Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques

Ihab S. Mohamed

Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques

Ihab Sami Mohamed Mohamed

European Master on Advanced Robotics

University of Genova, Italy

Supervisors:

Fulvio Mastrogiovanni, University of Genova

Stefano Rovetta, University of Genova

Renato Zaccaria, University of Genova

Co-supervisors:

Alessio Capitanelli, University of Genova

Krzysztof Mianowski, Warsaw University of Technology

In partial fulfillment of the requirements for the degree of

Robotics Engineering

September 18, 2017

Acknowledgements

Firstly, I would like to express my gratitude for my supervisors: Prof. Fulvio Mastrogiovanni, Stefano Rovetta, and Renato Zaccaria. Their continuous guidance and constant motivation throughout the period of thesis were priceless. I am grateful to Prof. Fulvio Mastrogiovanni for his immense patience and his valuable feedbacks in each and every step.

I convey my heartfelt gratitude to Prof. Stefano Rovetta for his assistance. I am inspired by his selfless to assist me even when approached in the mid work of the thesis.

I am thankful to Mr. Alessio Capitanelli and Mr. Antonello Scalmato for their valuable time and support. I convey my gratitude to and Qiao Yang, also from Singular Perception for their precious time to discuss and exchange ideas with me.

My heartfelt gratitude to Prof. Renato Zaccaria, EMARO Unige local coordinator for his tireless effort in ensuring the perfect environment for us to prosper. I sincerely thank him for his replies to all my queries on time.

My sincere thanks to Prof. Phillipe Martinet for accepting me as a student in the EMARO Program.

I am thankful to my dear wife without her support and patience I would not have embarked on this wonderful journey for the quest of knowledge.

Last but not the least, I would like to thank my family, friends especially the EMARO colleagues and every single soul who helped and cared for me throughout my Masters.

To all the Master and PhD students of Robotics Engineering at the University of Genova.

Abstract

The problem of developing an autonomous forklift that is able to pick-up and place pallets is not new. The same is true for pallet detection and localization, which pose interesting perception challenges due to their sparse structure. Many approaches have been presented for solving the problems of extraction, segmentation, and estimation of the pallet based on vision and Laser Rangefinder (LRF) systems. Here, the focus of attention is on the possibility of solving the problem by using a 2D LRF.

On the other hand, machine learning has become a major field of research in order to handle more and more complex detection and recognition problems. The aim of this thesis is to develop a new and robust system for identifying, localizing, and tracking the pallets based on machine learning approaches, especially Convolutional Neural Network (CNN)s. The proposed system is mainly composed of two main components: Faster Region-based Convolutional Network (Faster R-CNN) detector and CNN classifier for detecting and recognizing the pallets, and a simple Kalman filter for tracking and increasing the confidence of the presence of the pallet. For fine-tuning the proposed CNNs, the system is tested systematically on real world data containing 340 labeled object examples. Finally, performance is evaluated given the average accuracy over k-fold cross-validation. The computational complexity of the proposed system is also evaluated.

Finally, the experimental results are presented, using MATLAB and ROS, verifying the feasibility and good performance of the proposed system. The best performance is achieved by our proposed CNN with an average accuracy of 99.58% for a k-fold of 10. Regarding the tacking task, the experiments are performed while the robot was moving towards the pallet. Due to availability, the experiments are carried out by considering only one pallet, and consequently to check the robustness of our algorithm, an artificial data are generated by considering one more pallet in the environment. It is observed that our system is able to recognize and track the positive tracks (pallets) among other negatives tracks with high confidence scores.

Contents

List of Figures

List of Abbreviations

AGV Automated Guided Vehicle. 4, 5

AIRONE Automa Intelligente Robotico per Organizzazione Navette Elettriche. 3

ANN Artificial Neural Network. 11

CNN Convolutional Neural Network. iii–vi, 5–7, 10–16, 18, 20, 23, 25–30, 37, 39, 43–45, 50

DSP Digital Signal Processing. 8

EMARO European Master on Advanced RObotics. 38

EPAL European Pallet Association. 34

ETHNOS Expert Tribe in a Hybrid Network Operating System. 4

Faster R-CNN Faster Region-based Convolutional Network. iii, 7

FOV Field-of-View. 5, 7, 29, 30, 36

FPGA Field Programmable Gate Array. 8

ICP Iterative Closest Point. 10

LRF Laser Rangefinder. iii, 5, 7, 9, 50

MLP multilayer perceptrons. 12, 14

MPC Model Predictive Control. 51

PDA Personal Digital Assistant. 9

PMD Photonic Mixer Device. 9

R-CNN Region-based Convolutional Neural Netwroks. iv–vi, 11–14, 19, 20, 23, 25–28, 30, 39, 41, 43–45

ROI Regions of Interest. v, vi, 22–26, 28, 29, 39, 42–44, 46–49

ROS Robt Operating System. 37–39

RPN Region Proposal Network. 13, 20

RST Robotics System Toolbox. 37

SGD Stochastic Gradient Descent. 41, 43

SLAM Simultaneous Localization and Mapping. 5, 9

SVM Support Vector Machine. 12–14

TOF Time-of-Flight. 35

Chapter 1

Introduction

1.1 Robots for Loading and Unloading Containers

Many of the goods for sale in Europe and the US are made in Asia. Most of these goods cross the ocean in standardized shipping containers. To save transport costs, the majority of these goods are loaded on the container's floor and stacked to the ceiling without pallets. When the container arrives at a port, it is loaded onto a truck and sent to a distribution centre. On arrival, the contents of the container are typically unloaded by hand, sorted, and stacked onto pallets so that they can be stored in the warehouse. This manual and labour-intensive process can take several hours. Similarly, many long-haul parcel trucks are loaded floor to ceiling without pallets and require significant labour to unload.

In 2003, in an attempt to improve this process, DHL and its business and research partners developed a new robot prototype called *parcel robot*, which consists of a chassis, a telescopic conveyor belt, a 3D laser scanner, and a gripping system made up of an articulated robotic arm and a grabber[1]. The robot is positioned in front of a container to unload and uses its laser to scan all of the boxes. An integrated computer determines the optimal unloading sequence. The robot picks up a box and places it onto a conveyor that transports the item out of the container and into the sorting centre. The robot moves forward as it works until the entire container is unloaded. DHL never rolled out this concept across its network as in 2003 the technology was insufficiently mature for industrial uptake.

Nevertheless, DHL's innovative parcel robot proved that robot-based unloading is possible and several companies have since developed the concept further. In fact, a US company called Wynright currently offers a truck unloading robot

[1] https://www.youtube.com/watch?v=CsHfbdnyHAE

for sale[1]. Like the DHL's parcel robot, it unloads boxes onto an extendable conveyor belt at a rate of over 500 parcels per hour. Unlike the DHL's robot, it uses low-cost cameras to locate the boxes rather than more expensive laser scanners. Companies like Wynright are also developing trailer loading robots. This application adds further complexity to the algorithm determining the unloading sequence, because the system has to determine the best way to stack boxes of different shapes and weights to optimally fill the trailer without damaging any of the items.

1.2 Mobile Object Picking Robots

The opposite of the goods-to-picker system is a mobile robot that moves around traditional warehouse shelves and picks items just like a person would. Several startups are currently working on robots with these capabilities. IAM Robotics is a small US-based company currently developing a mobile manipulator using vision to navigate in existing warehouses, picking items from shelves, and placing them into an order tote. The system has been first field tested in a pharmaceutical warehouse in New York, where it was able to pick test orders from 40 items that it had never seen before[2].

Fetch Robotics is a well-funded startup developing a robot able to move around a warehouse and to pick items from shelves. Its primary robot, called Fetch, can extend its torso to reach upper shelves, while a small secondary robot, call Freight, helpfully holds the tote that Fetch will pick items into. Each Fetch robot can have several of these smaller Freight robots supporting the picking process. The agile Freight robots quickly move the totes around the warehouse from area to area while the slower Fetch robots can stay in one aisle and deal with picking items. This solution will create a hybrid goods-to-picker and manual picking concept.

Magazino is a German startup developing perception-driven mobile robots for intra/logistics. Their latest development is the picking robot TORU[3]. Using 2D and 3D cameras as well as Magazino's technology, TORU is expected to identify individual objects on a shelf, grasp an item securely, and place it precisely at its destination. TORU works alongside humans, providing just-in-time object delivery to the workbench or shipping station.

There are still technical challenges to overcome before these robots will be ready for widespread use. However, they have some key advantages over station-

[1] http://www.wynright.com/products/by-product-family/robotic-solutions/truck-and-container-loading-and-unloading/

[2] http://www.logisticsmgmt.com/article/the_robots_are_coming_part_iii

[3] https://www.youtube.com/watch?v=ahwnEnP-um8

ary goods-to-picker robots:

- They represent modular and scalable solutions.
- They can work alongside human co-workers, picking up easy items while humans can pick up more complicated products or focus on managing exceptions.

On the other hands, Typically, these kind of robots cannot be considered fully autonomous. They either rely on environmental signs, which have been purposely designed for them, such as artificial beacons, coloured strips on the floor or markers of various shapes, or are designed as transportation systems for specialized goods, or both.

1.3 AIRONE Project

Automa Intelligente Robotico per Organizzazione Navette Elettriche (AIRONE) project is a collaboration between Dibris and the food delivery company Sogegross SpA, Tortona Italy [1, 2]. The motivations and general objectives of the project are to develop and design an autonomous and low-cost electrical trans-pallet to deliver packages in a predefined warehouse, as shown in Figure 1.1. Typically, those kinds of robots that rely on environmental signs which have been purposely designed for them, such as artificial beacons, colored strips on the floor or markers of various shapes, cannot be considered fully autonomous robots. In our project, we would like to investigate methods and technologies allowing fully autonomous vehicles in warehouses, able to transport goods between different areas without relying on modifications of the environment to work properly.

In fact, this topic has received much attention both in scientific literature and in the startup ecosystem such as the Amazon Kiva system, the Knapp open shuttle, the Locus Robotics systems, the Swisslog CarryPick vehicle, and the GreyOrange Butler. For instance, the solutions from the Scallog system or the Hitachi Racrew are examples of commercial or quasi-commercial products for warehouse logistics automation. However, theses systems represent specialized robot solutions, which do not share almost anything with current, human-based solutions for warehouse logistics. The main objectives of AIRONE project are to overcome the drawbacks of the commercial robots such as:

- They are based on expensive hardware. We want to prove that a $10K$ euro robot can be used instead.
- They operate in unmanned space. Our project will prove that the robot has the capability of sharing the workspace with human-driven vehicles and walking workers.

Figure 1.1: The autonomous Trans-pallet for picking up and place pallets inside warehouses

- They use specialized and costly sensing technologies. We want to prove that several low cost distributed sensors can be used effectively.

The required tasks of the robot are receiving operation requests from a high-level warehouse management system, locating a pallet to pick up automatically, traveling to final destination. Finally, the robot reaches the put-down zone where manual displacements will be carried out. The put-down zone can be in an open space and/or in peculiar small corridors. The robot's architecture that has been designed to accomplish the required tasks is introduced in [3]. The software architecture presents different modules, these are integrated within a multi-agent based system, called Expert Tribe in a Hybrid Network Operating System (ETHNOS) [4, 5].

1.4 Thesis Objectives

Integrating Automated Guided Vehicle (AGV)s in industrial environments for transporting materials is becoming more widespread. Automation leads to cost and time reduction at installation time and during working time. However, this introduces many safety problems and the requirement for precision in localizing the AGV within the facility and other objects around it. Special attention should

be paid to the operation of loading and unloading pallets because incorrect positioning could lead to accidents. AGVs to engage pallets typically require that the location of the pallets be known a priori and that the pallets are accurately positioned. To perform pallet handling also when the pallet position is not precisely known in advance, e.g. when the loads are handled and placed by a human driver, only a few solutions have been presented.

A huge number of studies are concerned with pallet recognition, tracking, and reconstruction based on vision systems such. On the other hand, 2D laser range scans are extensively employed for the mobile robot Simultaneous Localization and Mapping (SLAM) problem and many approaches to it make use of techniques to optimally align two scans by using raw data [6] or extracted geometric features like lines [7, 8]. LRF have the advantage of producing reliable data with well known noise characteristic and are more accurate than stereo cameras at longer distances and are not influenced by illumination variability, have high angular resolution and good sampling rate, but 2D laser scans give only contour information about objects. For this reason some sensory systems include both cameras and one or more laser sensors [9]. Here, we focus the attention on the possibility of finding and localizing the pallets, by the only means of a 2D LRF under the assumption that the robot moves on a plane. Anyway, this constraint can be removed if a 3D LRF is used.

The aim of this research is to develop a robust method which is capable of automatically detecting and recognizing all the pallets in the laser Field-of-View (FOV). This objective is achieved using machine learning techniques, especially CNNs. To achieve this goal, several intermediate phases have been accomplished:

1. Firstly, the data preparation phase has been used to convert the acquired laser scanner raw data into 2D image in order to generate the training dataset.

2. Secondly, the training and testing phase has been used to describe the structure of the proposed network, its layers, and finally the network evaluation.

3. Thirdly, the tracking phase has been executed which aims to detect and keep tracking all pallets to increase the confidence that the pallet is present.

4. Finally, the experimental tests have been implemented to assess the system performance, by considering two pallets in the environment instead of having only one pallet.

The following research questions should be addressed in order to achieve the above mentioned research objectives.

1. Among the existing laser-based pallet detection methods, which algorithms can be suitably applied or extended to recognize all the pallets?

2. In case of using machine learning techniques, what is the best algorithm that can be used to reach our goal?

3. What is the structure of the network that can be used to perform high accuracy?

4. What are the factors that influence the performance of the algorithm used in this research?

1.5 Structure of Dissertation

The rest of this dissertation is organized as follows. Chapter 2 summarizes the previous relevant systems for solving the problems of extraction, segmentation and estimation of the pallet. Moreover, it gives a brief review of the main contributions of this research. In Chapter 3, a comprehensive review of the literature related to objects detection based on CNNs will be provided. Then, it summarizes the most important concepts related to CNNs including the structure of networks and essential layers. Chapter 4 provides details on the design of our pallets detection system and its main phases. Furthermore, it gives a comprehensive overview of the main steps of each phase. In Chapter 5, the experiment setup, the data used, the results of the training, and the results of tracking task will be presented. Chapter 6 concludes our project and indicates the future work.

Chapter 2

State of the Art

Summary

The goal of this chapter is to pay attention to the problem of loading and unloading pallets automatically without manual intervention. Moreover, it summarizes the previous relevant systems for solving the problems of extraction, segmentation and estimation of the pallet. These systems have been grouped into two main categories, vision-based systems and LIDAR-based systems. Finally, it gives a brief review of the thesis contributions.

2.1 Problem Statement

Generally, the topic of the detection and localization of the pallet, in order to load the pallet automatically, has received much attention in the scientific literature. Several approaches have been presented for finding robust solutions based on vision and laser rangefinder systems. We here focus the attention on the possibility of finding and localizing the pallets, by the only means of a 2D LRF under the assumption that the robot moves on a plane.

The main objective of this thesis is to use a laser rangefinder located on a robotic trans-pallet to detect the presence of the pallets, possibly more than one, in the laser FOV. This objective is achieved using CNNs, especially Faster R-CNN. Then, the pose of the pallet with respect to the robot reference frame must be estimated. Once the position of the pallet relative to the trans-pallet is obtained, a trajectory has to be generated to pick up the pallet.

2.2 Related Work

The specific problem of developing an autonomous forklift that is able to pick up and place pallets is not new [10]. The same is true for pallet detection and localization, which pose interesting perception challenges due to their sparse structure. Many approaches have been presented for solving the problems of extraction, segmentation and estimation of the pallet based on vision and laser rangefinder systems.

2.2.1 Vision-Based Systems

Most of these approaches are based on visual features extracted by image camera data. The first vision-based system for pallet recognition and pose estimation was presented by [11]. In [12] an image segmentation method based on color and geometric characteristics of the pallet is used to successfully detect and localize the pallet. However, this approach requires a good illumination condition and camera calibration. The method proposed in [13] attempted to estimate pallet pose using structured light method which based on a combination of range camera and a video camera. The main problem is that the accuracy of the measuring method using structured light quickly decreases with distance. The same goal has been achieved using the artificial visual features placed on the pallets without relying on strict camera calibration and illumination problems [10], [14]. However, It is difficult to place fiducial markers in all fields of activity and the markers may be fuzzy or broken as time goes by.

A model-based vision algorithm without any fiducial markers or specific illumination support has been implemented in [15]. This algorithm is based on the identification of the central cavities of the pallets in order to identify two pallet slots and estimate their geometric center in calibrated images. However, that system requires high prior knowledge of the pallet pose. A retrofitted autonomous forklift with the capability of stacking racks and picking up pallets placed with limited uncertainty was presented in [16]. The method is based on detection of specific reference lines for concurrent camera calibration and identification, and allows stacking of well-illuminated racks and localization of pallets in front of the vehicle and close to it. [17] described a more complex scheme which is on the basis of hierarchical visual features like regions, lines and corners using both raw and template-based detection. In [18], the authors presented an autonomous pallet handing method based on the line structured light sensor, where the design of the line structured light sensor based on embedded image processing board that contains a Field Programmable Gate Array (FPGA) and a Digital Signal Processing (DSP). They solved the problem that Hessian matrix decomposition based light stripe center extraction cannot run in real time. Then, they identified and local-

ized the pallet using the geometry structure of it based on model match method, and use position based visual servoing method driving the vehicle approach the pallet.

[19] presented a solution for automatic pallet detection by combining stereo reconstruction and object detection from monocular images. Moreover, the improvements and extensions for a stereo-camera sensor system that is responsible for autonomous load handling was presented, by the same authors, in [20]. The newly added functions tackle unloading operations using scene understanding obtained from the stereo disparity map. The paper from [21] discussed a method on how to identify a pallet using color segmentation in real time. The aim of the paper was to describe a complete recognition process in a robotic industrial application. A comparison between two common 3D camera technologies, the Photonic Mixer Device (PMD) and the Stereo Vision technique, was presented in [22]. The authors conclude that the PMD system had a greater accuracy than a stereo camera system. The paper [23] presented a solution for load detection and de-palletizing using a PMD camera. Moreover, other vision-based methods for pallet recognition and pose estimation have been presented in [24–29].

2.2.2 LIDAR-Based Systems

Typically, 2D LRF are extensively employed for the mobile robot SLAM problem. Recently, there are some effective methods to detect and localize pallets based on LRFs. In contract to camera based systems, these approaches do not suffer from distortion and illumination problems or object scaling problems which can result in false detection or misdetection of significant features. In general, the early work by Hebert et al. [30] describes techniques for scene segmentation, object detection, and object recognition with an outdoor robot using laser scan data. In [31], they authors presented a method for detecting and classifying the faces comprising 3D objects based on range data. A model-driven technique that leverages prior knowledge of object surface geometry to jointly classify and estimate surface structure was proposed in [32].

The paper from [33] used the acquired data from LRFs to detect and localize the pallet but could not deal with matching problems in the case of multiple targets. [34] used a fast linear program for segment detection, applied to pre-filtered points selected by man gestures on a Personal Digital Assistant (PDA) showing an image from a camera mounted on the forklift. The pallet is identified by the classification of detected segments belonging to its front face and the position of it is then computed. In [35], the authors presented two laser scanner based approaches which are independent of lighting conditions. The first approach uses pallets modified with reflectors to calculate the position and orientation of a pallet whereas the second approach uses only geometrical characteristics, since it is

a big project to place reflector marks in all pallets. Iterative Closest Point (ICP) algorithm has been used to match the laser point with the pallet model. The ICP algorithm is point-to-feature matching method. The main drawbacks of the ICP algorithm is that it needs the pallet approximately position, otherwise the interactive calculation is very complex.

In order to overcome the drawbacks of ICP algorithm, the paper from [36] presents a feature-to-feature matching method of the pallet based on the laser data, detecting the line segment, and then matching with the geometric model of the pallet. Moreover, other methods for 2D segmentation, feature detection, fitting, and matching have been presented in [37, 38]. A combined double-sensor system, laser and camera, for solving problem of identifying and localizing a pallet with large uncertainty prior was presented in [39]. [14, 40] integrated the vision output with odometry and realized smooth and non-stop transition from glob navigation to visual servoing. Typically, the maximum range of the visual detection methods between the vehicle and the pallets is $4m$ [41]. Some researchers, including multiple-view laser scanners, provided longer distances for the forklift configuration space, specifically $6m$ [34, 42].

2.3 Thesis Contributions

The main contributions of our proposed system for detecting and tracking the pallets can be summarized as follows:

- To the best of our knowledge, this is the first system to perform pallets detection and recognition based on deep learning (in particular, CNNs).
- Our proposed system has the capability of detecting and tracking more than one pallet.
- Our approach does not require a prior the approximate position of the pallet or even any modifications to the pallet. It can detect the pallet in variable positions.
- This method is applicable for detecting other objects and can be applied without major changes.

Chapter 3

Convolutional Neural Networks for Object Detection

Summary

The main aim of this chapter is to provide a comprehensive review of the literature related to objects detection based on CNNs. It gives a brief review of what is CNN. Then, it summarizes the most important concepts related to CNNs including the structure of networks and essential layers. Eventually, it briefly introduces the R-CNN family and the main difference between them.

3.1 Artificial Neural Networks

Even though computers are designed by and for humans, it is clear that the concept of a computer is very different from a human brain. The human brain is a complex and non-linear system, and on top of that its way of processing information is highly parallelized. It is based on structural components known as neurons, which are all designed to perform certain types of computations. It can be applied to a huge amount of recognition tasks, and usually performs these within 100 to 200 ms. Tasks of this kind are still very difficult to process, and just a few years ago, performing these computations on a CPU could take days [43].

Inspired by this amazing system, in order to make computers more suitable for these kinds of task a new way of handling these problems arose. It is called an Artificial Neural Network (ANN). An ANN is a model based on a potentially massive interconnected network of processing units, suitably called neurons. In order for the network and its neurons to know how to handle incoming information, the model has to acquire knowledge. This is done through a learning process. The connections between the neurons in the network are represented by weights, and

these weights store the knowledge learned by the model. This kind of structure results in high generalization, and the fact that the way the neurons handle data can be non-linear is beneficial for a whole range of different applications. This opens up completely new approaches for input-output mapping and enables the creation of highly adaptive models for computation [43]. The learning process itself generally becomes a case of what is called supervised learning, which is described in the next segment.

3.2 What is CNN?

Convolutional Neural Network (CNN, or ConvNet) is a type of artificial neural network inspired by biological processes [44]. In machine learning, it is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. It can be seen as a variant of multilayer perceptrons (MLP). In computer vision, a traditional MLP connects each hidden neuron with every pixel in the input image trying to find global patterns. However, such a connectivity is not efficient because pixels distant to each other are often less correlated. The found patterns are thus less discriminative to be fed to a classifier. In addition, due to this dense connectivity, the size of parameters grows largely as the size of an input image increases, resulting in substantial increases in both computational complexity and memory space usage.

However, these problems can be alleviated in CNNs. A hidden neuron in CNNs only connects to a local patch in the input image. This type of sparse connectivity is more effective to discover local patterns and these local patterns learned from one part of an image are also applicable to other parts of the image.

CNNs have been widely used for visual based classification applications. In recent years, a series of R-CNN methods are proposed to apply CNNs on object detection tasks [45–47]. In [45], the original version of R-CNN, R-CNN takes full image and object proposals as input. The regional object proposals could come from a variety of methods and in their work they use Selective Search [48]. Each proposed region is then cropped from the original image and wrapped to a unified 227×227 pixel size. A 4096-dimensional feature vector is extracted by forward propagating the subtracted region through fine-tuned CNN with *five convolutional layers* and *two fully connected layers*. With the feature vectors, a set of class-specific linear *Support Vector Machine (SVM)s* are trained for classifications.

R-CNN achieves excellent object detection accuracy, however, it has notable drawbacks. First, training and testing has multiple stages including fine-tuning CNN with *Softmax* loss, training SVMs and learning bounding-box regressors. Secondly, the CNN part is slow because it performs forward pass for each object

proposal without sharing computation. To address the speed problem, Spatial Pyramid Pooling network (SPPnet) [49] and Fast R-CNN [47] are proposed. Both methods compute one single convolutional feature map for the entire input image and do the cropping on the feature map instead of on the original image and then extract feature vectors for each region. For feature extraction, SPPnet pools the feature maps into multiple sizes and concatenate them as a spatial pyramid [50], while Fast R-CNN only use single scale of the feature maps. The feature sharing of SPPnet accelerates R-CNN by 10 to 100x in testing and 3x in training. However it still has the same multiple-stage pipeline as R-CNN. In Fast R-CNN Girshick propose a new type of layer, region of interest (RoI) pooling layer, to connect the gap between feature maps and classifiers. With this layer, they build an *semi* end-to-end training framework which only rely on full image input and object proposals.

All the mentioned methods rely on external object proposal input. In [46], the authors proposed proposal-free framework called Faster R-CNN. In Faster R-CNN, they use a Region Proposal Network (RPN), which slides over the last convolutional feature maps to generate bounding-box proposals in different scales and ratio aspects. These proposals are then fed back to Fast R-CNN as input. Another proposal-free work *You Only Look Once* is proposed in [51]. This network uses features from the entire image to predict object bounding box. Instead of sliding windows on the last convolutional feature maps, this network connects the feature map output to an 4096-dimensional followed by another full-connected $7 \times 7 \times 24$ tensor. The tensor is a 7×7 mapping of the input image. Each grid of the tensor is a 24-dimensional vector which encodes bounding boxes and class probabilities of the object whose center falls into this grid on the origin image. The YOLO network is 100 to 500x faster than Fast R-CNN based methods, though with less than 8% *mean Average Precision* (mAP) drop on VOC 2012 test set [52].

Some other specific R-CNN variants are also proposed to solve different problems. The paper from [53] presents a R-CNN based networks with triple loss functions combined for the task of keypoints (as representation pose) prediction and action classification of people. It also adapt R-CNN to use more than one region, but also contextual subregions for human detection and action classification called R*CNN [54]. In [55], the authors proposed DeepID-Net with deformation constrained pooling layer, which models the deformation of object parts with geometric constraint and penalty. Furthermore, a broad survey of the recent advances in CNNs and its applications in computer vision, speech and natural language processing have been presented in [56].

On the other hands, in general, there are some effective laser-based methods for object detection, estimation and tracking using machine learning approaches [57–60]. A multi-modal system for detecting, tracking and classifying objects in

outdoor environment was presented in [61].

3.3 Network Structures and Essential Layers

In this section, some important concepts related to general CNNs including the structure of networks and essential layers will be covered.

3.3.1 CNN Architectures

In general, the architecture of CNN can be decomposed into two stages, which are hierarchical feature extraction stage and classification stage. A typical architecture of CNN is shown in Figure 3.1. An input image is convolved by a set of trainable filters (kernels) each with a nonlinear mapping (e.g. ReLU [62]) to produce so-called *feature maps*. Each feature map containing special features is then partitioned into equal-sized, non-overlapping regions and the maximum (or average) of each region is passed to the next layer (sub-sampling layer), resulting in resolution-reduced feature maps with depth unchanged. This operation allows small translation to the input image, thus more robust features that are invariant to translations are more likely to be found [63]. These two steps, convolutions and subsampling, are alternated for two iterations in the CNN in Figure 3.1 and the resulting feature maps are fully connected with a MLP to perform classification. In some applications, the final fully connected layer that performs classification is replaced with other classifiers e.g. SVM. For example, the state-of-the-art object detector R-CNN [45] extracts high-level features from the penult final fully layer and feeds them to SVMs for classification [64].

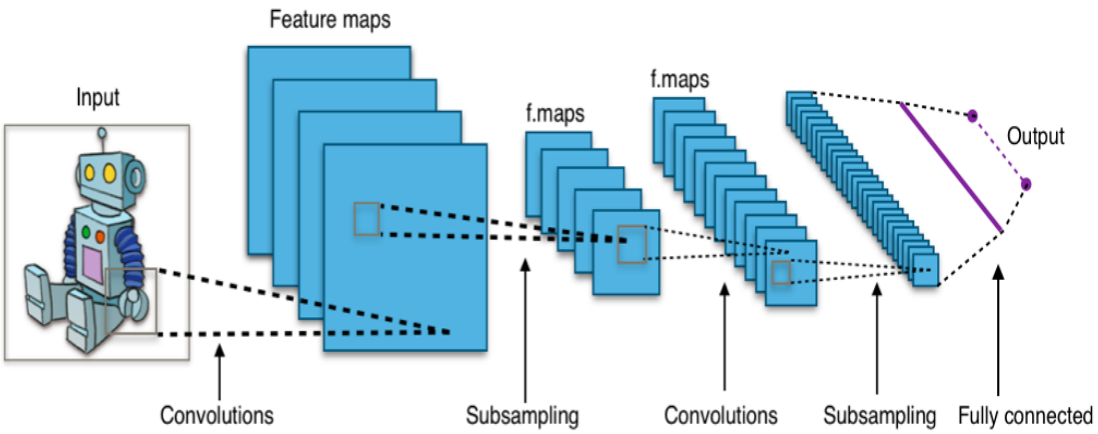


Figure 3.1: A typical architecture of CNN (Wikipedia).

3.3.2 Layers in CNNs

As mentioned in Section 3.3.1, CNNs are commonly made up of mainly three layer types: *convolutional layer, pooling layer (usually subsampling) and fully connected layer*. The explanations of these layers and the introduction of other auxiliary layers that are not shown in Figure 3.1 will be introduced.

- *Convolution Layer*

  The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of filters or kernels, which have a small receptive field, but extend through the full depth of the input volume or image. The convolution operation replicates a filter across the entire image field to get the response of each location and form a response feature map. Given multiple filters, the network will get a stack of features maps to form a new 3D volume.

  Officially, the convolution layer accepts a volume or image of size $W_1 \times H_1 \times D_1$ from previous layer as input data, where H_1, W_1, D_1 are image height, image width, and a number of channels (or depth) respectively. The layer defines K filters with the shape $F \times F \times D_1$ each, where F is the kernel size. The convolution of input volume and filters produces the output volume of size $W_2 \times H_2 \times K$, where the new volume's W_2 and H_2 are dependent on the filter size, stride and pad settings of the convolution operation. In general, the formula for calculating the output size, $W_2 \text{and} H_2$, for any given convolution layer is defined as:

  - width: $W_2 = \frac{(W_1 - F + 2P)}{S} + 1$
  - height: $H_2 = \frac{(H_1 - F + 2P)}{S} + 1$

  Where: K is the filter size, P is the padding, and S is the stride.

  For instance, Figure 3.2 illustrate a 2D version convolution where the $7 \times 7 \times 1$ input volume is convolved with one 3×3 filter. With 0 padding and 1 stride settings, it produces a $5 \times 5 \times 1$ output volume.

  - *Stride and Padding*
    There are two main parameters that must be tuned after choosing the filter size K in order to modify the behavior of each layer. These two parameters are the *stride* and the *padding*. *Stride*, S, controls how the filter convolves around the input volume. In the previous example,

[2] https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

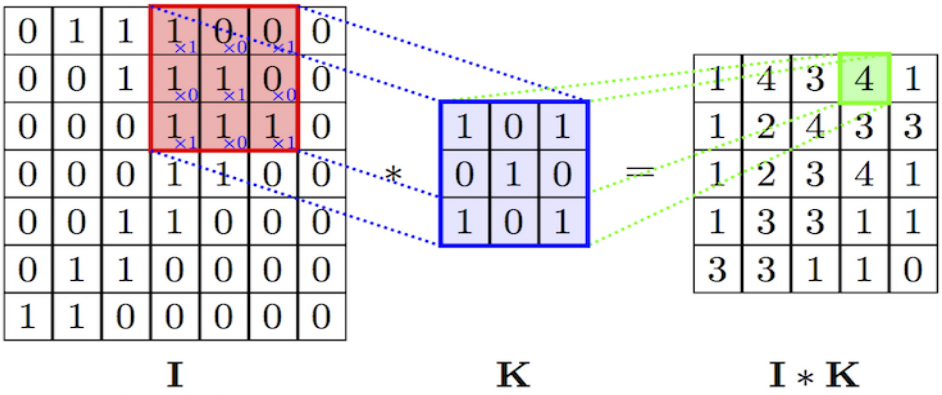


Figure 3.2: An example of convolution operation in 2D[2].

$S = 1$. This means that the filter convolves around the input volume by shifting one unit at a time. So, the amount by which the filter shifts is the stride. Moreover, as we keep applying convolution layers, the size of the volume will decrease faster than we would like. Therefore, in order to preserve as much information about the original input volume so that we can extract those low level features, the zero-padding must be applied. Let's say we want to apply the same convolution layer but we want the output volume to remain $7 \times 7 \times 1$, which is equal to the input size. To do this, we can apply a zero-padding of size 1 to that layer. Zero-padding pads the input volume with zeros around the border. The zero-padding is defined as:

$$P = \frac{K - 1}{2} \tag{3.1}$$

- *Pooling Layer*

  Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum (in case of *max-pooling*). The function of pooling layer is to reduce the spatial size of representation and hence reduce the amount of parameters and amount of computations in the network and also control overfitting. There are several non-linear functions to implement pooling such as *max-pooling, average-pooling and stochastic-pooling.* The pooling layer operates independently on every depth slice of the input and resizes it spatially. Pooling is an translation-invariance operation. The pooled image keeps the structural layout of the input image.

Formally a pooling layer accepts a volume of size $W_1 \times H_1 \times D_1$ as input and output a volume of size $W_2 \times H_2 \times D_1$. The output width W_2 and height H_2 are dependent on the kernel size, stride and pad settings, as shown in Figure 3.3. The produced output has dimensions:

- width: $W_2 = \frac{(W_1 - F)}{S} + 1$
- height: $H_2 = \frac{(H_1 - F)}{S} + 1$

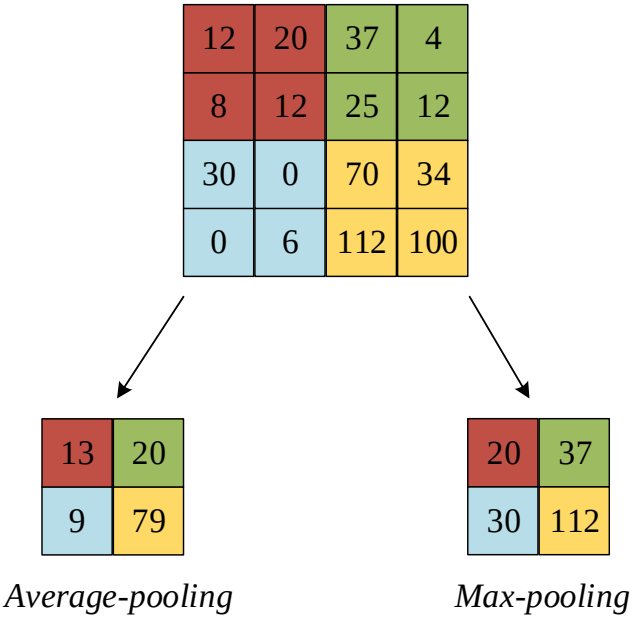


Figure 3.3: An example of pooling with a 2×2 filter and a stride of 2.

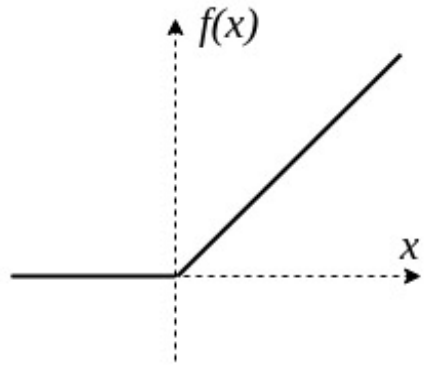


Figure 3.4: The ReLU activation function.

- *ReLU Layer*

  ReLU stands for Rectified Linear Units. ReLU is one of the most notable non-saturated activation functions, which can be used by neurons just like any other activation function. The ReLU activation function is defined as (Figure 3.4):

  $$f(x) = \max(0, x) \tag{3.2}$$

  ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. For that reason, after each convolution layer, it is convention to apply a ReLU layer immediately afterwards. The main reason that it is used is because of how efficiently it can be computed compared to more conventional activation functions like the *sigmoid function* $f(x) = |tanh(x)|$ and *hyperbolic tangent function* $f(x) = tanh(x)$, without making a significant difference to generalization accuracy. Many works have shown that ReLU works better than other activation functions empirically [65,66]. Moreover, the recently used activation functions in CNNs based on ReLU such as Leaky ReLU [65], Parametric ReLU [66], Randomized ReLU [67], and Exponential Linear Unit (ELU) [68] were introduced in [56].

- *Fully Connected Layer*

  Eventually, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have full connections to all neurons in the previous layer. It provides a form of dense connectivity and loses the structural layout of the input image. Fully connected layers are usually inserted after the last convolution layer to reduce the amount of features and creating vector-like representation.

- *Loss Layer*

  It is important to choose an appropriate loss function for a specific task. The loss layer specifies the learning process by comparing the output of the network with the true label (or target) and minimizing the cost. Generally, the loss is calculated by forward pass and the gradient of network parameters with respect to loss is calculated by the backpropagation. For multi-class classification problems, softmax classifier with loss is commonly used. Firstly, it takes multi-class scores as input, and uses softmax function to normalize the input and get a distribution-like output. Then, the loss

is computed by calculating the cross-entropy of the target class probability distribution and the estimated distribution. The softmax function is defined as:

$$y(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \tag{3.3}$$

Where:

- $0 \leq y(x)_i \leq 1$
- $\sum_{j=1}^{n} y(x)_j = 1$
- $i = 1, \ldots, n$ & n is the number of Classes.

The cross-entropy between the target distribution p and the estimation distribution q is given by

$$H(p, q) = \sum_i p_i \log q_i \tag{3.4}$$

The purpose of the softmax classification layer is simply to transform all the network activations in your final output layer to a series of values that can be interpreted as probabilities. The softmax function is also known as the normalized exponential function. The recently used loss layers (e.g. Hinge loss [69], L-Softmax loss [70], Contrastive loss [71,72], and Triplet loss [73]), were presented in [56].

3.3.3 Brief Introduction to R-CNN Variants

R-CNN stands for Region-based Convolutional Network. It takes full image and object proposals as input. The main R-CNN family includes R-CNN [45], Fast R-CNN [47] and Faster R-CNN [46]. Briefly speaking, R-CNN and Fast R-CNN rely on external region proposals generated by Selective Search [48]. R-CNN directly crop regions from the image and feeds them into the CNN for training and classification, while in Fast R-CNN the cropping takes place on the last convolutional feature maps. The advantage of Fast R-CNN is that it does not require external disk space for saving cropped regions, which achieves end-to-end training. Since the forward pass calculation is done only once for each image, the running time reduces a lot compared with original R-CNN. On the other hand, Faster R-CNN introduced RPN, which slides over the last convolutional feature maps to generate bounding-box proposals in different scales and ratio aspects. The RPN is a two-class classification which evaluate the *objectness* of the candidate boxes. The boxes with high *objectness* are selected as region proposals and then fed into the Fast R-CNN for finer object classification. Faster R-CNN achieves fully end-to-end training without relying on external proposals.

Chapter 4

System's Architecture

Summary

This chapter provides details on the design of our pallets detection system. It gives an overview of the detection pipeline and its implementation phases. Moreover, it gives a comprehensive overview of the main steps of each phase in order to recognize the pallets and keep tracking them.

4.1 Pallets Detection Pipeline

The pallet detection pipeline designed for our project is depicted in Figure 4.1. It consists of three sequential phases. The data preparation phase is mainly used to convert the acquired laser scanner raw data into 2D image as discussed in Section 4.2. Then, the training and testing phase takes as input the created 2D images. This phase describes the structure of the proposed network, its layers, and finally the network evaluation. Once the network is fine-tuned and tested, the tracking phase must be executed which aims to detect and keep tracking all pallets.

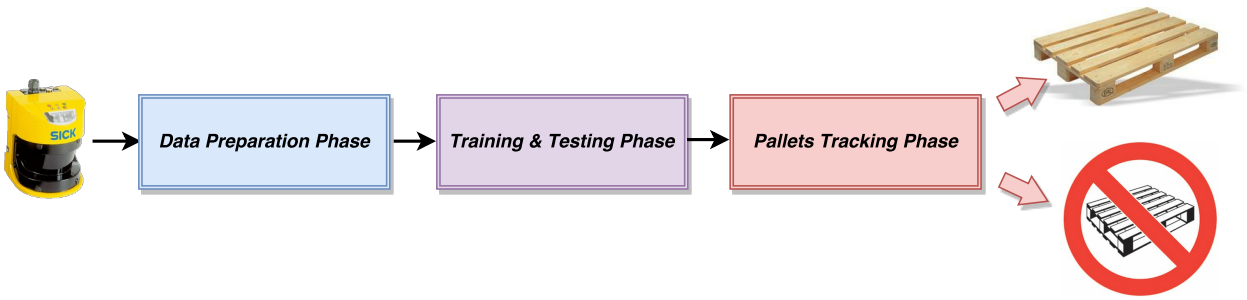


Figure 4.1: The proposed pallet detection pipeline.

4.2 Data Preparation Phase

The goal of this section is to give an overview of the main steps for creating an image from the laser scan ranges in order to use it with the proposed neural network. Moreover, it introduces how to extract the ROIs of the objects in the image and how to generate an artificial data from the original data.

4.2.1 Laser Data Acquisition

As mentioned before in Section 5.1.2, the laser that has been employed for the experiments is a SICK 2D laser scanner S3000 to recognize pallets at ground plane. Range data provided by laser sensor is typically in polar coordinates and can be denoted as

$$\{(r_1, \phi_1), \ldots, (r_i, \phi_i), \ldots, (r_N, \phi_N) | i = 1, \ldots, N\} \tag{4.1}$$

where r_i is the measured distance of an obstacle to the laser in the direction of ϕ_i (e.g. each distance r_i has an angle ϕ_i), as shown in Figure 4.2. The SICK divides its maximum angular range into constant steps and proceeds on each step a distance measurement by a laser beam (see Figure 4.2).

In our work according to the laser specifications, one scan is available as an array of 761 readings in polar coordinates captured relatively to the laser location. Then, the polar coordinates r and ϕ can be converted to the Cartesian coordinates x and y by using the trigonometric functions *sine* and *cosine*:

$$x = rcos(\phi) \tag{4.2}$$
$$y = rsin(\phi) \tag{4.3}$$

4.2.2 Image Creation

Upon receiving the scan, the distances are converted to XY points, relatively to the laser reference in which the X-axes represent the horizontal distance and the Y-axes constitute the vertical distance of the measured point (or the relative separation). The next step after converting the polar measurements, laser scans, of the laser into Cartesian coordinates is to generate RGB (or Grayscale) 2D images of the scene. Therefore to obtain these images, in a simple way, the X-value of each point provides information on their depth. On the other hand, the

[2] https://www.sick.com/media/dox/3/63/863/Operating_instructions_S3000_Safety_Laser_scanner_en_IM0011863.PDF

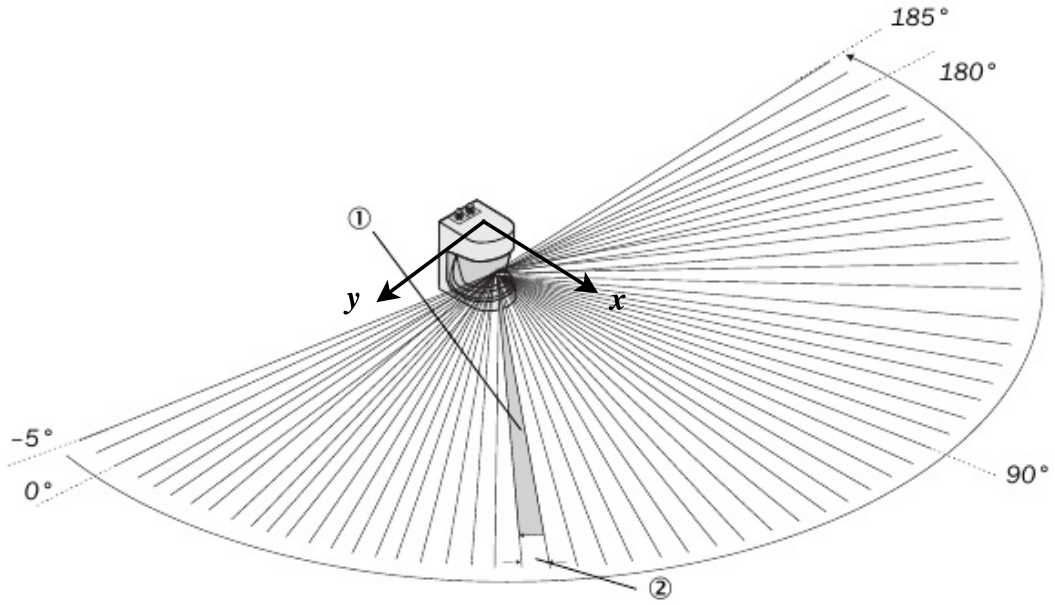


Figure 4.2: 190° as a Field-of-View (FOV) of SICK 3000 with angular resolution of 0.25° or 0.5° ①. As a result resolutions between $30mm$ and $150mm$ can be achieved ②. The first beam of a scan starts at $-5°$ [2].

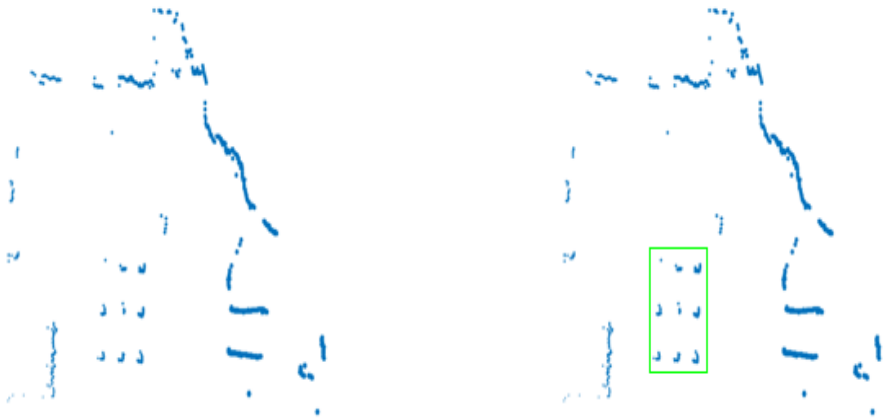
Y-value offers information on the separation of some points. After this point, we have 2D image which represents the laser scans at this moment. These images will be used by the CNN for object detection and tracking. Note that, the laser origin is at the center, whereas the image origin should always be on the top left corner.

4.2.3 ROI Extraction for Training

For training the R-CNN family, the bounding boxes, or ROIs, of the objects in the images should be defined in order to train the classifier. The ROI has the form of $[x_{min}, y_{min}, width, height]$, which specifies the current position of the bounding box in the image, as shown in Figure 4.3. In our experiments, the bounding boxes of the pallets in the images were manually generated using *Training Image Labeler App*. This App provides an easy way to specify ROIs in images by defining the locations of objects.

4.2.4 Data Augmentation

For training the network, generating artificial data has a number of advantages. Firstly, it is the most common and significant method to reduce over-fitting on



(a) Typical scan of Sick S3000. (b) Same scan after defining the ROI (pallet)

Figure 4.3: An example of image creation and ROI extraction

image data [74, 75]. Secondly, establishing and generating a real world dataset is time-consuming and very expensive, while it is quite easy to collect and generate raw (artificial) data recordings. Thirdly, it is highly time-consuming for humans to label the data for *ground truth* generation. On the other hands by using artificial data, the labels can be generated automatically. Besides the ability to generate arbitrary amounts of data, artificial data generation also allows us to control the class distribution [76].

In our implementation, for training the network, the original images are used for generating the artificial data by simply rotating each image by 90° and −90°. This increases the size of our training set to $3N$ instead of N, where N is the number of training dataset. Another method, that can be used in order to reduce the over-fitting, is so-called *dropout*. It randomly sets input elements to zero with a probability of 0.5 [77] and it only performs during the network training.

4.2.5 The Algorithm of Data Preparation Phase

The data preparation process can be summarized as a pseudo code, as illustrated in the Algorithm 1. This structure is typical for image creation and, of course, for on-line object detection. The first step is to acquire data from the laser and convert it into 2D image. For the network training, ROIs must be defined and the artificial data must be generated from the original image.

Algorithm 1 On-line Image Creation Algorithm

```
1: function READFRAME(ImSize)                    ▷ Where ImSize - image size
2:     Subscribe to the topic /scan
3:     Receive data from the subscriber as a ROS message
4:     Read laser scan ranges in Cartesian coordinates
5:     Convert the XY scan point into 2D image
6:     if Training Phase then
7:         Define ROIs in the image
8:         Generate an artificial data by rotating the image by 90°& − 90°
9:     else
10:        Break
11:    end if
12: end function
```

4.3 Training and Testing Phase

The proposed pallet detection method that is used in this work based on Fast R-CNN which is followed by CNN. For each network implementation, the training and testing phase are executed independently and sequentially.

4.3.1 Training and Testing the Faster R-CNN Detector

Generally, huge amounts of data is required for the training phase. The collection and preparation of this training dataset is quite time-consuming process for engineers. Therefore, artificial data is generated from the original dataset (see Section 4.2.4). For the classification task, the structure of the network designed for pallets detection is mainly based on Faster R-CNN, which provides state-of-art object detection performance (see Section 3.3.3). The main objectives of Faster R-CNN is to detect the ROIs in the images. Then, the last stage of the network is extended with CNN, a supervised network, for pallets tracking and bounding boxes (ROIs) classifications. The main reason that Faster R-CNN is followed by CNN is that the network is learning from patterns because the 2D laser gives only contour information about objects, as mentioned in Chapter 5.

In our system, the Faster R-CNN detector is composed of two convolution layers and two fully connected layers. The first layer is the image (input) layer which defines the type and size of the input layer. ReLU layers are inserted after the first and second convolution layers, while the last one is placed after the first fully connected layer. The second ReLU layer is followed by max-pooling layer. The last fully connected layer is followed by the softmax classification layer which is the last layer. The CNN classifier is quite simple in structure. It consists of one convolution layer and one fully connected layer. The convolution layer is followed

by only one ReLU layer which is followed by a max-pooling layer. On the other hand, the fully connected layer is followed by softmax classification layer. Figure 4.4 illustrates the detailed layer structure of Faster R-CNN and CNN.

The two networks have been fine-tuned independently with adopting different tuning strategies. The main steps for training the Faster R-CNN can be summarized in:

1. *Load* the image dataset which contains the ROI label for the pallets. These dataset should be split into a training set for training the detector, and the rest as a test set for evaluating the detector. In our work, 70% of the data are selected for training, and the rest used for testing.

2. After that, *configure* training options. The next step after creating the network structure is to define training options such as *max epoch, batch size, and number of iterations*.

   - *One epoch* is equivalent to one single pass (forward and backward) through your entire training dataset.
   - While, *batch size* is equivalent to the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you will need.
   - *One iteration* describes the number of times a *batch* of data passed through the algorithm.

   For instance, if the training dataset is 2000, and the batch size is 500, then it will take 4 iterations to complete 1 epoch.

3. Then, *train* Faster R-CNN object detector. Once the training options are defined, the detector can be trained.

4. Finally, *evaluate* the trained detector. The first step for detector evaluation is to run the detector on the test set. After collecting the detection results, the Average Precision (AP) and mean Average Precision (mAP) metrics are used for evaluation. Where, the AP is related to the area under the Precision-Recall curve for a class. However, the mAP is defined as the mean of these average individual-class-precision. *Recall* is a ratio of true positive (TP) instances to the sum of true positives and false negatives (FN) in the detector, based on the ground truth, $Recall = \frac{TP}{TP+FN}$. While, *Precision* is a ratio of TP instances to all positive instances of objects in the detector, $Precision = \frac{TP}{TP+FP}$. Since the detector is learning from patterns, the possibility of having FP is too high. For that reason, the mAP is applied to the selected bounding boxes that have a high confidence score or have a bounding box overlap greater than a certain threshold.

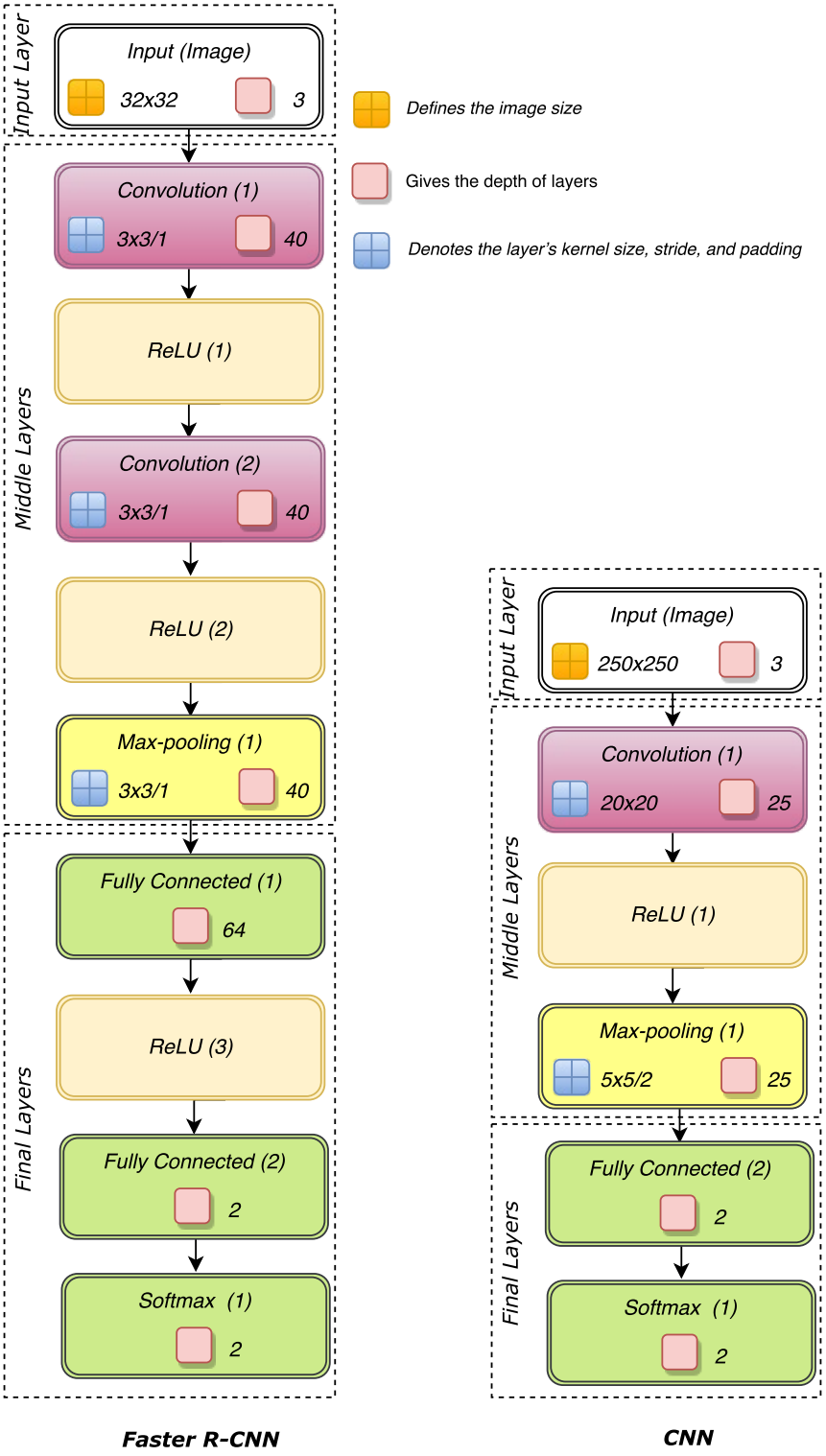


Figure 4.4: The structure of the network designed for pallets detection based on Faster R-CNN and CNN.

4.3.2 Training and Testing the CNN Classifier

Once the Faster R-CNN detector is fine-tuned and tested, the CNN classifier should be trained based on the strongest bounding boxes (ROIs) of the first detector. In a simple matter, this classifier takes the output images of the Faster R-CNN which have only the detected ROIs as a training dataset. Bearing this in mind, for the training CNN, the input image size is typically the size of the original images. After that, the main steps for training the CNN classifier is completely similar to the one mentioned before except the way of evaluating the performance of the network. Here, since it is a supervised task, the *k-fold cross-validation* is used for evaluation.

Cross-validation is one of the most significant tools, as it gives you an honest assessment of the true accuracy of the network [78]. In *k-fold cross-validation*, the original dataset is randomly partitioned into k equal sized subsets. Of the k subsets, a single subset is retained as the validation data for testing the model, and the remaining $k-1$ subsets are used as training data. Then, this process is repeated k times (the folds), with each of the k subsets used exactly once as the validation data. Finally, the results from the k folds can be averaged to produce a single estimation accuracy of the network.

4.3.3 The Algorithm of Training and Testing Phase

The algorithm for training and testing the acquired dataset based on Faster R-CNN and CNN is illustrated in the Algorithm 2 and Algorithm 3. Algorithm 2 shows the main steps for training the Faster R-CNN and testing the pretrained network and evaluate its performance. While Algorithm 3 illustrates how to perform training and testing the performance of CNN using k-fold cross-validation. Once the two networks are fine-tuned and tested, they can be used for performing the tracking phase.

Algorithm 2 Training and Testing Algorithm for Faster R-CNN

1: % Training & testing of Faster R-CNN detector
2: *Load* the image dataset
3: *Split* dataset into training set and test set
4: *Create* the Faster R-CNN layers
5: *Configure* training options, such as *max epoch, batch size, Image size*
6: *Train* Faster R-CNN object detector
7: *Evaluate* the trained detector ≡ Calculate the mAP

Algorithm 3 Training and Testing Algorithm for CNN

```
% Training & testing of CNN classifier
Load the dataset with only the strongest bounding boxes (ROIs).
Generate k-fold cross-validation indices
for i = 1 : k-fold do
    subset(i) ≡ testing set & (k − 1)subsets ≡ training set
    Create the CNN layers
    Configure training options
    Train CNN object detector
    Calculate the accuracy
end for
Calculate the average accuracy of the k-folds validation
```

4.4 Pallets Tracking Phase

In reality, the laser scanner is mounted on a moving robot. One of the main motivation goals is to detect all pallets within the laser scanner FOV, while the robot is moving, so that we can find a general solution for pallet recognition and position estimation. In order to achieve that goal, the tracking phase must be executed. Moreover, the most significant aim of this phase is to increase the confidence that there is a pallet in the predefined area. This tracking algorithm defers the decision for a track until sufficient confidence is available. Upon achieving sufficient confidence, the position of the pallet can be calculated for performing autonomous docking. The problem of pallet tracking can be divided into two parts:

1. Detecting objects (e.g. pallets) in each frame using the pretrained networks (see Section 4.3).

2. Associating the detections corresponding to the same object over time.

In our work, the association of detections to the same object (or track) is based only on motion, assuming that the robot is moving with constant velocity. Each track motion is estimated by a Kalman filter [79]. The filter is used for predicting the location of the track in each frame, and determine the likelihood of each detection being assigned to each track. For a given frame, some detections may be assigned to tracks, while other detections may remain unassigned. In addition, other tracks may remain unassigned. Then, the assigned tracks are updated using the corresponding detections, while the unassigned tracks are marked invisible. A new track begins using the unassigned detection. Each track keeps count of the number of unassigned consecutive frames and the recent average scores of the track. If the count exceeds a specified threshold or the average score less than a

certain threshold, the algorithm assumes that the object left the laser FOV or it is a false alarm, then, it deletes the track.

The tracking algorithm is composed by a set of steps which can be defined as functions. These functions are:

1. The *readFrame* function is used for acquiring the data from the laser scanner (frames) and then converting them into images, as illustrated in Algorithm 1.

2. The *initializeTracks* function is used to create array of tracks, where each track is a structure array representing a specific object. The aim of the structure is to preserve or keep the tracked object state. The state consists of information used for detections assignment, track termination, and display. The structure contains the following fields:

   - *id* refers to the ID of the track.
   - *color* refers to the track color for display the results.
   - *age* defines the number of frames since the track was initialized.
   - *totalVisibleCount* defines the total number of frames in the track where the object was detected.
   - *kalmanFilter* refers to the object Kalman filter which used for motion tracking. In our implementation, we track the center point of the object bounding box in the image.
   - *bboxes* represent a N-by-4 matrix of the bounding boxes of the object, where each row has a form of $[x_{min}, y_{min}, width, height]$ and N represents the total number of the bounding boxes belongs to the track.
   - *scores* represent a N-by-1 vector to record the confidence score from the pretrained detector.
   - *confidence* represents how confident we trust the track. It stores the maximum and the average detection scores in the past within a predefined time window.
   - *predPosition* stores the predicted bounding box in the next frame.

3. The *detectPallets* function returns the strongest *bounding boxes*, the *confidence scores*, and the *centroids* of the detected pallets. This function calls the previously trained Faster R-CNN detector and CNN classifier which performs filtering and non-maximum suppression on the raw output of the detector. The *centroids* represent the center point of the bounding boxes. It is a N-by-2 matrix with each row in the form of $[x, y]$.

4. The *predictNewLocationsOfPalletTracks* function uses the Kalman filter for predicting the centroid of each track in the current frame, and update its bounding box accordingly.

5. The *detectionToTrackAssignment* is used for assigning object detections in the current frame to existing tracks by minimizing cost. The cost is computed based on the overlap ratio between the predicted bounding box and the detected bounding box. The cost is minimum when the predicted bbox is perfectly aligned with the detected bbox (e.g. overlap ratio is one). The assignment problem is solved using the Hungarian algorithm, which minimizes the total cost [80].

6. The *updateAssignedTracks* function updates each assigned track with the corresponding detection. Moreover, it corrects the estimate of the object's location using the new detection. Consequently, it stabilizes the bounding box by taking the average of the size of recent k boxes on the track. It increases the age of and the total visibility of the track by 1. Finally, the function adjusts the confidence score of the track based on the previous confidence (detection) scores.

7. The *updateUnassignedTracks* function marks each unassigned track as invisible by setting the confidence score to 0 since we are not sure why it was not assigned to a track. It increases the age of the track by 1, and appends the predicted bounding box to the track.

8. The *deleteLostTracks* function deletes tracks that have been invisible for too many consecutive frames or have an average detection confidence score less than a predefined threshold. It also deletes recently created tracks that have been invisible for many frames overall. The purpose of this function is to recognize the false tracks which resulting from noisy detections because the 2D laser gives only contour information about objects.

9. The *createNewTracks* function creates new tracks from the unassigned detections. Assuming that any unassigned detection is a start of a new track.

10. The *displayTrackingResults* function is used for display purpose. It draws a colored bounding box and label ID for each track. It displays only the reliable tracks which have an average confidence score greater than a predefined threshold or their ages greater than half of the minimum length required for considering a track as a True Positive (TP).

There are some parameters must be defined and set to perform the tacking task using the mentioned algorithm:

- $ageThresh$ refers to the minimum length required for a track being true positive.
- $visThresh$ defines the minimum visibility value for a track being true positive.
- $MinConfidenceThresh$ is a threshold to determine if a track is true positive or false alarm by taking the recent average confidence up to $timeWindowSize$ frames.
- $timeWindowSize$ is a tuning parameter to specify the number of frames required to stabilize the confidence score of a track.

4.4.1 The Algorithm of Pallets Tracking Phase

The algorithm of pallets tracking can be considered as a function, *PalletTrackingUsingKalmanFilter*. This function is a collection of the previously mentioned functions, as illustrated in the Algorithm 4. These functions must be called each frame in order to perform the pallet tracking task.

Algorithm 4 Pallets Tracking Algorithm

```
1:  function PalletTrackingUsingKalmanFilter()
2:      Set the global parameters AgeThresh, MinConfidenceThresh, VisibiltyThresh,
                                  ImageSize, timeWindowSize, ...
3:      Load the pretrained designed networks
        % call initializeTracks function
4:      tracks = initializeTracks();
5:      for frame = 1 : NumOfFrames do
            % call readFrame function
6:          TestImage = readFrame();
            % call detectPallets function
7:          [bboxes, scores, centroids] = detectPallets();
            % call predictNewLocationsOfPalletTracks function
8:          predictNewLocationsOfPalletTracks();
            % call detectionToTrackAssignment function
9:          [assignments, unassignedTracks, unassignedDetections] =
                                                    detectionToTrackAssignment();
            % call updateAssignedTracks function
10:         updateAssignedTracks();
            % call updateUnassignedTracks function
11:         updateUnassignedTracks();
            % call deleteLostTracks function
12:         deleteLostTracks();
            % call createNewTracks function
13:         createNewTracks();
            % call displayTrackingResults function
14:         displayTrackingResults();
15:     end for
16: end function
```

Chapter 5

Experiments and Results

Summary

This chapter provides the detailed implementation of the proposed approach as well as the experimental results. It first gives an introduction to the experiment settings and the test environment. Then, it provides details on the collection of training data and testing data. Moreover, it explains how we implement the proposed approach, especially the network training and the evaluation methodology that is used to evaluate the pallet detection system. Finally, it analyses the experimental results particularly for the tracking task, by considering two different scenarios. The first scenario is performed by considering only one pallet, while the second one is performed by considering one more pallet in the environment.

5.1 Experimental Setup

5.1.1 EUR-Pallet

In this project, a standard Euro pallet will be manipulated. Euro pallet is the standard European pallet as specified by the European Pallet Association (EPAL). The size of the Euro pallets is $1200 \times 800mm$ with a height of $144mm$, as shown in Figure 5.1. In our research, we define as operating face of the pallet the one of narrow width. On that face there are two slots, each 227.5mm wide, as shown in Figure 5.1.

Following the standardization, most of the European industries switched over to use Euro-pallets with trucks, forklifts and high-rack warehouses optimized for their size. Using a standard size and shape pallet optimized for using in transformation tasks, reduces supply chain costs and increases the productivity. Nec-

[2] https://en.wikipedia.org/wiki/EUR-pallet

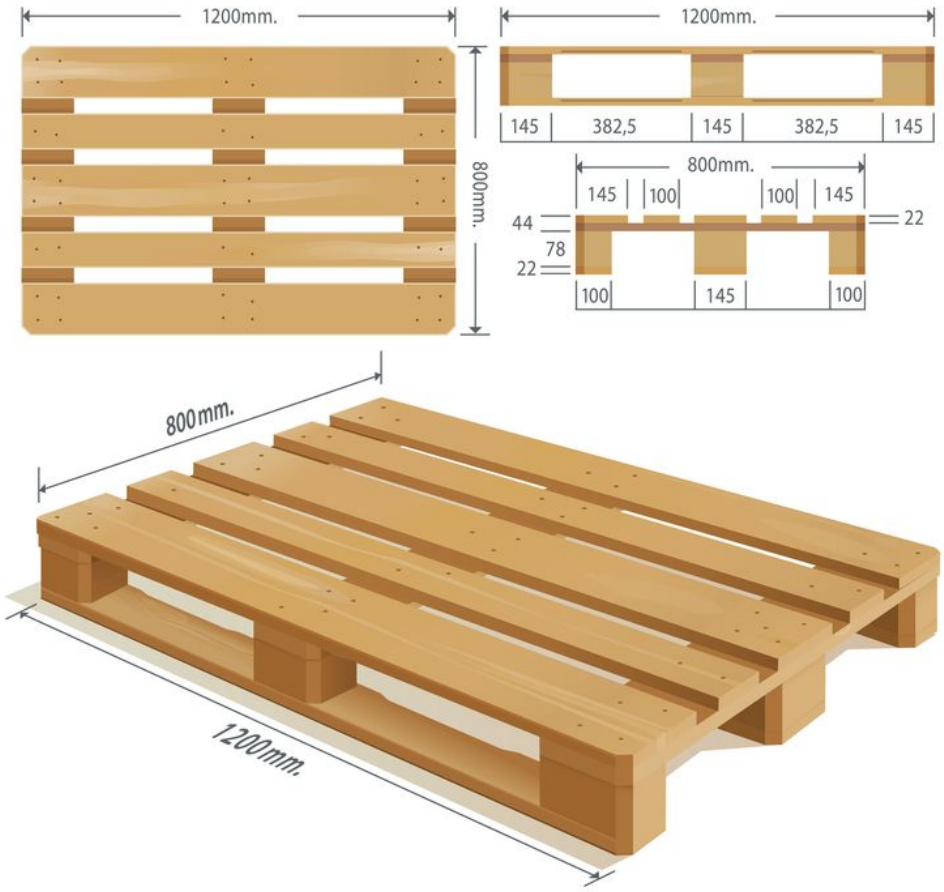


Figure 5.1: Geometric characteristic of Euro standard pallet [2].

essary industry adjustments required for the implementation of standard pallet size(s), and all relevant industries follow the standard nowadays. Due to this fact it is necessary to have pallet recognition algorithms which would a wide range of users.

5.1.2 Laser Scanner

In the context of mobile robotic applications, there are many ways of measuring distances, one of the most used approaches is Time-of-Flight (TOF) systems, which measures the delay until an emitted signal hits a surface and returns to the receiver, thus estimating the true distance from the sensor to the surface. This category of systems involves sensing devices such as a Laser Measurement System (LMS) or LIDAR, radars, TOF cameras, or sonar sensors, which send rays of light (for example laser) or sound (for example sonar) in the world, which

will then reflect and return to the sensor. Knowing the speed with which a ray propagates and using precise circuitry to measure the exact time when the ray was emitted and when the signal was returned, the distance can be estimated easily.

Figure 5.2: S3000 Professional laser scanner.

The experiments of detecting the pallets are done with the SICK 3000, S3000 Professional CMS, which is doing planar range scans with the aid of laser beams, as shown in Figure 5.2. The S3000 is mainly used as a safety unit and typically used to detect obstacles such as humans, and slow or stop the vehicle dependent upon obstacles distances to the vehicle. According to the product's manual[3], SICK 3000 measures its surroundings in two-dimensional polar coordinates. If a laser beam is incident on an object, the position is determined in the form of distance and direction. SICK 3000 has a range of 49m (20 m at 20% reflectivity), a resolution of 0.25 degree, and 16 Hz frequency with a empirical error of 0.003 meters. It has a maximum FOV of 190°, which is sufficient for performing our task. This produces high density 761 readings per frame.

[3] https://www.sick.com/ag/en/s3000-professional-cms-sensor-head-with-io-module/s30a-6011db/p/p31284

5.1.3 System Overview

Our system is composed of a SICK S3000 (see Section 5.1.2) connected to the PC through a RS422-USB converter. There are two different PCs are used in our experiments. The first PC is a Lenovo Z50-70 Laptop which is used for acquiring data from the laser scanner and performing the on-line pallets tracking. Its specifications are presented in Table 5.1. For training and testing the proposed networks, we used a high-performance computing PC which its specifications are presented in Table 5.2.

Table 5.1: The specifications of Lenovo Z50-70 Laptop

Name	Lenovo Z50-70
CPU	Intel® Core i5-4210U CPU & 1.70GHz 4
GPU	Nvidia Geforce
Memory	6 GB DRR3 RAM 1600 MHz
Storage	1000 GB 5400 RPM
OS	Ubuntu 16.04 LTS 64-bit

Table 5.2: The specifications of EMARO Lab PC

Name	EMAROLab PC
CPU	Intel(R) Core i7-4790 CPU & 3.60GHz
GPU	Nvidia Geforce GTX970
Memory	32 GB DDR3 RAM 4000 MHz
Storage	1000 GB 7200 RPM
OS	Ubuntu 14.04 LTS 64-bit

The experiments were mainly carried out in MATLAB and Robt Operating System (ROS) [81] on Ubuntu to verify our approach. The raw data is acquired from the laser scanner using ROS node, which is written in C++ language, so-called *s3000_laser_node* node. This node publishes *sensor_msgs/LaserScan* messages on the scan topic. The running ROS nodes can be displayed using the *rqt_graph*, as shown in Figure 5.3. *rqt_graph* provides a GUI plugin for visualizing the ROS computation graph[4]. To process that data, we write a node that subscribes messages on the scan topic. For the implementation of CNNs, this node is used for acquiring the training and testing dataset. In our experiment, the pallet recognition and tracking algorithms are written in MATLAB and implemented using Computer Vision System Toolbox. So, in order to perform on-line tracking

[4] http://wiki.ros.org/rqt_graph

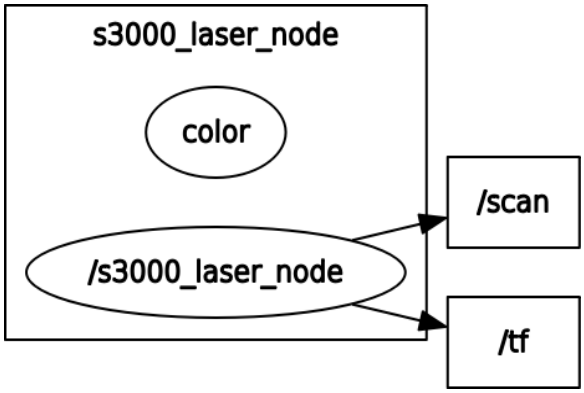


Figure 5.3: *rqt_graph* shows the node and their connections through topics

of the pallets, the interface between MATLAB and ROS should be carried out. It is carried out by the *Robotics System Toolbox (RST)*[5]. RST enables you to communicate with a ROS network, interactively explore robot capabilities, and visualize sensor data. You can develop, test, and verify your robotics algorithms and applications on ROS-enabled robots and robot simulators such as Gazebo and V-REP. Once the interface between MATLAB and ROS is established, you can read the laser scan ranges by subscribing to the laser scan topic.

5.2 Dataset

It is now time to have a look at dataset used to verify our approach. The dataset is collected in European Master on Advanced RObotics (EMARO) Lab at the University of Genoa[6]. The test area is a typical indoor environment with different obstacles, such as walls, robots, and offices, which helped us to acquire the proper data. Figure 5.4 shows the test environment. In our experiments, the total number of the captured data by the laser scanner where the pallet is present is 340 scenes (examples). Where, the scene is the acquired data by the laser scanner in each execution step.

The data association is performed considering these assumptions:

- The scene must be different at each time, in terms of the distance and angle of the laser scanner to the pallet, and the dynamic environment which may change.

[5] https://it.mathworks.com/hardware-support/robot-operating-system.html
[6] http:www.robotics.ingegneria.unige.it/

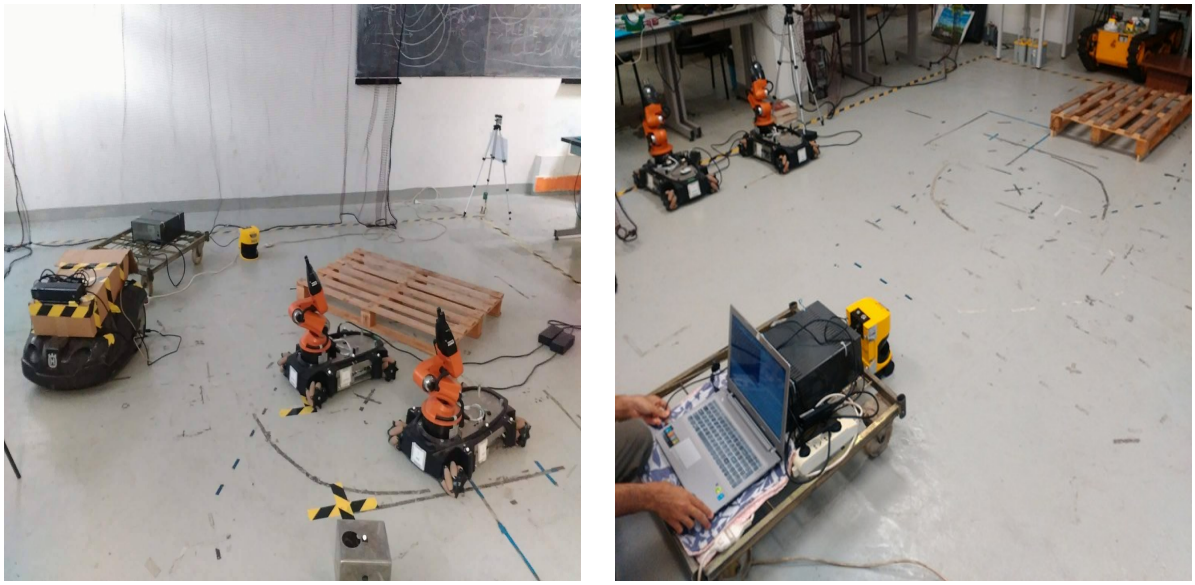


Figure 5.4: The test environment at EMARO Lab.

- We define as operating face of the pallet the one of narrow width. On that face there are two slots, each 227.5mm wide, as shown in Figure 5.1.
- Bearing this in mind, if there is an obstacle in front of the operating face of the pallet, this scene will not be considered as a pallet class. In other words, it will be enough just to see the front view, and no need to give the whole pallet as the model to be detected.

In order to avoid some problems presented in realistic situations, such as data association mistakes, road irregularities, vehicle vibrations, and so on, the data was collected under user supervision. This task is done by *rviz* node to increase the confidence that the pallet in the scene, as shown in Figure 5.5. The *rviz* node comes with a standard package *rviz* in ROS[7]. All the scans collected during the entire pallet detection process can be conveniently visualized and shown inside this interface. Finally, all the captured data must be converted into images as mentioned before in Section 4.2. Figure 5.6 presents some examples of our dataset which will be used for training and testing the proposed network.

The bounding boxes of the pallets in the images were manually generated using *Training Image Labeler App*. The 340 original images are used for generating the artificial data by simply rotating each image by 90° and −90°. So, the data for training and testing is totally 1020 examples.

[7] http://wiki.ros.org/rviz/Tutorials

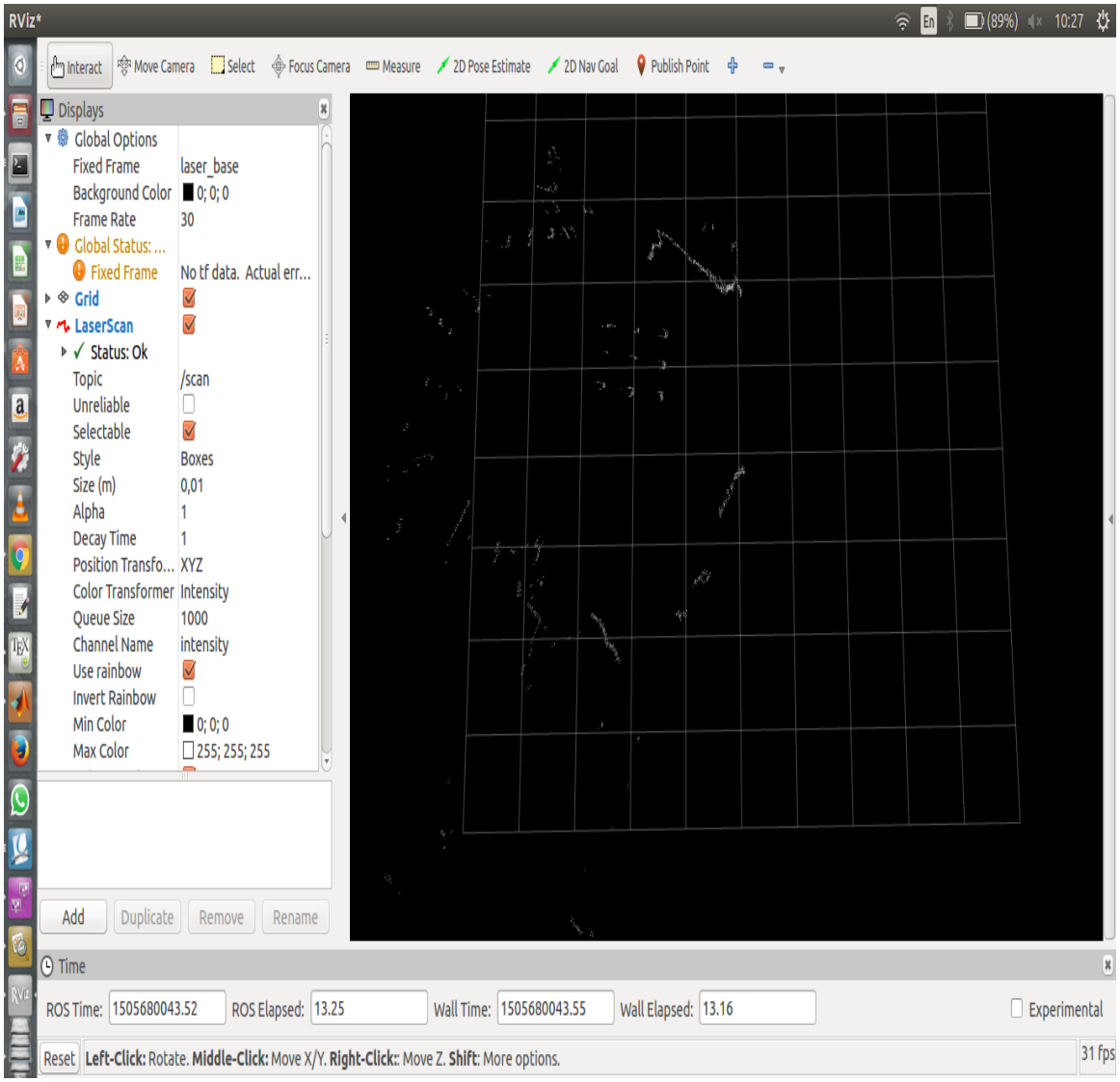


Figure 5.5: User interface of *rviz* node.

Figure 5.6: Examples from the dataset used for training and testing.

5.3 Training and Evaluation Process

Some data preprocessing is required before network training. The original images are resized to 250×250 images. The maximum detection range of the laser is adjusted to $6m$ which is compatible with the test area. The data is stored in a table. The first column contains the path to the images file. The remaining columns contain the ROI labels for pallets.

In our experiments, the proposed network architecture for the pallets detection and classification that we use is the same as the one introduced in Figure 4.4, with exactly the same parameters. We create the CNN layer by layer using *Neural Network Toolbox* functionality, starting with the *imageInputLayer* function, which defines the type and size of the input layer. For classification task using CNN, the input size is typically the size of the original training images (e.g. 250×250 image). For detection task using Faster R-CNN, the CNN needs to analyze smaller sections of the image, so the input size must be similar in size to the smallest object in the data set.

5.3.1 Faster R-CNN Detector

5.3.1.1 Training Process

As mentioned previously, the Faster R-CNN detector is composed of three main layers. They are input layer, middle layers, and final layers. The input layer consists of only the image input layer with a size of 32×32 and a depth of 3. While, the middle layers are made up of two convolution layers, two ReLU layers, and one max-pooling layer. The two convolution layers are created with 40 filters, each with a height and width of 3, a stride of 1 in the horizontal and vertical directions, and a padding of 1. Whereas, the max-pooling layer creates pooling regions of size $[3, 3]$ and a stride of 1. The final layers are composed by two fully connected layers, one ReLU layer, and finally softmax classification layer. The first fully connected layer is a fully connected layer with 64 output neurons. The output size of this layer is an array with a length of 64, which represents the most significant features in the image. The last fully connected layer must produce outputs that can be used to measure whether the input image belongs to one of the object classes or background. Table 5.3 shows all the intermediate layers and their parameters of the Faster R-CNN for the detection task.

To train the network, we split the collected data into training set and testing set, where the training set consists of 714 images and the validation set consists of 306 images. Because of the development of these models being based on GPU-computing for faster and more efficient training, the GPU played a very important role for the results. Stochastic Gradient Descent (SGD) algorithm is used to train

Table 5.3: All the intermediate layers and their parameters of the Faster R-CNN

Name	Type	Kernel size	Stride	Padding	Layer depth
ImLayer	Image Input	-	-	-	3
conv1	Convolutional	3	1	1	40
relu1	ReLU	-	-	-	-
conv2	Convolutional	3	1	1	40
relu2	ReLU	-	-	-	-
pool1	Max-pooling	3	1	0	40
fully1	Fully connected	-	-	-	-
relu3	ReLU	-	-	-	-
fully2	Fully connected	-	-	-	-
soft1	Softmax	-	-	-	-

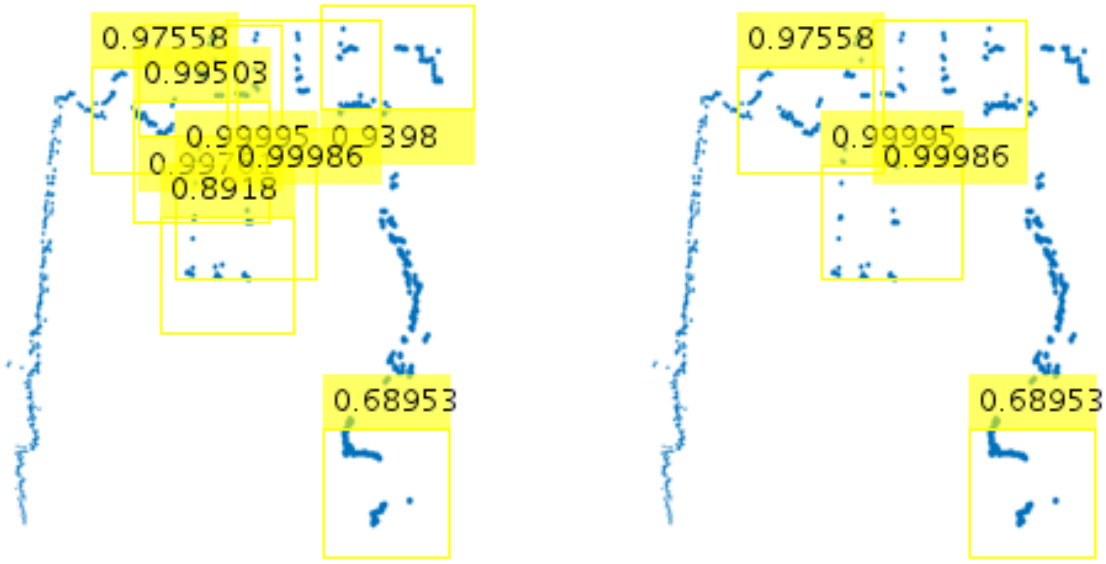
the network. The SGD algorithm updates the parameters (weights and biases) so as to minimize the error function by taking small steps in the direction of the negative gradient of the loss function [82]. To fine-tune the network, the initial learning rate is preferred to be small. We actually tried using $10^{-3}, 10^{-4}$, and 10^{-6} as the initial learning rate. We finally chose the initial learning rate of 10^{-6} which achieves faster convergence and high performance of detection. We trained the network for 20 epochs, which takes an average of 45 minutes on the EMARO Lab PC, and 6 hours in Lenovo Labtop.

There are two significant parameters must be fine-tuned in order to improve the detection task. They are *PositiveOverlapRange* and *NegativeOverlapRange*. The *PositiveOverlapRange* defines the bounding box overlap ratios for positive training samples, while the *NegativeOverlapRange* defines the bounding box overlap ratios for negative training samples. The best values for these parameters should be chosen by testing the trained detector on a validation set. We set *PositiveOverlapRange* to $[0.6, 1]$ and *NegativeOverlapRange* to $[0.1, 0.3]$. Bounding boxes are additionally filtered using non-maximum-suppression with threshold of 0.3. The reason for that is because the network is learning from patterns. Figure 5.7 displays the detected ROIs and their corresponding scores before and after suppression.

5.3.2 CNN Classifier

5.3.2.1 Training Process

As mentioned previously, the CNN classifier is composed of three main layers. They are input layer, middle layers, and final layers. The input layer consists of



(a) The detected ROIs before suppression.

(b) The detected ROIs after suppression.

Figure 5.7: The detected ROIs and detection scores before and after suppression.

only the image input layer with a size of 250 × 250 and a depth of 3 which is exactly the size of the original training images. While, the middle layers are made up of one convolution layer, one ReLU layer, and one max-pooling layer. The convolution layer is created with 25 filters, each with a height and width of 20, a stride of 1, and a padding of 1. Whereas, the max-pooling layer creates pooling regions of size $[5, 5]$ and a stride of 2. The final layers are composed by one fully connected layer, and finally softmax classification layer. The fully connected layer produces outputs that can be used to measure whether the input image belongs to the pallet object class or background (non-pallet). Table 5.4 shows all the intermediate layers and their parameters of the CNN for the classification task.

As mentioned previously in Section 4.3.2, the CNN classifier takes the output images of the Faster R-CNN with only the detected ROIs as a training dataset. Figure 5.7b shows the final output of the Faster R-CNN. It has 4 ROIs each one considered as an image with the same size as the original images, as shown in Figure 5.8. In this case, the dataset consists of 950 images, where the positive class (C_1), where the pallet is present, consists of 450 images and the negative class (C_0) consists of 500 images. Furthermore, SGD algorithm is used to train the CNN network. For fine-tuning the network, we actually tried different values

Table 5.4: All the intermediate layers and their parameters of the CNN

Name	Type	Kernel size	Stride	Padding	Layer depth
ImLayer	Image Input	-	-	-	3
conv1	Convolutional	20	1	1	25
relu1	ReLU	-	-	-	-
pool1	Max-pooling	5	2	0	25
fully2	Fully connected	-	-	-	-
soft1	Softmax	-	-	-	-

of the initial learning rate α. We finally chose the initial learning rate of 0.03 which achieves faster convergence and high performance ,as illustrated in Table 5.5. We trained the network with two different values of epochs (*e.g. MaxEpochs = 5 or 10*) with a mini-batch size of 50.

5.3.2.2 Evaluation Process

Since it is a supervised task, the *k-fold cross-validation* is used for evaluation. We trained the CNN with different values of k-fold K. We set K to 5 and 10. The results of the both cases are summarized in Table 5.5. We can observe from Table 5.5 that our classifier has an average success rate of 99.58% in the case of $\alpha = 0.03$ with $K = 10$. In general, the classifier achieves a high accuracy within short a time of training.

Table 5.5: The testing phase of CNN with different values of α and K.

α	k-fold K	*MaxEpochs*	*Average Accuracy (%)*	Training time (sec)
0.03	5	5	99.26	761
0.03	10	5	**99.58**	**1551**
0.003	5	5	95.47	770
0.003	10	5	95.68	1553

5.4 Pallets Tracking Process

According to the fine-tuning results in Table 5.5, the last phase of our system can be tested. Bearing this in mind, pallets are detected from the environment using Faster R-CNN, then tracked with a simple *Kalman* filter. The objective of the tracking experiment is to increase the confidence that the pallet is present.

(a) ROI *No. 1*

(b) ROI *No. 1*

(c) ROI *No. 3*

(d) ROI *No. 4*

Figure 5.8: The 4 detected ROIs from Faster R-CNN, Figure 5.7b, each one considered an input image to CNN, with a size of 250 × 250, for training the network.

In order to achieve our goal, we performed 4 experiments in terms of the distance and angle of the laser scanner to the pallet, and the dynamic environment which may change. These experiments were performed while the robot was moving towards the pallet. In a simple matter, we acquired data from the laser scanner while the robot was moving. In total for each trajectory, we acquired 10 scenes (frames). Then, between each two sequential frames, we generated an artificial data from the original one. So, the acquired data for each trajectory is 40 frames.

Due to availability, the experiments were carried out by considering only one pallet in the environment. Consequently to check the robustness of our algorithm, an artificial data was generated by considering one more pallet in the environment. The following parameters were used for performing the tracking task:

- $ageThresh = 10$
- $timeWindowSize = 6$
- $MinConfidenceThresh = 0.35$
- $visThresh = 0.6$

5.4.1 Experiment Results Considering only One Pallet

Based on the predefined parameters and the fine-tuned networks, the experiments were implemented for each track using *PalletTrackingUsingKalmanFilter* function, as illustrated in Algorithm 4. It is observed that the proposed algorithm be able to detect and track the positive track (pallet) among other negatives tracks. Figure 5.9 and Figure 5.10 show the results of the pallet tracking task, considering only the first 24 frames of the acquired frames. Each frame, in the Figures, displays the detected ROIs (see Frame No. 1, the box in yellow color) of the designed network for detecting the pallets. As observed in the Frame No. 6, the tracking algorithm displays the first track which has an average confidence of 0.9999 (T1).

As we can see in Frame No. 8, another track is considered (T2). However, by time passing, its average confidence became less than $MinConfidenceThresh = 0.35$. Consequently, the algorithm decided to delete it. This means that it was a false alarm. By Frame No. 17, we can infer that track T1 represents positive track (pallet) with an average confidence of 0.9857, and consequently, we can estimate the position of the pallet in order to perform automatic docking to the pallet. It is observed that the total time for processing the forty frames is 11 sec. This means that the on-line tracking can be carried out without any delay.

5.4.2 Experiment Results Considering Two Pallets

Based on the same conditions and the fine-tuned networks, the experiments were implemented by considering two pallets. It is observed that the algorithm is able to recognize both and keep tracking them, as shown in Figure 5.11. Moreover, it is observed that the total time for processing the forty frames is 12 sec.

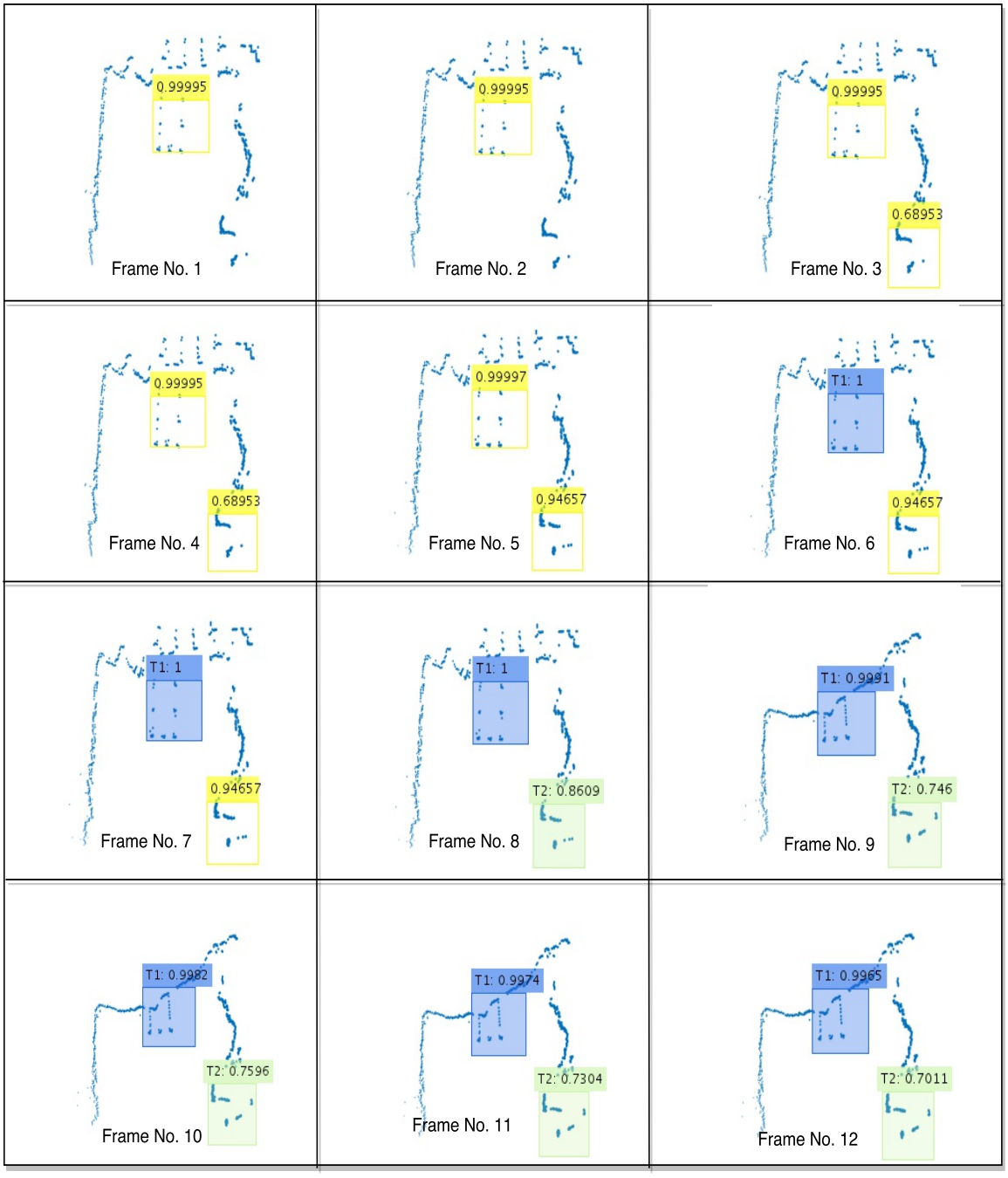


Figure 5.9: The detected ROIs for each frame with displaying only the reliable tracks (Frames No. $1 - 12$).

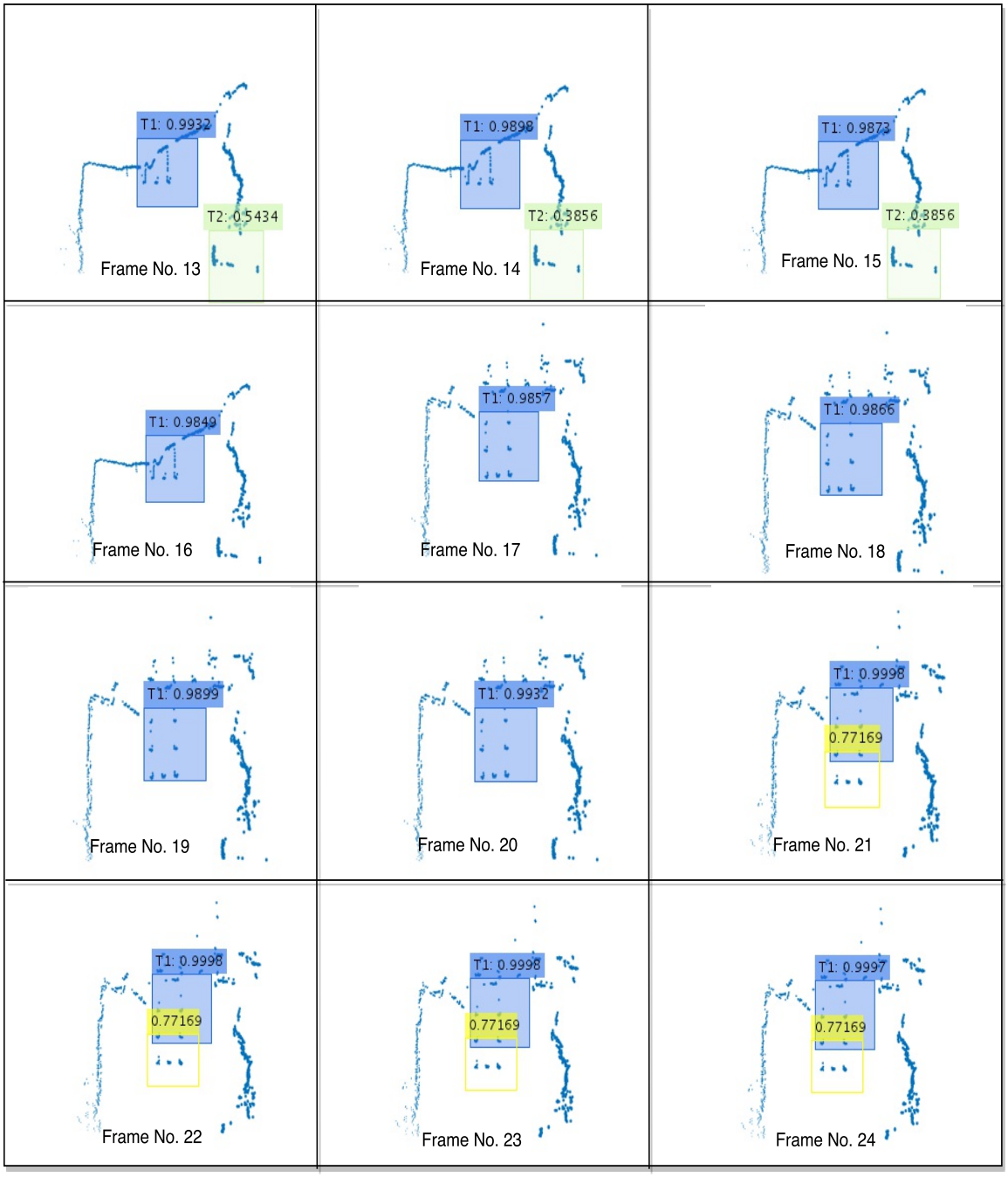


Figure 5.10: The detected ROIs for each frame with displaying only the reliable tracks (Frames No. 13 – 24).

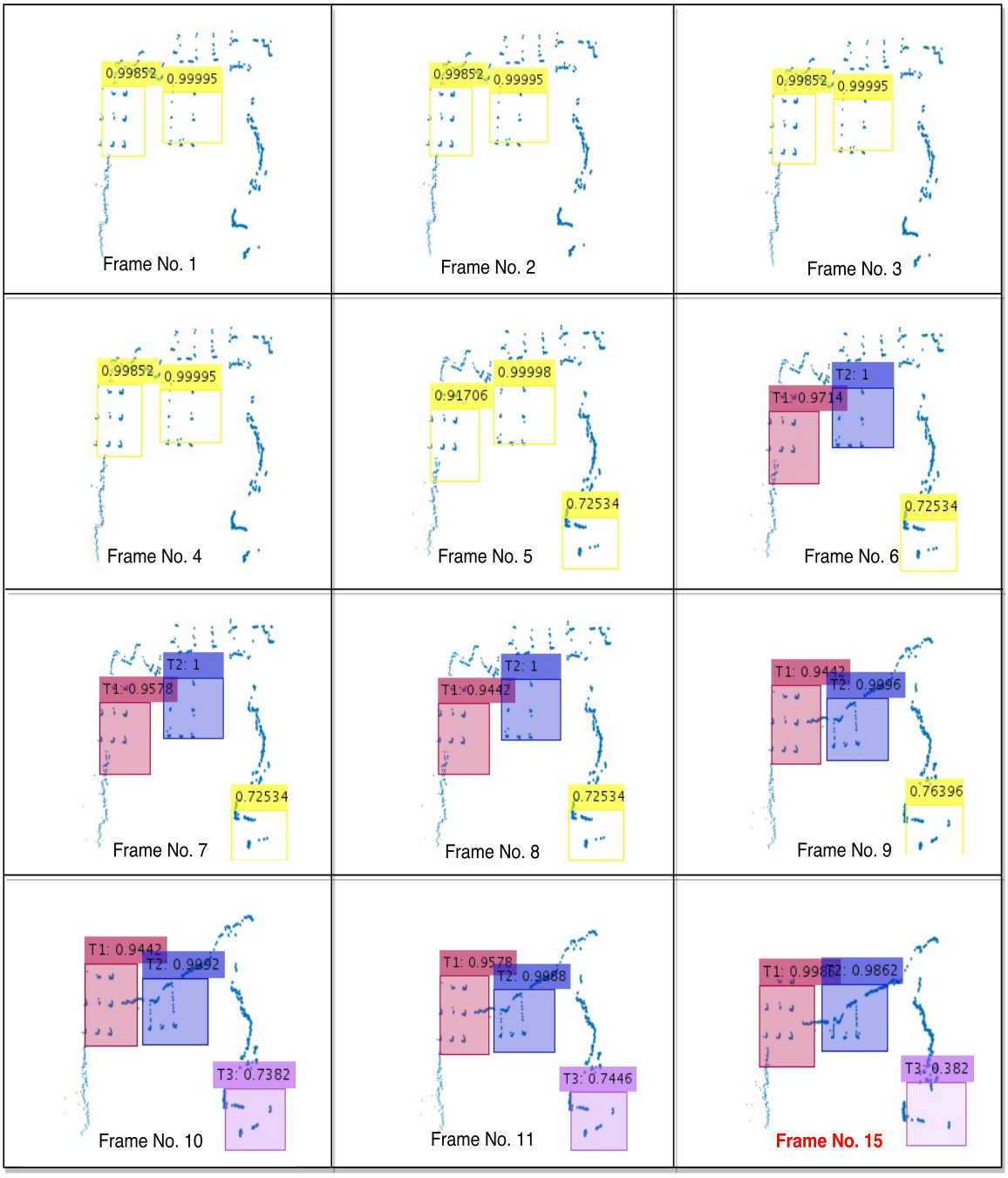


Figure 5.11: The detected ROIs for each frame, considering two pallets.

Chapter 6

Conclusions and Future Work

The main objectives of this thesis was to present the solution to the problem of identifying, localizing, and tracking the pallets based on laser rangefinder using machine learning techniques, in order to load the pallet automatically. The solution is mainly based on detecting the pallets using CNNs and tracking them using a simple Kalman filter. In fact, we do not need to have a robust tracker, because it is used just to reduce the false positive rate and to maintain the objects (pallets) in the scene labelled. To the best of our knowledge, this is the first work that applies CNNs for pallets detection using a 2D LRF.

The designed and developed pallet detection system was presented and tested on real world dataset. A comprehensive overview of the detection pipeline and its implementation phases were given. The problem of pallet tracking was presented which can be divided into two parts: (1) Detecting objects (e.g. pallets) in each frame using the pretrained networks, (2) Associating the detections corresponding to the same object over time.

The experiment settings, the test environment, and details on the collection of training data were introduced for testing our system. Moreover, the network training and the evaluation methodology, that is used to evaluate the pallet detection system, were presented for performing the pallet tracking task. Finally, the experimental results particularly for the tracking task, by considering only one pallet as well as considering one more pallet in the environment, were analyzed. It is observed that the proposed system has the capability of recognizing and tracking the positive tracks (pallets) among other negatives tracks with high confidence scores.

Here, we are going to list some possible directions for future work:

- We first need to perform on-line tracking for the pallets.
- Then, we need to estimate the position of the pallet with respect to the robot reference frame.

- Consequently, we need to integrate the proposed pallet detection system into trans-pallet vehicle and test them in real environment.
- Finally, once the position of the pallet relative to the vehicle is obtained, a trajectory has to be generated to pick up the pallet. This task can be managed by Model Predictive Control (MPC).

Appendix A

Marker-Based Localization System

Being autonomous is one of the most important goals in mobile robots. One of the fundamental works to achieve this goal is giving the robot the ability to find its own correct pose. Localization is a well studied problem in mobile robotics since the information about the robot's pose is essential for many applications. Over the last 20 years, several indoor localization techniques have been proposed and they have demonstrated the capability to robustly estimate the pose of robots in a large variety of application scenarios. Especially in industrial applications, a key requirement is the high localization and positioning accuracy of the robot on the factory floor. Precise positioning typically is a prerequisite to perform docking or mobile manipulation tasks such as pick and place.

The second topic that has been covered in this thesis is regarding the robot localization. It presents a mobile robot self-localization method with the use of artificial markers deployed to an environment. A specialty of the markers is that the position and rotation of the camera relative to the marker can be determined from only one marker [83]. In previous works, two indoor localization approaches have been used to provide a very precise estimation of the robot position, as presented in [3, 84, 85]. The experimental tests are carried out on the Laptop Webcam, using ARUCO markers.

A.1 Literature Review

Several approaches have been introduced to solve this problem. The commonly used sensors for localization are RFID [86], laser scanners [87,88], cameras [89–93], or GPS receivers [94]. Different probabilistic approaches have been successfully applied to localize robots with respect to a given map in a robust manner, often relying on techniques such as extended Kalman filters (EKF) [95–97], or particle filters [98–100]. There exist also approaches based on Monte-Carlo localization (MCL) [101]. In [101] the authors introduce the Monte Carlo Localization method, where they represent the probability density involved by maintaining a set of samples that are randomly drawn from it. By using a sampling-based representation they obtain a localization method that can represent arbitrary distributions. Vision-based MCL was first introduced by Dellaert et al. [102].

A novel method based on the harmony search (HS) algorithm for robot localization through scan matching is presented in [103]. In [104] an overview is presented of the most recent developments in sensor technology and methodology for indoor positioning, mapping and new applications enabled by indoor location information. In [105] inertial sensors are integrated with a 2D laser scanner to obtain the position of a mobile robot. The work in [106] presents a global localization system for an indoor autonomous vehicle equipped with odometry sensors and a radio-frequency identification (RFID) reader to interrogate tags located on the ceiling of the environment. Sensors and methods for indoor localization have been comprehensively reviewed in these books [107–110].

A.2 Marker-Based Localization System

This section presents firstly a visual localization system based on artificial markers which are placed in the environment and are used to determine the relative position of the robot with respect to these markers. The visual system can include one or two cameras and should detect some markers in the environment by use of computer vision tools [111–113]. These markers may be artificial or natural [83]. A specialty of the markers is the possibility to determine the relative position of the robot to a single marker. It is when a special type of 2D marker is used. If we know the calibration parameters of the camera lens, we can get the translation and rotation of the marker towards the image plane [114]. Moreover, using markers for robot localization does not add a new cost to the system like in beacons.

Several types of 2-D marker systems such as ARSTudio, ReacTIVision, CyberCode, Intersense, ARToolkit, ARTag, and ArUco were primarily developed for

the purpose of augmented reality [115]. In this work, the system ArUco markers will be used in visual localization. The main benefit of these markers is that a single marker provides enough information to obtain the camera pose. In [83] experiment results are described. The results prove that this system can be reliably employed in visual localization of mobile robots. The main features of ArUco library and how it works are presented in [114, 116].

References

[1] S. S. Dibris, "Automa Intelligente Robotico per Organizzazione Navette Elettriche (AIRONE)." http://www.research.softeco.it/airone.aspx, 2017. [Online; accessed 06-Sep-2017]. 3

[2] G. ROBOT, "GENOVA ROBOT." http://www.genovarobot.com/index.php/projects, 2017. [Online; accessed 06-Sep-2017]. 3

[3] F. Capezio, F. Mastrogiovanni, A. Scalmato, A. Sgorbissa, P. Vernazza, T. Vernazza, and R. Zaccaria, "Mobile robots in hospital environments: an installation case study.," in *ECMR*, pp. 61–68, 2011. 4, 52

[4] M. Piaggio, A. Sgorbissa, and R. Zaccaria, "Ethnos-ii: A programming environment for distributed multiple robotic systems," in *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*, pp. 10–pp, IEEE, 1999. 4

[5] M. Piaggio, A. Sgorgbissa, and R. Zaccaria, "Ethnos: a light architecture for real-time mobile robotics," in *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 3, pp. 1292–1297, IEEE, 1999. 4

[6] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3, pp. 2743–2748, IEEE, 2003. 5

[7] L. Zhang and B. K. Ghosh, "Line segment based map building and localization using 2d laser rangefinder," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 3, pp. 2538–2543, IEEE, 2000. 5

[8] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation,"

in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1, pp. 1304–1311, IEEE, 2003. 5

[9] E. Schulenburg, T. Weigel, and A. Kleiner, "Self-localization in dynamic environments based on laser and vision data," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, pp. 998–1004, IEEE, 2003. 5

[10] M. Seelinger and J.-D. Yoder, "Automatic pallet engagment by a vision guided forklift," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 4068–4073, IEEE, 2005. 8

[11] G. Garibotto, S. Masciangelo, M. Ilic, and P. Bassino, "Service robotics in logistic automation: Robolift: vision based autonomous navigation of a conventional fork-lift for pallet handling," in *Advanced Robotics, 1997. ICAR'97. Proceedings., 8th International Conference on*, pp. 781–786, IEEE, 1997. 8

[12] J. Pages, X. Armangué, J. Salvi, J. Freixenet, and J. Martí, "A computer vision system for autonomous forklift vehicles in industrial environments," in *Proc. of the 9th Mediterranean Conference on Control and Automation MEDS*, pp. 1–6, 2001. 8

[13] J. Nygards, T. Hogstrom, and A. Wernersson, "Docking to pallets with feedback from a sheet-of-light range camera," in *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3, pp. 1853–1859, IEEE, 2000. 8

[14] M. M. Aref, R. Ghabcheloo, and J. Mattila, "A macro-micro controller for pallet picking by an articulated-frame-steering hydraulic mobile machine," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6816–6822, IEEE, 2014. 8, 10

[15] G. Garibott, S. Masciangelo, M. Ilic, and P. Bassino, "Robolift: a vision guided autonomous fork-lift for pallet handling," in *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, vol. 2, pp. 656–663, IEEE, 1996. 8

[16] W. Kim, D. Helmick, and A. Kelly, "Model based object pose refinement for terrestrial and space autonomy," 2001. 8

[17] R. Cucchiara, M. Piccardi, and A. Prati, "Focus based feature extraction for pallets recognition," in *BMVC*, 2000. 8

[18] S. Wang, A. Ye, H. Guo, J. Gu, X. Wang, and K. Yuan, "Autonomous pallet localization and picking for industrial forklifts based on the line structured light," in *Mechatronics and Automation (ICMA), 2016 IEEE International Conference on*, pp. 707–713, IEEE, 2016. 8

[19] R. Varga and S. Nedevschi, "Vision-based autonomous load handling for automated guided vehicles," in *Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on*, pp. 239–244, IEEE, 2014. 9

[20] R. Varga, A. Costea, and S. Nedevschi, "Improved autonomous load handling with stereo cameras," in *2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*. 9

[21] G.-z. Cui, L.-s. Lu, Z.-d. He, L.-n. Yao, C.-x. Yang, B.-y. Huang, and Z.-h. Hu, "A robust autonomous mobile forklift pallet recognition," in *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, vol. 3, pp. 286–290, IEEE, 2010. 9

[22] C. Beder, B. Bartczak, and R. Koch, "A comparison of pmd-cameras and stereo-vision for the task of surface reconstruction using patchlets," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007. 9

[23] F. Weichert, S. Skibinski, J. Stenzel, C. Prasse, A. Kamagaew, B. Rudak, and M. Ten Hompel, "Automated detection of euro pallet loads by interpreting pmd camera depth images," *Logistics Research*, vol. 6, no. 2-3, pp. 99–118, 2013. 9

[24] S. Byun and M. Kim, "Real-time positioning and orienting of pallets based on monocular vision," in *Tools with Artificial Intelligence, 2008. ICTAI'08. 20th IEEE International Conference on*, vol. 2, pp. 505–508, IEEE, 2008. 9

[25] G. Chen, R. Peng, Z. Wang, and W. Zhao, "Pallet recognition and localization method for vision guided forklift," in *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on*, pp. 1–4, IEEE, 2012. 9

[26] J.-Y. Oh, H.-S. Choi, S.-H. Jung, H.-S. Kim, and H.-Y. Shin, "Development of pallet recognition system using kinect camera," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 9, no. 4, pp. 227–232, 2014. 9

[27] J.-L. Syu, H.-T. Li, J.-S. Chiang, C.-H. Hsia, P.-H. Wu, C.-F. Hsieh, and S.-A. Li, "A computer vision assisted system for autonomous forklift vehicles

in real factory environment," *Multimedia Tools and Applications*, pp. 1–21, 2016. 9

[28] D. Holz and S. Behnke, "Fast edge-based detection and localization of transport boxes and pallets in rgb-d images for mobile robot bin picking," in *ISR 2016: 47st International Symposium on Robotics; Proceedings of*, pp. 1–8, VDE, 2016. 9

[29] R. Varga and S. Nedevschi, "Robust pallet detection for automated logistics operations.," in *VISIGRAPP (4: VISAPP)*, pp. 470–477, 2016. 9

[30] M. Hebert, "Outdoor scene analysis using range data," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3, pp. 1426–1432, IEEE, 1986. 9

[31] R. Hoffman and A. K. Jain, "Segmentation and classification of range images," *IEEE transactions on pattern analysis and machine intelligence*, no. 5, pp. 608–620, 1987. 9

[32] T. S. Newman, P. J. Flynn, and A. K. Jain, "Model-based classification of quadric surfaces," *CVGIP: Image Understanding*, vol. 58, no. 2, pp. 235–249, 1993. 9

[33] L. Baglivo, N. Bellomo, G. Miori, E. Marcuzzi, M. Pertile, and M. De Cecco, "An object localization and reaching method for wheeled mobile robots using laser rangefinder," in *Intelligent Systems, 2008. IS'08. 4th International IEEE Conference*, vol. 1, pp. 5–6, IEEE, 2008. 9

[34] M. R. Walter, S. Karaman, E. Frazzoli, and S. Teller, "Closed-loop pallet manipulation in unstructured environments," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 5119–5126, IEEE, 2010. 9, 10

[35] D. Lecking, O. Wulf, and B. Wagner, "Variable pallet pick-up for automatic guided vehicles in industrial environments," in *Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on*, pp. 1169–1174, IEEE, 2006. 9

[36] Z. He, X. Wang, J. Liu, J. Sun, and G. Cui, "Feature-to-feature based laser scan matching for pallet recognition," in *Measuring Technology and Mechatronics Automation (ICMTMA), 2010 International Conference on*, vol. 2, pp. 260–263, IEEE, 2010. 10

[37] C. Premebida and U. Nunes, "Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications," *Robotica*, vol. 2005, pp. 17–25, 2005. 10

[38] R. Bostelman, T. Hong, and T. Chang, "Visualization of pallets," in *Optics East 2006*, pp. 638408–638408, International Society for Optics and Photonics, 2006. 10

[39] L. Baglivo, N. Biasi, F. Biral, N. Bellomo, E. Bertolazzi, M. Da Lio, and M. De Cecco, "Autonomous pallet localization and picking for industrial forklifts: a robust range and look method," *Measurement Science and Technology*, vol. 22, no. 8, p. 085502, 2011. 10

[40] M. M. Aref, R. Ghabcheloo, A. Kolu, M. Hyvonen, K. Huhtala, and J. Mattila, "Position-based visual servoing for pallet picking by an articulated-frame-steering hydraulic mobile machine," in *Robotics, Automation and Mechatronics (RAM), 2013 6th IEEE Conference on*, pp. 218–224, IEEE, 2013. 10

[41] M. M. Aref, R. Ghabcheloo, A. Kolu, and J. Mattila, "A multistage controller with smooth switching for autonomous pallet picking," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 2535–2542, IEEE, 2016. 10

[42] M. R. Walter, M. Antone, E. Chuangsuwanich, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, Y. Friedman, J. Glass, J. P. How, *et al.*, "A situationally aware voice-commandable robotic forklift working alongside people in unstructured outdoor environments," *Journal of Field Robotics*, vol. 32, no. 4, pp. 590–628, 2015. 10

[43] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks*, vol. 2, no. 2004, p. 41, 2004. 11, 12

[44] Y. LeCun and M. Ranzato, "Deep learning tutorial," in *Tutorials in International Conference on Machine Learning (ICML13)*, Citeseer, 2013. 12

[45] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014. 12, 14, 19

[46] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015. 12, 13, 19

[47] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015. 12, 13, 19

[48] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013. 12, 19

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European Conference on Computer Vision*, pp. 346–361, Springer, 2014. 13

[50] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2, pp. 2169–2178, IEEE, 2006. 13

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016. 13

[52] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015. 13

[53] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik, "R-cnns for pose estimation and action detection," *arXiv preprint arXiv:1406.5212*, 2014. 13

[54] G. Gkioxari, R. Girshick, and J. Malik, "Contextual action recognition with r* cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1080–1088, 2015. 13

[55] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, *et al.*, "Deepid-net: Deformable deep convolutional neural networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2403–2412, 2015. 13

[56] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, and G. Wang, "Recent advances in convolutional neural networks," *arXiv preprint arXiv:1512.07108*, 2015. 13, 18, 19

[57] T. BARBIÉ, R. TANAKA, R. KABUTAN, and T. NISHIDA, "Real time object position estimation by convolutional neural networks," 13

[58] A. Teichman and S. Thrun, "Practical object recognition in autonomous driving and beyond," in *Advanced Robotics and its Social Impacts (ARSO), 2011 IEEE Workshop on*, pp. 35–38, IEEE, 2011. 13

[59] W. Xiaoa, B. Valletb, K. Schindlerc, and N. Paparoditisb, "Simultaneous detection and tracking of pedestrian from panoramic laser scanning data," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 295–302, 2016. 13

[60] A. M. Pinto, L. F. Rocha, and A. P. Moreira, "Object recognition using laser range finder and machine learning techniques," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 1, pp. 12–22, 2013. 13

[61] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, "A lidar and vision-based approach for pedestrian and vehicle detection and tracking," in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pp. 1044–1049, IEEE, 2007. 14

[62] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010. 14

[63] I. Goodfellow, A. Courville, and Y. Bengio, "Deep learning, book in preparation for mit press (2016)," *URL http://www. deeplearningbook. org*, 2016. 14

[64] M. Braun, Q. Rao, Y. Wang, and F. Flohr, "Pose-rcnn: Joint object detection and pose estimation using 3d object proposals," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pp. 1546–1551, IEEE, 2016. 14

[65] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013. 18

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015. 18

[67] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015. 18

[68] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015. 18

[69] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the twenty-first international conference on Machine learning*, p. 116, ACM, 2004. 19

[70] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks.," in *ICML*, pp. 507–516, 2016. 19

[71] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 539–546, IEEE, 2005. 19

[72] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2, pp. 1735–1742, IEEE, 2006. 19

[73] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015. 19

[74] C.-A. Brust, S. Sickert, M. Simon, E. Rodner, and J. Denzler, "Convolutional patch networks with spatial prior for road detection and urban scene understanding," *arXiv preprint arXiv:1502.06344*, 2015. 23

[75] Z. Teng, J.-H. Kim, and D.-J. Kang, "Real-time lane detection by using multiple cues," in *Control Automation and Systems (ICCAS), 2010 International Conference on*, pp. 2334–2337, IEEE, 2010. 23

[76] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, and V. N. Murali, "Deeplanes: End-to-end lane position estimation using deep neural networksa," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 38–45, 2016. 23

[77] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012. 23

[78] S. Arlot, A. Celisse, *et al.*, "A survey of cross-validation procedures for model selection," *Statistics surveys*, vol. 4, pp. 40–79, 2010. 27

[79] E. V. Cuevas, D. Zaldivar, and R. Rojas, "Kalman filter for vision tracking," 2005. 28

[80] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics (NRL)*, vol. 2, no. 1-2, pp. 83–97, 1955. 30

[81] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, 2009. 36

[82] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006. 42

[83] A. Babinec, L. Jurišica, P. Hubinskỳ, and F. Duchoň, "Visual localization of mobile robot using artificial markers," *Procedia Engineering*, vol. 96, pp. 1–9, 2014. 52, 53, 54

[84] M. Piaggio, A. Sgorbissa, and R. Zaccaria, "Navigation and localization for service mobile robots based on active beacons," *Systems Science*, vol. 27, no. 4, pp. 71–83, 2001. 52

[85] M. Piaggio, A. Sgorbissa, and R. Zaccaria, "Autonomous navigation and localization in service mobile robotics," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4, pp. 2024–2029, IEEE, 2001. 52

[86] B. F. D. Hähnel and D. Fox, "Gaussian processes for signal strength-based location estimation," in *Proceeding of Robotics: Science and Systems*, Citeseer, 2006. 53

[87] E. Ivanjko, A. Kitanov, and I. Petrovic, *Model based Kalman filter mobile robot self-localization*. INTECH Open Access Publisher, 2010. 53

[88] Y.-J. Lee, B.-D. Yim, and J.-B. Song, "Mobile robot localization based on effective combination of vision and range sensors," *International Journal of Control, Automation and Systems*, vol. 7, no. 1, pp. 97–104, 2009. 53

[89] H. Andreasson, A. Treptow, and T. Duckett, "Localization for mobile robots using panoramic vision, local features and particle filter," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 3348–3353, IEEE, 2005. 53

[90] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke, "Metric localization with scale-invariant visual features using a single perspective camera," in *European Robotics Symposium 2006*, pp. 195–209, Springer, 2006. 53

[91] P. Elinas and J. J. Little, "σmcl: Monte-carlo localization for mobile robots with stereo vision," in *Proc. of Robotics: Science and Systems (RSS)*, 2005. 53

[92] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1697–1702, IEEE, 2012. 53

[93] N. Crombez, G. Caron, and E. M. Mouaddib, "Using dense point clouds as environment model for visual localization of mobile robot," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2015 12th International Conference on*, pp. 40–45, IEEE, 2015. 53

[94] F. Capezio, A. Sgorbissa, and R. Zaccaria, "Gps-based localization for a surveillance ugv in outdoor areas," in *Robot Motion and Control, 2005. RoMoCo'05. Proceedings of the Fifth International Workshop on*, pp. 157–162, IEEE, 2005. 53

[95] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 376–382, 1991. 53

[96] E. Kiriy and M. Buehler, "Three-state extended kalman filter for mobile robot localization," *McGill University., Montreal, Canada, Tech. Rep. TR-CIM*, vol. 5, p. 23, 2002. 53

[97] H. Ahmad and T. Namerikawa, "Extended kalman filter-based mobile robot localization with intermittent measurements," *Systems Science & Control Engineering: An Open Access Journal*, vol. 1, no. 1, pp. 113–126, 2013. 53

[98] J. M. Pak, C. K. Ahn, Y. S. Shmaliy, and M. T. Lim, "Improving reliability of particle filter-based localization in wireless sensor networks via hybrid particle/fir filtering," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 5, pp. 1089–1098, 2015. 53

[99] H. Nurminen, A. Ristimaki, S. Ali-Loytty, and R. Piché, "Particle filter and smoother for indoor localization," in *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pp. 1–10, IEEE, 2013. 53

[100] J. Röwekämper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard, "On the position accuracy of mobile robot localization based on particle filters combined with scan matching," in *Intelligent Robots and*

Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 3158–3164, IEEE, 2012. 53

[101] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1322–1328, IEEE, 1999. 53

[102] F. Dellaert, W. Burgard, D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, pp. 588–594, IEEE, 1999. 53

[103] M. Mirkhani, R. Forsati, A. M. Shahri, and A. Moayedikia, "A novel efficient algorithm for mobile robot localization," *Robotics and Autonomous Systems*, vol. 61, no. 9, pp. 920–931, 2013. 53

[104] K. Khoshelham and S. Zlatanova, "Sensors for indoor mapping and navigation," 2016. 53

[105] Y. Gao, S. Liu, M. M. Atia, and A. Noureldin, "Ins/gps/lidar integrated navigation system for urban and indoor environments using hybrid scan matching algorithm," *Sensors*, vol. 15, no. 9, pp. 23286–23302, 2015. 53

[106] E. DiGiampaolo and F. Martinelli, "Mobile robot localization using the phase of passive uhf rfid signals," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 1, pp. 365–376, 2014. 53

[107] J. Borenstein, H. Everett, L. Feng, *et al.*, "Where am i? sensors and methods for mobile robot positioning," *University of Michigan*, vol. 119, no. 120, p. 27, 1996. 53

[108] J. Borenstein, L. Feng, and H. Everett, *Navigating mobile robots: Systems and techniques*. AK Peters, Ltd., 1996. 53

[109] H. Everett, *Sensors for mobile robots: theory and application*. AK Peters, Ltd., 1995. 53

[110] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016. 53

[111] O. Marques, *Practical image and video processing using MATLAB*. John Wiley & Sons, 2011. 53

[112] M. Sukop, M. Hajduk, J. Varga, and M. Vagaš, "Image processing and object founding in the robot soccer application,-2012," *International Scientific Herald*, vol. 3, no. 2, pp. 203–206, 2012. 53

[113] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008. 53

[114] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. 53, 54

[115] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 590–596, IEEE, 2005. 54

[116] A. website, "Automa Intelligente Robotico per Organizzazione Navette Elettriche (AIRONE)." https://www.uco.es/investiga/grupos/ava/node/26, 2017. [Online; accessed 06-Sep-2017]. 54