The proposed voting system tries to accommodate the missing information, but in comparison to the YOLO approach, the bounding boxes of the proposed system a lot of times fail to include the object with respect to the height dimension. That bottlenecks severely in many cases the accuracy in detection. Sensor synchronization also adds additional challenges and limits the reliability.

In terms of real time detection speed comparison, given the specific hardware specs used in this thesis, the proposed system is a lot faster that YOLOv3 but a bit slower than the tiny YOLO model.

5.2 Future Work

The proposed perception system, as it is, with the use of transfer learning can be adapted in theory to any specialized application. Following the intuition of the proposed perception and navigation system, a lot of space is given to new approaches, especially in the fusion application part, some of which are described below.

First and foremost, the use of a 3D LIDAR sensor will greatly enhance the accuracy

in object localization from the point cloud and can enable more specialized approaches to point cloud segmentation and clustering using deep learning.

Another approach of the fusion of the two sensors can be accomplished using more complex methods. Fusing the data prior to the final inference, i.e. early fusion, and using the fused data as input to an inference model can enhance the reliability of the system in terms of data synchronization.

As far as the navigation approach used in this thesis, a combination of advanced SLAM techniques enhanced by the use of the detection system in terms of Visual Servoing, can increase the applicability of the system as a whole, in many complex dynamic environments and enable the robots accurate and reliable interaction with the workspace.

Bibliography

[1] D. Schacter, D. T. Gilbert, and D. M. Wegner, *Psychology (2nd Edition)*. New York: Worth, 2011. [Online]. Available: http://www.amazon.com/Psychology-Daniel-L-Schacter/dp/1429237198/ref=sr_1_1?s=books&ie=UTF8&qid=1313937150&sr=1-1

[2] Wikipedia contributors, "Perception — Wikipedia, the free encyclopedia," 2004, [Online; accessed 10-August-2021]. [Online]. Available: https://en.wikipedia.org/wiki/Perception#cite_note-2

[3] M. J. A. Lynn Abbott, Ruoxi Jia. Machine perception research - ece - virginia tech. [Online]. Available: https://ece.vt.edu/research/area/perception

[4] M. Tatum. What is machine perception? [Online]. Available: https://www.easytechjunkie.com/what-is-machine-perception.htm

[5] A. Serov, "Subjective reality and strong artificial intelligence," *CoRR*, vol. abs/1301.6359, 2013. [Online]. Available: http://arxiv.org/abs/1301.6359

[6] C. Premebida, R. Ambrus, and Z.-C. Marton, "Intelligent robotic perception systems," in *Applications of Mobile Robots*, E. G. Hurtado, Ed. Rijeka: IntechOpen, 2019, ch. 6. [Online]. Available: https://doi.org/10.5772/intechopen.79742

[7] Wikipedia contributors, "Sensor fusion," 2004, [Online; accessed 05-August-2021]. [Online]. Available: https://en.wikipedia.org/wiki/Sensor_fusion

[8] C. Suchocki, "Comparison of time-of-flight and phase-shift tls intensity data for the diagnostics measurements of buildings," *Materials*, vol. 13, no. 2, 2020. [Online]. Available: https://www.mdpi.com/1996-1944/13/2/353

[9] C. Ye and J. Borenstein, "Characterization of a 2d laser scanner for mobile robot obstacle negotiation," in *Proceedings 2002 IEEE International Conference*

on Robotics and Automation (Cat. No.02CH37292), vol. 3, 2002, pp. 2512–2518 vol.3.

[10] E. Optics. Imaging electronics 101: Understanding camera sensors for machine vision applications. [Online]. Available: https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-camera-sensors-for-machine-vision-applications/

[11] P. Corke, *Robotics, Vision and Control Fundamental Algorithms In MATLAB® Second, Completely Revised, Extended And Updated Edition*, ser. Springer Tracts in Advanced Robotics. Springer International Publishing, 2017, vol. 118.

[12] C. Stachniss. Camera parameters – extrinsics and intrinsics. [Online]. Available: https://www.ipb.uni-bonn.de/

[13] S. S. Kenji Hata. Cs231a: Computer vision, from 3d reconstruction to recognition. [Online]. Available: https://web.stanford.edu/class/cs231a/

[14] Wikipedia contributors, "Direct linear transformation," 2004, [Online; accessed 02-August-2021]. [Online]. Available: https://en.wikipedia.org/wiki/Direct_linear_transformation

[15] T. V. Haavardsholm, I. Dyrdal, T. Opsahl, and R. Smestad. Maskinsyn (machine vision) - university of oslo. [Online]. Available: https://www.uio.no/studier/emner/matnat/its/nedlagte-emner/UNIK4690/v18/lectures/lecture_01/

[16] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[17] B. Caramiaux and A. Tanaka, "Machine learning of musical gestures: Principles and review," 2013. [Online]. Available: http://research.gold.ac.uk/id/eprint/14645

[18] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[19] A. Gunawan, "A faster algorithm for dbscan," 2013.

[20] E. Schubert, J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Dbscan revisited, revisited: Why and how you should (still) use dbscan," *ACM Trans. Database Syst.*, vol. 42, pp. 19:1–19:21, 2017.

[21] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[22] C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Springer, 2018. [Online]. Available: https://books.google.gr/books?id=AsTswQEACAAJ

[23] M. Shinozuka and B. Mansouri, "4 - synthetic aperture radar and remote sensing technologies for structural health monitoring of civil infrastructure systems," in *Structural Health Monitoring of Civil Infrastructure Systems*, ser. Woodhead Publishing Series in Civil and Structural Engineering, V. M. Karbhari and F. Ansari, Eds. Woodhead Publishing, 2009, pp. 113–151. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781845693923500049

[24] F. F. Li. Convolutional neural networks for visual recognition. [Online]. Available: http://cs231n.stanford.edu/

[25] Wikipedia contributors, ""convolutional neural network"," 2004, [Online; accessed 17-August-2021]. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network

[26] J. Brownlee. A gentle introduction to object recognition with deep learning. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/

[27] G. K. Karagiannidis, *Telecommunication Systems*. Tziolas, 2009.

[28] A. Gjendemsjø and et al, *Information and Signal Theory*, ser. Science Textbooks Books. free-ebooks, 2014.

[29] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, R. Hovey, G. Kendrick, and R. B. Fisher, "Chapter 21 - deep learning for coral classification," in *Handbook of Neural Computation*, P. Samui, S. Sekhar, and V. E. Balas, Eds. Academic Press, 2017, pp. 383–401. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128113189000211

[30] A. Likas and G. Theofanis, "Multiyolo: Learning new yolo categories without full retraining," 2020.

[31] G. Karimi. Introduction to yolo algorithm for object detection. [Online]. Available: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

[32] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics Modelling, Planning and Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer-Verlag London, 2009.

[33] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, "Principles of robot motion: Theory, algorithms and implementations."

[34] R. W. contributors. Rosaria package summary. [Online]. Available: http://wiki.ros.org/ROSARIA

[35] T. Sebastian, B. Wolfram, and F. Dieter, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005. [Online]. Available: http://www.amazon.de/gp/product/0262201623/102-8479661-9831324?v=glance&n=283155&n=507846&s=books&v=glance

Short Biography

Spyridon Syntakas was born in Ioannina,Greece and since 2017 holds a Diploma in Electrical and Computer Engineering from the Department of Electrical and Computer Engineering of the Polytechnic School of Aristotle University of Thessaloniki. After fulfilling the compulsory military service as a private in Research and Informatics Corps in 2019, he enrolled in the Graduate Program of the Department of Computer Science and Engineering of University of Ioannina, and is persuiting a MSc Degree entitled ”Data and Computer Systems Engineering”. Since July 2021, he is a PhD candidate at the Department of Computer Science and Engineering with research topics in the broad scientific fields of Robotics and Artificial Intelligence. His specific research interests are Intelligent Perception focusing on Machine and Deep learning approaches, Automated Control of Robotic Systems and Analysis and Fusion of sensory data. He is a member of the Technical Chamber of Greece, making him a licensed Engineer.