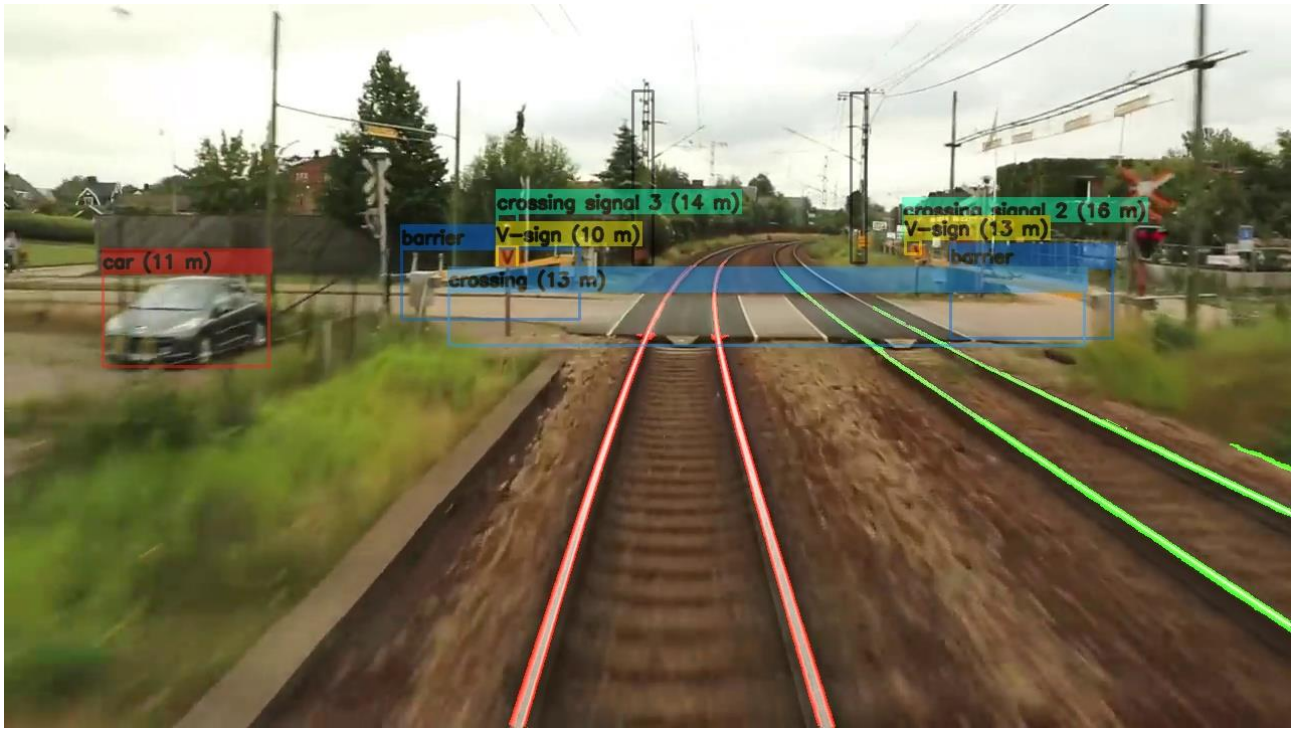
CHALMERS
UNIVERSITY OF TECHNOLOGY



Implementation of a Vision System for an Autonomous Railway Maintenance Vehicle

Track and Object Detection with YOLO, Neural Networks and Region Growing

Master's thesis in Engineering Mathematics and Computational Science, and Complex Adaptive Systems

Albin Warnicke
Jesper Jönsson

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2021
www.chalmers.se

MASTER'S THESIS IN ENGINEERING MATHEMATICS AND COMPUTATIONAL SCIENCE, AND COMPLEX ADAPTIVE SYSTEMS

Implementation of a Vision System for an Autonomous Railway Maintenance Vehicle

Track and Object Detection with YOLO, Neural Networks and Region Growing

ALBIN WARNICKE
JESPER JÖNSSON

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2021

Implementation of a Vision System for an Autonomous Railway Maintenance Vehicle
Track and Object Detection with YOLO, Neural Networks and Region Growing
ALBIN WARNICKE
JESPER JÖNSSON

Master's thesis 2021:11
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
The figure shows the developed vision system applied at a railway crossing. The detected track at which the train is traveling is colored in red with the detected sidetrack in green. The eight objects detected in the image are marked by bounding boxes.

Chalmers Reproservice
Gothenburg, Sweden 2021

Implementation of a Vision System for an Autonomous Railway Maintenance Vehicle
Track and Object Detection with YOLO, Neural Networks and Region Growing
Master's thesis in Engineering Mathematics and Computational Science, and Complex Adaptive Systems
ALBIN WARNICKE
JESPER JÖNSSON
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

Railway infrastructure is often expensive to maintain. To improve efficiency and lower these costs, the use of autonomous railway vehicles for such maintenance has begun to be explored. A railway vehicle requires several components to achieve complete automation, including systems for navigation, decision-making, and sensors such as cameras. This project aims to develop the vision system used by an autonomous track trolley under development at Chalmers University of Technology. The proposed vision system can detect railway tracks and switches by a region-growing algorithm based on the image intensity gradient. Object detection is achieved by the use of a YOLOv4-tiny neural network and is developed to detect persons, vehicles, railway signs and signals, road crossings and catenary support poles. The signal and speed sign messages are further classified by additional convolutional neural networks. The vision system is implemented as a ROS node on a single-board computer, a NVIDIA Jetson Nano, and is running in real-time at up to 15 FPS.

The vision system is accurate and robust enough to be used as a prototype in simple environments. The track that the vehicle is traveling on was detected in 98.4 % of the evaluated video frames, with the sidetracks correctly identified in 70-80 % of the time. Several of the considered objects were detected with 90-100 % accuracy, for example vehicles and road crossings. Other objects, particularly railway switches and incoming tracks, were however only correctly recognized in about 60 % of their occurrences. Signals and speed signs were detected with high accuracy.

Some features can be improved or added before the vision system can be applied to a complete autonomous railway vehicle. The main limitation of the implemented object detection is the lack of large training datasets. With more available video data, datasets with an increased number of labeled objects and greater diversity could be created. Utilizing the full capabilities of larger datasets would eventually require the use of more complex neural networks. The currently used hardware however limits the possible methods to simpler algorithms. The track detection algorithm can serve as a base for further improvement, with the region growing based on the image intensity gradients not being robust enough to handle large variations in lighting and environment conditions. An approach with semantic segmentation neural networks is instead suggested to achieve robust track detection.

Keywords: Autonomous vehicles, Railway, Object detection, Computer vision, Track detection, Machine learning, Artificial neural networks, Convolutional neural networks, YOLO, You Only Look Once, Region growing

Preface

This report is the result of a master's thesis project carried out at the Department of Mechanics and Maritime Sciences at Chalmers University of Technology during the spring of 2021. The work is a part of the project group led by Krister Wolff that is developing an autonomous railway service vehicle. The project is a collaboration with and financed by the Swedish Transport Administration. In this master's thesis project, the authors have investigated the possibilities of a vision system that will be applied to the autonomous vehicle. Such a vision system is proposed, implemented, and evaluated.

Acknowledgements

We would greatly like to thank our supervisor and examiner Krister Wolff for his contribution to this work, from proposing and introducing the project to helping us finalize our results.

We would further also want to thank the train enthusiast Jan Kivisaar for sharing his railway videos publicly and letting us use them in this project.

Göteborg, June 2021
Albin Warnicke
Jesper Jönsson

Nomenclature

ATC	Automatic Train Control
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
FPS	Frames Per Second
GPS	Global Positioning System
HOG	Histogram of Oriented Gradients
OpenCV	OPEN source Computer Vision library
MS COCO	Microsoft Common Objects in COntext - Dataset for object detection
PASCAL VOC	Pattern Analysis, Statistical Modelling And Computational Learning Visual Object Classes - Dataset for object detection
RailSem19	Semantic segmentation dataset from railway environments
ReLU	Rectified Linear Unit - Neural network activation function
RGB	Red Green Blue - Color format
ROS	Robot Operating System
YOLO	You Only Look Once - Neural network for object detection

Contents

1 Introduction

Maintenance of railway infrastructure is often expensive, but necessary to keep the train traffic running. Disruptions in railway traffic are costly to companies that get their goods and personnel delayed, and reduces the public inclination towards traveling by train. In the end, delays lead to reduced usage of railway traffic and a potential increase in other transports that have greater emissions of greenhouse gases [Eur08]. Cost- and time-efficient maintenance might instead lead to train traffic becoming more attractive for public and goods transportation.

One way to potentially improve the maintenance is to utilize autonomous railway vehicles. Such an autonomous vehicle could be used to automatically collect data through inspection rounds, detect faults, and perform transportation tasks or simple maintenance, all by itself. An important advantage of autonomous vehicles is that they can operate during inconvenient working hours when the railway traffic is low, or depart from locations where there is no available maintenance crew nearby. Automatic metros have been deployed and used successfully in a closed and controlled environment, but autonomous applications on open tracks are quite more difficult to realize [Tre+18]. The development of autonomous maintenance vehicles may also be used as a testbed, and the first step towards the realization of autonomous passenger and freight trains.

Currently, a project group at the Department of Mechanics and Maritime Sciences at Chalmers, led by Krister Wolff, has started constructing a fully autonomous railway vehicle based on an existing track trolley. The project is a collaboration with and financed by the Swedish Transport Administration. The project is financed by the Swedish Transport Administration. To develop a fully autonomous railway vehicle, it is necessary to utilize computer vision and sensor fusion. This includes using sensors such as cameras, lidar, sonar, and GPS to give the autonomous vehicle the abilities needed for navigation, localization, decision-making, and detection of faults and obstacles.

1.1 Purpose and Scope

The purpose of this project is to develop a generalizable vision system for an autonomous track trolley, for use in the inspection and maintenance of railway infrastructure. The vision system should be capable of detecting the railway tracks and switches, road crossings, light signals, and signal boards that can be used for navigation and decision-making, but also vehicles and pedestrians on or close to the tracks. It should also locate the positions of the catenary support poles, which can be used to identify faults in the more complex parts of the catenary system. The proposed vision system will be implemented on a single-card computer connected with cameras and a GPS receiver on the track trolley. Furthermore is the output from the vision system to be combined with GPS data to enable localization of detected objects. The vision system will communicate information to other components of the autonomous vehicle, but will not perform any actual decision-making or driving of the vehicle by itself.

1.2 Limitations

Most importantly must the methods used in the implementation be fast enough to run in real-time, at least 10 Frames Per Second (FPS), on a portable lower-end device. The vision system is developed focusing on a low-speed vehicle with a short braking distance, such as the available track trolley which has a maximum speed of 16 km/h. This implies that the vision system should be focused on detecting objects in the vicinity of the vehicle, and not far away. Within the scope of this project, no data from railways were collected by the authors. Thus the implemented programs were developed based on already available videos. The video data used for development and evaluation is captured on high-speed trains, traveling at speeds up to 160 km/h. This implies that much of the functionality of the proposed vision system is optimized to work at these greater traveling speeds. Due to the lack of train velocity data, several speed-dependent features that the authors had in mind to eliminate the need for fixed-speed optimization were not implemented.

This project should be considered as a pilot study to investigate and propose solutions for a vision system applicable in a simple scenario. Thus, the focus has not been on obtaining optimal performance for the implemented systems, instead rather on demonstrating the proof of concept. Several objects occurring in railway traffic have been omitted regarding detection, partly because of them too seldom occurring in the available video data, but also due to their relevance to the intended application. The focus has instead been

on detecting the objects that are of the most relevance and importance to be able to detect. Similarly, the implemented programs were not developed to optimally deal with complex scenarios such as large railway yards, extreme weather, and very bad lighting conditions.

1.3 Motivation and Contribution

Some research has been performed on different topics related to automatic analysis of the railway environment. A brief review of the work related to the intended application of this project is presented in section 2.3. The integration of such developed systems into an autonomous railway vehicle is although still to be investigated. By in this project proposing, implementing, and evaluating an approach to a vision system for such a vehicle, insight is gained on the prospects and remaining challenges of a complete solution. This project thus serves as a stepping stone in the progress of creating the fully autonomous railway maintenance vehicle developed by Chalmers and the Swedish Transport Administration. The development of the autonomous vehicle might in the future even contribute to the introduction of autonomous trains in passenger and freight traffic.

2 Background

The vision system created in this project is intended for a maintenance vehicle applied on railways in Sweden. The railway infrastructure is not globally standardized, and light signals and signal boards are often country-specific. The important properties of the Swedish railway infrastructure are therefore here presented. This chapter will also give a brief review of previous research related to this project, which has mainly been conducted with focus on foreign railways. The track trolley, for which the vision system was developed, is first introduced.

2.1 The Alumicart Track Trolley

The implemented systems were developed with a four-seated Bance Battery Electric Alumicart [R B11] in mind for application, see Figure 2.1. In the figure it is seen that the vehicle is developed and equipped for use in maintenance work. The Alumicart is a 320 kg, 203 cm long track trolley capable of carrying four passengers and an extra payload, with a total weight of 500 kg. The vehicle has with its four electric motors a maximum speed of 16 km/h and continuous speed control in both directions. It is powered by three pairs of 12 V batteries. It is expected by the manufacturer that the vehicle can travel at least 100 km before the batteries have to be recharged. The Alumicart is braked by regenerative braking whilst in motion and is equipped with axial disc brakes on two of the engines for static braking. In addition to marker lights, the vehicle is equipped with a 70 W halogen spotlight mounted between the front seats together with two 70 W halogen floodlights on the front of the track trolley. The Alumicart is not equipped with any ATC system, see section 2.2.1.

Figure 2.1: *The four-seated Bance Battery Electric Alumicart for which the vision system was developed.*

2.2 Swedish Railway Infrastructure

To enable safe, automatic maneuvering of a railway vehicle it needs a vision system capable of detecting and recognizing some of the common objects in the railway traffic. These objects include various railway traffic light signals and signal boards, as well as road crossings. Some objects are not vital for maneuvering, but still of interest to detect or monitor. An example of this is the catenary system and its support structure, which can be monitored to detect faults and future risks.

Further on, light signals and signal boards will be referred to as signals and signs respectively. The use of the term 'signal' should thus not be confused with its use in some of the referenced literature, where it may refer to both light signals and signal boards [Swe20b]. In the Swedish railway network, the signals and signs are normally placed to the left of the railway track for which they apply. For a single-track railway, signals and signs may be placed on either side of the track. In the case of a multi-track railway, the normal placement of the applying signs and signals are; to the left of the leftmost track, to the left of possible intermediate tracks, and to the right of the rightmost track [Swe19a]. Thus a double-track railway will normally have signs and signals on both sides of the tracks.

2.2.1 Automatic Train Control

Modern trains are equipped with so-called ATC systems, the term used in Sweden for Automatic Train Control and protection systems. Most railway infrastructure in Sweden today utilizes the ATC-2 version of the system. This system is incompatible with the European standards, but the plan is to replace the current system over the next decades. ATC is used to obtain information regarding the current railway route digitally, directly to the train cockpit. The Swedish ATC system is based on emitters that are placed along the tracks. Such an emitter is activated when a train passes above it and then sends information to the train computer. The system conveys information about the applying speed limit, operation mode, drive permission, and various actions that the driver may have to perform. The ATC system can also slow down the train when traveling faster than allowed [Swe20a; Swe20c]. ATC may thus be considered as a step towards autonomous railway traffic.

2.2.2 Railway Signals

There are mainly five types of signals used in the Swedish railway traffic; main signals, distant signals, road crossing signals, distant road crossing signals, and dwarf signals, all illustrated in Figure 2.2. Dwarf signals mainly occur at railway yards, complex environments that are not the focus of this project. Thus the dwarf signals are not taken into consideration in this project and are therefore not further described. Observe that, beyond the five signal types mainly used and here accounted for, there are additional signals, combinations of signals and signs, and extra rules that apply to less common situations that may occur in the railway traffic. For an extensive description of these anomalies, see the currently applying rule modules from the Swedish Transport Administration [Swe]. The main signal, distant signal, road crossing signal, and distant road crossing signal can each convey several different messages depending on which of their lamps that are lit, the colors of the lights, and if the lights are flashing or fixed. The four signals mainly convey messages of the current speed limit, drive permission, and information about the upcoming signal messages. The latter is important since trains often have low acceleration and have to brake long in advance of a decreased speed limit or stop. For a detailed description of the possible signal messages, see Appendix A.1.

(a) (b) (c) (d) (e)

Figure 2.2: *Examples of the five signal types. From the left; (a) main signal, (b) distant signal, (c) road crossing signal, (d) distant road crossing signal, and (e) dwarf signal. Images from: Mysid, Wikipedia.*

2.2.3 Railway Signs

In railway traffic, there are in addition to signals also signs that convey the messages required for maneuvering the vehicles. Among others, there are signs conveying information regarding the speed limit, warnings, sound signals, connectivity requirements for the catenary systems, and how to approach train stations. Several of the railway signs are not directly applicable to the intended application of this project, and others are not common enough to be of practical interest, see the further comment on this in section 4.2.1. Thus a selection of the signs of interest was performed, and the chosen signs are displayed in Figure 2.3. The selected signs convey messages of the current speed limit or are related to road crossings and main signals. As for signals, additional signs and rules regarding signs can be found in the currently applying rule modules from the Swedish Transport Administration [Swe].

Several signs convey messages of the speed limit for the railway vehicle. Among these is the speed sign in Figure 2.3 (a), showing the speed limit in units of km/h. Vehicles equipped with ATC may obey increased speed limits compared to those without. Such speed limits are indicated by the four ATC speed signs in Figure 2.3 (b)-(e). The triangular warning sign, seen in Figure 2.3 (f) and (g), conveys information of an upcoming main signal, train station, speed limit change, or road crossing. The warning sign always comes in combination with another sign, defining the specific message. In this project, only the two combinations in Figure 2.3 (f)

and (g) are considered. These are the warnings of an upcoming road crossing and main signal respectively, hence the supplement V-sign and the sign resembling the distant signal [Swe19b].

(a) (b) (c) (d) (e) (f) (g)

Figure 2.3: *The signs, and combinations of signs, that are considered in this project. From the left; (a) speed sign conveying the speed limit in units of km/h, (b) speed sign with a greater speed limit for vehicles equipped with ATC, (c) start of route with reduced ATC speed, (d) start of route with increased ATC speed, (e) warning of an upcoming ATC speed reduction, (f) warning of an upcoming road crossing, and (g) warning of an upcoming main signal. Images from: Frederik and Markus Tellerup, Järnväg.net, redistributed with permission.*

2.2.4 Catenary Support Structure

The catenary is a system of overhead power lines, supplying the trains with electricity via a pantograph on top of the vehicle. In the Swedish railways, these wires are hanging roughly 5.5 m above the ground. The wires are held up by the support structure characterized by a green, brown, or gray metal pole. These poles are normally placed at a distance of 55-60 m apart from each other. In Sweden, there are annually 300-400 disturbances in the catenary and among these damaged catenary components are one of the most common causes [Swe18]. Automatically monitoring the catenary to detect faults and possible future risks, such as bird nests built on the support structure, could be used to prevent unnecessary train traffic delays.

2.3 Related Work

The car industry is rapidly developing towards making self-driving vehicles a reality. Large amounts of money are invested into the business by car manufacturers that aspire on taking the lead in this race [Blo20]. The automatic analysis of the car traffic environment has as a consequence been widely studied, including detection and recognition of common objects. The railway industry is yet behind in the development of autonomous vehicles. Even though these two applications share several challenges, almost none of the suggested solutions for autonomous cars have been implemented and tested in railway traffic. Following are brief descriptions of earlier research within the automatic analysis of the railway environment that is related to the scope of this project.

2.3.1 Previous Work in Object Detection

Much of the existing research within the field of automatic object recognition in a railway environment is focused on detecting and recognizing signs and signals. For these challenges, the approaches have mainly been based on classical image processing techniques with pattern matching [NUN09; MLG06], SIFT features [NU10], and detection of simple geometric shapes [GP19; Agu+16]. These methods are often greatly dependent on certain protrusive features in the objects to detect and are thus not optimal for the detection of general objects of any color and shape.

For the detection of a greater variety of objects common in railway traffic, several approaches with artificial neural networks have been proposed. The implemented neural networks have in a railway environment shown to be able to detect and recognize objects such as trains, pedestrians, helmets, tools, and directions of railway tracks [Ye+18; Ye+20a; Ye+20b; LZY18]. A similar approach was demonstrated by Karagiannis et al. [KOP19], implementing a neural network able to detect and recognize ten different signals and signs from the Australian railway network. These methods have all been evaluated on individual images, free of temporal context. There still does not exist any generalizable method for real-time object detection demonstrated in a railway environment.

2.3.2 Previous Work in Track Detection

Autonomous detection and monitoring of railway tracks have been investigated for a wide variety of applications. To potentially increase the safety of traveling by railway, different methods for automatic inspection of track

fasteners and detection of rail surface defects have been proposed [TIM20; Wei+20; Wei+19]. To detect faults on a larger scale, for example buckled tracks, Mittal and Rao [MR17] demonstrated that artificial neural networks can be used for such tasks. Mittal and Rao, as well as Ye et al. [Ye+18], also showed that neural networks can be deployed to identify railway switches, marking the position of these by bounding boxes. To obtain a more detailed description of the tracks, Wedberg [Wed17] applied a neural network to thermal images for the prediction of the track traction on a greater distance ahead of the vehicle. To achieve similar results as Wedberg, Qi et al. [QTS13] used Histogram of Oriented Gradients (HOG) features to identify the pixels in an image that form the rails. The method is based on pixel-wise region growing, utilizing that the image regions with rails have certain HOG features. Qi et al. implemented the algorithm such that it could be run in real-time at speeds up to 10 FPS for 1600 x 1200 pixel images. The track detection demonstrated by Qi et al. is at a level of detail such that the rails found can be used to predict the traveling path at a railway switch. An image segmentation approach to track detection is proposed by Li et al. [Li+20]. Li et al. applied a neural network named RailNet to pixel-wise find the segments of the images that contain tracks. Thus, the individual rails are not automatically separated.

2.3.3 Previous Work in the Analysis of the Railway Catenary System

There exist several studies of different approaches on how to automatically detect anomalies in the railway catenary. For a detailed review of these, see Liu et al. [LWL19]. Some of the proposed methods focus on the detection of faults in the complex components of the catenary or the pantograph, while only a few have demonstrated the detection of the support structure. For this, Arastounia [Ara15] proposed a method using 3D lidar data. Tang et al. [TJC14] instead used images from a camera mounted on the front of a railway vehicle. The approach of Tang et al. is to first detect the position of the supporting pole arm and thereafter apply a separate algorithm on the found catenary to detect possible faults in its more complex components. The method to detect the pole arm is based on detecting separate lines using the probabilistic Hough transform and looking for the intersections of these lines that correspond to those typical for a support arm. This method showed to be advantageous in the cases of large amounts of interference in the images. A drawback of this approach is the computationally demands, with Tang et al. demonstrating the proposed method at 5 FPS, too slow for practical applications [TJC14].

2.3.4 Railway Traffic Datasets

One reason why the research field of autonomous railway vehicles has not been studied as extensively as for cars is the lack of publicly available datasets that are needed for most applications of Artificial Neural Networks [Nak+19; KOP19]. Even though there exist several datasets with annotated images containing objects that occur in railway traffic, there exist few with railway-specific objects of interest. In 2019, Zendel et al. [Zen+19] presented what is claimed to be the first public dataset for semantic scene understanding covering the rail domain, RailSem19. The RailSem19 dataset is based on video streams obtained from the online community of railway enthusiasts that have uploaded videos recorded from the cabin of the train driver. The images extracted from these videos were manually annotated with pixel-wise semantic labels. The used labels include classes such as 'rail-track', 'traffic-light', 'traffic-sign', 'person', 'car', 'sky' and 'vegetation' among others. By the knowledge of the authors, there exists no other public railway dataset related to the purpose of this project.

3 Theory

This chapter begins with a description of the basics of image and video representation and processing. This covers the image processing used explicitly in the implemented vision system, but also introduces some of the tools applied by the artificial and convolutional neural networks described thereafter. The features of the major convolutional neural network used in this project, the YOLO object detector, are described in more detail. Lastly, the basics of GPS localization and a method for obtaining distances in images are introduced.

3.1 Image Representation and Processing

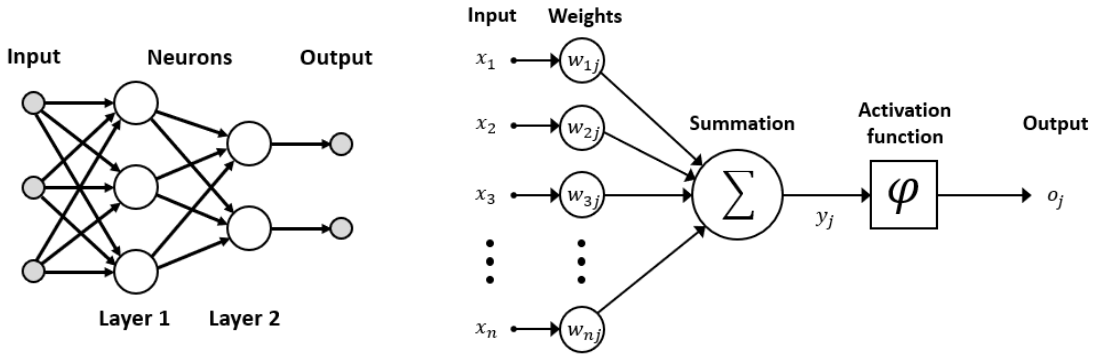
A video film consists of a stream of images, either in color or grayscale. An 8-bit grayscale image is represented by a matrix where each matrix element, called a pixel, takes a value in the range [0, 255], representing its intensity. Thus, each pixel can represent either black, white, or certain levels of gray in-between. A color image instead consists of three color channels and is often represented in the RGB format, where the channels individually represent the red, green, and blue components of the image. Since an image can be represented as a matrix, its pixel intensity derivatives can be computed. A greater derivative indicates a great difference in the intensity of the pixel relative to its neighborhood. By combining the two one-dimensional derivatives along the horizontal and vertical directions the image gradient is formed. The magnitude of the gradient at a certain pixel, d, is obtained by the pixel-wise computation,

$$d = \sqrt{d_x^2 + d_y^2}, \tag{3.1}$$

where d_x and d_y are the one-dimensional derivatives at that pixel. The discrete derivatives can be obtained by convolving the image with a differentiation operator, for example a Sobel filter. A convolution means that a two-dimensional matrix kernel is passed over an image, for each pixel summing the neighborhood pixel values weighted by the elements of the kernel, resulting in a modified image. To reduce the impact of noise in an image it may be blurred. Image blurring can also be performed by convolving the image with a filter, for example of uniform or Gaussian type. An image may also be resized or cropped. Cropping an image means that a specific region of pixels is extracted, without further modifying the pixel values. Resizing an image will however affect the pixel values since many pixels will be merged in the case of downsizing, and new pixels will be created when upsizing an image. In either case, some kind of interpolation will be applied. For a comprehensive introduction to image processing, the reader is referred to Gonzalez and Woods [GW08]. Several algorithms applicable in digital image processing, including the ones mentioned above, are provided by the open-source software library OpenCV. OpenCV has C++, Python, Java, and Matlab interfaces and is compatible with Windows, Linux, Android, and Mac OS [Bra00].

3.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a computational system inspired by biological neurons. An ANN consists of connected artificial neurons, often ordered into layers. The basic idea of such an ordered network is that the output values from a previous layer serve as input to the next, where they are processed. The new output is thereafter sent to the neurons in the next layer. Figure 3.1 (a) shows a simple fully connected neural network with five artificial neurons, ordered into two layers. That the layers are fully connected means that the whole output from the previous layer is available as input to all neurons in the layer thereafter. The conceptual idea of an ANN as the one in Figure 3.1 (a) is that input is fed into the first layer of the network, with the network serving as a black box, providing output from its last neuron layer. The output is generated by propagating the input through the layered structure.



(a) (b)

Figure 3.1: *(a) Schematic overview of a fully connected ANN with five artificial neurons ordered into two fully connected layers. Adaptation of original image from: Offnfopt, Wikipedia, CC BY-SA 3.0. (b) Schematic overview of an artificial neuron. Adaptation of original image from: Christoph Burgmer, Wikipedia, CC BY-SA 3.0.*

Figure 3.1 (b) illustrates the processes within a single typical artificial neuron of index j within its neuron layer. In the figure, the n individual input elements x_i are multiplied by their corresponding weights, w_{ij}, before being summed to form y_j. The sum y_j is then passed to the activation function φ, producing the neuron output o_j. Common activation functions are the Rectified Linear Unit (ReLU) and softmax. The ReLU function is defined as

$$\varphi(y_j) = \max(0, y_j). \tag{3.2}$$

The softmax function is an activation function that is commonly used in the last neuron layer in a classification network. Such a neural network classifies the input, for example an image, into one or several of some predetermined classes. The softmax function is defined as

$$\varphi(y_j) = \frac{\mathrm{e}^{y_j}}{\sum_{i=1}^{N} \mathrm{e}^{y_i}}, \tag{3.3}$$

with N being the number of possible classes. Observe from expression (3.3) that the value of $\varphi(y_j)$ is by the summation dependent on all the neurons in the layer. The output from the softmax function may be interpreted as the probability of the input belonging to a specific class.

An ANN is trained by adjusting the neuron weights such that the network produces the desired output. Within the field of supervised learning, the adaption of the weights is commonly done by backward propagation. In backward propagation, an input is first propagated forward through the network, just as when the network is used for prediction. The error is defined as the difference between the predicted and correct output. The errors are then propagated backward through the network to adjust the weights. The next time the same input is run through the network, the results will probably be more accurate. The training is proceeded by iterating this process with a large number of training samples until the neural network is hopefully accurate enough to make satisfactory predictions. Training the neural network too much may however have negative effects due to overfitting. An overfitted network has specialized in correctly predicting the training data at the expense of generalizability to unseen input. A large and diverse dataset of training samples is therefore needed to successfully train a robust neural network. A more extensive description of ANNs can be found in the compendium by Mehlig [Meh21].

3.2.1 Convolutional Neural Networks

The most basic task of image classification is to classify the motif of an image, for example by picking the most correct out of several possible object classes. While fully connected ANNs can in theory be used for image classification, these are in general outperformed by neural networks with convolutional layers. A convolutional neural network (CNN) [ON15; AMA17] takes three-dimensional image representations as input, including the color depth. The base of a CNN is the convolutional layers that apply matrix convolutions, individually traversing their input with one or several kernels. The kernel matrix can be interpreted as a pattern or feature, and the output of such a convolution represents how well that feature is present in the input. The output from

a convolutional layer is three-dimensional, with the depth as the number of convolutional kernels. By combining convolutional layers it is possible to achieve networks that scan images for features on different scales.

Apart from the convolutional layers, CNNs often also consist of fully connected and pooling layers. A pooling layer down-samples the output from a convolutional layer by extracting the maximums or means of regions in its input. The fully connected layers are often used as the last layers of CNNs. Figure 3.2 shows a typical CNN with both convolutional and fully connected layers. Since the kernels are reused in every convolution within a neuron layer, CNNs often have much fewer weights to train compared to pure fully connected ANNs. This reduces the model complexity and improves the training speed and often also the propagation speed. The qualitative performance in image classification tasks is typically greater for CNNs compared to pure fully connected ANNs. This is mainly due to the usage of the feature-recognizing kernels.

There are among other things two ways to improve the performance of an ANN; batch normalization and data augmentation. Batch normalization is applied within the ANN and normalizes the input data to a layer. This reduces the risk of losing information from saturation by the activation function and improves the training speed [Meh21, p. 148]. Augmentation is used to modify the training data to make it more diverse, and thus indirectly making the neural network more robust to changes in the input. Augmentation alters the already available data and can be viewed as a way to obtain almost new training samples for free. This is an alternative to the often costly and time-consuming process of collecting more annotated data. If the data consists of images, augmentation can be done by cropping, rotating, or shifting the hue, saturation, or brightness of the images. Some more advanced augmentations can be found in [BWL20].

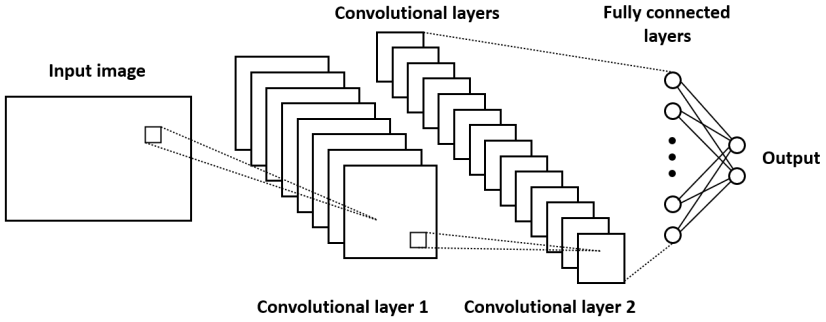
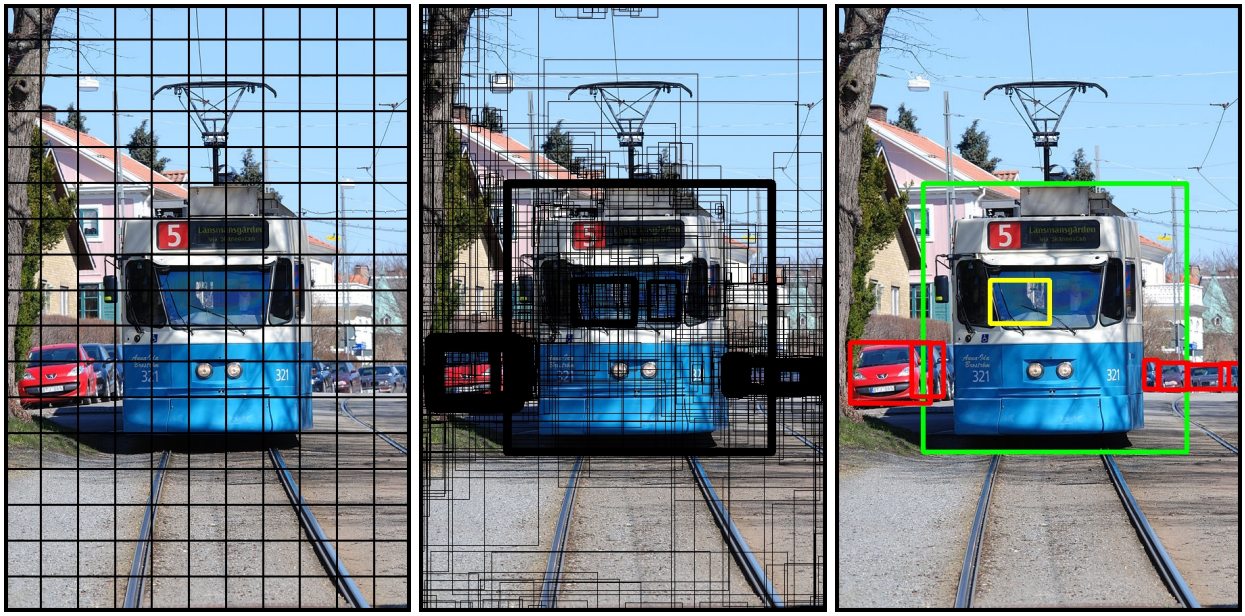


Figure 3.2: *Schematic overview of a CNN with convolutional layers and one succeeding layer of fully connected neurons. Adaptation of original image from: Phung, Rhee, ResearchGate, CC BY.*

3.2.2 YOLO Object Detector Based on a CNN

Artificial neural networks are often used for the object detection task of finding where within an image that a certain object is. One way to find the position of an object is to scan the image with a CNN at multiple scales. This is however a time-consuming process that in most cases cannot run in real-time. Another approach is used by the You Only Look Once (YOLO) neural network [Red+16], which works just like its name indicates. YOLO only scans the image once and directly outputs bounding boxes confining the objects it finds. It also classifies the bounding boxes into one of several predetermined classes.

The procedure of the detection and classification is as follows. The YOLO network first divides the image into a grid, illustrated in Figure 3.3 (a). Each grid cell is responsible for detecting the objects with centers within that cell. Every grid cell predicts a certain number of bounding boxes, regardless of whether they are believed to contain objects or not. Each bounding box has a confidence score representing the certainty that it contains an object. Apart from the confidence, each box is predicted with a class probability for each object that the network can detect. The geometry of each predicted bounding box is represented by four values; the two image coordinates of its center, together with its width and height. Figure 3.3 (b) shows examples of predicted bounding boxes. In the figure, bounding boxes of too low confidence have been removed to increase visibility. The final classifications are obtained by filtering the bounding boxes based on their confidence values, removing those with a confidence lower than a predefined threshold. After this, the non-maximum suppression technique is applied to remove overlapping bounding boxes, further described in section 3.2.3. The remaining boxes are labeled with the class of greatest probability, and an example result is illustrated in Figure 3.3 (c). In the image, eight objects of three different classes have been detected.



(a) (b) (c)

Figure 3.3: *Illustration of how the YOLO neural network detects objects in an image. (a) The cell grid. (b) The proposed bounding boxes, with the line widths proportional to the corresponding bounding box confidence score. (c) The final, filtered predictions with the individual object classes represented by the colors of the bounding boxes. Adaptation of original image from: Historiker, Wikipedia, CC BY-SA 3.0. Lines and boxes have been drawn on the original image.*

There currently exist four main versions of the YOLO neural network, each successor introducing new features. The above description of YOLO is consistent with the most recent, the fourth version. The first version, YOLOv1, consists of both convolutional and fully connected layers, and can only predict one class type per grid cell. YOLOv2 is a pure CNN, without fully connected layers, and also utilizes batch normalization. The second version introduces multi-scale training, which means that the size of the input is changed during training. This leads to increased robustness for detections of objects in images of different sizes. Changing the input size is possible since YOLOv2 only consists of convolutional layers, which can easily be scaled without adjusting their weights. YOLOv2 and later versions also classify each bounding box separately. YOLOv3 is larger and more accurate, but slower. YOLOv4 has an improved augmentation compared to the previous versions. The first three versions augment the training images by random cropping, rotations together with hue, saturation, and exposure shifts. YOLOv4 also includes augmentation by a mix-up of two images with different opacity, images pasted onto each other, mosaics of several images, blur, and more. YOLOv4 also uses a genetic algorithm to select hyper-parameters during the first training iterations. In addition to the four main versions described above, each version comes with a smaller variant, named YOLO-tiny. The YOLO-tiny networks have fewer layers than their larger versions and therefore lower capacity and accuracy, but are instead much faster and can be run in real-time on lower-end devices. For further details about the differences between the YOLO versions, see the three first versions from Redmon and Farhadi [Red+16; RF16; RF18] and the improved YOLOv4 by Bochkovskiy et al. [BWL20].

3.2.3 Non-maximum Suppression

Since one or more of the YOLO grid cells can propose several bounding boxes of the same class, a single object may get detected more than once. Therefore the Non-Maximum Suppression (NMS) [Red+16] technique is used by YOLO for choosing the most correct bounding box, and removing overlapping ones. After the removal of bounding boxes of too low confidence, as described earlier, the NMS algorithm acts on the bounding boxes one class at a time. For each class, the bounding boxes are ranked by their confidence scores, starting with the one of greatest confidence. This bounding box is compared with all other bounding boxes of the same class, removing those that have an Intersection Over Union (IOU) greater than a predefined threshold. IOU is a measurement of overlap and is computed by dividing the intersection area of two regions by the area of their

union. When all bounding boxes with enough overlap of the box of greatest confidence have been removed, that bounding box is saved. Thereafter the same procedure is repeated with the bounding box of the greatest confidence of the remaining ones until all boxes have either been saved or removed.

3.3 Datasets for Training Neural Networks

The neural networks used in this project are trained by supervised learning. The networks have therefore been trained to detect annotated objects in a specific dataset. The training dataset thus limits what the neural network will be able to recognize. There exist several annotated image datasets for public use, for example the PASCAL VOC and MS COCO datasets.

3.3.1 PASCAL VOC Dataset

The Pattern Analysis, Statistical Modelling And Computational Learning Visual Object Classes (PASCAL VOC) [Eve+] challenge is a competition and workshop that was running annually between 2005 and 2012. The principal challenge is based on the detection and classification of objects in images supplied by the competition hosts. The 2012 PASCAL VOC classification and detection datasets consist of 11 530 color images of different sizes, altogether containing 27 450 annotated objects. The annotated objects are divided into 20 different classes, including 'person', 'bicycle', 'bus', 'car', 'motorbike', and 'train'. Each object is annotated with its class in addition to the position and size of a rectangular bounding box.

3.3.2 MS COCO Dataset

The Microsoft Common Objects in COntext (MS COCO) dataset is as the PASCAL VOC dataset supplied as the base for a still active annual competition, the MS COCO Object Detection Task [COC]. The most recently provided MS COCO dataset was published in 2017 and consists of 123 000 annotated images with objects divided into 80 classes, including 'person', 'bicycle', 'motorcycle', 'car', 'train', 'truck', and 'bus'. As for the PASCAL VOC dataset, these object annotations contain a class label and information about the bounding boxes.

3.4 Projective Distances in Images

Being able to estimate actual distances in images is useful, but also often difficult. Following are methods, proposed by the authors, for measuring distances in images by adapting existing concepts to the railway environment. In a photography, the width in-between two objects running in parallel away from the camera, for example a pair of railway rails, decreases linearly with the distance to the lens. At some height from the image bottom, the so-called vanishing point occurs, where the objects virtually intersect each other [Sze11]. To exemplify, the vanishing point of a railway track is normally the horizon. The image track width in units of pixels, w, at the image height y_i, can thus be expressed as

$$w(y_\mathrm{i}) = \begin{cases} \frac{w_\mathrm{bottom}}{y_\mathrm{horizon}}(y_\mathrm{horizon} - y_\mathrm{i}) & \text{for } y_\mathrm{i} \leq y_\mathrm{horizon} \\ 0 & \text{otherwise,} \end{cases} \tag{3.4}$$

with the y-axis pointing upwards and $y_\mathrm{i} = 0$ at the image bottom. The height of the horizon, y_horizon, is provided in units of pixels. Here w_bottom is the track width at the bottom of the image. Observe that above the horizon, the track width is interpreted to be zero since it does not exist there.

The photographic linear magnification formula from optics [Ope20] states that,

$$M = \frac{d_\mathrm{i}}{d_\mathrm{o}} = \frac{h_\mathrm{i}}{h_\mathrm{o}}, \qquad \Longrightarrow \qquad d_\mathrm{o} = \frac{d_\mathrm{i}}{h_\mathrm{i}} h_\mathrm{o}, \tag{3.5}$$

where M is the magnification relating the actual heights and distances to those in the image. Here h_o is the height of an object on distance d_o from the camera lens, with h_i as the height of the object projected onto an image on distance d_i from the lens. Figure 3.4 illustrates the definitions of these four variables. Observe that the heights used in the photographic linear magnification formula can be exchanged by any measurement of

extent, such as width. The units of all four variables are in meters, but the unit of h_i can also be converted to pixels by a constant k.

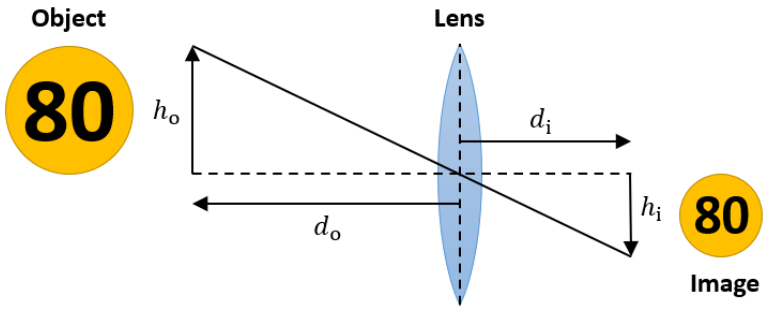


Figure 3.4: *Illustration of the principle of the photographic linear magnification by the optics of a camera. Adaptation of original image from: Redbobblehat, Wikipedia.*

The photographic linear magnification formula states that the actual distance to an object can be calculated if for example its real width or height is known. In the case of a railway track, the actual width of the track is always constant. The projected track width at any height in the image is also known from expression (3.4). Thus the actual distance to an object standing on the ground can be obtained by the second expression in (3.5) combined with the formula for the track width in (3.4). Altogether, the distance in meters to an object in an image can be computed by,

$$d(y_\text{i}) = \frac{C_1}{w(y_\text{i})} = \frac{C_2}{y_\text{horizon} - y_\text{i}}, \qquad \text{with} \qquad C_2 = C_1 \frac{y_\text{horizon}}{w_\text{bottom}}, \qquad \text{and} \qquad C_1 = k d_\text{i} h_\text{o}. \tag{3.6}$$

The introduced constant C_1 needs to be calibrated only once for a specific camera, while y_horizon and w_bottom are adjusted for each fixed camera position. Observe that this formula is only applicable to get distances to objects standing on the ground in a flat environment.

The distance to an object can be estimated even if it is not standing on the ground, assuming that the width of the object, w_o, is known. This approach although also requires knowledge about the image width in pixels of the object, w_i. The distance to the object is then obtained from the second expression in (3.5),

$$d(w_\text{i}) = C_3 \frac{w_\text{o}}{w_\text{i}}, \qquad \text{with} \qquad C_3 = k d_\text{i}, \tag{3.7}$$

where it suffices to calibrate C_3 once for all objects.

3.5 GPS Localization

The Global Positioning System (GPS) is a utility owned by the United States Government and operated by the United States Space Force [Nata]. The space segment of the GPS infrastructure is composed of at least 24 satellites simultaneously, orbiting Earth in a medium orbit. The satellites transmit one-way signals to receiver devices that can compute their position, velocity, and local time with accuracy depending on the number of satellites that are monitored. The constellation of the satellites is designed such that there should always be at least four satellites visible simultaneously at each point on the surface of the Earth. This is necessary since a GPS receiver requires four GPS satellites to make an accurate estimation of the position. The location accuracy from the GPS is typically within a few meters [Natb]. The speed-accuracy is however greater, with the error less than some centimeters per second. The accuracy however depends on the surrounding environment, and for example buildings and tunnels may disrupt the connections.

4 Methods

The vision system developed in this project is mainly based on two algorithms that were developed specifically for this application; track detection and object detection. These two parts are responsible for robustly extracting information from an input image regarding the visible tracks and various objects.

4.1 Track Detection Algorithm

The track detection algorithm is divided into four main parts; preprocessing of the input image, region growing to detect rails, pairing of rails to form tracks, and lastly searching for possible railway switches and incoming tracks. Figure 4.1 shows a flowchart of the track detection with an overview of the processes included in the four parts of the algorithm. Following is a distinction made between the track that the vehicle is running on, further on referred to as the main track, and the sidetracks.

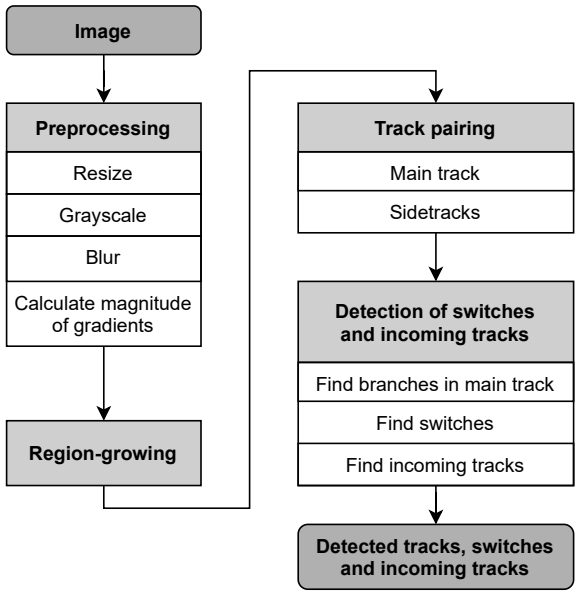


Figure 4.1: *Flowchart of the track detection algorithm structure, which is divided into four main steps; preprocessing of the image, region growing to find rails, pairing of rails into tracks, and lastly detection of railway switches and incoming tracks.*

4.1.1 Image Preprocessing

The input image to be analyzed is first scaled to 1280×720 pixel resolution. The image is thereafter converted to grayscale and blurred by Gaussian blur and additional uniform blur along the vertical axis. The Gaussian blurring is applied on the bottom half of the image to reduce image noise in objects close to the camera while keeping the details of objects far ahead. The vertical blur is used to exaggerate the main track rails, and is only applied in a pyramidal-shaped region around the main track to not affect the detection of non-vertical sidetracks. After blurring, the image derivatives in the vertical and horizontal directions are computed with Sobel filters and combined to form the gradient image. The magnitudes of the gradients are then computed by expression (3.1) in section 3.1.

4.1.2 Region growing

The track detection was implemented with inspiration from the proposed method of Qi et al. [QTS13], which uses the concept of region growing to extract the individual rails in an image. Compared to the approach of Qi et al., the region-growing algorithm implemented for this specific application is applied on the magnitudes of the gradients image instead of on HOG features. The algorithm is here also expanded to enable detection of

rails starting on the image sides. The proposed region-growing algorithm processes a single video frame at a time, without carrying any information in-between frames.

The concept of the implemented region-growing algorithm for rail detection is presented in Algorithm 1. The conceptual idea is the following: Each rail is grown from a starting seed, a pixel on the bottom or either side of the image. The neighboring pixels of that seed are each added to the rail if they fulfill a certain condition. The neighbors of a pixel are here defined as the pixels in the row above within a maximum horizontal distance away from it, see the example in Figure 4.2. By the definition of neighbors, the rails are forced to grow upwards, which is naturally inspired from that rails run vertically in the ego-perspective of trains. The condition that needs to be fulfilled for a pixel to be added to the rail is that its gradient should have a magnitude greater than a certain threshold. Thus the proposed region-growing algorithm detects the rail edges, and not the whole rails. The threshold is adaptive, mainly to deal with varying lighting conditions, and is based on the maximum and mean values of the gradient magnitudes within a region on the bottom of the image. The adaptive threshold is also depending on the position of the considered pixel, and thus varying thresholds are used in different image regions. The pixels that are added to the rail will then act as new base points for rail growing, with their pixel neighbors evaluated next. Thus, the algorithm continues row by row until no more pixels are added or all pixels have already been evaluated. Thereafter the region growing is restarted from a new seed, representing another rail. Only those pixels that have not already been assigned to a rail are then evaluated. So the algorithm continues until all seeds have been grown from, and the result is an image with connected components of pixels belonging to different rails. The region-growing algorithm used also applies some extra conditions on the possible rails found to remove those who clearly do not correspond to real rails. For example cannot the pixels that are above the specified horizon be rails and are thus not taken into consideration in the region growing.

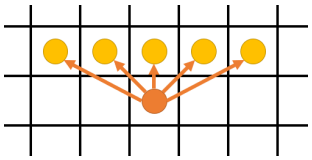


Figure 4.2: *Five neighbor pixels, in yellow, within the maximum horizontal distance of two pixels from the main pixel, in orange. Adaptation of original image from: Qi et al. [QTS13].*

Algorithm 1: Concept of the region-growing algorithm for rail detection. Details of the functions ComputeAdaptiveThreshold() and GetPixelNeighbors() are described in more detail in the text.

Input : Magnitudes of gradient image: imageGradients, list of starting seeds: seedList
Output : Image with classified pixels: classifiedImage

```
classifiedImage.SetToZero();
currentRail = 0;
foreach seed ∈ seedList do
    start = 0, end = 0;
    growList.Clear();
    growList.Add(seed);
    while start ≤ end do
        currentPixel = growList[start];
        neighborsList = GetPixelNeighbors(currentPixel);
        foreach neighborPixel ∈ neighborsList do
            neighborGradient = imageGradients.At(neighborPixel);
            threshold = ComputeAdaptiveThreshold(imageGradients, neighborPixel);
            if neighborGradient ≥ threshold and classifiedImage.At(neighborPixel) == 0 then
                classifiedImage.At(neighborPixel) = currentRail;
                growList.Add(neighborPixel);
                end++;
            end
        end
        start++;
    end
    currentRail++;
end
```

4.1.3 Track Pairing

The possible rails found by the region-growing algorithm are thereafter analyzed to pair them into tracks. First, all combinations of the possible rails found are evaluated pair-wise. If the distance between the starting seeds of the two rails is within a specified interval, then the rails form a potential track. Assuming that the camera is mounted fairly centered above the rails, the main track is taken to be the rail pair closest to the middle of the image. The rest of the potential tracks are then labeled as sidetracks, such that no rail can be paired with more than one other. Unpaired rails are discarded. An example of the resulting rail pairs is seen in Figure 4.3, with the main track and two sidetracks. Since signs and signals only apply to specific tracks, a track layout was introduced to know the current track status of the vehicle. Based on what sidetracks that are detected, the vehicle is assumed to be running on either a single, middle, left, or right track. If a train is detected, see section 4.2, it is assumed that it is running on an existing sidetrack. This information is included when deciding on the current track status.

Figure 4.3: *Example of detected rail pairs forming valid tracks, with the main track colored in red and the two sidetracks in green. Original image credit: Jan Kivisaar [Kiv12], modified with permission.*

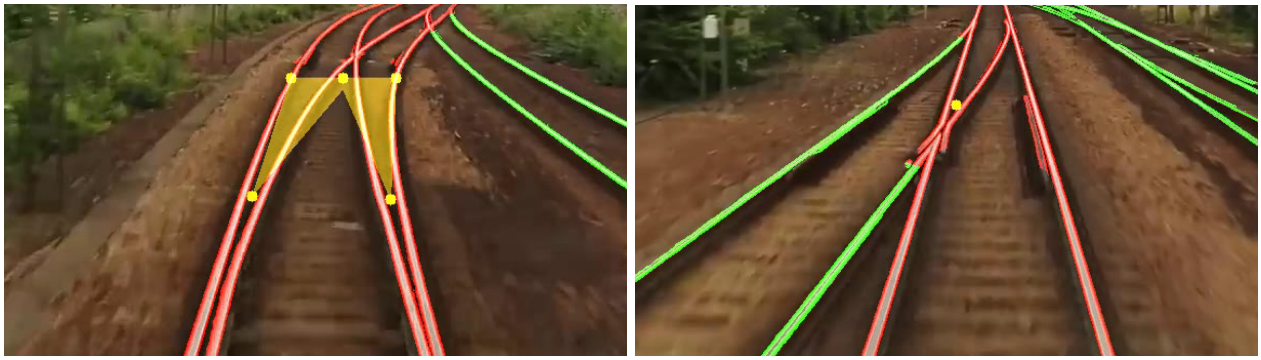
4.1.4 Detection of Switches and Incoming Tracks

A railway track may split up into two at a railway switch. At these switches, the two rails forming a track will branch into four, see the example in Figure 4.4 (a). The detection of switches is therefore based on finding so-called branching of the rails. Only the main track is checked for branching. Each rail in the main track is individually processed to find the points where it branches into two rails. At a certain distance past the two branching points of a railway switch, the right branch from the left rail and the left branch from the right rail intersect. At the intersection height in the image, the distance between the two branches of each rail has grown to be equal to the width of the track. The implemented algorithm finds the separation points where this happens. If a branching point and the two separation points are found for both rails, a switch is assumed to have been detected. These four points, as well as the point of intersection, are marked with yellow dots in Figure 4.4 (a) for a detected switch. In the video frames after a switch has been detected, the algorithm searches for branching of only a single main track rail. Such branching occurs in one of the rails when the two tracks separate. If such a branch is found on the left rail of the main track, then the new sidetrack is to the left of the main track and vice versa.

In a similar way as for the switches, left or right tracks incoming from the sides of the main track are detected by searching for branching of only one of the two main track rails. The main concept is as follows; if the left rail branches, then there is an incoming track from the left, and vice versa. This is visualized in Figure 4.4 (b), where there is an incoming track from the left, and thus branching of the left main track rail.

4.1.5 Track Warning Zone

When the main track has been identified, a vehicle and pedestrian warning zone is defined. The warning zone is visualized in Figure 4.5. The extent of the warning zone is based on the width, length, and curvature of the detected main track and thus changes from frame to frame. The warning zone width is at each image row a



(a) (b)

Figure 4.4: *(a) Example of a detected railway switch. The branching points are marked by the two lower yellow points. The upper three points illustrate the intersection and separation of the two new tracks. (b) Example of a detected incoming sidetrack from the left. The branching of the left main track rail is marked by a yellow dot. Original image credit: Jan Kivisaar [Kiv12], modified with permission.*

multiplication factor times the width of the main track. The warning zone is divided into three sub-regions parallel with the main track, one inner and two outer regions. The warning classification of a detected vehicle or pedestrian can thus be abstracted to if it is within the inner or an outer warning zone.

Figure 4.5: *The warning zone, with the inner and outer regions distinguished by different colors. Original image credit: Jan Kivisaar [Kiv12], modified with permission.*

4.2 Object Detection Algorithm

The object detection is performed in two steps. First, a YOLOv4-tiny neural network, originally developed by Redmon and Farhadi [Red+16] and further improved by Bochkovskiy et al. [BWL20], is used to detect objects in an image. Five of the object classes detected by the YOLO neural network, the speed sign and the four considered types of signals, are then further processed by other classification neural networks to extract their messages. To be able to train the YOLO and the five classification neural networks, six datasets were created.

4.2.1 Datasets for Training the Neural Networks

As explained in section 3.2, the neural networks used in this project learn to detect and classify objects by training on annotated image data. Some objects of interest for an autonomous railway vehicle to detect are found in the PASCAL VOC and MS COCO datasets described in section 3.3. Many of the images in these two datasets are although of a context not relevant to be able to analyze. Therefore, images from both the PASCAL VOC and MS COCO datasets were manually extracted by the authors to form new, refined data. The refined PASCAL VOC and MS COCO datasets contained 1505 and 7738 images respectively, with a combined total of 65 633 objects of the seven classes; bicycle, bus, car, motorbike, person, train and truck. The exact distribution of these annotated objects is presented in Table 4.1.

Figure 4.6: *An example of a typical annotated image in the dataset. The image visualizes a road crossing with barriers on both sides of the single track, hindering a person on a motorcycle and a car from entering the track. In connection to the crossing there is also a road crossing signal with the associated V-sign. There are also two catenary support poles and a warning sign marked. Original image credit: Jan Kivisaar [Kiv12], modified with permission.*

In addition to the seven object types extracted from the PASCAL VOC and MS COCO datasets, there are many objects common in Swedish railway traffic that do not occur in any appropriate and publicly available datasets. Using data from foreign railway infrastructure would not suffice since the vision system is to be applied on Swedish railways. Therefore, new image data was collected and annotated by the authors. This data is based on images extracted from video streams captured from the ego-perspective of trains. All the collected videos are shared online by the train enthusiast Jan Kivisaar [Kiv12], and are used with permission. A total of seven videos, with more than five hours of data were collected. The captured environments are all from railways in Sweden and recorded in different light and weather conditions. None of the videos are captured in snowy terrain or darkness. The videos used are all of HD resolution, 1280 × 720 pixels. Programs were created by the authors to manually extract and annotate individual frames from the videos.

First railway data for the training of the YOLO neural network was created. For this image data the annotations are on the same bounding box format as for the PASCAL VOC and MS COCO datasets. A typical example of an annotated railway image in the YOLO dataset is provided in Figure 4.6, which contains eleven objects with marked bounding boxes. For the YOLO dataset, the authors collected and annotated 2892 images with bounding boxes for a total of 16 113 objects divided into 19 classes. The distribution of annotated objects among these classes is provided in Table 4.1. The data collected by the authors was then merged with the refined data extracted from PASCAL VOC and MS COCO to form the whole YOLO dataset. For each of the railway images two copies were added, the original and a vertically mirrored image.

As seen in Table 4.1, there are object classes in the YOLO dataset that are represented by only a few hundred samples, for example the ATC speed up sign class. Those classes are underrepresented in the dataset

compared to the number of bounding boxes of for example the car and pole classes. The 19 considered object classes are all among the most frequently occurring objects in the available video material. As stated in section 2.2, there are more objects that are of interest to detect for an autonomous vehicle applied in the Swedish railway environment. These are although rare in the obtained videos, and would thus be represented by only a few annotated images, and therefore risk to be practically unusable.

Table 4.1: The number of occurrences for each of the 19 classes in the YOLO dataset. The 'Authors' data refers to the images collected and annotated by the authors while 'Others' refers to data from the PASCAL VOC and MS COCO datasets.

Class	**ATC speed down sign**	**ATC speed up sign**	**Barrier**	**Bicycle**	**Bus**	**Car**	**Road crossing**
Authors	367	174	449	0	12	801	421
Others	0	0	0	1634	3622	23 797	0
Class	**Distant road crossing signal**	**Distant signal**	**Main signal**	**Motorbike**	**Person**	**Pole**	
Authors	232	484	2038	7	294	8547	
Others	0	0	0	2547	27 146	0	
Class	**Road crossing signal**	**Speed sign**	**Train**	**Truck**	**V-sign**	**Warning sign**	**Total**
Authors	362	348	278	58	428	813	16 113
Others	0	0	2945	3942	0	0	65 633

In addition to the YOLO dataset described above, five smaller classification datasets were also created. These five datasets were used to train the neural networks for the classification of the messages of the signals and the speed sign. These classification datasets are thus each associated with the main signal, distant signal, road crossing signal, distant road crossing signal, and speed sign respectively. Instead of being annotated with bounding boxes, each image in these classification datasets only contains a single object and was labeled with a description of its lit lights or conveyed speed limit. Regarding signals, samples were collected for each possible combination of lit lights that form the 14 possible signal messages listed in Appendix A.1. For each of the four signal classification datasets, samples without any lights lit were also added. Regarding the speed signs, instead of combinations of lit lights, signs with different speed limits were collected. Images of various 'false' objects were added to all five datasets. The labels used and the distribution of these are presented in Appendix A.2 for each of the classification datasets. The images of the five datasets were augmented by creating three copies of different brightness for each image. All images were thereafter rotated around their centers by random angles between -20° and +20°. Ten images from the five classification datasets are presented in Figure 4.7. In the figure it is seen that the rotations introduce black edges around the original images.

Figure 4.7: *Examples of augmented images from the five classification datasets containing samples of main signals, distant signals, road crossing signals, distant road crossing signals, and speed signs. The images have here been resized to be square. Original image credit: Jan Kivisaar [Kiv12], modified with permission.*

4.2.2 YOLO Object Detection Neural Network

The YOLOv4-tiny network used in the project is built upon 21 convolutional layers and several max-pooling layers, routing layers to skip connections, and up-sampling layers to make defections on a smaller scale. The network uses a grid of size 19×19 and predicts three times as many possible bounding boxes. The YOLO network was trained with the Darknet framework from Alexey [Ale21] for 250 000 iterations on the YOLO dataset. Darknet [Red13] is an open-source framework for neural networks written in C and CUDA. The input image size of the YOLO network was 608×608 pixels. The neural networks were trained with the built-in data augmentation in Darknet, including random rotation, saturation, exposure, hue, and resizing of the network. The YOLO CNN was after training first converted to the ONNX format and thereafter converted to the TensorRT format by the code provided by Jung [Jun21]. TensorRT [NVI16] is a high-performance framework from NVIDIA built on CUDA used to optimize and deploy neural networks in real-time applications.

4.2.3 Classification Neural Networks

To obtain the messages of the signals and the speed signs found by the YOLO neural network, each of these objects is further processed by a classification neural network. For each of these five objects, the authors thus created a neural network and trained it on the corresponding classification dataset with PyTorch. Before being classified by its corresponding neural network, the image containing only the detected object is extracted from the original video frame by cropping it based on the object bounding box. The cropped image is thereafter resized to 32×32 pixels. The images of distant road crossing signals and the speed signs are both converted to grayscale since the color information is assumed to not be needed for their classification.

The five classification neural networks are all of the same structure, except for the number of color channels in the input and the number of classes in the output. The complete layout of the neural networks is presented in Table A.6 in Appendix A.3. The five neural networks consist of three convolutional layers with intermediate max-pooling layers, followed by two fully connected layers. The ReLU activation function is applied after all layers, except for the last, after which the softmax function is used. The output from the classification networks is thus interpreted as class probabilities. The final classification is taken to be the class of greatest probability.

The five classification networks were as mentioned trained on the corresponding classification datasets presented in section 4.2.1. Thus, the neural networks are in theory capable of classifying all possible signal combinations that can occur on Swedish railways. As can be seen in Table A.5 in Appendix A.2, the dataset for speed signs only contains a total of 18 classes, including 'False'. Thus there are several speed limits that may occur in railway traffic that the vision system cannot identify correctly.

4.2.4 Detection of Catenary Support Structure

A first step in monitoring the railway catenary is to locate the poles of the support structure. The used YOLO neural network is trained to detect these poles fairly close to the camera. A method was developed to keep track of individual detected poles over several video frames. The conceptual idea of the method is here described.

All detected poles are first characterized based on their side of the main track. For each of the two sides, only the pole closest to the camera is considered. Each of these two is compared to the corresponding pole from earlier video frames. Based on the pixel distance between these, a decision is made of whether the current pole is the same as the one being tracked since earlier. If so, the position of the tracked pole is updated. If the position of the tracked pole has not been updated for a certain time, then that pole is assumed to have been passed by the vehicle and is considered detected.

4.3 Temporal Robustness of Detected Objects

Since the object and track detection algorithms are not always perfectly accurate for individual frames, robust detection of objects over time is necessary to avoid false detections. The methods implemented in this project for robust object detection are of varying kinds. All are although based on the same concept, which is illustrated by the flowchart in Figure 4.8. The idea is to count the number of video frames in which an object is detected, and only consider it robustly detected if it has been detected for enough time. The number of video frames since the object was last detected is stored, and when it has not been detected some time, its occurrence counter is reset.

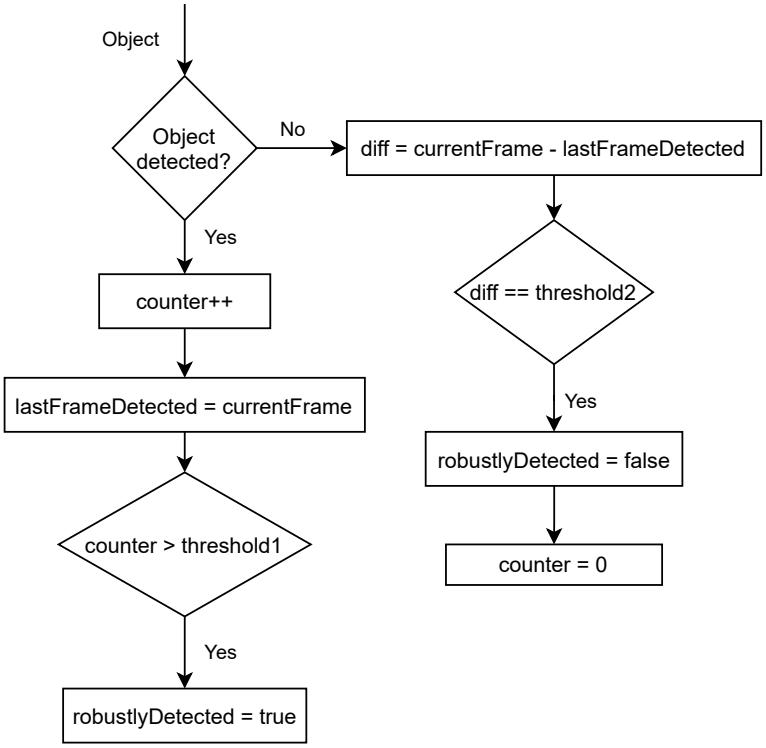


Figure 4.8: *Schematic flowchart of how the object detection is made temporal robust.*

5 Implementation

The vision system was together with GPS and localization functionalities implemented as Robot Operating System (ROS) nodes. ROS [ROS] is a framework for creating robot software and is based on executable programs called nodes that communicate via messages. Each message has a topic that a node can publish to, and a node can also subscribe to the specific topics that are relevant to it. This chapter will first introduce the functionalities of the vision, GPS, and localization nodes and end with describing the hardware used to run the developed programs.

5.1 Vision Node

All code for the vision node was written in C++ in Microsoft Visual Studio and is available on GitHub [LINK]. The programs were also to a great extent created using functions and classes from the OpenCV C++ library. The structure of the implemented vision system is summarized in the flowchart in Figure 5.1. An image obtained from a video stream is first passed to both the track detection algorithm and the YOLO neural network. To increase the speed, these are executed in parallel with the YOLO object detection network running on the GPU. The objects found by the YOLO neural network that can be further analyzed, the speed signs and the four signals, are passed as cropped images to the corresponding classification neural network. All found and classified objects and messages are then post-processed to achieve robust detection as described in section 4.3.

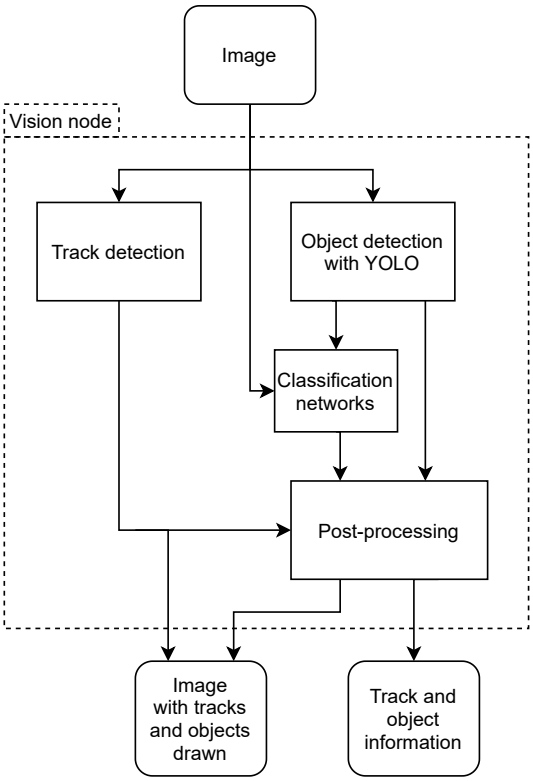


Figure 5.1: *Flowchart of the vision node structure with the track and object detection programs running on parallel threads. Information from the two parts is then merged in the post-processing.*

The post-processing is of varying kinds for the different objects. Common to all is that they have some implementation of temporal robustness. For some objects the post-processing utilizes information from both the track and object detection. An example is seen in the detection of pedestrians and vehicles, for which their bounding boxes are compared to the warning zone to know if the objects are too close to the main track. The process of identifying the signs and signals on the applying side of the main track is another example, here using the detected track status and the position of the main track. Flashing signals are recognized by having robustness measurements combining counters from the signal with and without the flashing lights lit. The distance to each detected object is computed by the two approaches described in section 3.4, with the approach used depending on if the object is standing on the ground or has a known width. It is assumed that vehicles, pedestrians, poles, road crossings, and barriers are always positioned on the ground and that all sign and signals have known widths.

Lastly, the found tracks and objects can be displayed, as in the example in Figure 5.2. Various information is also published via ROS to be available for the other parts of the autonomous vehicle responsible for localization, decision-making, and steering. The information from the vision node is published to the ROS topic *vision_topic*. This information includes the current speed limit, signal and sign messages, warnings regarding upcoming main signals and road crossings, and the track status etc. The published information in full is presented in Table A.7 in Appendix A.4. A subset of this information is displayed in the upper left corner of the image in Figure 5.2. The detected speed limit is displayed by the illustration of a speed sign in the upper right corner of the image.

Figure 5.2: *Example of the output image from the vision system, with identified tracks, bounding boxes of detected objects, and the warning zone. Each bounding box is described by its associated class name and its estimated distance from the vehicle, in units of meters. The main track is drawn in red with the sidetrack in green. The currently interpreted speed limit is shown in the upper right corner, with other information shown within the blue box on the upper left. Original image credit: Jan Kivisaar [Kiv12], modified with permission.*

5.2 GPS and Localization Nodes

A GPS node was developed in Python to fetch GPS data and publish it to the topic *gps_topic* via ROS. The GPS data includes the latitude and longitude coordinates, the speed in units of knots, the number of satellites that the GPS module is connected to, and the current time. The GPS node only publishes data when the GPS module is connected to at least four satellites.

A third ROS node, the localization node, was developed to store information from the traveled route for future analysis. Thus the localization node subscribes to both *vision_topic* and *gps_topic* and combines their information. The current version of the localization node stores the fetched and combined information from each processed video frame as a block of text in a text file. The text file can then be loaded and processed at later times.

5.3 Hardware

To enable the vision system to be used by the autonomous vehicle without requirements on any additional hardware, the developed programs were installed and run on an NVIDIA Jetson Nano single-board computer. To the computer, an Adafruit Ultimate GPS Breakout was connected. When videos captured in real-time are to be analyzed, cameras can be connected directly to the Jetson Nano. Thus the full vision capabilities can be utilized by just providing power to the Jetson Nano and fetching the published vision information via ROS by an Ethernet or Wi-Fi connection. The Jetson Nano does not have built-in Wi-Fi support but can be connected to an external Wi-Fi module, for example the Intel Dual Band Wireless-AC 8265 [Int16], to enable both Wi-Fi and Bluetooth connections. The camera setup was tested by connecting two 12.3 megapixel SONY IMX477 camera modules from Arducam [Ard20] to the Jetson Nano via its CSI interface. The two cameras can with 4032×3040 resolution capture the rear and front view of the track trolley in 30 FPS.

5.3.1 NVIDIA Jetson Nano

The NVIDIA Jetson platform [NVI19] is a class of small and energy-efficient, single-board computers with built-in NVIDIA GPUs, specifically developed for artificial intelligence, computer vision and robotics applications. The smallest NVIDIA Jetson module, the Jetson Nano, has a 128-core Maxwell GPU, a Quad-core ARM CPU at 1.43 GHz, and 4 GB LPDDR4 RAM. Figure 5.3 shows an image of the Jetson Nano. The Jetson platform has its own operating system based on Ubuntu and includes some high-performance libraries for GPU usage, for example CUDA, cuDNN, and TensorRT.

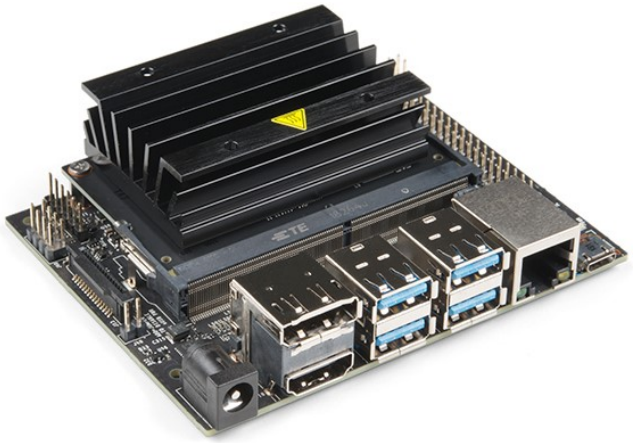


Figure 5.3: *The NVIDIA Jetson Nano. Image credit: SparkFun Electronics, Wikipedia Commons.*

5.3.2 Adafruit Ultimate GPS Breakout

The Adafruit Ultimate GPS Breakout [Lad21] is a GPS receiver breakout board based on an MTK3339 GPS module that can track up to 22 satellites simultaneously. The breakout has an integrated ceramic patch antenna, with the possibility to add an external active antenna for improved performance. It has an update frequency at up to 10 Hz and position and velocity accuracy of 3 m and 0.1 m/s respectively.

6 Evaluation and Results

The implemented vision node runs on the NVIDIA Jetson Nano in up to 15 FPS. To evaluate the performance of the implemented solutions, two approaches were used, distinguishing between the detection of tracks and the more abstract detection of objects. This difference is mainly due to the introduction of the robust detection of objects over time, described in section 4.3.

6.1 Evaluation Methodology

The evaluation was performed on eight videos from the ego-perspective of the train, all captured on railways in Sweden. None of the evaluation videos were used in the creation of the training datasets. As for the railway images in the training datasets, all videos used for evaluation are provided by Jan Kivisaar [Kiv12]. The eight videos are of varying duration, with six of them having lengths between eight and twelve minutes. The two others are of three and 38 minutes lengths respectively. Six of the videos were captured in the same resolution, 1280×720 pixels, and two with resolution 1080×1920 pixels. All videos were captured in daylight, without impact from rainfall or snow. No evaluation was performed when the train was traveling inside of a tunnel, mainly due to insufficient lighting conditions, but also since almost no such data was provided in the training datasets.

6.1.1 Method for Evaluating the Track Detection

The evaluation of the track detection algorithm is divided into two parts. The detection of switches and incoming tracks is evaluated with an approach similar to the one for object detection and is discussed further in section 6.1.2. The evaluation of the detection of the actual main and sidetracks is performed automatically frame by frame with annotated videos. Each video is annotated by the authors with intervals labeled by the track status of the train; single, left, right, or middle track. It is also assumed that the main track is existing and detectable in every video frame. The evaluation procedure of the track detection is based on comparing the track status detected by the implemented programs and the actual track status from the annotations. The number of video frames where the main track is detected is also counted.

6.1.2 Method for Evaluating the Object Detection

Due to the temporal robustness introduced in section 4.3, all objects have to be identified over several video frames before being declared robustly detected. Flashing signals are examples of where this approach is necessary since their messages cannot be correctly classified from only a single moment. Thus, the full capabilities of the vision system cannot be evaluated via frame-wise evaluation. Instead, a manual, context-based test is introduced for evaluation of the object detection.

The evaluation was performed by running the vision system on the eight evaluation videos, with manual monitoring by the authors during runtime. The encountered objects were manually counted while simultaneously monitoring the information conveyed by the vision system. An object was only accounted for once, independent of the number of video frames occurring. For all objects, the false detections and missed objects were also counted. Since the vision system is focused on detecting objects in the vicinity, vehicles were only taken into consideration if they appeared within the main track warning zone. Since the objects of type bicycle, bus, motorbike, and truck never appeared more than once each over the eight videos, these were grouped with the car class to form a joint object class called 'vehicle'.

In addition to what is mentioned above, only signals and signs that apply to the main track are considered for the evaluation. Due to the low resolution of the available videos, the small supplement version of the ATC speed signs and the distant signal sign, see Figure 2.3 (b) and (g), were omitted in the evaluation since they were not always distinguishable even for the authors. The stand-alone ATC speed up and down signs seen in Figure 2.3 (c), (d), and (e) were although used. In addition to the detection of the speed sign and the four considered signals, the number of their messages correctly interpreted by the vision system was counted. Both the incoming tracks and switches are related to additional information; the side that the track is coming from and what track that was switched to. These were accounted for in the same way as for the speed signal and sign messages.

6.2 Evaluation Results of the Track Detection

The results from the evaluation of the track detection are presented in Table 6.1. The table shows the total number of annotated video frames with the main track and each track status, as well as the part of these being correctly identified. The detection of the main track was successful in 98.4 % of the video frames, while the left, right, and middle track statuses were correctly identified in 77.1, 74.0, and 70.7 % of the time respectively. The single track status was detected correctly in 85.4 % of the time.

Table 6.1: Evaluation results of the main track and track status detection in the eight videos. The total number of frames and the part correctly detected in percentage are presented for the main track and the single, left, right, and middle track statuses.

	Total number of video frames					**Part correctly detected (%)**				
Video	**Main**	**Single**	**Left**	**Right**	**Middle**	**Main**	**Single**	**Left**	**Right**	**Middle**
Video 1	11 900	9180	1910	420	390	97.4	60.3	40.7	18.8	60.3
Video 2	54 320	41 620	2510	8230	1960	97.8	93.8	65.9	80.5	71.6
Video 3	15 100	720	14 380	0	0	98.9	10.3	79.4	-	-
Video 4	15 400	11 940	1050	2370	40	99.9	88.5	68.6	70.3	12.5
Video 5	11 500	0	11 430	0	70	99.8	-	76.8	-	100.0
Video 6	14 690	0	7140	7140	410	98.1	-	90.4	68.4	73.4
Video 7	4100	0	3860	0	240	99.3	-	70.9	-	27.5
Video 8	6530	2930	250	2400	950	98.1	51.0	95.6	81.8	83.4
Total	133 540	66 390	42 530	20 560	4060	98.4	85.4	77.1	74.0	70.7

6.3 Evaluation Results of the Object Detection

The evaluation results of the object detection are presented in Table 6.2. For each type of object, its total number of occurrences in all eight evaluation videos is provided. The number of objects missed, false detections, and messages falsely interpreted are also listed. The detection and classification accuracy, which is the part of the objects that are correctly detected and classified, was calculated by

$$\{\text{Accuracy (\%)}\} = \frac{\{\text{Occurrences}\} - \{\text{Missed}\} - \{\text{False messages}\}}{\{\text{Occurrences}\}} \cdot 100. \tag{6.1}$$

Observe that the false detections are not considered when calculating the accuracy.

As seen in Table 6.2, some objects were up to 100 % correctly detected and interpreted, while the detection of others was less successful. The four types of signals were almost always detected. The messages of the signals were however sometimes classified wrongly. Similar tendencies are seen for the speed signs. The other four types of signs were also rarely missed. The road crossings, barriers, and trains were never missed, while persons and vehicles were sometimes falsely detected. Regarding the detection of poles, roughly five and eight percent of them were missed with about as many false detections as missed ones. The objects for which the track detection was responsible, incoming tracks and switches, were both often missed. False detections of incoming tracks were common, while the directions of the switches were often wrongly interpreted.

6.4 Demonstration of the GPS and Localization Nodes

The implemented functionalities of the GPS and localization nodes were demonstrated by having the vision node processing an evaluation video at a fixed speed of 1 FPS. Simultaneously, the GPS and localization nodes were run during a walk with the hardware setup around the Johanneberg Campus. A map was thereafter generated from the information stored by the localization node. Figure 6.1 shows the map generated from the walk. For the demonstration of the concept, only catenary poles and the detected track status were included in the stored localization data. As seen in the figure, the detected track status varies between the single, middle, and right track.

Table 6.2: The results of the evaluation of the object detection on the eight evaluation videos. For each object item its number of occurrences is listed together with missed objects and false detections. Those objects that convey messages that may need to be interpreted have the extra evaluation metric of the number of false messages. The detection and classification accuracy is based on the number of occurrences, missed objects, and possible false messages.

Object	**Occurrences**	**Missed**	**False detections**	**False messages**	**Accuracy (%)**
Main signal	103	4	14	7	89.3
Road crossing signal	41	5	0	9	65.9
Distant signal	15	1	0	2	80.0
Distant road crossing signal	30	1	0	2	96.7
Speed sign	47	3	0	18	55.3
V-sign	75	3	5	-	96.0
Warning sign	101	0	3	-	100.0
ATC speed down	24	0	4	-	100.0
ATC speed up	8	1	0	-	87.5
Crossing	55	0	6	-	100.0
Barrier	42	0	0	-	100.0
Train	21	0	9	-	100.0
Vehicle	127	9	21	-	92.9
Person	138	7	13	-	94.9
Pole (left)	2174	105	130	-	95.2
Pole (right)	2164	171	90	-	92.1
Incoming track	47	18	34	1	59.6
Switch	44	8	0	8	63.6

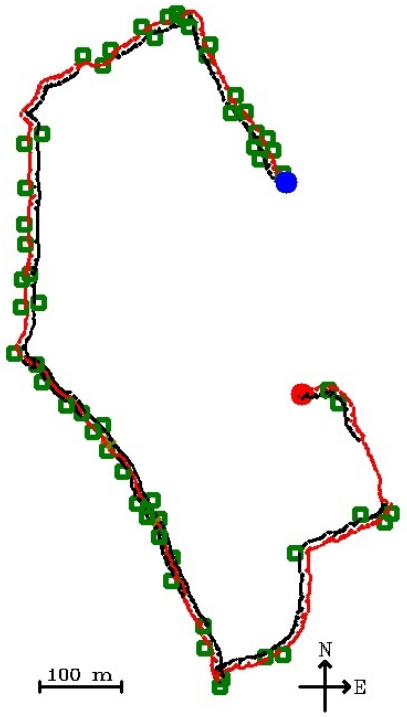


Figure 6.1: *The map generated from the data gathered by the localization node during a walk around the Johanneberg campus. The red and blue dots are the start and end positions of the walk respectively. The green squares represent the detected catenary support poles. The red line illustrates the detected main track while the black lines represent detected sidetracks.*

7 Discussion and Future Work

The results from the evaluation of the track and object detection showed that the implemented features have great potential, and may successfully be used for a prototype railway vehicle. There are however still limitations and drawbacks with the proposed vision system, and thus room for improvement. In this chapter, an in-depth review of the implemented solution is presented. Based on this, improvements of the vision system are suggested. Lastly, since this master's thesis aims to contribute to a project developing an autonomous vehicle for railway maintenance, some implemented and suggested vision features for railway maintenance are presented.

7.1 Performance of the Track Detection

As seen in Table 6.1 in section 6.2, has the implemented track detection been shown to successfully detect both the main track and possible sidetracks. The main track was detected in 98.4 % of the video frames, enough to be of real use to an autonomous vehicle. The correct track status was correctly identified at least 70 % of the time. The track status was detected with varying success in the eight videos. This indicates that variations in conditions had a great impact on the algorithm. The authors assess that many of the wrong detections were due to difficult lighting conditions, rail conditions, and ground materials.

The track detection is built upon the region-growing algorithm that uses image gradients to find possible rails. This approach to track detection thus requires connected components with a distinct contrast between the rails and the background. This is often the case when the rail is shiny and the background is dark. In a video recorded on a sunny day, the background could instead become almost as bright as the rails, resulting in that the ground was included in the detected tracks. Shadows from the rails by bright sunlight from the side were also problematic since the shadow edge was detected as a rail. The videos captured on cloudy days, without distinct shadows were thus the best suited for detecting the tracks.

Another difficult situation is when the vehicle travels in forests where the trees block most of the sunlight. In those cases, the darkened rails were difficult to distinguish from the background. Similar problems were also encountered for rusty, browned rails. The adaptive threshold used in the region-growing algorithm was implemented partly to deal with such situations, and successfully handled some variation in lighting conditions. The implemented adaptive threshold also came with drawbacks. By having an elastic threshold, the performance was affected in other situations than intended. The adaptive threshold was sometimes set too low or high, thus including too much of the background as false rails or not detecting any tracks at all. The adaptive threshold was especially affected by some ground materials, with gravel being particularly troublesome when the train was standing still. The high contrast in the gravel was then mistakenly detected as rails. The motion blur created when the vehicle was moving eased the detection task since it reduced the contrast in the background. The vertical blur was implemented to simulate motion blur and successfully improved the detection of the main track. The vertical blur however made it more difficult to detect the track when it was not running vertically, for example at curves and railway switches.

Rails are often not as shiny and prominent when seen from the side. This implies that sidetracks were much more difficult to detect than the main track. This is confirmed by the results in Table 6.1, with the left and right track status less successfully detected compared to when traveling on a single track. To correctly identify the middle track status is even harder since it requires correct detections of at least two sidetracks. To ease the detection of the sidetracks, the threshold in the region growing was lowered outside of the region with the main track. This however also led to a greater risk of detecting false sidetracks. Particularly parallel objects such as fences or drawings on the train station platforms were falsely detected as sidetracks. Therefore the track status was almost always wrongly identified when the train was traveling next to such objects. A measurement of the presence of such false detections is the difference between the correctly detected main track and the single track status in Table 6.1. By a rough estimation, 13 % of the single track status was misclassified by detections of false sidetracks.

It is seen in Table 6.2 in section 6.3 that both railway switches and incoming tracks were quite troublesome to detect. Many of the missed switches were due to that all parts of the rails were not detected, partly because of the vertical blur. The changing speed of the train also heavily impacted the detection of the switches. The robustness parameters were not implemented with a vehicle speed dependence, as would have been optimal. This means that the incoming tracks and switches will be missed if the train is traveling too fast. The compromise between being able to detect such objects at high speed and avoiding false detections at lower speeds resulted in neither fully solving any of the problems.

7.1.1 Suggested Improvements of the Track Detection

Despite the main track was successfully identified in 98.4 % of the time, the track detection algorithm must be improved to robustly detect sidetracks. The main problem of the implemented algorithm is its lack of robustness to varying light and background conditions. Various traditional image processing techniques could potentially be used to normalize the video frames and reduce the effect of differences in brightness etc., and would be of interest to investigate. An already more robust way to detect the rails is proposed by Qi et al. [QTS13], using HOG features instead of just the image intensity gradients. Computing the HOG features is however a very computationally demanding process. This approach was briefly tested by the authors but was abandoned due to the increased runtime of the algorithm. The track detection algorithm based on region growing from HOG features takes roughly three times longer than the approach suggested by the authors. Therefore, such a method based on HOG features would require improved hardware to be able to run in real-time.

A possibly more promising approach is to apply a semantic segmentation neural network to detect the tracks. Such an approach could also be used to detect and recognize other large objects, such as train station platforms. Semantic segmentation is often used for the vision systems of autonomous cars but has also been demonstrated for images from a railway environment by Li et al. [Li+20]. Using semantic segmentation would probably solve the issues of false detections of fences and drawings on the train station platforms. A useful semantic segmentation neural network must be trained to detect tracks in a wide variety of situations. This requires a large dataset, where the RailSem19 [Zen+19] might serve as a start. There although often appear reasons to use a dataset custom-made for the intended application, and creating an extensive dataset is a time-demanding process. An additional approach to track detection would be to utilize the best properties from a combination of different methods, such as region growing and semantic segmentation.

An advantage in the detection of railway tracks compared to car roads is the constant track width. The potential of this geometric constraint was however not fully utilized in the implemented solution. The possibilities of using the constant track width already in the region-growing algorithm should therefore be investigated. By this approach, more accurate tracks can be expected, with fewer false rail pixels. It may also be possible to detect and recognize individual components of the tracks directly in the growing phase by utilizing the geometric constraints. This may improve the detection of switches and incoming tracks. Nevertheless, there is a risk that an algorithm using geometric constraints may be very complex and computationally demanding. The algorithm may also have problems with recognizing sidetracks or even the main tracks in sloping terrain where the distant formula (3.6) from section 3.4 may not be defined properly, which is further reviewed below.

7.2 Performance of the Object Detection

The object detection algorithm used is in many aspects closer to a complete solution compared to the track detection. As seen in Table 6.2 in section 6.3, has the implemented object detection algorithm been proved to successfully be able to detect all objects for which it is responsible. A majority of the objects are detected correctly with more than 90 % accuracy, and some of them are even detected correctly every time. This suggests that the used approach for object detection is well-suited for the intended application. There are however still some flaws and drawbacks of the proposed object detection solution.

The capabilities and performance of the object detection are limited to the content of the training datasets. The railway training data that was annotated by the authors included hundreds to thousands of bounding boxes for each object class. For all classes except for the catenary support poles, the number of bounding boxes is still quite low compared to those of the vehicles and persons from the MS COCO and PASCAL VOC datasets. Due to the lack of available video data, many of the annotated bounding boxes also had to be generated from the same object, but from different video frames. Thus the variation in the training datasets needed to achieve the vital robustness is not on the desired level. A recurring reason for false detections of railway traffic signs is that encountered car traffic signs were often detected to be railway signs. The same problem occurred with the existing railway traffic signs that the YOLO network was not trained to detect. Both these problems may be explained by all of the dataset images from a railway environment being taken only from situations where there exist objects that are of interest to detect. Thus, the focus was not on including complex images with so-called 'false' objects, for example car traffic signs, and such objects are therefore underrepresented in the training data.

Similar problems are also present for the minor datasets used to train the classification neural networks. The distribution of the different messages from speed signs and the four signals is far from even. Above all, the speed sign classification network was only trained to be able to classify some of the possible speed limits. This

lack of data may reduce the performance of the classification networks since the rarely occurring speed signs and signal settings were often misclassified.

The results from the evaluation of the object detection can in some cases be misleading. The occurrences of persons and vehicles were overrepresented by individual objects being part of a group, such as people on a crowded train station platform or cars on a parking lot. Due to this, a missed object occurring alone would only have a small effect on the final evaluation statistics. Of all the missed vehicles, only one was in connection to a road crossing and dangerously close to the main track. The training data does not include any cars driving over the main track, and therefore the YOLO network may have learned that such cannot happen.

Since more than 8000 poles were annotated in the YOLO dataset, it may be unexpected that the detection of these seems to not be better than the detection of other objects. Many poles were missed when the train was traveling in forests. The green and brown catenary poles were then often camouflaged against their background, and hardly distinguishable even to the authors. Other dark places, often just before or after bridges and tunnels, were also problematic. The accuracy of the algorithm was greatly increased by a distinct contrast between the poles and their background, with the sky as a good example. Another issue, more fundamentally important, is when two poles are positioned very close to each other. The temporal robustness of the pole detection is optimized to account for poles with a farther distance in-between them and was therefore often counting two such poles as one. This is particularly problematic in the more complex cases of railway yards and train stations where there almost always are catenary poles in the background, belonging to other tracks. Such poles resulted in errors similar to that of two poles standing close to each other. The false detections of poles mainly came from the same pole being counted twice. This was mainly due to that the pole was first detected and then lost for enough time to be counted once before being detected again. Overall, the authors assess that the detection of poles works very well at low to fairly high speeds, in simple environments.

The estimation of the distances to the detected objects, obtained by the formulas (3.6) and (3.7) in section 3.4, works fairly well. The major limitation is that the formulas are only defined for use in a flat environment. Particularly persons and vehicles in an elevated area will appear to be closer or farther away than they are. The track inclination is however often rather small and therefore rarely a problem.

7.2.1 Suggested Improvements of the Object Detection

By the existing flaws in the implemented version of the object detection, there is still plenty of room for improvement. One of the more obvious enhancements would be to use a larger object detection neural network, such as the main YOLO network instead of YOLO-tiny. The YOLO-tiny versions are less accurate than the larger ones. On the other hand would a more complex neural network require more powerful hardware, and it would not have been fast enough to run in real-time on the NVIDIA Jetson Nano.

With a more complex neural network, the number of classes that it may successfully be able to detect and recognize increases. As stated earlier, several false detections came from interchanges of the considered signs and signals and those that the network was not supposed to detect. A natural solution to this problem would be to include all signs and signals used in Swedish railway traffic in the dataset. To increase the robustness against false detections, it would also be beneficial to add a large number of images from the railway environment that does not necessarily contain objects of interest to detect. The amount of work in such a process is although deterrent and would require more video material than currently available. With more video data the dataset diversity for the currently considered objects would also be possible to increase. This would be vital in achieving reliable object detection.

7.3 Possible Features for a Maintenance Vehicle

The long-term goal is to use the vision system on an autonomous railway maintenance vehicle. Most of the implemented features can be used for navigation, localization, and decision-making, but some can also be used particularly for maintenance work. An example is the analysis of damaged switches and catenary components. With similar approaches, the vision system can currently detect the position of switches and catenary support poles to enable further analysis of the state of their more complex components.

Currently, the vision system has no explicit detection of obstacles on the tracks, except for vehicles and persons. The maximal and minimal length of the detected main track rails might although be used for this cause. If the distance for both rails is small, it is an indication that something might be blocking the track. It could be an undetected vehicle at a road crossing, a large animal, or a fallen tree. The autonomous vehicle may then send an image of the situation to a remote operator for manual decision-making. Other maintenance

features that a future vision system could possess are to detect broken rails and missing rail fasteners. These two applications would although probably require the introduction of several cameras directed to explicitly monitor the rails. These features would also require special datasets for the intended applications, including images from situations with faults to detect. Such images may be difficult to obtain.

8 Conclusions

This project resulted in a vision system that can be used for an autonomous railway vehicle. The developed program can be run in real-time on an NVIDIA Jetson Nano, without any further hardware requirements. The information interpreted by the vision system is accessible via the ROS interface. The vision system is capable of detecting railway tracks and switches and several objects including railway signs and signals, vehicles, and persons. The track detection is based on a region-growing algorithm that uses the magnitude of the image gradients to find the pixels of the rails. The track detection algorithm was successfully used to detect the main track while other solutions are required to robustly detect also the sidetracks. Other approaches to track detection should therefore be considered, such as region growing using HOG features or semantic segmentation by a neural network.

The object detection is based on a YOLOv4-tiny neural network. The messages from signals and speed signs are further processed by classification neural networks. The object detection showed great potential for robust detections. A future vision system would require scaled-up object detection based on a neural network trained on a more diverse dataset, capable of identifying a greater variety of objects. The YOLO-tiny network is limited and future vision systems should use larger object detection neural networks of greater capacity. Furthermore, the parameters used for temporal robustness should be implemented with dependence on the speed of the vehicle. The vision system is well suited for simple navigation and decision-making. Some of the functionality required for a complete autonomous maintenance vehicle is however still missing. Both the track and object detection proved that their conceptual purposes are vital and realizable for an autonomous railway vehicle.

References

[Agu+16] D. Agudo et al. "Real-time railway speed limit sign recognition from video sequences". *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE. 2016, pp. 1–4.

[Ale21] Alexey. *AlexeyAB/Darknet*. Mar. 2021. URL: https://github.com/AlexeyAB/darknet (visited on 03/26/2021).

[AMA17] S. Albawi, T. A. Mohammed, and S. Al-Zawi. "Understanding of a Convolutional Neural Network". *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[Ara15] M. Arastounia. Automated recognition of railroad infrastructure in rural areas from LiDAR data. *Remote Sensing* **7**.11 (2015), 14916–14938.

[Ard20] Arducam. Datasheet IMX477 HQ Camera Module (2020).

[Blo20] Bloomberg. The State of the Self-Driving Car Race 2020. *Bloomberg.com* (2020). URL: https://www.bloomberg.com/features/2020-self-driving-car-race/ (visited on 03/25/2021).

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[BWL20] A. Bochkovskiy, C. Wang, and H. M. Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[COC] COCO Consortium. *COCO - Common Objects in Context*. URL: https://cocodataset.org/#home (visited on 03/18/2021).

[Eur08] European Commission. Directorate-General for Energy and Transport. *Modern Rail Modern Europe: Towards an Integrated European Railway Area*. LU: Publications Office, 2008. URL: https://data.europa.eu/doi/10.2768/22613 (visited on 03/25/2021).

[Eve+] M. Everingham et al. *The PASCAL Visual Object Classes Homepage*. URL: http://host.robots.ox.ac.uk/pascal/VOC/index.html (visited on 03/18/2021).

[GP19] M. Guerrieri and G. Parla. Automated Railway Signs Detection. Preliminary Results. *Transport and Telecommunication* **20**.1 (2019), 12–21.

[GW08] R. Gonzalez and R. Woods. *Digital Image Processing, third edition*. Prentice Hall, 2008.

[Int16] Intel Corporation. Intel Dual Band-AC 8265 (2016).

[Jun21] J. K. Jung. *Jkjung-Avt/Tensorrt_demos*. Mar. 2021. URL: https://github.com/jkjung-avt/tensorrt_demos (visited on 03/26/2021).

[Kiv12] J. Kivisaar. *Jan Kivisaar*. https://www.youtube.com/c/JanKivisaar/featured. 2012.

[KOP19] G. Karagiannis, S. Olsen, and K. Pedersen. "Deep learning for detection of railway signs and signals". *Science and Information Conference*. Springer. 2019, pp. 1–15.

[Lad21] Lady Ada. Adafruit Ultimate GPS. *Adafruit Learning System* (2021).

[Li+20] H. Li et al. "RailNet: An Information Aggregation Network for Rail Track Segmentation". *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.

[LWL19] S. Liu, Q. Wang, and Y. Luo. A review of applications of visual inspection technology based on image processing in the railway industry. *Transportation Safety and Environment* **1**.3 (2019), 185–204.

[LZY18] J. Li, F. Zhou, and T. Ye. Real-world railway traffic detection based on faster better network. *IEEE Access* **6** (2018), 68730–68739.

[Meh21] B. Mehlig. *Machine learning with neural networks*. 2021. arXiv: 1901.05639 [cs.LG].

[MLG06] R. Marmo, L. Lombardi, and N. Gagliardi. "Railway sign detection and classification". *2006 IEEE Intelligent Transportation Systems Conference*. IEEE. 2006, pp. 1358–1363.

[MR17] S. Mittal and D. Rao. Vision based railway track monitoring using deep learning. *arXiv preprint arXiv:1711.06423* (2017).

[Nak+19] M. C. Nakhaee et al. "The recent applications of machine learning in rail track maintenance: A survey". *International Conference on Reliability, Safety, and Security of Railway Systems*. Springer. 2019, pp. 91–105.

[Nata] National Coordination Office for Space-Based Positioning Navigation and Timing. *GPS.gov*. URL: https://www.gps.gov/systems/gps/ (visited on 03/23/2021).

[Natb] National Coordination Office for Space-Based Positioning Navigation and Timing. *GPS.Gov: GPS Accuracy*. URL: https://www.gps.gov/systems/gps/performance/accuracy/ (visited on 05/21/2021).

[NU10] B. T. Nassu and M. Ukai. "Automatic recognition of railway signs using sift features". *2010 IEEE Intelligent Vehicles Symposium*. IEEE. 2010, pp. 348–354.

[NUN09] N. Nagamine, M. Ukai, and B. T. Nassu. Detection of Slow-Speed-Notifying Signals Using Image Recognition from the Driver's Cabin. *Quarterly Report of RTRI* **50**.3 (2009), 162–167.

[NVI16] NVIDIA Corporation. *NVIDIA TensorRT*. Apr. 2016. URL: https://developer.nvidia.com/tensorrt (visited on 03/26/2021).

[NVI19] NVIDIA Corporation. *Jetson Nano Developer Kit*. Mar. 2019. URL: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (visited on 03/18/2021).

[ON15] K. O'Shea and R. Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE].

[Ope20] OpenStax. *The Simple Magnifier*. [Online; accessed 2021-03-25]. Nov. 5, 2020. URL: https://chem.libretexts.org/@go/page/4878.

[QTS13] Z. Qi, Y. Tian, and Y. Shi. Efficient railway tracks detection and turnouts recognition method using HOG features. *Neural Computing and Applications* **23**.1 (2013), 245–254.

[R B11] R. Bance & CO LTD. *Battery Electric Alumicarts Operating Manual*. 2011.

[Red+16] J. Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[Red13] J. Redmon. *Darknet: Open Source Neural Networks in C*. http://pjreddie.com/darknet/. 2013.

[RF16] J. Redmon and A. Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV].

[RF18] J. Redmon and A. Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[ROS] ROS team. *About ROS*. URL: https://www.ros.org/about-ros/ (visited on 03/18/2021).

[Swe] Swedish Transport Administration. *Regelmoduler i Trafikbestämmelser för järnväg*. URL: https://www.trafikverket.se/for-dig-i-branschen/Arbetsmiljo-och-sakerhet/sakerhet-pa-jarnvag/trafikbestammelser-for-jarnvag--ttj/regelmoduler-i-trafikbestammelser-for-jarnvag/ (visited on 03/25/2021).

[Swe18] Swedish Transport Administration. *Kontaktledning och störningar*. 2018. URL: https://www.trafikverket.se/for-dig-i-branschen/teknik/anlaggningsteknik/Elkraftsystemet/Kontaktledning-och-storningar/ (visited on 03/15/2021).

[Swe19a] Swedish Transport Administration. *Modul 3 Signaler - Gemensamma regler*. 2019. URL: https://www.trafikverket.se/for-dig-i-branschen/Arbetsmiljo-och-sakerhet/sakerhet-pa-jarnvag/trafikbestammelser-for-jarnvag--ttj/regelmoduler-i-trafikbestammelser-for-jarnvag/.

[Swe19b] Swedish Transport Administration. *Modul 3HMS Signaler - System H, M, S*. 2019. URL: https://www.trafikverket.se/for-dig-i-branschen/Arbetsmiljo-och-sakerhet/sakerhet-pa-jarnvag/trafikbestammelser-for-jarnvag--ttj/regelmoduler-i-trafikbestammelser-for-jarnvag/.

[Swe20a] Swedish Transport Administration. *ATC – tågskyddssystem*. 2020. URL: https://www.trafikverket.se/for-dig-i-branschen/teknik/anlaggningsteknik/signalteknik/atc--tagskyddssystem/ (visited on 03/25/2021).

[Swe20b] Swedish Transport Administration. *Modul 1 Termer*. 2020. URL: https://www.trafikverket.se/for-dig-i-branschen/Arbetsmiljo-och-sakerhet/sakerhet-pa-jarnvag/trafikbestammelser-for-jarnvag--ttj/regelmoduler-i-trafikbestammelser-for-jarnvag/.

[Swe20c] Swedish Transport Administration. *Modul 3E Signaler - System E2 och E3*. 2020. URL: https://www.trafikverket.se/for-dig-i-branschen/Arbetsmiljo-och-sakerhet/sakerhet-pa-jarnvag/trafikbestammelser-for-jarnvag--ttj/regelmoduler-i-trafikbestammelser-for-jarnvag/.

[Sze11] R. Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer, 2011.

[TIM20] J. J. Tai, M. S. Innocente, and O. Mehmood. FasteNet: A Fast Railway Fastener Detector. *arXiv preprint arXiv:2012.07968* (2020).

[TJC14] P. Tang, W. Jin, and L. Chen. "Visual abnormality detection framework for train-mounted pantograph headline surveillance". *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2014, pp. 847–852.

[Tre+18] D. Trentesaux et al. "The autonomous train". *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. IEEE. 2018, pp. 514–520.

[Wed17] M. Wedberg. *Detecting Rails in Images from a Train-Mounted Thermal Camera using a Convolutional Neural Network*. 2017.

[Wei+19] X. Wei et al. Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study. *Engineering Applications of Artificial Intelligence* **80** (2019), 66–81.

[Wei+20] X. Wei et al. Multi-target defect identification for railway track line based on image processing and improved YOLOv3 model. *IEEE Access* **8** (2020), 61973–61988.

[Ye+18] T. Ye et al. Automatic railway traffic object detection system using feature fusion refine neural network under shunting mode. *Sensors* **18**.6 (2018), 1916.

[Ye+20a] T. Ye et al. Autonomous railway traffic object detection using feature-enhanced single-shot detector. *IEEE Access* **8** (2020), 145182–145193.

[Ye+20b] T. Ye et al. Railway traffic object detection using differential feature fusion convolution neural network. *IEEE Transactions on Intelligent Transportation Systems* (2020).

[Zen+19] O. Zendel et al. "RailSem19: A dataset for semantic rail scene understanding". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

A Appendix

A.1 Signals in the Swedish Railway Infrastructure

In this project four types of signals are of interest to detect and recognize; main signals, distant signals, road crossing signals, and distant road crossing signals. These four types of signals convey messages regarding speed limits, drive permissions, and information about the next upcoming signals. The specific messages are determined by the lit lamps and their colors. Following are descriptions of the four signal types and their messages. The texts are mainly based on two modules from the Swedish Transport Administration [Swe19a; Swe19b].

A.1.1 Main Signal

The main signal consists of two to five light openings, with each lamp of a single color; green, red, or white. The first, third, and fifth lamps from the top are green, while the second and fourth are red and white respectively. The main signal can convey up to seven different messages, all listed in Figure A.1. The message is determined by the combination of the lit lamps and if the lights are fixed or flashing. The main signal messages are either 'Stop' or 'Proceed', the latter also providing a speed limit. The main signal may also convey information about the message to expect from the next upcoming main signal, for example 'Expect to stop' as in Figure A.1 (d). Note that a main signal with fewer lamps than five will not be able to show all the listed messages. For example will the main signal with two lamp openings, the two upper ones, only be able to convey 'Proceed 80 km/h' or 'Stop', shown in Figure A.1 (a) and (b).

(a) (b) (c) (d) (e) (f) (g)

Figure A.1: *The seven main signal messages. A flashing light is illustrated by a star around the intended lamp opening. The possible messages; (a) Proceed 80 km/h, (b) Stop, (c) Proceed 40 km/h with caution, (d) Proceed 80 km/h and expect to stop, (e) Proceed 80 km/h and expect to proceed 80 km/h, (f) Proceed 40 km/h and expect short route, and (g) Proceed 80 km/h and expect to proceed 40 km/h. Images from: Mysid and Bobjork, Wikipedia.*

A.1.2 Distant Signal

The distant signal conveys information regarding the message to expect of the next upcoming main signal. The distant signal has either two or three light openings, and the lamps are all single-colored with the first and third lights from the top in green, and the second lamp in white. These convey up to three different messages, see Figure A.2 (a)-(c), all with flashing lights. As for the main signal, the distant signal with two lamps can only convey the messages using the upper two lamps, see Figure A.2 (a) and (b).

A.1.3 Road Crossing Signal

The road crossing signal is used in connection with railway road crossings and always occurs together with the V-sign. The road crossing signal tells if the railway vehicle is allowed to pass the crossing or not. To convey these two messages, the road crossing signal has a single light opening. The lamp is two-colored, with either white or red light to signal for 'Proceed' or 'Stop before crossing' respectively, see Figure A.2 (d) and (e).

A.1.4 Distant Road Crossing Signal

The distant road crossing signal conveys information regarding the message to expect of the next upcoming road crossing signal. The distant road crossing signal conveys information of either expected 'Proceed' or 'Stop before crossing' at the next road crossing. The distant road crossing signal consists of three lamp openings, all with yellow lights. These simultaneously have fixed or flashing lights for the two messages respectively, see Figure A.2 (f) and (g).

(a) (b) (c) (d) (e) (f) (g)

Figure A.2: *The messages of the distant signal, road crossing signal, and distant road crossing signal. Flashing lights are illustrated by a star around the intended light opening. The signals messages, from the left; (a) Expect to stop, (b) Expect to proceed 80 km/h, (c) Expect to proceed 40 km/h, (d) Proceed (e) Stop before crossing, (f) Expect to pass crossing, and (g) Expect to stop before crossing. Images from: Mysid and Bobjork, Wikipedia.*

A.2 Classification Datasets

In Tables A.1, A.2, A.3, A.4, and A.5 are the used classes listed for the datasets created for classification of the main signal, distant signal, road crossing signal, distant road crossing signal and speed sign messages respectively. In these five classification datasets, each image only contains a single object, belonging to one of the classes used in that dataset. For each class, the number of samples of that specific class before augmentation is listed in the tables. The 'False' class contains images of objects other than the concerned signal or speed sign while the 'No light/lights' class being images of signals without any visible lights. The speed sign classes are characterized by the number written on the sign, indicating the speed limit in units of km/h.

Table A.1: The number of occurrences of each class in the message classification dataset for main signals. The class names '1', '2', '13', '14', and '135' refer to the indices of the lit lamps in the main signals with 1 as the index of the most upper lamp.

Class	**False**	**No lights**	**1**	**2**	**13**	**14**	**135**	**Total**
Main signal occurrences	209	680	564	491	72	228	4	2248

Table A.2: The number of occurrences of each class in the message classification dataset for distant signals. The class names '1', '2', and '13' refer to the indices of the lit lamps in the distant signals with 1 as the index of the most upper lamp.

Class	**False**	**No lights**	**1**	**2**	**13**	**Total**
Distant signal occurrences	102	493	30	172	63	860

Table A.3: The number of occurrences of each class in the message classification dataset for road crossing signals.

Class	**False**	**No light**	**Red light**	**White light**	**Total**
Road crossing signal occurrences	209	128	50	178	565

Table A.4: The number of occurrences of each class in the message classification dataset for distant road crossing signals.

Class	**False**	**No lights**	**Lights**	**Total**
Distant road crossing signal occurrences	42	36	254	332

Table A.5: The number of occurrences of each class in the message classification dataset for speed signs. The class name refers to the speed limit conveyed by the sign, in units of km/h.

Class	**False**	**10**	**40**	**60**	**65**	**70**	**80**	**85**	**90**	
Speed sign occurrences	214	5	26	85	11	185	28	27	84	
Class	**100**	**105**	**110**	**120**	**125**	**130**	**140**	**150**	**160**	**Total**
Speed sign occurrences	77	40	56	70	18	238	54	11	14	1243

A.3 Classification Neural Network Structure

The neural networks created by the authors to classify the messages of the main signal, distant signal, road crossing signal, distant road crossing signal, and speed sign were all of the same structure. The network layout, presented in Table A.6, is the same for all the five classification neural networks, except for the number of color channels in the input, C, and the number of object classes in the output.

Table A.6: The layout of the classification networks that are used to classify the found signal and speed sign messages. The structure is the same for all five classification networks, except for the number of color channels in the input, $C \in \{1, 3\}$, and the number of output classes.

Operation layer		**Number of filters**	**Size of each filter**	**Stride**	**Size of output**
Input image					$32 \times 32 \times C$
Convolution layer	Convolution Batch normalization ReLU	32	$3 \times 3 \times C$	1×1	$32 \times 32 \times 32$
Pooling layer	Max pooling	1	2×2	2×2	$16 \times 16 \times 32$
Convolution layer	Convolution Batch normalization ReLU	64	$3 \times 3 \times 32$	1×1	$16 \times 16 \times 64$
Pooling layer	Max pooling	1	2×2	2×2	$8 \times 8 \times 64$
Convolution layer	Convolution Batch normalization ReLU	64	$3 \times 3 \times 64$	1×1	$16 \times 16 \times 64$
Linear layer	Fully connected ReLU Batch normalization	-	-	-	64
Linear layer	Fully connected Softmax	-	-	-	{Number of classes}

A.4 Vision Node Information

The implemented vision node communicates with other parts of the autonomous railway vehicle by publishing information on its own ROS topic, the *vision_topic*. The information is updated for each video frame processed by the vision node. A complete list of the published information is given in Table A.7. Each item of information is described by its data type and an additional brief explanation.

Table A.7: The information published by the vision node. For each item of information, its type of either integer, boolean, string, or list kind is presented together with an additional explanation.

Information	**Type**	**Explanation**
Frame number	INT	The index of the current video frame.
Track status	INT	Integer code for the current track status of the vehicle. 0 = Single track, 1 = Left track, 2 = Right track, 3 = Middle track.
Speed limit	INT	The current speed limit in km/h.
Main signal message	INT	Integer code for the message of the detected main signal. 0 = None, 1 = Go 80, 2 = Go 40, 3 = Go 40 gently, 4 = Stop, 5 = Short path to stop.
Upcoming main signal message	INT	Integer code for information about the upcoming main signal. 0 = None, 1 = Go 80, 2 = Go 40, 4 = Stop, 5 = Short path to stop.
Road crossing message	INT	Integer code for the message of the detected road crossing. 0 = None, 1 = Upcoming, 2 = Detected, 3 = Go, 4 = Stop.
Upcoming road crossing message	INT	Integer code for information about the upcoming road crossing. 0 = None, 2 = Detected, 3 = Go, 4 = Stop.
Max view distance	INT	The estimated length in units of meters of the farthest seen main track rail.
Min view distance	INT	The estimated length in units of meters of the shortest seen main track rail.
Distance to road crossing	INT	The estimated distance in units of meters to the seen crossing.
Distance to main signal	INT	The estimated distance in units of meters to the detected applying main signal.
Distance to road crossing signal	INT	The estimated distance in units of meters to the detected applying road crossing signal.
Distance to speed sign	INT	The estimated distance in units of meters to the detected applying speed sign.
Switch detected	BOOL	Tells if a switch is detected.
Switched lane status	INT	Integer code for the detected change of lane status at switch. 0 = No info, 2 = Switched right, 3 = Switched left.
Incoming track status	INT	Integer code for the side of the incoming track. 0 = No info, 4 = Incoming from left, 5 = Incoming from right.
Road crossing detected	BOOL	Tells if a road crossing is detected.
Barriers detected	BOOL	Tells if a road crossing barrier is detected.
ATC speed up sign detected	BOOL	Tells if an ATC speed up sign is detected.
ATC speed down sign detected	BOOL	Tells if an ATC speed down sign is detected.
Warning sign detected	BOOL	Tells if a warning sign is detected.
Distant signal sign detected	BOOL	Tells if a distant signal sign is detected.
V-sign detected	BOOL	Tells if a V-sign is detected.
Left pole counter	INT	The number of poles counted to the left of the main track.
Right pole counter	INT	The number of poles counted to the right of the main track.
Object list	List<Object>	List with information regarding the detected vehicles and persons.
Errors	STR	Error message.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY