Master Thesis

Automated railway object mapping using imagery & point clouds

Submitted by

Anastasios Vogiatzis

Master Thesis

Automated railway object mapping using imagery & point clouds

Submitted by

Anastasios Vogiatzis
MSc Student
TU Delft

Under the guidance of

Roderik Lindenbergh / Chris Sullivan
Associate Professor / Software Team Lead at Fugro RailData

Department of Applied Earth Sciences
Fugro
A Blaeulaan 60A, 3528 AD Utrecht, P.O. Box 63, Prismastraat 4, 2631 RT Nootdrop,
The Netherlands

AUTOMATED RAILWAY OBJECT MAPPING USING IMAGERY & POINT CLOUDS

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the master degree of

Applied Earth Science, track Geosciences & Remote Sensing

by

Anastasios Vogiatzis

March 2021

The work in this thesis was made in the:

Europe Rail Software Development team
Geoscience & Remote Sensing
Civil Engineering & Geosciences
Delft University of Technology

Supervisors:	Dr. R.C. Lindenbergh
	Team Lead C. Sullivan
	Dr.ir. A.A. Verhagen
	Dr. H. Ledoux
	Ph.D. Candidate M. Fragkiadakis
Co-readers:	ir. L. Amoureus
	Ph.D. Kaixuan Zhou

ABSTRACT

Everything around us is rapidly changing. Whole new blocks of buildings are built, huge infrastructural projects are constructed and so on. Hence, there is a need of a reliable and up-to-date inventory of the area and the objects of interest for mapping and monitoring assets and their changes. An answer of this upcoming need is an automated inventory of infrastructure using Remote Sensing and artificial intelligence (AI) techniques.

Rail sector shares the same need for fast and reliable inspection on its infrastructure. Monitoring frequently the condition of the railway infrastructure, can improve the maintenance efficiency and the avoidance of hazards. The traditional monitoring techniques are costly, time consuming and in some cases dangerous, due to their reliance on the physical presence of the inspector. Hence, new state-of-the-art techniques that are able to frequently and without putting in risk human lives, inspect the condition of the railway and its infrastructure.

This master thesis aims at developing an efficient workflow for combining $2D$ imagery and $3D$ light detection and ranging (LiDAR) point clouds for the automated detection and localization of the railroad infrastructural objects into $3D$ world coordinate system, for monitoring the railway infrastructure.

Using deep learning (DL) methods in imagery we detected and mapped, approximately the 60% of the railroad equipment of our interest (i.e. light signals and equipment boxes). These detected equipment were analysed with stereoscopic techniques to retrieve their position in $3D$ world coordinate system. That led to the automated creation of a geographical information system (GIS) map having the positional and class information of railway equipment. Once the detected objects were mapped, then the point cloud data were automatically cropped into voxels including the same objects. Hence, using various sophisticated machine learning (ML) techniques, the points referring to the objects were classified. Furthermore, combining the positional information provided via $2D$ analysis with $3D$ point clouds, the vertical position was refined and the height of the mapped objects was estimated. Lastly, the positional information estimated from the $2D$ analysis enhanced, the unsupervised ML classification in point clouds. The product of this classification, has the potential to be used as training data to Fugro's point cloud classifier.

The proposed workflow and methods are based on rail environment using *RILA* - a state of the art mobile mapping systems (MMS) which has multiple sensors able to record the accurate position of the train and they can track and capture the railroad and the environment next to it accurately up to a millimeter level.

ACKNOWLEDGEMENTS

Many people were involved actively or passively for the well of this project. I would like to thank the following people who have helped me undertake this research:

My parents who raised me being creative and artistic. Although the start of this project was vague and a bit surrealistic, a clear product delivered at the end.

My main supervisors Dr. Roderik Lindenbergh - Chris Sullivan, for believing in me, my skills and motivation, and guided me throughout the whole project, supporting me in technical and non technical manners.

Luc Amoureus, with his math skills and attention to detail, helped me georeferencing the cameras correctly and to evaluate our results.

Special thanks to Manolis Fragkiadakis who helped me with the setup of the laptop and the Deep Learning part of the thesis.

My supervisors Sandra Verhagen and Hugo Ledoux, who invested time on the shape, structure and quality of the report in order to be academically valid.

Help of high importance was the guidance I received from Kaixuan Zhou and Carvalho Diego on defining the model of the triangulation.

Stavros Korokithakis, who at the initial level of the thesis convinced me to use the method of triangulation with one camera for $3D$ reconstruction.

My partner Argyro Bitaki for supporting my decisions in every phase of the master's journey.

Even though the majority of my friends never understood the subject of this thesis, I would like to thank all of them, for the - unrelated with the project - time we spent together.

Finally, the people who thought that white noise helps concentration and they uploaded various natural or man-made sounds to the web.

Delft, $8th$ of March 2021 - Anastasios Vogiatzis

He who fights, can lose. He who doesn't fight, has already lost - **Bertolt Brecht**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

LIST OF ABBREVIATIONS

1 INTRODUCTION

The inspection of the railway infrastructure was involving one or more field operators (see Figure 1.1). This technique was dangerous, time consuming and costly [16]. Automation and remote sensing techniques on the inspection of infrastructure became feasible due to recent technological advancements. Nowadays, we have the potential to process and store big amount of data in real time and with minimum computational efforts. In addition, with the help of AI we are able to lower the risks of hazardous situations (e.g. driving assistant) [17] as well as to visually inspect potential structural damages [18]. Taking advantage of these new technologies, the inspection of the railway infrastructure can be now performed by a remote operator .

Figure 1.1: Monitoring the railway infrastructure by means of GNSS and leveling. The acquisition of the geo-data made by the physical presence of an inspector (Figure taken from [1]).

In this project, the acquisition of the input data was derived by cameras, geographical navigation satellite systems (GNSS) and mobile laser scanner (MLS), that were mounted in Fugro's mobile mapping system(s) (MMS) RILA (see Figure 1.2). RILA was adjusted on the tail of the train, to acquire data of the railroad and the surroundings.

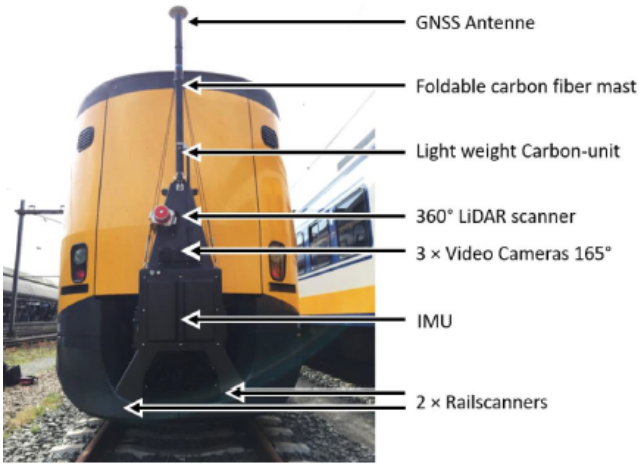


Figure 1.2: Illustration of the main sensors and parts of Fugro's MMS RILA. For the acquisition of geo-data, RILA consists of 3 LiDAR, 3 cameras, GNSS reciever and an inertia measurement unit IMU (Figure taken from [2]).

This thesis investigates the potential and advantages of combining 2*D* imagery and 3*D* MLS data. The output of this research would be the creation of a suitable workflow creating a geographic information systems (GIS) map that provides information about the position of the side rail infrastructure (i.e. light signals and equipment boxes). In more detail, 2*D* video frames capturing a part of railway in Ely (see Figure 1.3), were used to detect the mentioned objects by means of convolutional neural network (CNN). Once the 2*D* video frames were georeferenced, stereoscopic techniques were used to retrieve the 3*D* position of the detected objects in the world coordinate system, as to be located on a geographic map. The 2*D* object detection via CNN involves the creation of 2*D* bounding boxes around the detected objects. However, inaccuracies in the height of the created bounding boxes were noticed. To compensate for these inaccuracies, 3*D* LiDAR point clouds were involved to refine the vertical plane and the height information of the detected mapped objects. First, the 3*D* volumes-voxels of the detected objects were cropped, based on their estimated position retrieved from the previous 2*D* analysis. Then, the voxels were classified, using machine learning (ML) techniques, either as "**ground**", "**object**" or "**other**". Lastly, based on the classified points referring to the objects, the refinement of the vertical plane and their height was made.

Automation in mapping railway infrastructural objects (i.e. light signals and equipment boxes), is a proper definition of this thesis scope. Automation in the sense that the user involves in placing the input 2*D* video frames captured via RILA and the software returns the output. The output is a .csv file that includes the object-class, position and height information of the railway infrastructural objects, and classified 3*D* voxels that include the aforementioned three classes. The last output of the thesis, unlocked the potential of using the automatically classified voxels as training data for Fugro's point cloud classifier.

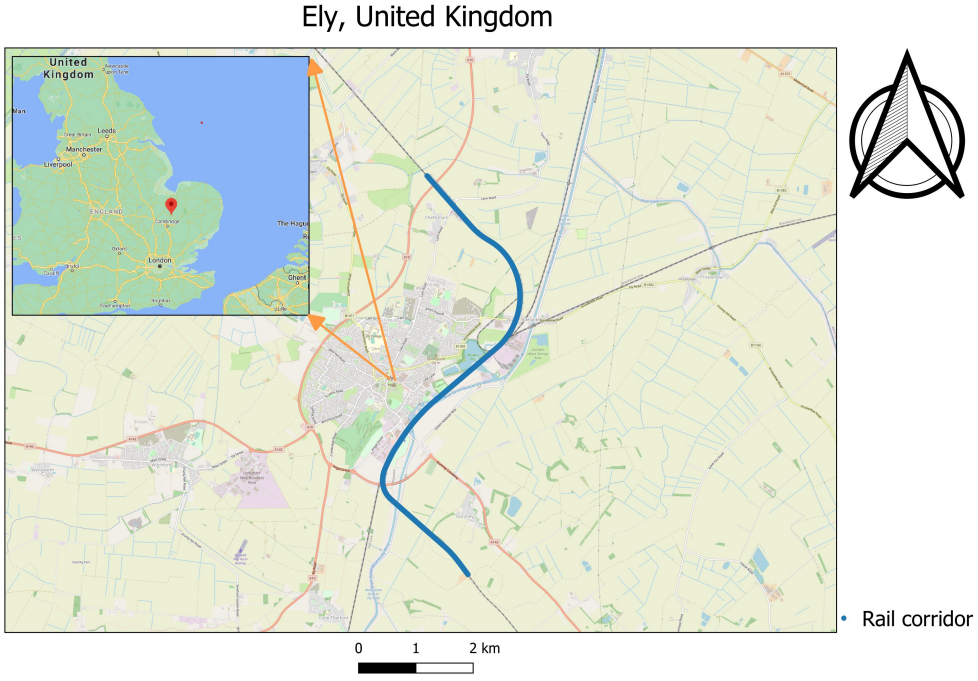


Figure 1.3: Ely, city in the United Kingdom. In blue the part of the railroad that the data of this thesis were acquired.

1.1 MOTIVATION

Railroad transportation represents a huge proportion of the travel needs in the globe. Hence, the need of a fast, reliable and safe transportation is increasing. In the Netherlands, the railway sector is an economic pillar of outmost importance with a long history that begun at 1839. In addition, *"The maintenance of tracks and track side infrastructures is considered to be one of the most hazardous jobs in the rail industry"* [16]. The aforementioned need has led to safer and more efficient methods, using remote sensing techniques for monitoring the railway infrastructure, having a subjective diagnosis of its condition for maintenance and inventory reasons, regardless of the weather conditions. *"A semi-automatic or fully automatic inspection will reduce its subjectivity and will save public resources while improving the road safety"* [19].

The previous decades, the inspection of the rail and the track side infrastructure was made with inspectors being physically present (Figure 1.1). Lately, this has changed and modern automatic remote sensing techniques have replaced the field worker with a remote operator (Figure 1.2). Automation in monitoring railways and the infrastructure next to it, brought a revolution in the inspection of the sector, minimising the delivery time of the results. As a consequence, the inspection frequency increased, leading to faster and safer diagnosis of the railway condition [20]. Most of the time, airborne light detection and ranging (LiDAR) and aerial photography are used to extract the rail infrastructure, the railway itself as well as the catenary wire masks [21], [22].

More recently, terrestrial MMS have been involved in monitoring the railway. Due to the higher resolution that they provide, we are able to detect and classify smaller infrastructural objects, such as light signals and equipment boxes, that they are not observed in the railroad in high frequency. RILA is Fugro's terrestrial MMS for the acquisition of the railway geo-data. This research is dealing with the investigation of using RILA's $2D$ and $3D$ data to detect and map in $3D$ world coordinate system, the aforementioned infrastructural objects.

1.2 SCOPE OF THE STUDY

There is a need for automated, safe and more frequent inspection of the infrastructural objects that are next to the rail. The current state-of-the-art methods for the automated monitoring of the railway infrastructure mainly involve MLS for the segmentation and classification of the $3D$ point cloud data. This technique is less capable in the recognition of the infrastructural objects, where the involvement of video cameras is more suitable even for smaller objects. *"The distribution of* $3D$ *LIDAR point clouds become more and more sparse as the distance from the scanning center increases, which brings difficulties for a* $3D$ *LIDAR to detect specific objects in the classification step"* [23].

The objective of this study is the investigation and the creation of an effective workflow to combine georeferenced $2D$ video imagery and LiDAR data, for an automated inventory of the track side infrastructure.

Hence, this thesis proposes a reliable method that can deal with this need and investigates the feasibility of combining $2D$ video imagery with $3D$ point clouds, to acquire accurate $3D$ position of the signage next to the rail. In a nutshell, this study approaches the problem by means of object detection in $2D$ video imagery using deep learning (DL) techniques. After the objects of our interest are detected, using the stereoscopic method of triangulation, their position in $3D$ world coordinates is retrieved. Having their positional information in xy plane, the correspondingly $3D$ volumes-voxels are cropped from the point cloud and the objects are classified using various ML techniques. Lastly, the height of the objects is retrieved by $2D$ and $3D$ fusion.

1.3 RESEARCH QUESTIONS

Taking into account all the above and considering that *Fugro rail* operates the innovative and state-of-the-art *RILA*4.0, this thesis deals with the following research questions.

Main question:

- What would be an effective workflow to combine 2D and 3D data to map the infrastructural railway objects (i.e. light signals and equipment boxes), with an accuracy of a meter?

Sub-questions:

- What are the properties of the input data?
- What are the existing techniques for object detection in 2*D* imagery and what are their pros and cons?
- How to estimate the 3*D* position of the detected objects in 2*D* imagery?
- What are the requirements of the inspection regarding the accuracy and precision of the position of the signage?
- To what extent the feature detection from 2*D* imagery can be complementary to the 3*D* point cloud classification, and how it might aid in improving the accuracy of the classified point cloud?

Validation questions:

- What methods can be used to describe the quality of the results? How can we validate the results?
- Can this method be linked to Fugro's OnePortal system? Is it robust?

2 RELATED WORK

This chapter presents an overview of the theoretical knowledge and background, that this study is using to approach its research. The literature review consists of introducing the mobile mapping systems (MMS) and Fugro's MMS RILA, the current methods for object detection in imagery and the stereoscopic techniques, as well as unsupervised classification techniques in point clouds. Lastly, the chapter deals with some existed studies illustrating fusion methods between $3D$ and $2D$ data.

2.1 MOBILE MAPPING SYSTEMS – RILA

MMS, due to its mobility and multi-discipline, is a highly expanded method for the collection of geo-spatial data [24], [25]. Typically, a MMS consists of various optical and ranging sensors such as cameras, light detector and ranging systems (LiDAR), radars, global navigation satellite systems (GNSS), inertial measurement unit (IMU) systems etc. that are adjusted into a land-based, water-based, hand-based and aerial-based vehicles. Hence, the MMS are able to acquire georeferenced data for mapping, monitoring and other purposes with a remarkable detail and having a big coverage depending on the application.

For monitoring of the rail infrastructure, Fugro uses the state of the art RILA MMS. *"RILA uses a sophisticated GPS measurement system, combined with inertial measurement units, laser scan technology and video cameras to collect the X, Y and Z position of the track, the rail profile and parameters such as track gauge and cant"* [26] (see Figure 1.2).

2.1.1 LiDAR

Laser light is one of the most important measuring tools for acquiring geo-data, in civil engineering and geo-sciences. The reason of being such an important sensor is that it provides a direct method for data collection with a millimeter accuracy [27]. The way that the sensor calculates the distances is by measuring the travel time of the propagated laser light beam when it hits an object. Depending on the application, different wavelengths of lights are used. Typically, in topographic measurements, near-infrared laser is used to acquire the geo-data [28]. The product of a LiDAR survey is a point cloud - a collection of points in $3D$ space, having X, Y and Z coordinates and other attributes, representing $3D$ shape of the objects.

2.1.2 Differential Global Navigation Satellite System (DGNSS)

"DGNSS is a code-based relative positioning technique that employs two or more receivers simultaneously tracking the same satellites"[29]. DGNSS involves a fixed-based location with well know position that improves the standalone GNSS position data, providing positional corrections and eliminating pseudorange errors (satellite clock, atmospheric, receiver noise) [30] (see Figure 2.1).

The positional accuracy of ranges from sub-meter level to meter depending mainly on the distance between the fixed-based receiver and the user (rover), and the performance of the receiver [[31], [32]].

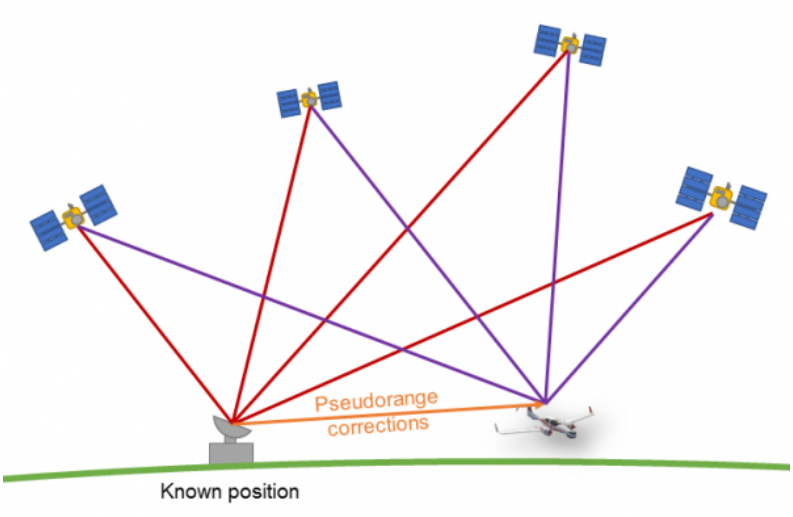


Figure 2.1: The complex of DGNSS. Left: fixed-base station with known position, right: The user (rover). The distance between the two is the baseline (Figure taken from [3]).

2.1.3 Inertial measurement unit (IMU)

An IMU device, uses accelerometers and gyroscopes to measure the angular velocity, body's specific force and orientation. These electronic devices are mainly used in maneuver airplanes and drones, autonomous vehicles and in robotics [[33], [34]].

IMU systems are of high importance in supporting GNSS in applications where GNSS is incapable of providing reliable positional measurements. *"while GPS can provide precise-long-term position information in open areas, the GPS signal could be blocked or attenuated by obstacles in urban areas resulting in GPS signal outages"*[34].

2.2 OBJECT DETECTION METHODS IN 2D IMAGERY

There are two major categories of methods in object detection and object classification/identification. The oldest one involves various computer vision algorithms for feature extraction in 2*D* imagery, while the other category uses sophisticated machine learning (ML) and deep learning (DL) techniques like the convolutional neural networks (CNN). The latter is also used for real time applications like automated driving and robotics [35].

The first category of object detection in 2*D* video imagery with the use of classical approaches is discussed in this paragraph. The object detection can be done using many existing techniques like the temporal/frame differencing, a method which involves the calculation of the changes between different 2*D* frames [36]. In addition, background subtraction which can be done by detecting an object, its trajectory, and make a prediction of its behaviour [37], is also a well known technique for object detection. Lastly, clustering based techniques are used, like the optical flow method. For the sake of accuracy, the use of multiple 2*D* frames to identify an object, reduces the false positives, hence it is better to have multiple frames instead of a single image [37], [36].

Regarding the use of DL techniques like CNN for object detection and inventory in 2*D* imagery, multiple studies exist and propose their own architecture. CNN is an image recognition and classification technique that passes the input images though convolutional layers (hidden layers) and filters (kernels) for feature extraction, and classifies the objects that are present in them [38]. Object detection algorithms are state-of-the-art computer vision techniques for locating instances of objects in 2*D* imagery and video [4]. The majority of the studies are based on the pedestrian

detection in urban areas, while a percentage of them are dealing with the identification of signage in rural areas for self driving [35]. Another study, combines some of the image processing techniques with the CNN, in order to return more reliable and fast results for inventory purposes [39]. The quality of all ML techniques rely on the architecture of the CNN models, but most importantly on the input data that the user provides to train the classifier.

In various applications such as street monitoring, DL methods are used. More specifically, you only look once YOLOv3 - a CNN architecture [38], that creates bounding boxes when identify a class. The reason that CNN used instead of classical ML techniques, is mainly the benefit of localization of the detected object (see Figure 2.2). In addition, while classical ML techniques outperform when having small training datasets, deep networks have achieved accuracy that is far beyond.

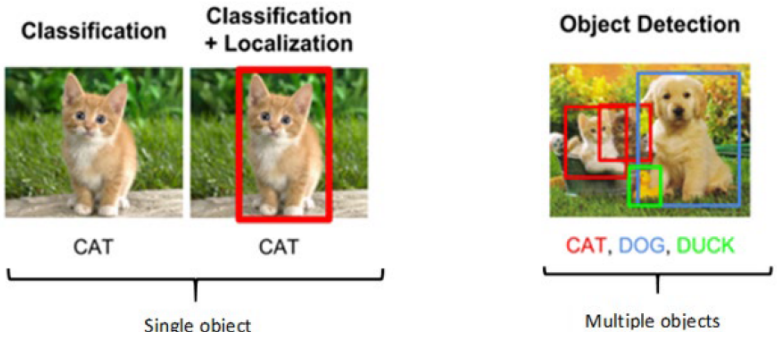


Figure 2.2: Classification vs Object Detection (Figure taken from [4]).

2.2.1 Deep Neural Network - YOLOv3

You **O**nly **L**ook **O**nce - YOLO, is a single neural network algorithm for object detection with 106 hidden convolutional layers. Its architecture permits small object detection, which is convenient in applications where the objects appeared in various scales due to the varying distances from the cameras. In more detail, YOLOv3 downsamples the input 2*D* video frames by the factor of 32, 16 and 8 (see Figure 2.3), improving the detection of an object in different scales/distances from the camera. Another important advantage of YOLO compared to other convolutional neural networks is its time performance. *"The state-of-art version (YOLOv3) not only has high detection accuracy and speed, but also performs well with detecting small targets"* [40].

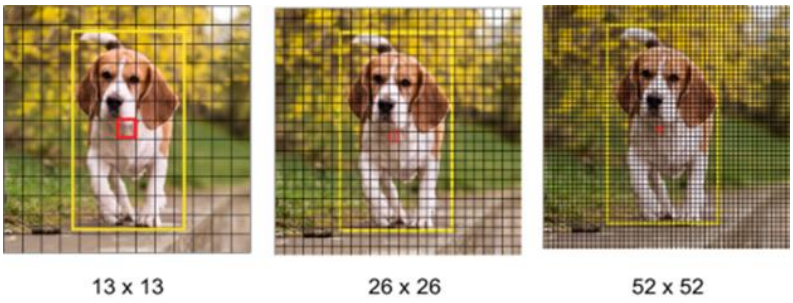


Figure 2.3: Scaling procedure of the an input image to optimize the detection of objects in multiple scales (Figure taken from [5]).

In more detail, the algorithm resizes the input image to 416*x*416 and then downsamples its size 3 times in different scales 8-16-32 [5] (see Figure 2.4). In every scale, a single convolutional network predicts multiple boxes, one for every grid shell and calculates the class probabilities for those boxes. These confidence scores are made by using logistic regression and reflect how confident the model is, that the bound-

ing box contains an object, and how accurate it thinks every box is [41]. Finally, $((52x52) + (26x26) + 13x13))x3 = 10647$ bounding boxes created [42]. To get rid of boxes with a low score and to find the best bounding box, a score-thresholding and the non-maximum suppression technique are used (see Figure 2.5), that ignores redundant and overlapping bounding boxes (see Figure 2.6).

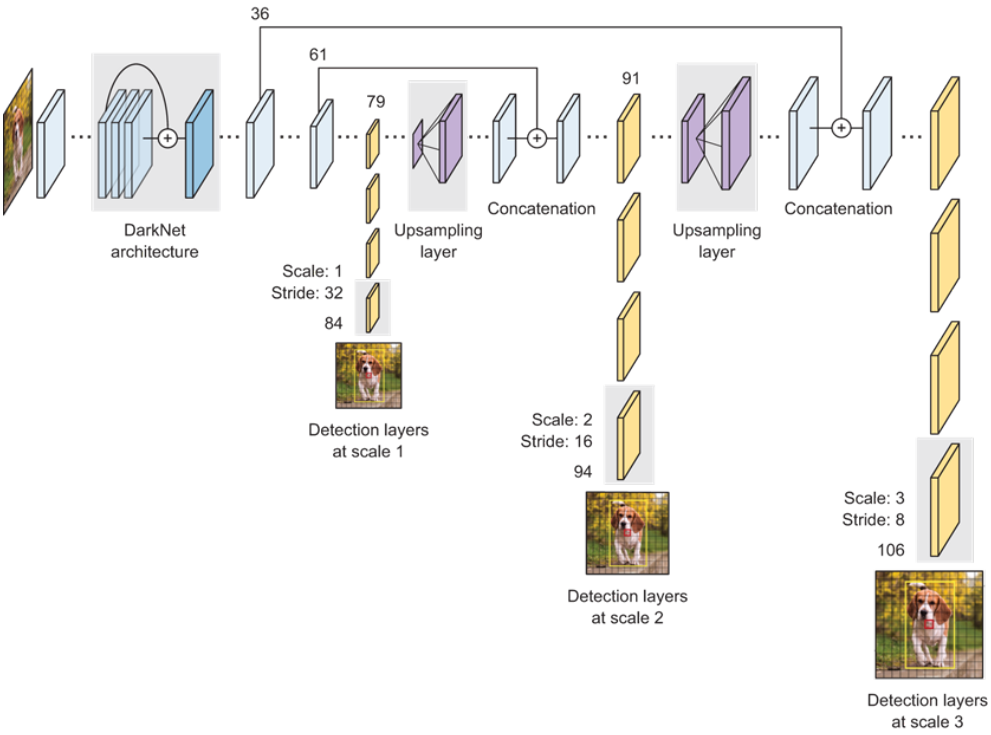


Figure 2.4: The architecture of the CNN YOLOv3, for object detection (Figure taken from [5]).

Figure 2.5: Choosing the best bounding box using the non-maximum suppression technique (Figure taken from [6]).

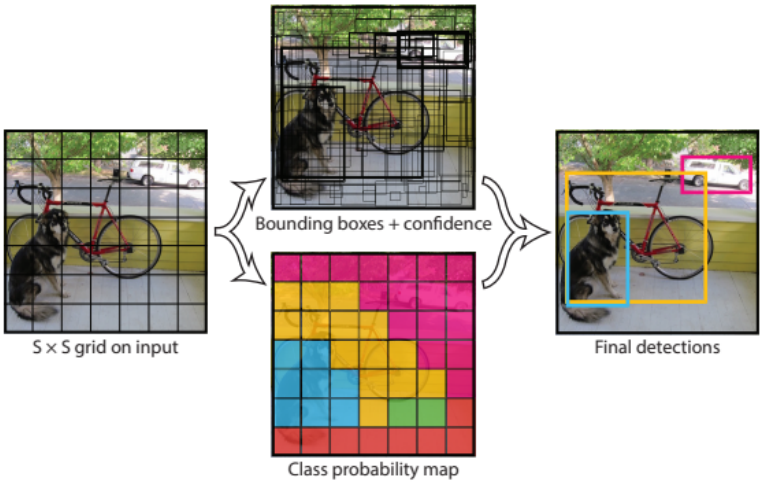


Figure 2.6: The 3 steps that YOLOv3 uses to create the bounding boxes in all 3 scales (Figure taken from [5]).

The training procedure of the classifier does not differ much from the main detection part; the ground truth knowledge makes it different. In more detail, 2 new statistical/metric tools are present in every iteration that the classifier uses during the training procedure. The first tool is the loss function or mean square error (MSE) (see Equation 2.1). The main principal of the loss function is the intersection over union (IoU) among the predicted and the ground truth bounding boxes, while the confidence of the predicted object is based on the center of each bounding grid cell. The second tool that YOLOv3 uses to access the quality of the trained weights is the mAP [43], which is is the average precision calculated for all the classes. It is also important to note that in some papers, the use average precision (AP) refers to mAP interchangeably. To understand better the mAP as a metric for the performance of a 2*D* detector, we need to first review the metrics **precision** and **sensitivity/recall** (see Figure 2.7). Hence, the metric mAP is defined as the summation of all the precision of the classes, divided by the number of the classes (see Table 2.1).

$$\mathcal{MSE} = \frac{1}{\mathcal{N}} \sum_{t=1}^{\mathcal{N}} (\mathcal{Y}(t) - \widehat{\mathcal{Y}}(t))^2 \tag{2.1}$$

Where N is the number of input labels and $\mathcal{Y}(t), \widehat{\mathcal{Y}}(t)$the ground truth and predicted corners of bounding boxes respectively [44]. $\mathcal{MSE}$ refer to the mean square error.

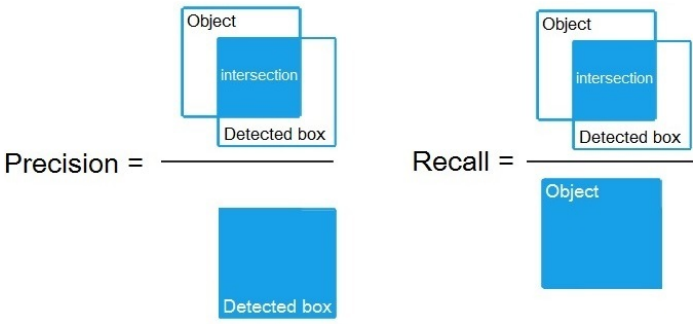


Figure 2.7: A cartoon showing the main metrics for the quality assessment of CNN (Image source).

	TP = true positive TN = true negative	FP = false positive FN = false negative
Precision	$\frac{TP}{TP+FP}$	
Sensitivity/Recall	$\frac{TP}{TP+FN}$	
mAP	$\frac{1}{N}\sum_{i=1}^{N}\frac{TP}{TP+FP}$	

Table 2.1: 2*D* detector's metrics.

2.3 ORIENTATION OF 2*D* PIXELS INTO 3*D* SPACE

There are multiple ways to localize a pixel into 3*D* space. These methods are using computer vision and stereoscopic techniques for the creation of *depth maps*, as well as some newer techniques like the well known *dense image matching* [45] and other photogrammetric techniques [46]. Hence, in this section some of the most prominent methods are presented.

In literature, the terms *depth map* and *disparity map*, are referring to the correlation of every pixel in 2*D* imagery with their distances from the camera. *"By using the principle of triangulation, the disparities of a number of* 3*D points mapped to pixels in two images are computed. Then the visual information of depth is also recovered"* [47]. Furthermore, advances in computer vision allow us to generate reliable 3*D* point clouds from depth images. In more detail, the study [48] uses and presents a workflow on how to generate georeferenced 3*D* point clouds (pseudo-light detection and ranging LiDAR) by using the depth images and the information of the relative position of the cameras.

The oldest photogrammetric technique is the *stereo matching*. With the technique of stereo we can combine two or more images (multi view) for the creation of the well known depth-imagery that gives us an illusion of the 3rd dimension [49]. This technique requires the calibration of the camera (intrinsic and extrinsic), with the latter to be the key for the creation of the depth images. In more detail, using the method of triangulation/3D reconstruction and having knowledge of the position of and pose of the camera (calibrated camera), it is possible to triangulate distances (see Figure 2.8) and make estimations of the distance to points in the world [50], [51], [52].

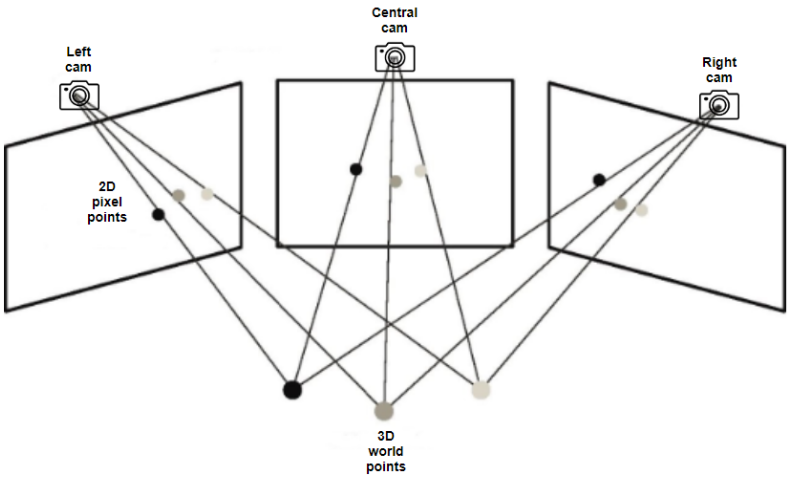


Figure 2.8: Triangulation of 3 cameras with known relative distances.

For the triangulation to be performed we need to have knowledge about the pinhole camera model (Figure 2.9) and the intrinsic and extrinsic parameters of

the camera(s) (see Section 2.3.1). Equation 2.2 shows the forward projection model, from a point in world coordinates to 2D pixel (projection matrix). Hence, creating the back projection we are able to project the 2D pixels back to 3D world coordinates. This procedure is also known as 3D reconstruction.

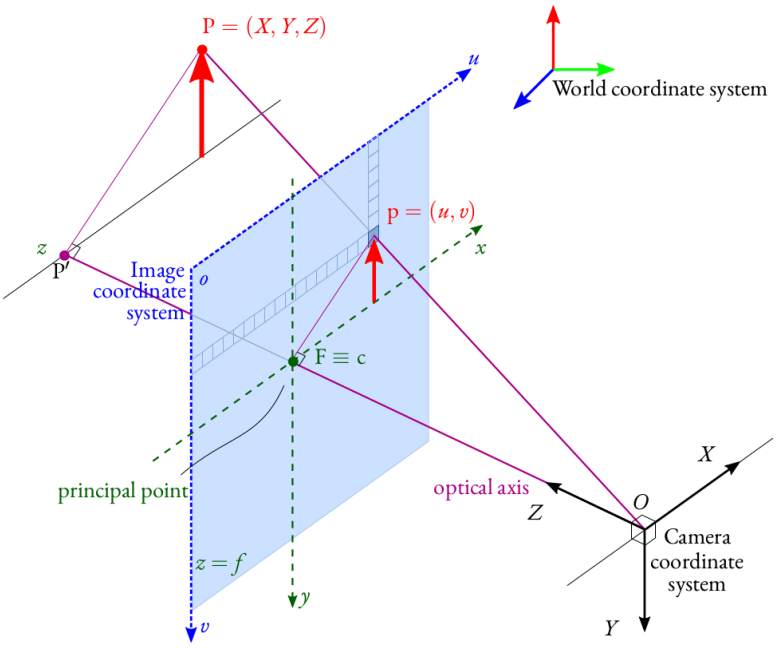


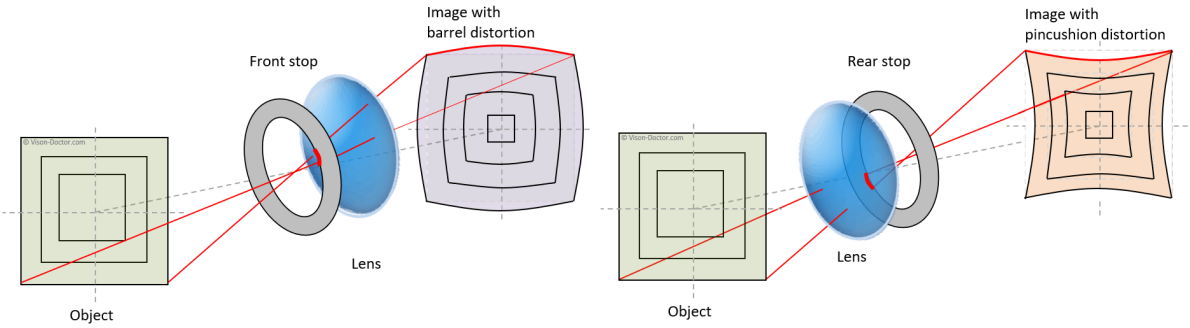
Figure 2.9: Pinhole camera model. An illustration of the forward projection (Figure taken from [7]).

$$\mathrm{P} = \mathrm{K}[\mathrm{R} \mid \mathrm{t}]\mathrm{W}, \quad \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

Where P is the projected point vector in 2D, K the camera matrix (see equation 2.3), R the rotation matrix (see also equation 2.5), t the translation vector (*East*, *North*, *Down*) and *W* a point in world coordinates. The multiplication of the camera matrix K with the transformation matrix [R | t] is called projection matrix.

2.3.1 Camera Calibration

"In geometrical camera calibration the objective is to determine a set of camera parameters that describe the mapping between 3-D reference coordinates and 2-D image coordinates" [53]. When a pinhole camera projects the 3D coordinates of a point to 2D image plane, it introduces distortion (see Figure 2.10). Furthermore, transforming from the real positions to pixels involves scaling and translation, since the size of the pixel is not standard (e.g. 1 [mm]) and since the principal point does not correspond to the center of the image sensor respectively. Hence, performing camera calibration is essential to extract the camera properties and in particular the intrinsic and extrinsic parameters of it (see Figure 2.11).



(a) Barrel distortion. **(b)** Pincushion distortion.

Figure 2.10: Lens distortion types (Image source).

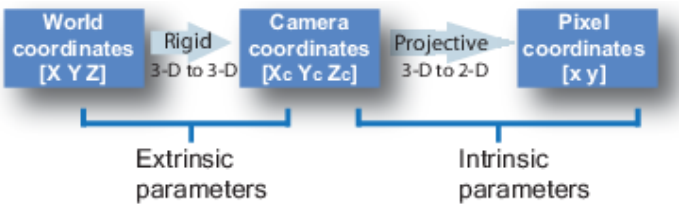


Figure 2.11: Forward model of the extrinsic/intrinsic camera parameters (Figure taken from [8]).

Intrinsic : The pinhole camera model defines the geometric relationship between a $3D$ point and its $2D$ corresponding projection onto the image plane. When using a pinhole camera model, this geometric mapping from $3D$ to $2D$ is called a perspective projection. The retrieved information we get when we perform perspective projection is the *focal length* of camera's lens which is the optical distance from the point where light rays converge to form a sharp image of an object to the sensor. In addition, we retrieve information about the *principal point*, which is defined as the spot of the image plane which the perspective center is projected, and the *radial/tangential* distortions, which are positional "shifts" of the projected points into the image plane (see Figure 2.12). After we retrieve the intrinsic parameters of the pinhole camera, we are able to construct the camera matrix (see equation 2.3) that is used to undistort the image as well as to create the projection matrix Equation 2.2.

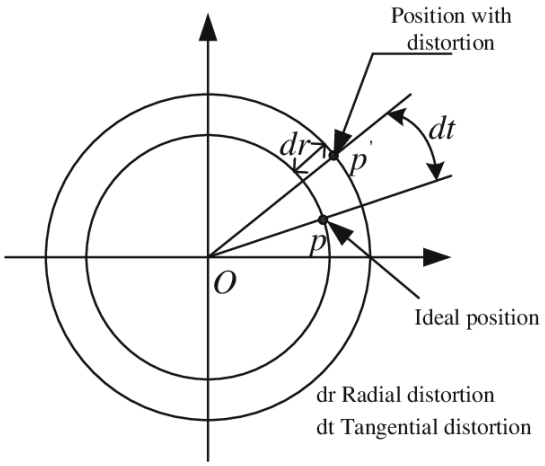


Figure 2.12: Intrinsic camera parameters. Radial and tangential distortions in $2D$ image plane (Figure taken from [9]).

$$\boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Where f_x, f_y the focal length of the camera and c_x, c_y the principal points of the lens.

Extrinsic : As opposed to the intrinsic parameters that describe internal parameters of the camera (focal distance, radial lens parameters), the extrinsic parameters indicate the external position (see equation 2.4) and orientation of the camera in a reference system (see equation 2.5). They are used to transform 3D coordinates to a camera coordinate frame. In addition, extrinsic parameters can be used to describe the relationship between multi-camera systems. In more details, the pose parameters of the camera are defined as the *pitch*, *yaw or heading* and *roll* (see Figure 2.13).

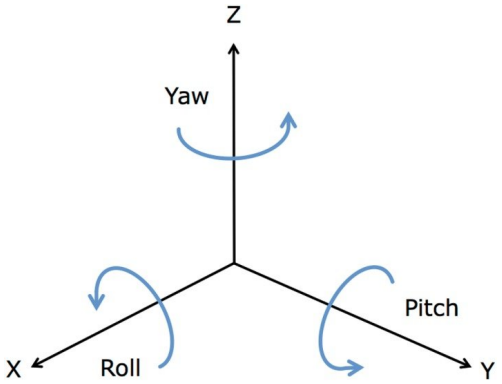


Figure 2.13: Extrinsic camera parameters - Pose (Figure taken from [10]).

$$\boldsymbol{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{2.4}$$

Where t is the translation vector and t1,t2,t3 are the easting, northing and altitude in [m] respectively.

$$\boldsymbol{R} = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix} \tag{2.5}$$

Euler angles rotation formula used for the construction of the above matrix. Where α, β and γ are the yaw, pitch and roll in [rad] respectively.

There are two main approaches for camera calibration, the photometric-calibration which involves 3D or planar objects with known geometry, and the self-calibration which is using constraints on the camera properties and the image scene. The latter one is mainly used in 3D modeling [54]. Among the most famous, easy to implement and fast methods is the use of control points like the well known checkerboard which has been used in computer vision the last decades. The latter method uses pattern recognition algorithms such as Delaunay triangulation to identify control points on the image [55] and then by the use of linear regression techniques to calculate the distortion that the lens introduces. Furthermore, other techniques such as the one that has been proposed by the [56], propose the use of non-linear methods to calculate the distortion of the image by using circular control points.

2.4 UNSUPERVISED POINT CLOUD CLASSIFICATION

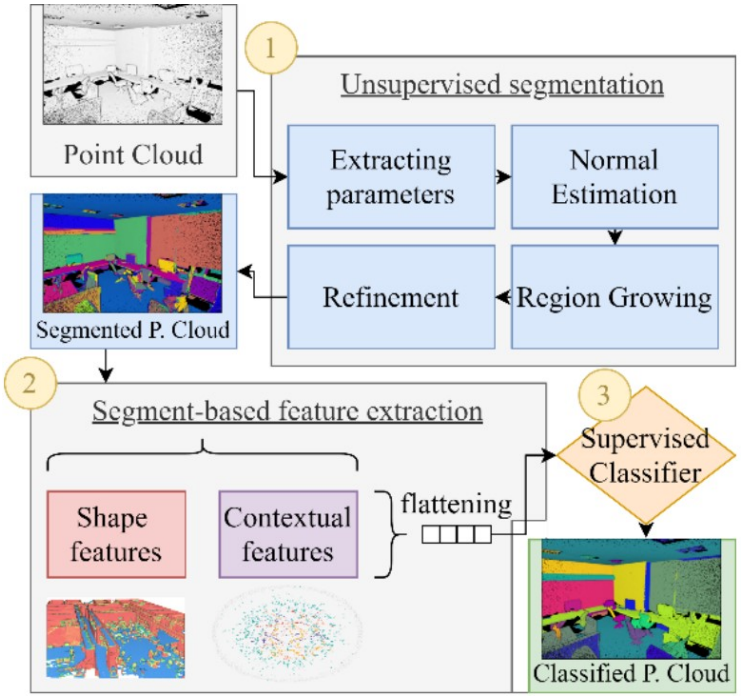


Figure 2.14: Workflow of how unsupervised aid to supervised classification (Figure taken from [11]). First, using the region growing algorithm, an initial segmentation of the point cloud is made. This segmentation is used to extract geometric features to further refine the classification of the data.

Unsupervised classification techniques, need the minimum involvement of the user with no use of training data, while they use the point density and the geometric properties of the unstructured point cloud (topology) (see Figure 2.15). Thus, unsupervised methods are used mainly as the initial segmentation of the raw unstructured point cloud data, to aid the final object-based classification (see Figure 2.14). Depending the application, unsupervised classification methods might be suitable because they are low in computational complexity. *"Automatic shape segmentation is thus valuable to avoid labour intensive labelling"* [11].

2.4.1 Geometric Feature Extraction

"Ground objects can be regarded as a combination of structures of different geometries. Generally, the structural geometries can be grouped into linear, planar and scatter shapes (see Figure 2.15). A good segmentation of objects into different structures can help to interpret the scanned scenes and provide essential clues for subsequent semantic interpretation" [57]. This can be done using only the raw X, Y, Z attributes of any point cloud [58].

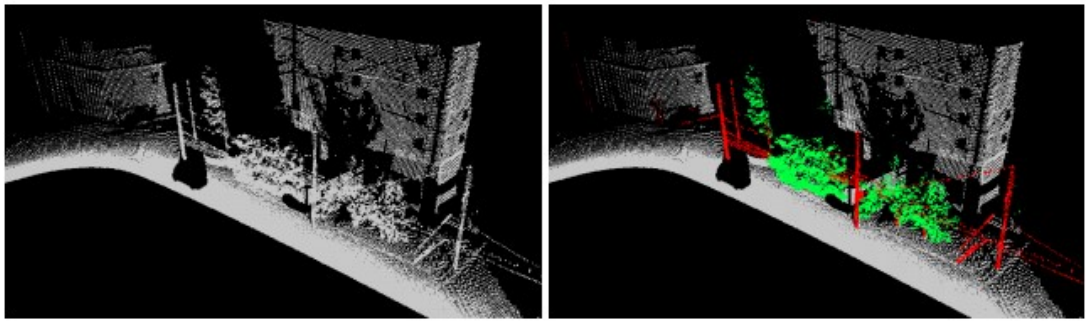


Figure 2.15: Unsupervised classification. Left: Raw point cloud. Right: Segmented point cloud based on the linearity (red), planarity (gray) and scattering (green) geometric features [12].

"Feature extraction from a range of scales is crucial for successful classification of objects of different size in 3D point clouds with varying point density" [15]. For the geometric properties of the point cloud to be retrieved, the use of the local properties of a neighbour area of the points should be investigated. *"The unstructured nature of 3D point cloud makes it necessary to recover neighbourhood information before meaningful features can be extracted"* [15]. After the investigation of the neighbourhood of the points, the construction of the covariance matrix of the created eigenvalues is essential to derive the local properties of the points. *"The features used in the separation of different objects are important for successful point cloud classification. Eigen-features from a covariance matrix of a point set with the sample mean are commonly used geometric features that can describe the local geometric characteristics of a point cloud and indicate whether the local geometry is linear, planar, or spherical"* [59]. Hence, extracting the maximum eigenvalues out of the covariance matrix, we can combined them to create the following features (see Table 2.2). Based on the created geometric features we are able to classify unstructured point clouds.

[60] suggests a pre-segmentation approach using multi conditional random field (CRF) classifier in order to define at a high-level the structure of a voxel.Those super voxels are defined by using clustering techniques such as *k-means*. This method can be defined as "weakly" supervised segmentation.

	Formula
Linearity	$L_\lambda = \frac{\lambda_1 - \lambda_2}{\lambda_1}$
Planarity	$P_\lambda = \frac{\lambda_2 - \lambda_3}{\lambda_1}$
Scattering	$S_\lambda = \frac{\lambda_3}{\lambda_1}$
Omnivariance	$O_\lambda = \sqrt[3]{\lambda_1 \lambda_2 \lambda_3}$
Anisotropy	$A_\lambda = \frac{\lambda_1 - \lambda_3}{\lambda_1}$
Change of curvature	$C_\lambda = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$
Mean elevation	$M\_e = \frac{\sum_{i=1}^{n} Z}{n}$
Elevation difference	$E\_d = Z_{\max} - Z_{\min}$

Table 2.2: Geometric features. Where $\lambda_1 > \lambda_2 > \lambda_3$ and Z the eigenvalues with the higher values of the covariance matrix and the vertical direction respectively [15].

2.4.2 Principal Component Analysis (PCA)

"*PCA is an orthogonal transformation technique used to convert a set of points with possibly correlated variables to another set of points with linearly uncorrelated variables called principal components in which the first principal component has the largest variance and each successive component is orthogonal to the preceding components*" [59]. Hence, applying PCA to the created geometric features, the components with low variance - high covariance, become redundant, leading to a smaller subset of components having higher discriminative power.

2.5 EXISTING PIPELINES FOR FUSING $2D - 3D$

A resent study focused on RGB $2D$ imagery and LiDAR data using a CNN for autonomous vehicle environment, showed remarkable results [61]. The study was based on the MMS MENGSHI, a system consisting of multiple visual and ranging sensors. The pipeline of this study starts with the creation of an image that includes the distance information in every pixel between the camera and the object (depth-image), from LiDAR data. Then, it continues with the object extraction in $2D$ imagery that is based on ground truth data. Lastly, these images and the combination of red, green, blue, depth (RGB-D) were used to train the classifier. Hence, the inputs of the classifier are objects consisted of four channels (R,G,B,Depth).

[19] is one of the most important studies on the object detection and object recognition for the self driving in urban areas. The data of the study derived by *LYNX* MMS comprised of RGB cameras and LiDAR. In this paper, the fusion between $2D$ and $3D$ data was made for the road sign recognition by using the $2D$ imagery as input for CNN. In more detail, the workflow started by segmenting and clustering the $3D$ point cloud for sign detection. The use of a digital surface model DSM and the property of high intensity due to reflectivity of the signage were used for the segmentation of the point cloud. To further classify and isolate the points that represent the signage, the Density-based spatial clustering of applications with noise (DBSCAN) clustering method was used. Next, having knowledge about the relative position between the vehicle and the cameras as well as knowing the trajectory of the vehicle, the clusters were re-projected on $2D$ images. Finally, using hierarchical classification techniques, the semantics of the signage obtained and stored.

In [21] and [62], a different workflow was used for the fusion of $2D$ and $3D$ data. Airborne high resolution imagery used for the detection of the rail infrastructure. Then, the position of the objects that have been detected in airborne imagery were used as a mask to spatially select matching point from airborne LiDAR point cloud dataset. After this, the implementation of the Random sample consensus (RANSAC) algorithm was used to accurately approximate the linear rail infrastructures (e.g. rails), so to make the best match of the point by keeping the linear segments having the less outliers.

Lastly, the study [63] combines photogrammetry point clouds that have been created by overlapping airborne imagery with the stereo pair technique (see Section 2.2), with LiDAR point clouds, to increase the accuracy of the 3D city models.

3 METHODOLOGY FOR OBJECT MAPPING

This chapter deals with the input data, workflow, methods and tools that were used in this project for the creation of the proposed workflow. After introducing the data used for this research, the chapter illustrates the steps made in 2*D* and 3*D* analysis, for mapping of the infrastructural objects (Figure 3.1).

3.1 WORKFLOW

Figure 3.1 illustrates the workflow of this project in a nutshell.

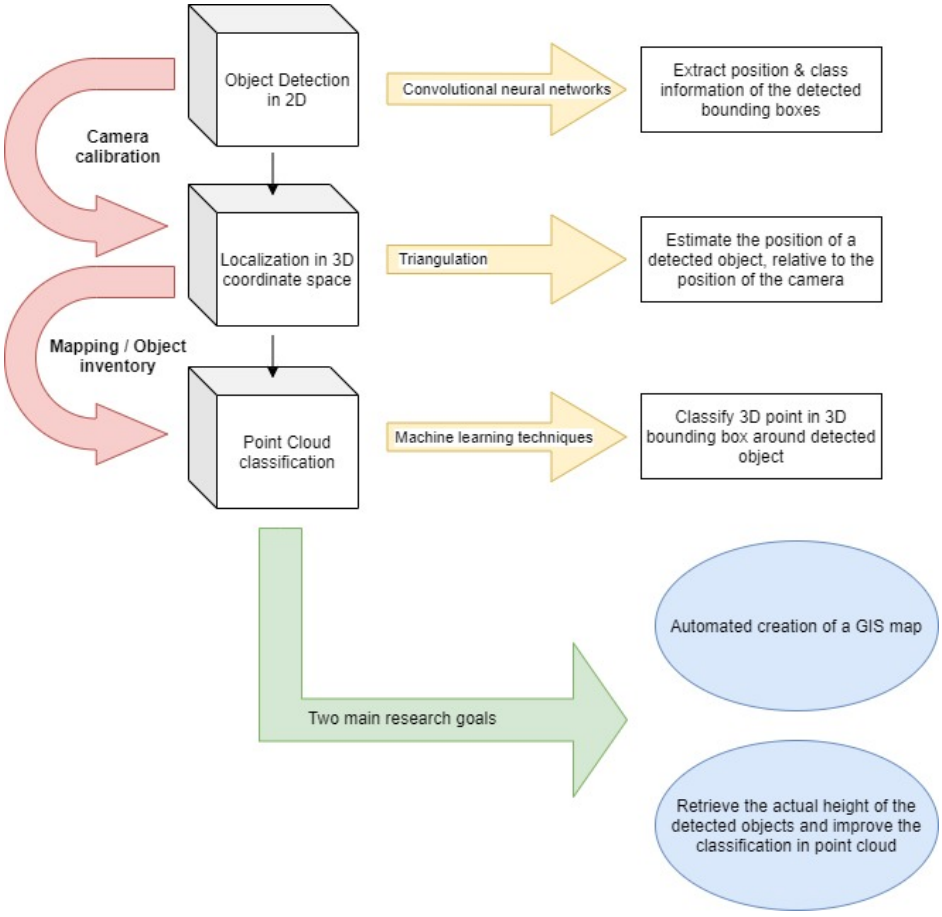


Figure 3.1: Workflow for mapping railway objects. The first 2 boxes refer to 2*D* image analysis while the last box to 3*D* point cloud analysis.

- The workflow starts with the preparation of the data as explained in Section 3.3, to be suitable as an input to train the 2*D* classifier. Next step is the training of the 2*D* classifier in Section 3.4 and the optimization of the 2*D* detector network in Section 3.4.1.
- The first intermediate step of a camera calibration (see Section 3.6) was necessary to re-project the estimated central coordinates of the created bounding

boxes (2D pixels) into 3D world coordinates. Here, multiple functions were created to transform the translation and pose information of the center of the mass of the IMU system to a world coordinate system. The same procedure was followed for the camera(s). Hence, mapping the forward projection from 3D coordinates to 2D image plane, we were able to perform the backward projection (from 2D to 3D).

- After we georeferenced the cameras in Section 3.6, we were able to move to the second box (Section 3.7). By combining pairs of 2D frames that include the central coordinates of the created bounding boxes referring to the same object, we retrieved the actual position of the objects in the 3D world coordinate system. The triangulation method was implemented to reconstruct the 3D position of the detected objects - from 2D pixels to 3D world coordinate system.

- The final intermediate step as described in Section 3.8, is the mapping/object tracking and inventory. This step deals with the central coordinates triangulation and the inventory of each individual detected object. The product of this step a geographical information system (GIS) map having positional and class information of all detected objects.

- After the creation of the object inventory, 3D light detection and ranging (LiDAR) point cloud data used to refine the vertical information. In Section 3.9 it is described how the 3D point cloud data are cropped based on the positional results from the previous step, and divided into classes like terrain, other and object, using machine learning (ML). Next, points that classified as *class: object* are used to refine the height information of each object.

3.2 INPUT DATA

The input data consist of 2D imagery, position data - .csv file - including the extrinsic information of the inertial measurement unit (IMU) system of every 2D video frame, and 3D light detection and ranging (LiDAR) data.

2D **Imagery:** The video frames are 2D images taken via three pinhole cameras (Section 2.3.1) that are mounted on Fugro's mobile mapping system (MMS) **RILA** (Section 2.1) which is installed at the tail of the train. The 2D imagery data were compiled as moisac of the three synchronized 2D frames under the same frame index. Hence, the raw imagery data is a tessellation of all cameras, where each image has a dimension of 4096x4096 pixels (see Figure 3.2). The video frames used in this thesis, are taken in 2019 and they capture 2.2km of railroad in the area of Ely in the United Kingdom (see Figure 1.3).

Figure 3.2: Raw 2*D* imagery data. A tessellation of raw images of the 3 cameras mounted on Fugro's MMS RILA. The dimensions of every individual 2*D* video frame are 2016*x*2016 pixels.

Position data: A csv file having positional, orientational and time information of the center of the mass of the RILA's IMU system, at the time of capturing the 2*D* video frame data. In more detail the file consist of ten columns, as specified in Figure 3.3, *frame index/longitude/latitude/easting/northing/altitude/heading/pitch/roll/time*, of the central of mass of the IMU system, and multiple rows (as much as the video frames).

	A	B	C	D	E	F	G	H	I	J	K
1	Path	Longitude	Latitude	Easting	Northing	Altitude	Heading	Pitch	Roll	CaptureTime	
2	frames/ep11-201002303-20190430-092646/frame_0756.jpg	0.27260879	52.43134713	554601.5283	283912.145	6.319065886	137.0884864	2.075714244	-1.69479346	04/30/2019 09:27:55	
3	frames/ep11-201002303-20190430-092646/frame_0757.jpg	0.272628172	52.43133366	554602.8929	283910.689	6.32592466	137.059368	2.078411849	-1.66003985	04/30/2019 09:27:55	
4	frames/ep11-201002303-20190430-092646/frame_0758.jpg	0.272647552	52.4313202	554604.2574	283909.2332	6.333034151	137.0332656	2.062187679	-1.65743476	04/30/2019 09:27:55	
5	frames/ep11-201002303-20190430-092646/frame_0759.jpg	0.272666932	52.43130674	554605.6219	283907.7783	6.340349518	137.030891	2.075061129	-1.67491468	04/30/2019 09:27:55	
6	frames/ep11-201002303-20190430-092646/frame_0760.jpg	0.272686313	52.43129329	554606.9864	283906.324	6.347780689	137.0506953	2.078135191	-1.69017312	04/30/2019 09:27:55	
7	frames/ep11-201002303-20190430-092646/frame_0761.jpg	0.272705688	52.43127985	554608.3505	283904.8699	6.355419284	137.0560786	2.081084835	-1.70905148	04/30/2019 09:27:55	
8	frames/ep11-201002303-20190430-092646/frame_0762.jpg	0.272725053	52.4312664	554609.714	283903.4157	6.363188515	137.0366448	2.09212742	-1.70632353	04/30/2019 09:27:55	
9	frames/ep11-201002303-20190430-092646/frame_0763.jpg	0.272744417	52.43125295	554611.0773	283901.9618	6.371153521	137.0168381	2.075059462	-1.69340723	04/30/2019 09:27:55	
10	frames/ep11-201002303-20190430-092646/frame_0764.jpg	0.272763786	52.43123952	554612.441	283900.509	6.379518178	137.0305175	2.058363428	-1.68200264	04/30/2019 09:27:55	
11	frames/ep11-201002303-20190430-092646/frame_0765.jpg	0.272783162	52.43122609	554613.8051	283899.0572	6.388034088	137.0796122	2.047132013	-1.70196716	04/30/2019 09:27:55	
12	frames/ep11-201002303-20190430-092646/frame_0766.jpg	0.272802535	52.43121267	554615.169	283897.606	6.396229826	137.1490633	2.03772342	-1.70604988	04/30/2019 09:27:55	
13	frames/ep11-201002303-20190430-092646/frame_0767.jpg	0.272821895	52.43119924	554616.532	283896.1544	6.404180429	137.1887482	2.018315836	-1.71906068	04/30/2019 09:27:55	
14	frames/ep11-201002303-20190430-092646/frame_0768.jpg	0.272841228	52.43118581	554617.8932	283894.7014	6.411813565	137.145541	2.014985419	-1.72202103	04/30/2019 09:27:55	

Figure 3.3: Capture of .csv file, including the position data, the extrinsic and the time information of the RILA's IMU system, for every video frame.

3*D* LiDAR **Point Cloud:** A point cloud dataset acquired via RILA, that captures the railway and its surroundings. As mentioned in Section 2.1, the MMS RILA consists of multiple LiDAR systems (see Figure 1.2). Because we are interested in the railroad surroundings, the 3*D* point cloud data acquired via the 360° LiDAR scanner - not via the two railscanners. The accuracy of the point cloud data is up 2*cm* level in the horizontal and 3*cm* in the vertical plane [26].

3.3 PREPARATION OF THE DATA

The preparation of the data is split into two parts. The first part consist of labeling of the data (see Section 3.3.1) in order to train the convolutional neural network

(CNN), while the second part consists of cropping/undistorting (see Section 3.3.2) the 2*D* video frames (see Figure 3.4).

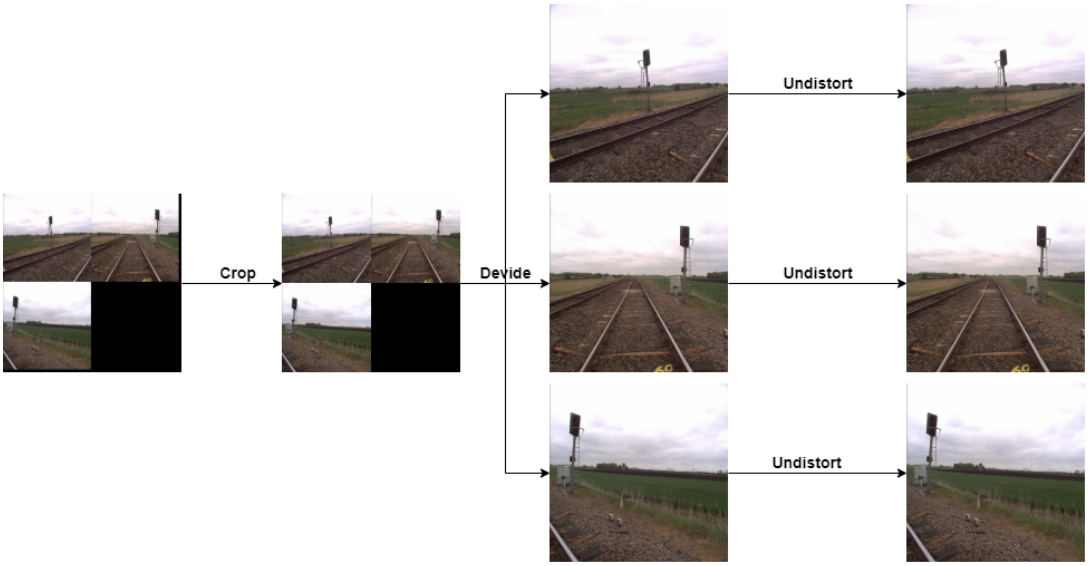


Figure 3.4: Preparation of the raw imagery data consists of cropping (left), dividing (center) and undistorting (right).

3.3.1 Labeling Data - Training Data for detecting equipment boxes and traffic signals

Here we describe how we obtain training data for training our classifier to detect objects in the 2D imagery. *"Every artificial intelligence system needs big data for training. In particular, artificial intelligence for object detection requires a lot of images for training"* [64]. As a rule of thumb, an effective training dataset consist of 2000 labeled images for every class, having different scale, rotation and illumination conditions [65].

We want to detect two classes of objects and therefore we also train for these two classes. The first class consist of equipment boxes (see top left in Figure 3.5) and the second of traffic signals (see bottom left in Figure 3.5). The choice was made due to the shape and structure differences among these two classes as well as their frequent appearance in the railroad. Another reason for this choice was their similarity to other infrastructural object near the rail. Hence, the results of these two objects can be used as a quality indicator of the performance of the neural network of you only look once (YOLO). In more detail, the class *"box"* has many similarities to planar objects such as buildings or warehouses next to the railroad, while the *"signal"* class can be confused with poles.

Labeling the training data was done manually by myself using the open source software application *labelimg* [66]. The number of labeled images is 995, with the majority of them having more than one object to be labeled (see Figure 3.5).

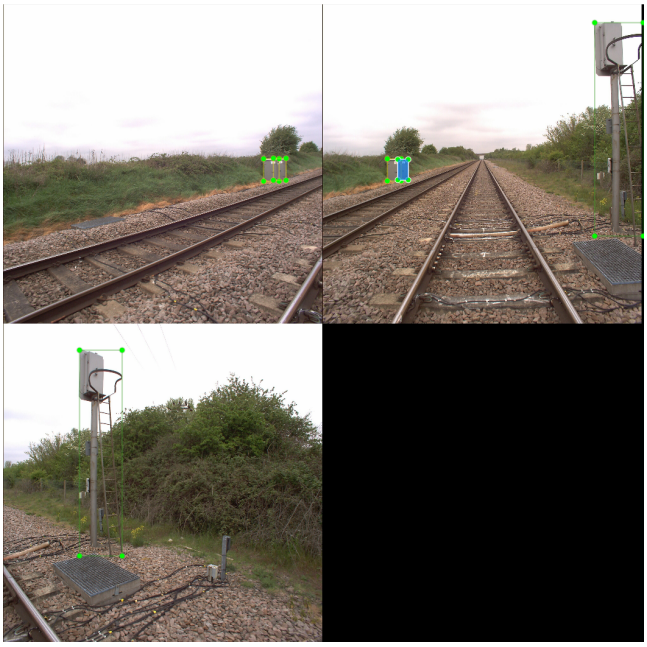


Figure 3.5: Creation of training data - Labeling procedure on raw input data. Top-left: labeled equipment box; Bottom-left: labeled signal; On top right, both objects classes are present.

Note that the quality of the training data (labeling), dominates the performance of the 2D detector. Hence, wrong labeling will lead to bad performance of the detector and visa-versa, [[67] [68]].

3.3.2 Cropping - Undistorting the Input Data

In Figure 3.2, the raw 2D video frames is a tessellation of 3 frames. In order to use the triangulation method to re-project the central coordinates of the created bounding boxes to 3D space, the 2D images should be reconstructed to their original form as acquired by the cameras. Hence, the raw images are cropped and split into three distinct images (see Figure 3.4) with sizes of 2016x2016 pixels, to become suitable input data for the 2D detector. Lastly, using the information of the intrinsic parameters of the camera(s) (Section 2.3.1), the images are undistorted such that the 2D coordinates of the created bounding boxes are corrected from shifts caused by lenses distortions.

Figure 3.6 and Figure 3.7, illustrate the undistortion of an image using 2 different methods. The methods differ in their choice of the scaling parameter *alpha*. The *alpha* value ranges from 0 to 1, where zero *alpha* eliminates unwanted black pixels which also cause the loose of some image information, while an alpha value of one *alpha* value retains all image pixels by introducing some extra black pixels at the edges [69]. In Figure 3.6, there are no black pixels on the edges, while there is a loss of pixel-information compared to Figure 3.7. In this application, the 2nd method is the preferably method for undistortion, as no valuable information is lost when applying it.

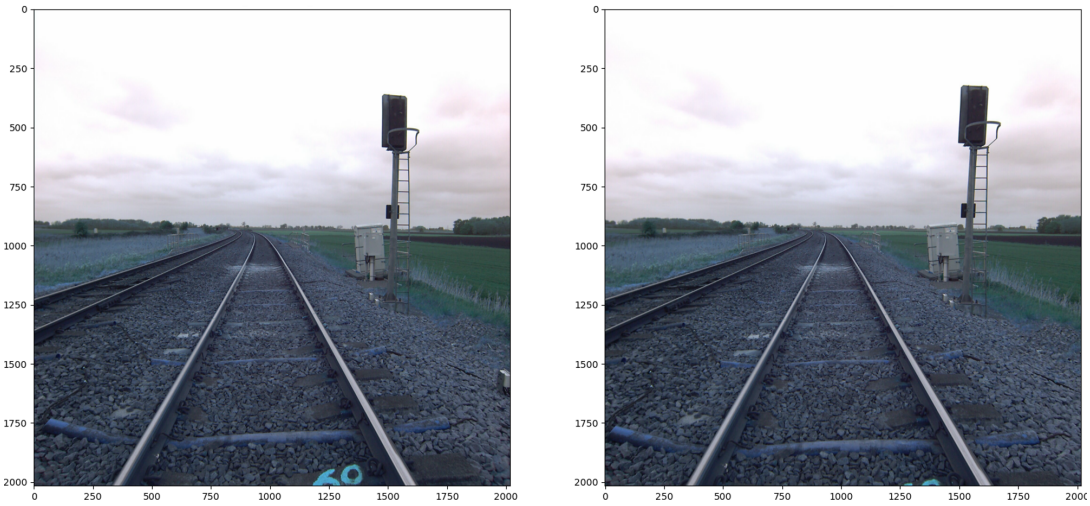


Figure 3.6: Distorted (left) and undistorted (right) image, due to lens distortions. $Alpha = 0$.

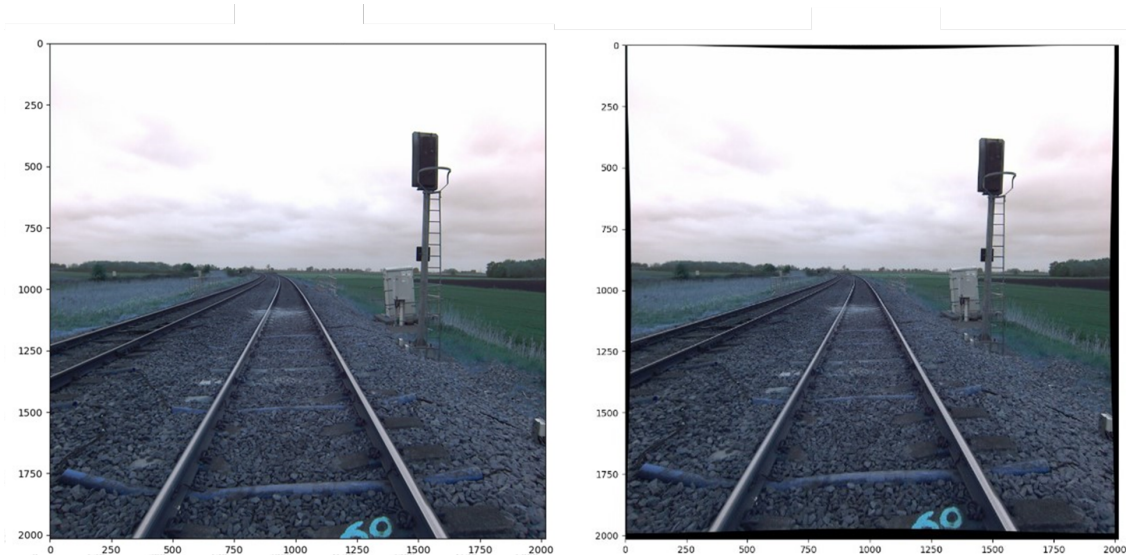


Figure 3.7: Distorted (left) and undistorted (right) image, due to lens distortions. $Alpha = 1$.

3.4 TRAINING THE YOLO 2D OBJECT DETECTION

"Good" training of the deep neural network, is essential for its performance as $2D$ object detector. The metrics that are used in Section 2.2.1 as quality indicators of the training procedure of YOLO, are the *loss function* (see Equation 2.1) and the mean Average Precision (mAP) (see Table 2.1); both metrics were analysed in Section 2.2.1. An ideal convergence through the training procedure of the loss function and the mAP is to be stabilized bellow 1 $pixel^2$ and above 90% respectively [70], [65]. Many factors play a role for this to happen. The major factors are illustrated in detail in Section 3.4.1.

In our case, the output of the training procedure showed promising results as the created weights end up having an *average loss* = 0.1687 $pixel^2$ and a mAP above 95% (see Figure 3.8).

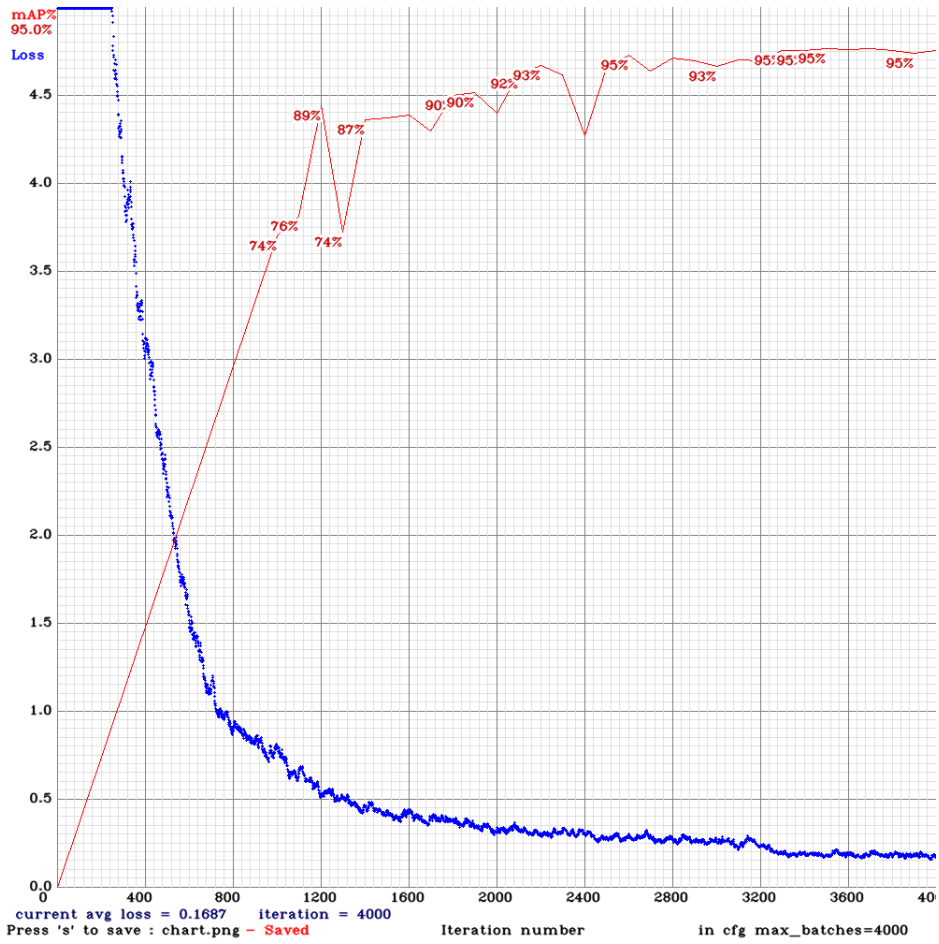


Figure 3.8: Plot that indicates the performance of the training procedure of YOLO's weights for 2D object detection. As the 2D detector is being trained, the mAP (in red) is increasing and hence, it eliminates false positives. The decreasing trend of the loss function (in blue) indicates that the 2D detector's predicted bounding boxes are in line with the ground truth based on the intersection over union IoU principal. Both curves show sharp changes at the beginning of the training while they stabilize at the end.

3.4.1 Optimization of the Yolo Neural Network Model

For detection and localization of the data, you only look once (YOLO)v3 is used. The reason for that is that YOLOv3 detects and localizes objects in 2D pixel space with high accuracy and speed. In addition it is able to perform well on small objects (see Section 2.2.1). This was suitable for this application, because its architecture permits detection in various scales. Due to the movement of the train cameras, the railway objects appeared in multiple scales, hence, the use of YOLO was convenient. The 2D position in pixels (i.e. center of the bounding boxes) of the detected objects are later used to reconstruct the 3D world coordinate position of the objects.

The repository used to train the classifier was imported from darknet_for_colab and was used on a local machine. The key factors that were customized in order to have most fruitful results were:

- The maximum number of batches/iterations, which indicates how many times the training runs thought the input data. As a rule of thump, 2000 batches for every class is needed for the optimization of a training [65].
- The number of steps. The first step should be 80% of the max batches (iterations), while the second should be 90% of it. Steps, present checkpoints

that define how the learning rate should change. In this way the 2D detector avoids over fitting on the training data.

- The number of convolutional filters of YOLO that extract image features (e.g. line detectors), tuned up with the following equation: Number of filters $=$ (classes $+5$) $*$ 3. Hence, in this case the value of the convolutional filters is 21.

- Subdivision of the input images. Depending on the scale of the objects that we want to detect, we can tune-up the subdivisions of the images so the network could perform to large-medium-small scale objects. As mentioned before, in this application we have multiple 2D video frames as a sequence that include the objects of our interest in all scales. Hence, the 2D input images were subdivided by $32, 16, 8$ (see Figure 2.4).

3.5 OBJECT DETECTION

The YOLO deep neural network (Section 2.2.1) is used for the detection of the railroad infrastructural objects in 2D images. As input for the 2D detector, undistorted 2D video frames (see Figure 3.6) captured via the 3 mounted cameras in RILA, in a part of the railway that did not took part in the training procedure were used. Except of the 2D images with created bounding boxes, the output of the detection includes .csv files with information on the specific camera(s) that captured the 2D video frames. This information includes their ID frame and 5 coordinates of every bounding box (4 corners and the central coordinate). After this step, georeference of the camera (Section 3.6) and the method of triangulation used to project 2D point into 3D world coordinate space (Section 3.7).

3.6 EXTRINSIC CALIBRATION – FRAME GEOREFERENCING

To be able to reconstruct the 3D position of a 2D pixel to 3D world coordinate system, the camera(s) captured the images should georeferenced - to know its position and pose in world coordinate system. The intrinsic and extrinsic parameters (Section 2.3.1) of the three cameras were provided in an .xml file. Regarding the extrinsic parameters (translation and rotation), the central camera were defined as the center of the local coordinate system. Hence, the *pitch, yaw, roll* and the *offset* of the other cameras were taken with respect to the central camera. In addition, for every 2D video frame, the translation and orientation of the IMU system with respect to the British coordinate reference system ($OSGB36 - OSTN15$) was provided in the .csv file (Section 3.2). Hence, having knowledge of the fixed position and orientation of the central camera with respect to the IMU, we were able to georeference the cameras with respect to the British coordinate system.

In more detail, the transformation matrix of the central point of the IMU was created with respect to the world coordinates system. The transformation matrix is a combination of a rotation matrix and translation vector (see equation 3.1). Constructing the transformation matrix of the central camera with respect to the IMU and multiplying it with IMU's transformation matrix, the transformation matrix of the camera with respect to the British coordinate reference system was created. Hence, with this procedure we georeferenced the central camera. Creating also the transformation matrices of the central camera with respect to the other cameras, we georeference the remaining cameras as well. Figure 3.9 illustrates in a nutshell the complex transformations of the extrinsic parameters of the cameras in order to be

georeferenced, which allowed us to proceed to the next step of the triangulation as described in Section 3.7.

$$\boldsymbol{M} = \left(\begin{array}{ccc|c} R_{11} & R_{12} & R_{13} & X \\ R_{21} & R_{22} & R_{23} & Y \\ R_{31} & R_{32} & R_{33} & Z \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{3.1}$$

Where R_{ij} are the elements of the rotation matrix (see equation 2.5), X,Y,Z the elements of the translation matrix (see equation 2.4) and M the transformation matrix.

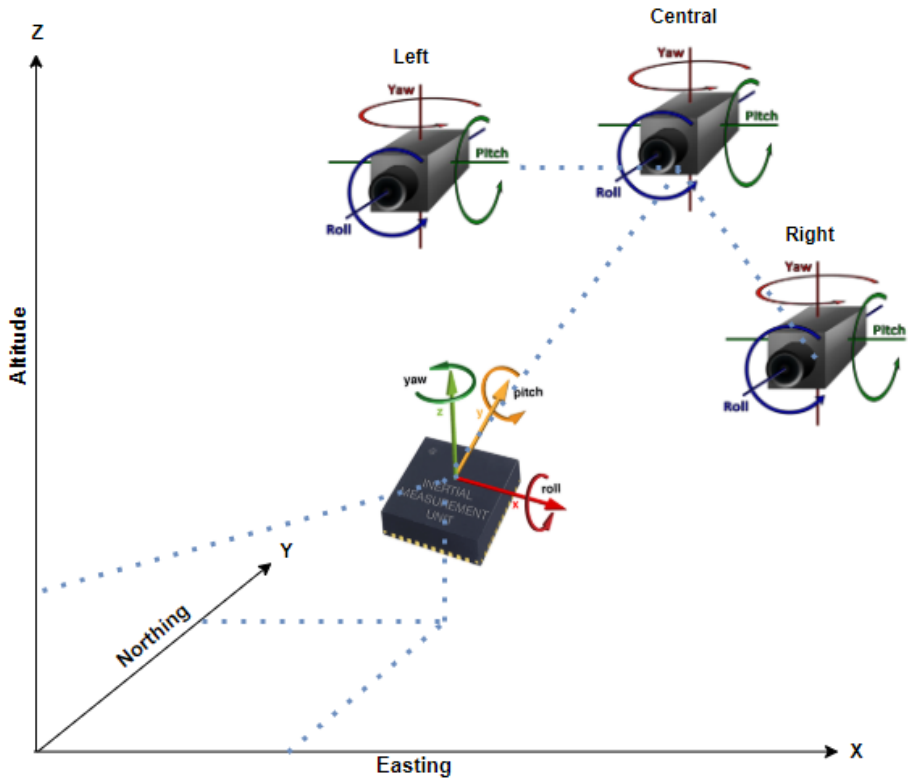


Figure 3.9: A cartoon showing the complex extrinsics of the cameras and IMU system mounded on the Fugro's MMS RILA.

3.7 TRIANGULATION – 3D RECONSTRUCTION

As mentioned in Section 2.3, the method of triangulation was used to back-project a 2D pixel into 3D coordinate space (*reverse perspective projection*). Having all the needed information of the coordinates of the detected objects, the OpenCV's build-in function *cv2.triangulatePoints* was used to perform the 3D reconstruction. In more detail, the function's technique is based on the intersection of the reconstructed 3D vectors from the camera(s) to the object in one or two 2D frame(s) in case two or one camera(s) involved respectively.

As input to the function, a pair of the created projection matrices (see equation 3.2) of the cameras as well as their projection points (bounding boxes central points) that relate to the same 2D video frame were used (see Figure 3.10 and Figure 3.11). In order to be consistent with Python's library, the following steps were followed. The translation vectors (see Equation 2.4), initialized in North-East-Down reference

frame and then they were related with the 3D world frame East-North-Up, by applying additional frame transformation.

$$\text{Proj}_M = \text{K}^*\text{M} \tag{3.2}$$

Where K is the camera matrix (see equation 2.3) and M the transformation matrix (see equation 3.1). The construction of the $Proj_M$-projection matrix, is needed for the projection of the 2D pixels to 3D world and visa versa.

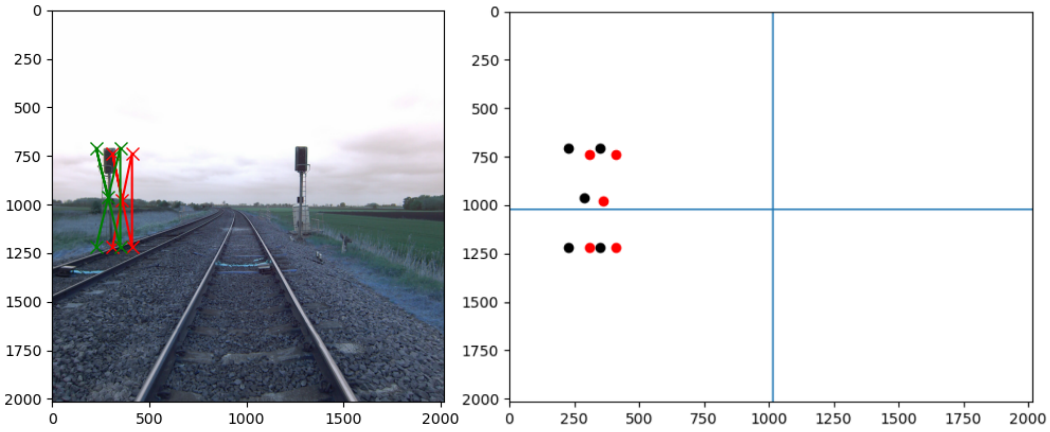
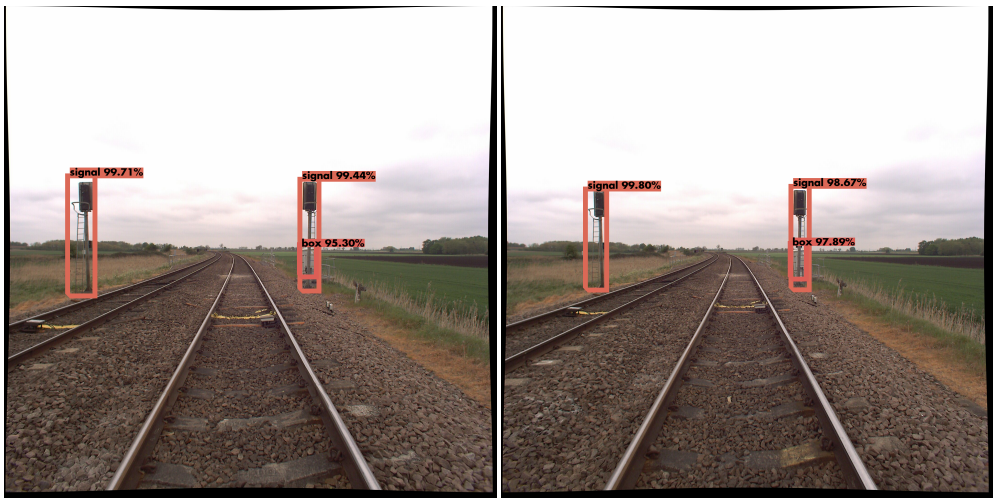


Figure 3.10: The 2D coordinates of the detected bounding boxes of 2 consecutive 2D frames. The coordinates of the corners and the center of the bounding box of the 2nd frame are shown in red.



(a) 2D frame 2386. **(b)** 2D frame 2387.

Figure 3.11: Two consecutive 2D video frames captured via the central camera. In Figure 3.10, the bounding boxes' coordinates of the left signals of both 2D frames are illustrated.

In case there is a sequence of 2D video frames that include one or more objects captured only by one camera, a pair of projection matrices can be replaced by the projection matrix of that camera. In the place of the two projected points of the same frame, 2 different 2D frames captured by the same camera (see Figure 3.12) are used. Hence, we triangulate distances and make estimations of the distance to points in the world, either from multiple cameras or multiple 2D frames from one.

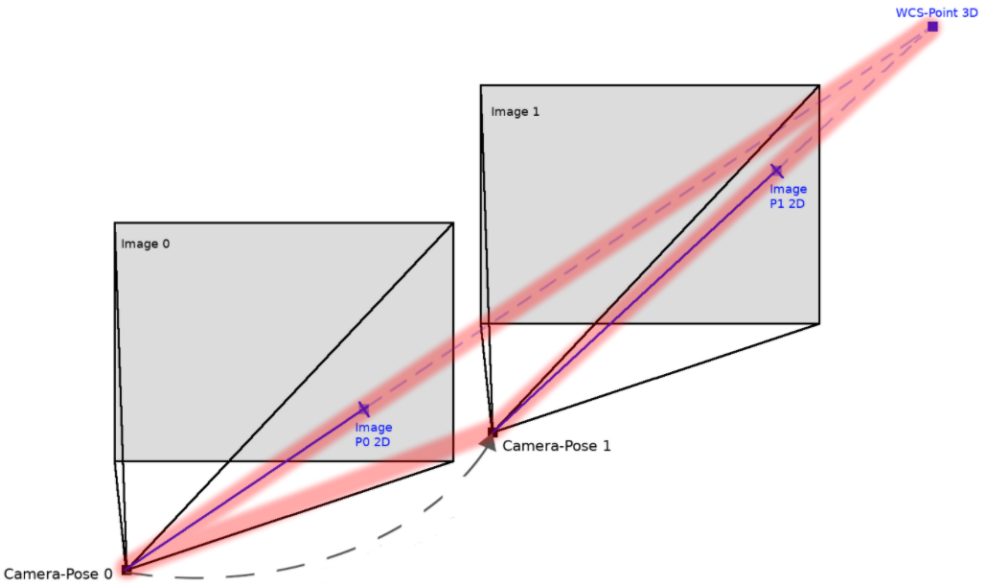


Figure 3.12: 3D reconstruction-triangulation, using 2 frames captured by the same camera (Figure taken from [13]).

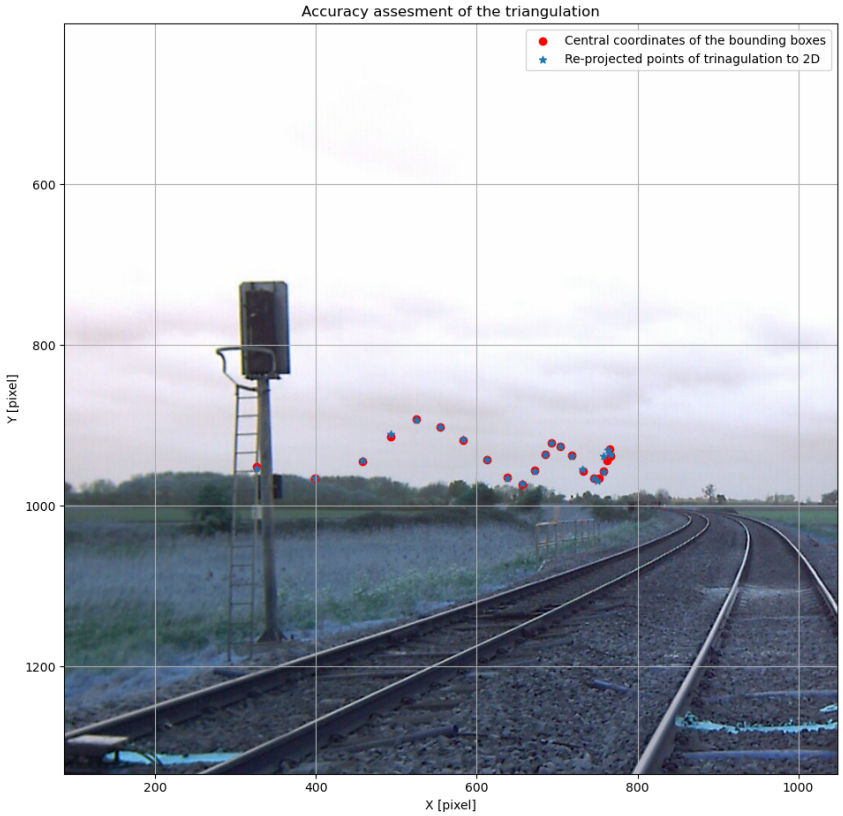


Figure 3.13: Quality assessment of the triangulation. Back-projection of 3D reconstructed points to 2D image plane pixels. The RMSE between them was 1.575 pixels.

Figure 3.10 and Figure 3.11, illustrate an example of bounding boxes' coordinates of only two consequently 2D video frames. In reality, this left light signal appeared in more than two 2D video frames and hence, combining pairs of them, we are able to reconstruct more than once the position of the signal in world coordinate system. Furthermore, the method of triangulation was used, considering the central coordinates of the bounding boxes of 23 video frames that captured and included

the same left signal. In order to asses the calibration and triangulation procedures, we first back-project the created $3D$ coordinates to $2D$ pixels (see Figure 3.13). In red, the bounding boxes' central coordinates of the left signal of 23 consecutive frames, while in blue star, the forward projection (see Figure 2.11) of the triangulations that were made combining the central coordinates of the bounding boxes, back to $2D$ image plane. As it can be seen, the re-projected points fell into the projected points having RMSE 1.575 pixels, which is an indicator of the accuracy of the camera calibration. Another outcome of this figure is the sinusoidal pattern of the $2D$ central coordinates of the detected bounding boxes. This can be justified by the wobbling movement of the train and consequently, the mobile mapping system (MMS) RILA.

Next, outliers from the estimated positions removed, using an iterative algorithm. The algorithm, in every iteration keeps the subset of points that the value of their horizontal plane is every time inside the standard deviation (std) (see Algorithm 3.1). This iterative procedure ended when the number of triangulated points reached the minimum of 4 (see Figure 3.14). The minimum number of estimated positions decided to be 4 after trial and error.

Algorithm 3.1: ITERATIVE OUTLIER FILTERING ($\mathcal{L}$, $\mathcal{L}\_clear$)

Input: A list with the estimated positions of an object $\mathcal{L}$
Output: A list with the Easting, Norhing and Altitude of the object $\mathcal{L}$_clear

$i = 0$
while $length(L) \geq 4$ **do**
  if $|x(i) - \overline{x} \leq std(x)|$ *or* $|y(i) - \overline{y} \leq std(y)|$ **then**
    $L\_clear \leftarrow L.drop(L(i))$
  $i = i + 1$
$x \leftarrow median(L\_clear(x))$
$y \leftarrow median(L\_clear(y))$
$z \leftarrow median(L\_clear(z))$

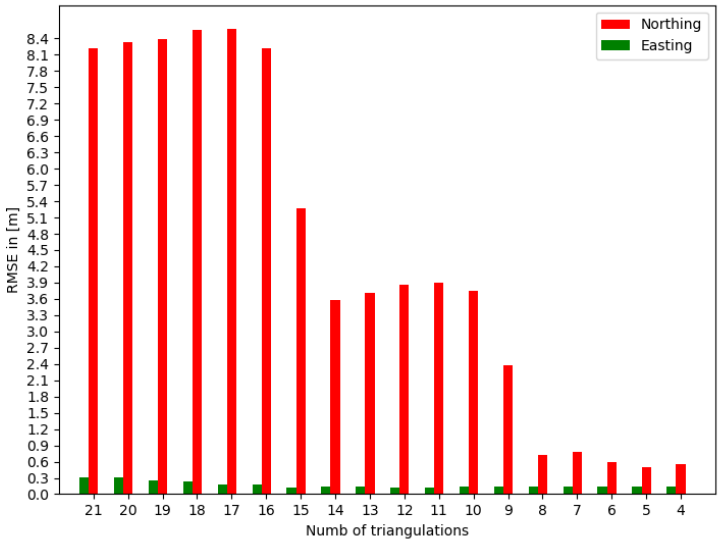


Figure 3.14: Bar chart illustrating the RMSE between the calculated positions and the ground truth. It can be seen that the RMSE decreases during the outlier filtering process. In addition, the *Northing* offset is higher due to the "South-North" direction of the train at this part of the railway (see Figure 3.16).

The Figure 3.15 is a scatter plot of the remaining 4 triangulations. Again, it can be seen that the Northing variations were higher due to the "South-North" direction of the train in these 2D frames. Hence, this "South-North" direction of the train cased sharpest intersection angles of the triangulation vectors and as a consequence the triangulations were less accurate.

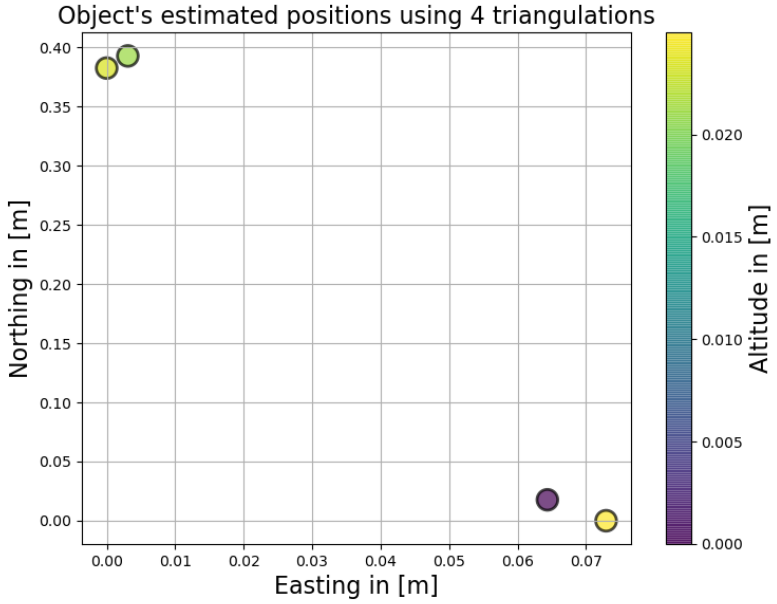


Figure 3.15: Scatter plot illustrating the position of 4 triangulations. It can be seen that the *Northing* range ($\approx 40cm$) is approximately 6 times higher than the *Easting* range ($\approx 7cm$). Lastly, the altitude deviates less at ($\approx 2cm$). The minimum value-coordinate extracted from the others to make the new local coordinate system easier to read.

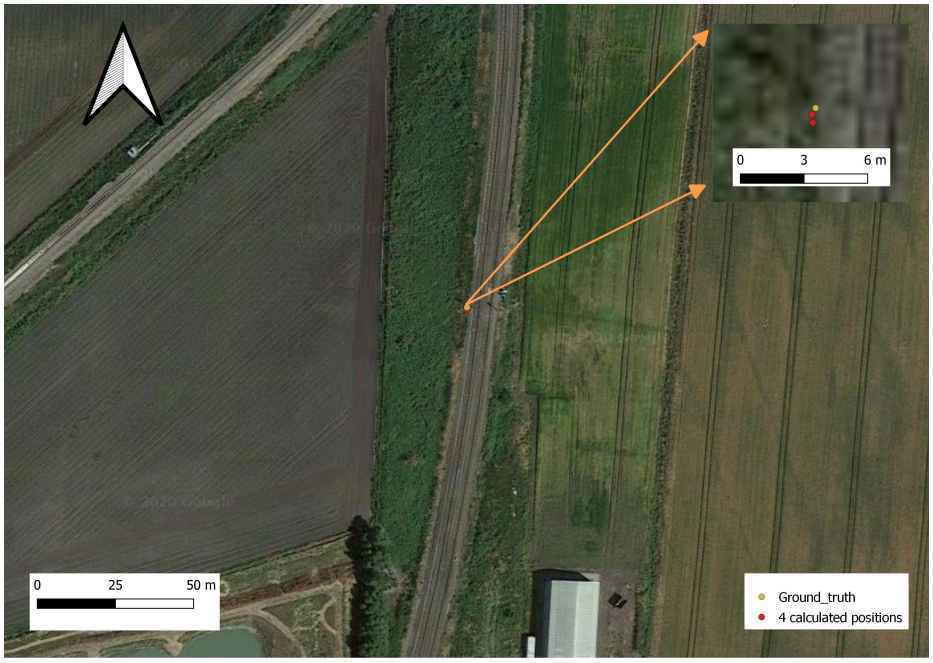


Figure 3.16: The projection of the 4 estimated coordinates made via triangulation, into *QGIS*.

Figure 3.16 illustrates the 4 estimated positions in the British local coordinate system, using the *QGIS*. It can be seen that all the points fell on the actual position of the left signal with a RMSE $\approx$ 30 cm. The median value of the the remained coordinates was used as the final estimated position of signal (Figure 3.17). The acceptance value of the difference between estimated positions and ground truth, was set by Fugro to be $1m$.

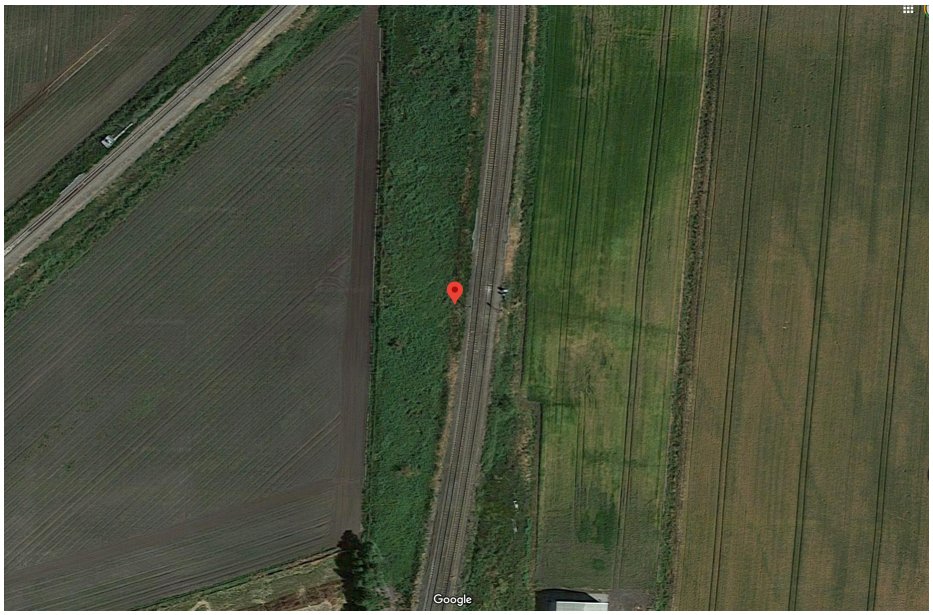


Figure 3.17: The final output coordinate of the $3D$ reconstruction, projected in google maps.

The procedure made in Section 3.5 and Section 3.7, can be seen at Figure 3.18. In the Section 3.8, the shame pattern followed, this time for all the detected objects.

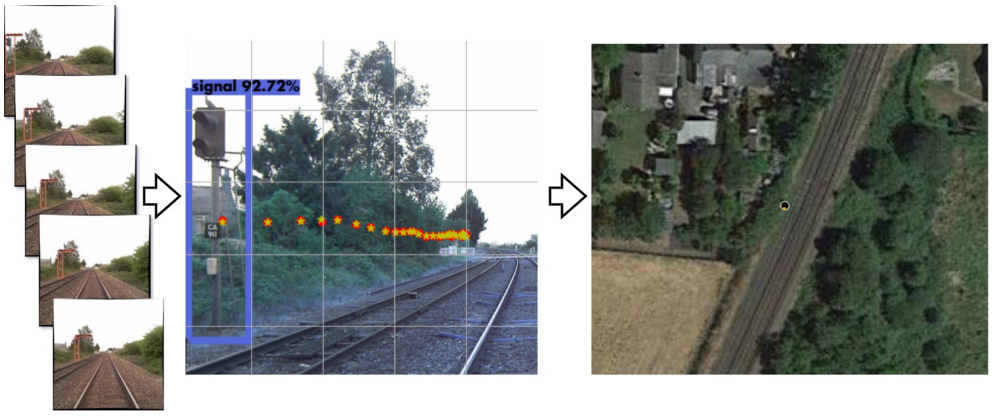


Figure 3.18: The procedure followed to map a single infrastructural object into world coordinate system. Sequence of undistored and georeferenced $2D$ video frames including the same detected light signal (left); triangulation/$3D$ reconstruction of the central coordinates of the bounding boxes (center); final $3D$ projection into world coordinate system - GIS map (right).

3.8 MAPPING – AUTOMATED OBJECT INVENTORY

The object inventory was crucial for the automation of the mapping of the detected infrastructural objects. The procedure of the $3D$ reconstruction of the position of one infrastructural object was illustrated in Section 3.7. There, the choice of the frames

that will be involved in the triangulation was made manually, knowing the presence of that specific object in these 2D frames. In this section, the automation of this procedure is illustrated. Automation in the sense that all the available video frames are involving in the creation of Ely's GIS map and hence, the 3D reconstruction of all detected objects is considered. The selection of the 2D frames that include the bounding boxes referring to the same object, made by using conditional statements (see Algorithm 3.2).

Algorithm 3.2: INVENTORY ($\mathcal{DF}, \mathcal{CSV}, \mathcal{W}$)

Input: List of csv files $\mathcal{CSV}$, the size of the frame-window $\mathcal{W}$
Output: Dataframe with all coordinates of every object $\mathcal{DF}$

```
for i ← csv_first to csv_last do
    previous_frame ← csv_frame_number
    pixel_threshold ← 120pixels
    threshold_step ← 15pixels
    j ← i + 1
    frame_count ← 0
    while frame_count < W & j < length_csv do
        current_frame ← csv_frame(j)
        if previous_frame ≠ current_frame then
            pixel_threshold ← pixel_threshold + threshold_step
            frame_count ← frame_count + current_frame − previous_frame
            previous_frame ← current_frame
        if csv_previous_class = csv_current_class then
            if csv_previous_coordinates − C_current_coordinates <
              pixel_threshold then
            DF ← append_csv
        j = j + 1
```

First, the selection of the 2D frames that will involve in the triangulation was made. Hence, a frame-window of value 4 found that returned the position of the majority of the infrastructural objects that are present in the Ely's railway area. The algorithm counts as individual objects all the classes that are present in the 1st 2D frame and searches if they appeared also in the next 3 2D frames based on more conditional statements. The second selection was based on a pixel-wise window. Hence, the algorithm is checking if the central coordinates of the same class object of the next 2D frames appeared in a certain pixel-window. It is worth to mention here that the pixel-wise window deviates based on the combination of the 2D frames. Hence, the area that the algorithm was checking to find a central coordinate, differ between the $1st$ - $2nd$ and the $1st$- $4th$ frame due to the motion of the train.

Furthermore, all the possible combinations of pairs of 2D frames were made. To remove the outliers of hundreds of triangulations referring to the same object, more conditional statements were used. First, a removal made to the results that their z value referring to altitude, was outside of the range of -2 and 50000 [m]. These numbers define the lowest and the highest elevation of railways worldwide [71], so, anything below or above these numbers would be an outlier. In addition, Z score formula (see Equation 3.3) was used for the second phase of removing outliers. The triangulations kept, were inside the range of 95% of the std of the position.

After the outlier removal, the median coordinates of all directions in every individual object kept and created the object inventory.

$$z_i = \frac{x_i - \bar{x}}{s} \tag{3.3}$$

Where $x_i, \bar{x}$ the individual triangulations and the sample mean respectively, while, S and z_i, the std and the Z score of the samples respectively.

3.9 POINT CLOUD CLASSIFICATION

The $3D$ point cloud analysis was made to refine the vertical position of the detected infrastructural objects. Although the $2D$ image analysis returned accurate results in horizontal plane (see Section 3.7), the vertical estimations were inaccurate (Chapter 3). The inaccuracy of the estimations in the vertical plane in $2D$ analysis, was a consequence of the non precisely created heights of the bounding boxes (see Figure 4.2). Hence, involving the centimeter accurate $3D$ point cloud in positional mapping of the infrastructural objects, increased the confidence and the accuracy of the results in the vertical plane and as a consequence, returned valuable height estimations of the detected objects.

This section illustrates the techniques and methodology used for the classification of the $3D$ point cloud. Instead of performing ground filtering (Section 3.9.1) and unsupervised classification algorithms (Section 3.9.2) to massive $3D$ point cloud data, it was decided to sub-sample the point cloud into $3D$ volumes-voxels. Hence, based on the estimated position of the infrastructural objects (Section 3.8), point cloud was cropped in voxels that include the area of the mapped objects. The length and width of the voxels decided to be $4x4[m]$ so to have a higher confidence that the voxels include the infrastructural objects. Having the object class information of the estimated positions and the knowledge of the height of the objects, the height of the voxels chose to be 4.5 and $3[m]$ referring to lights signals and equipment boxes respectively.

The major reason we cropped the $3D$ point cloud data into voxels, was to extract the height information of the mapped objects. To do this, it was needed to classify the points that refer to objects so to calculate their height. Their height was estimated by the absolute difference between the higher and lower elevation of the classified points as *object*. To be able to classify the points that refer to the mapped objects, ground filtering algorithms were performed (Section 3.9.1). After the voxels classified as ground and non-ground points, unsupervised classification performed into the non-ground points, to further distinguish the non-ground points as *object* and *other*. In this way we were able to refine the height of the mapped objects. There were also minor reasons that we analysed voxels instead of all the data. The first reason is that we are able to use the classified points as *object*, as training data for supervised classification. In this way, a further refinement can be done on the already classified $3D$ point cloud data of Fugro. Second, having only a small part of the $3D$ point cloud, ground filtering performe better in distinguishing the ground and non-ground points (Section 3.9.1), due to the smaller elevation variations of small $3D$ point cloud voxels. Lastly, having only the non-ground points of the $3D$ voxels allow us to use object-based classification techniques, based on the geometric properties of the objects (Section 3.9.2). The $3D$ point cloud analysis in a nutshell can be seen in Figure 3.19.

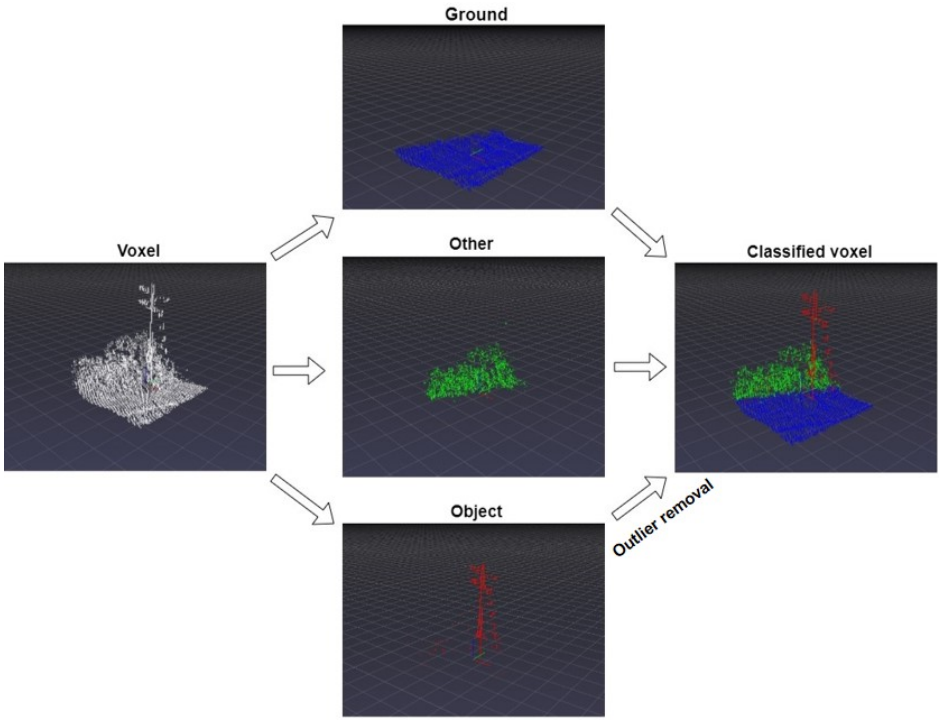


Figure 3.19: Followed steps for voxel classification. The 3*D* point cloud were cropped automatically into voxels based on the estimated positions. Then, ground filtering performed on voxels. Based on the principal component analysis (PCA) on non-ground points, further classification was made creating the classes *other* and *object*.

3.9.1 Ground Filtering - Digital Surface Model (DSM) of the Voxels

The ground filtering algorithms were performed on the created voxels. Having filtered the ground points, further improve the performance of the unsupervised classification such as the PCA. Hence, by better classifying the points that refer to the infrastructural objects, we more accurately estimated the height of them.

The main reason that filtering the ground points improved the PCA, is that it increased the discriminative power, of created geometric features (Section 2.4.1). In more detail, having only the DSM of the points, geometric features like the *planarity*, can be used to distinguish the vegetation from other more planar surfaces. The terrain next to the railroad is flat (planar) hence, differentiating the ground from the facade of an equipment box using the mentioned feature, is not trivial.

Initially, an iterative method was used for the ground filtering using the properties of a triangulated irregular network TIN, specifically the **Delaunay Triangulation**. The method is based on the greedy insertion of ground points into a TIN [72]. In more detail, this method creates a TIN that includes the lower points in every grid cell. After the creation of a TIN, the distance and the maximum angle of the points outside the TIN, from the created triangles, are checked. If the conditional statement is met, the algorithm re-creates the TIN including the checked point. The resulted TIN, is the DSM of a 3*D* point cloud voxel. The pseudo-code of the method (Algorithm 7.1) as well as the flowchart (Figure 7.7) can be found in the appendix Section 7.1.2.

Due to the complexity of the above method and because it is time-consuming, the alternative algorithm of Cloth Simulation Filter (CSF) [14] was used for the ground filtering. This technique was faster and performed better compared to the previous algorithm, mainly due to its simplicity. In a nutshell, the algorithm flips a terrain upside down (see Figure 3.20) and works as if a virtual cloth is placed on it. The shape of this virtual cloth presents the digital terrain modeling (DTM) of the points

while the remaining points are considered as the DSM. The steps that the algorithm follows are illustrated in the flowchart (Figure 3.21).

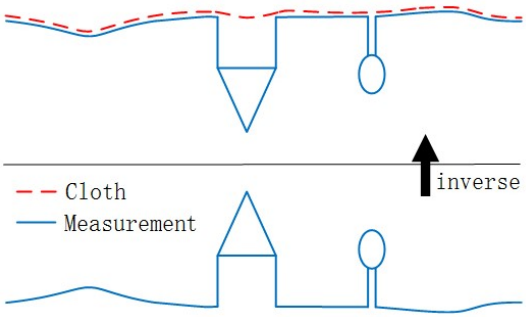


Figure 3.20: CSF mechanism for creating DSM (Figure taken from [14]).

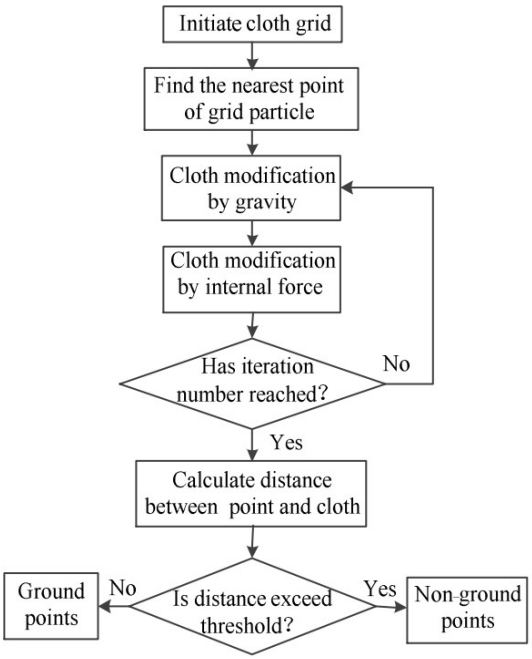


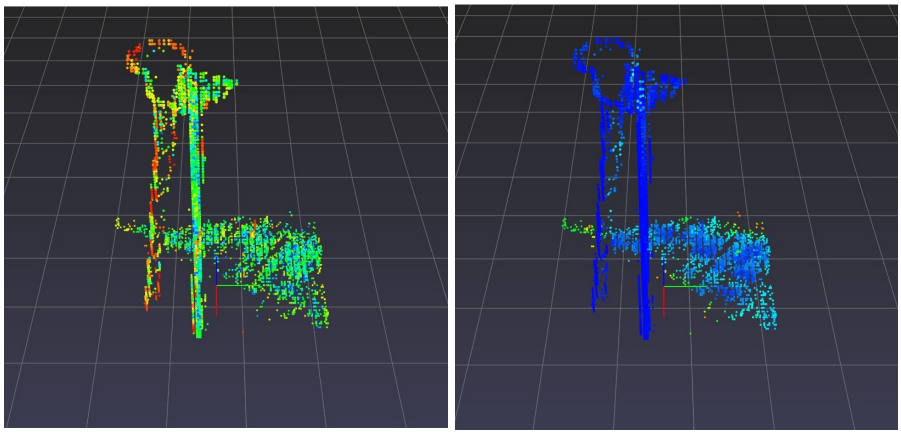
Figure 3.21: CSF flowchart (Flowchart taken from [14]).

3.9.2 Geometric Feature Analysis

PCA was used in 3*D* point cloud analysis to further classify the non-ground points (DSM) into the 2 new classes - *object* and *other*. The points that classified as *object*, were used to calculate the actual height of the detected objects.

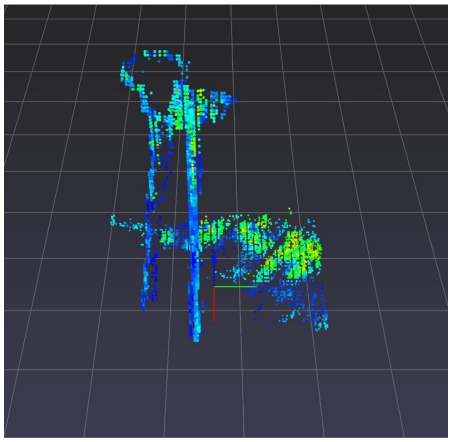
In more detail, having classified all the voxels as ground and non-ground points, we performed PCA on the non-ground points (DSM). At first, *Nearest Neighbors* performed to extract local geometrical features based on the structure of the 3*D* point cloud [73]. The use of k-dimensional tree (k-d tree) was used for the organization of the points in such way to accelerate the search of the *Nearest Neighbors* algorithm. Using the created covariance matrix, we extracted the 3 eigenvalues with the higher values and by combine them [12], the 8 geometric features Table 2.2 discussed in Section 2.4.1, were created.

To choose the geometric feature(s) with the most discriminative power, visual inspection Figure 3.22 and PCA Figure 3.23 were used.

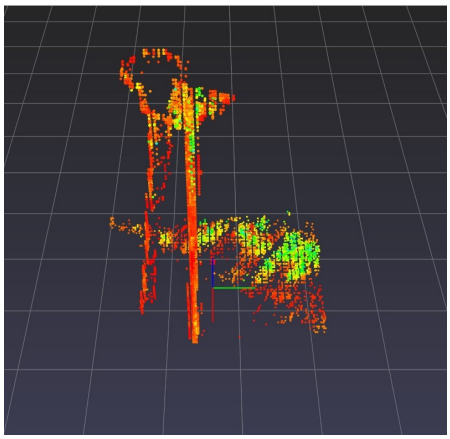


(a) Linearity

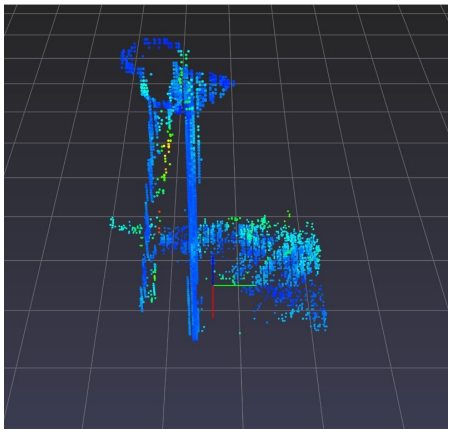
(b) Omnivariance



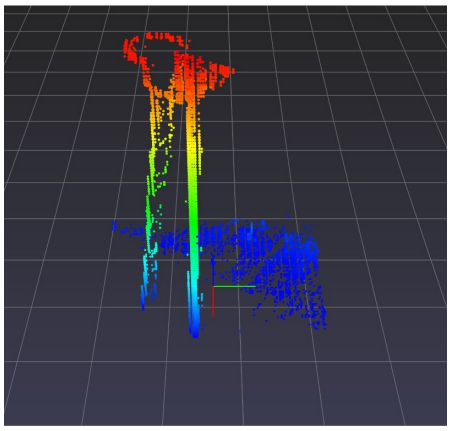
(c) Change of Curvature



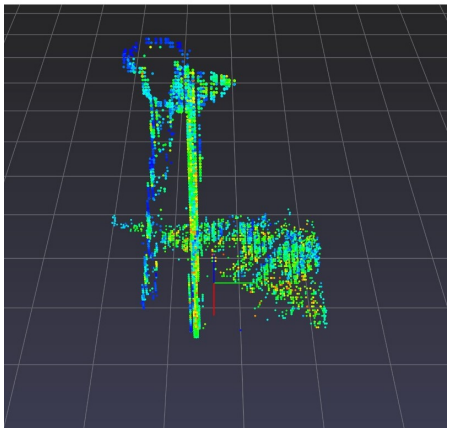
(d) Anisotropy



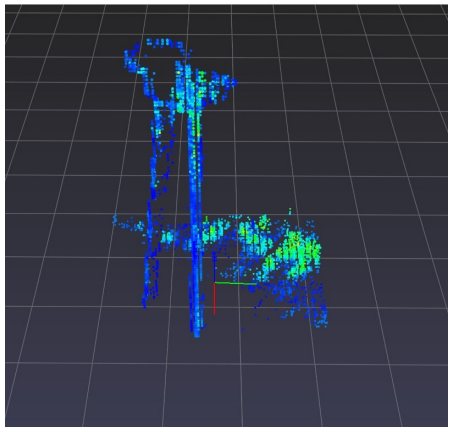
(e) Elevation difference



(f) Mean Elevation



(g) Planarity



(h) Scattering

Figure 3.22: 8 geometric features of a typical infrastructural signal. The color range follow the same order as the visual color spectrum. *Bluish* ≈ 0, *Reddish* ≈ 1.

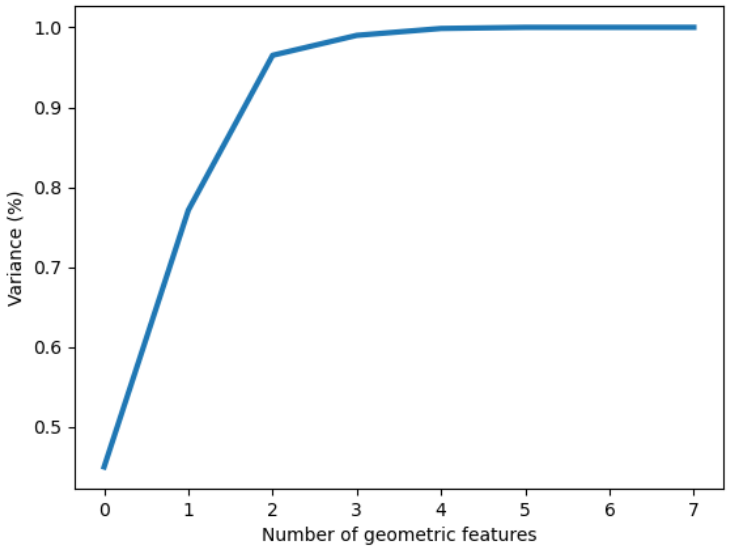


Figure 3.23: PCA analysis for feature reduction. One feature can approximately reach the 80% of the discriminative power.

From the Figure 3.23 it can be seen that the use of 1 geometric feature, returned approximately the 80% of the variance. To define the geometric feature that has the higher variance, visual inspection was used. Furthermore, looking the Figure 3.22, it can be concluded that the geometric feature *omnivariance* which is an indicator of the local density of the points, returned the higher discriminative power compared to the other features (see also Figure 3.24). Hence, using this feature we were able to distinguish the non-ground points as class *object* and class *other* (see Figure 3.28).

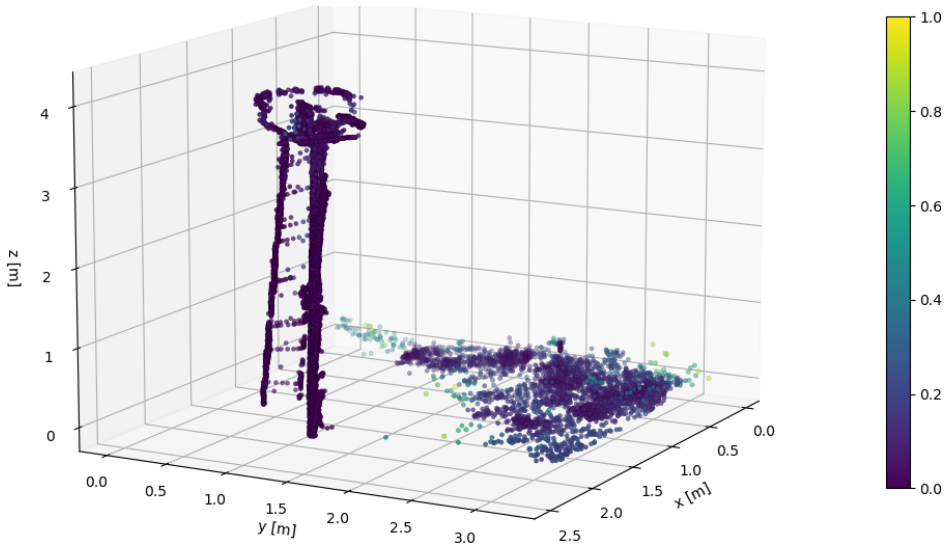


Figure 3.24: Discriminative power of the geometric feature *omnivariance*. The vegetation has higher values, while the infrastructure ("object" and "other") have relatively low.

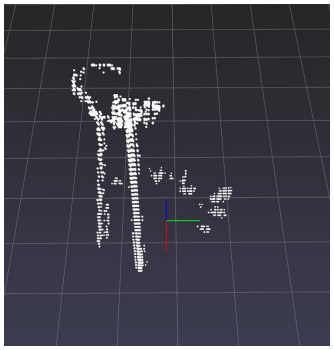


Figure 3.25: Class: Object

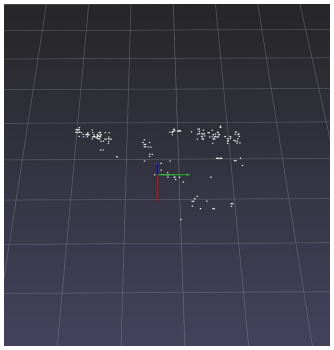


Figure 3.26: Class: Other

Figure 3.27: Signal.

Figure 3.28: Classification of the non-ground points of a signal based on the geometric feature *omnivariance*.

The same procedure followed for the classification of the box's non-ground points into *object* and *other*, while in that case, the geometric feature *scattering* was used (from appendix Section 7.1.1 see Figure 7.2 and Figure 7.6). The reason that *scattering* returned the higher discriminative power among all the geometric features, is the nature of the silhouette of the equipment box which has flatter surfaces compared to signals. Hence, the surroundings (vegetation, soil, other), have a different pattern compared to the relatively flat surfaces of a box (see Figure 3.29).

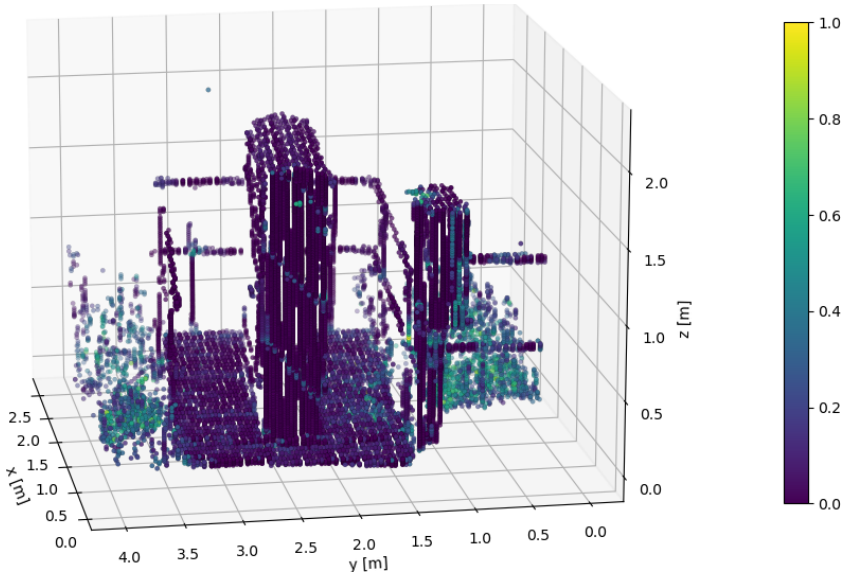


Figure 3.29: Discriminative power of the geometric feature *scattering*. The vegetation has higher values, while the infrastructure ("object" and "other") have relatively low.

After the points that refer to the infrastructural objects were classified, we used this information to adjust correctly the z coordinate of the detected objects and as a consequence to calculate the actual height of them (see Section 4.3).

3.9.3 Outlier Removal

To accurately calculate the height of the detected objects, we referred to the classified point cloud data (Section 3.9.2). Hence, a precise classification of the points referring an infrastructural railway object was crucial, while every non-well classified 3*D* point cloud led to errors. Figure 3.31 shows that although PCA performed into

the voxel's non-ground points (DSM), the classification results were imperfect and outliers were present.

Furthermore, to overcome this, the creation of a triangulated irregular network (TIN) was created with the use of Delaunay triangulation. In more detail, a TIN was created based on the points classified as *object*. First, a threshold value was used as an indicator of which edges were sufficiently long. This threshold was defined as the mean value of edges. Considering the length of all the edges of the triangles, vertices were considered as outliers and removed, when they were linked to more than two long edges (Figure 3.32). Although this technique sufficiently performed, the points consisting the class *object*, continue to have outliers (see Section 5.4).

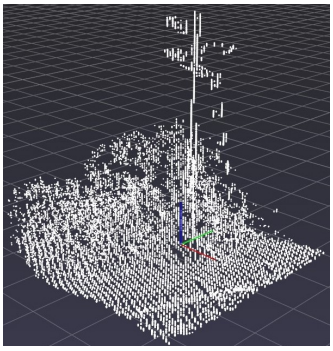


Figure 3.30: Voxel.

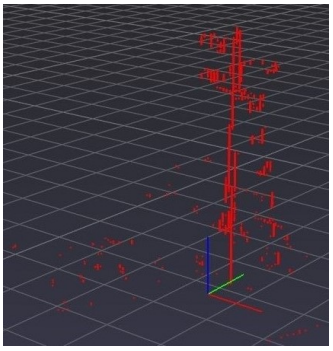


Figure 3.31: Class: Object.

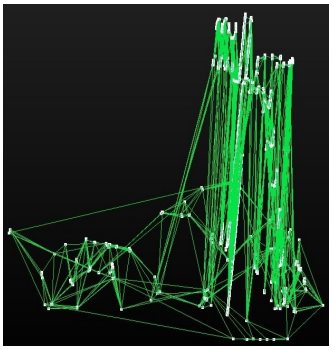


Figure 3.32: TIN *object*.

Figure 3.33: TIN of the class: *object* created Figure 3.32, to remove the outliers. Vertices whose triangle's edges were long are considered as outliers.

3.10 WORKFLOW AUTOMATION – SOFTWARE PACKAGE

A big part of this thesis was the automation of the developed workflow into a software tool. For the software to become reproducible, shareable and open to changes (e.g. include more object classes), many steps were followed. This steps including unit testing of the methods of the software's modules and packaging it so to be able to run in different environments. To further assist the reproducibility and ability of the developed tool to be shared, we have compiled it as well as its necessary dependencies into a software container. This step will elevate potential barriers of adoption and assist users for its further expansion.

4 RESULTS

This chapter presents the outcomes of this research. Section 4.1 starts with showing the results of the image analysis, ending at the 1_{st} outcome - the creation of a geographic information systems (GIS) map of the detected infrastructural objects. In Section 4.2 the results of the analysis of the point clouds around each detected object are presented - the contribution of the point cloud analysis on the refinement of the estimation of height and the vertical position of the detected objects. Section 4.4, illustrates the main sources of the errors of the positional results.

4.1 DETECTION PERFORMANCE

To evaluate the performance of 2*D* detection, a sequence of 245 raw 2*D* video frames capturing part of the railroad near Ely (United Kingdom) were used. From these 245 images the two objects introduced in Section 3.2 - equipment boxes and light signals - were detected using the method based on you only look once (YOLO), as described in Section 2.2.1 and Section 3.5. Looking at the results (see table Table 4.1), the sensitivity/recall metrics in both classes is above 60%. Where sensitivity/recall, is a metric that shows to what percentage the 2*D* detection detects an object correctly (see Figure 2.7). The performance of 2*D* detection and the Table 4.1 were made manually by visually inspecting the objects comparison to all input frames.

From this test dataset we conclude that the 2*D* detector mistakenly counted as signal one different object (false positive), while in the class of *box* the 2*D* detector performed precisely in all 2*D* frames.

	Box	Signal
True Positives - TP	68	63
False Positives - FP	0	1
False Negatives - FN	43	37
Precision - TP / (TP + FP)	1	0.98
Sensitivity/Recall - TP / (TP + FN)	0.61	0.63
Mean AV. Prec - mAP		0.99

Table 4.1: 2*D* detector's performance based on convolutional neural networks (CNN) metrics. High recall and precision implies that 2*D* detection method detected correctly ground truth objects.

The created bounding boxes did not always fit well the height borders of the silhouette of the detected objects. Hence, the estimated z coordinates are not reliable. Some extreme examples of the bad performance of the 2*D* detector can be found in the appendix Figure 4.2.

4.2 3D RECONSTRUCTION

This section illustrates the first outcome of the workflow - the creation of a GIS map. As mentioned in Section 3.8 and Section 3.7, first the object inventory was made and then the coordinates of every individual detected object were retrieved. The

reconstructed coordinates of the objects detected inside the 2D video frames were compared with ground truth locations. The output is a GIS map with the position and class information of the detected objects.

Figure 4.1 illustrates the creation of a GIS map that covers 2.2 Km railroad in the area of Ely. The majority of the light signals and equipment boxes were mapped correctly, especially in the low density areas, while in areas with high density of objects, the performance of both the 2D detection and 3D reconstruction was lower. The RMSE of the x,y difference between the estimated and ground truth position of objects was approximately 30cm.

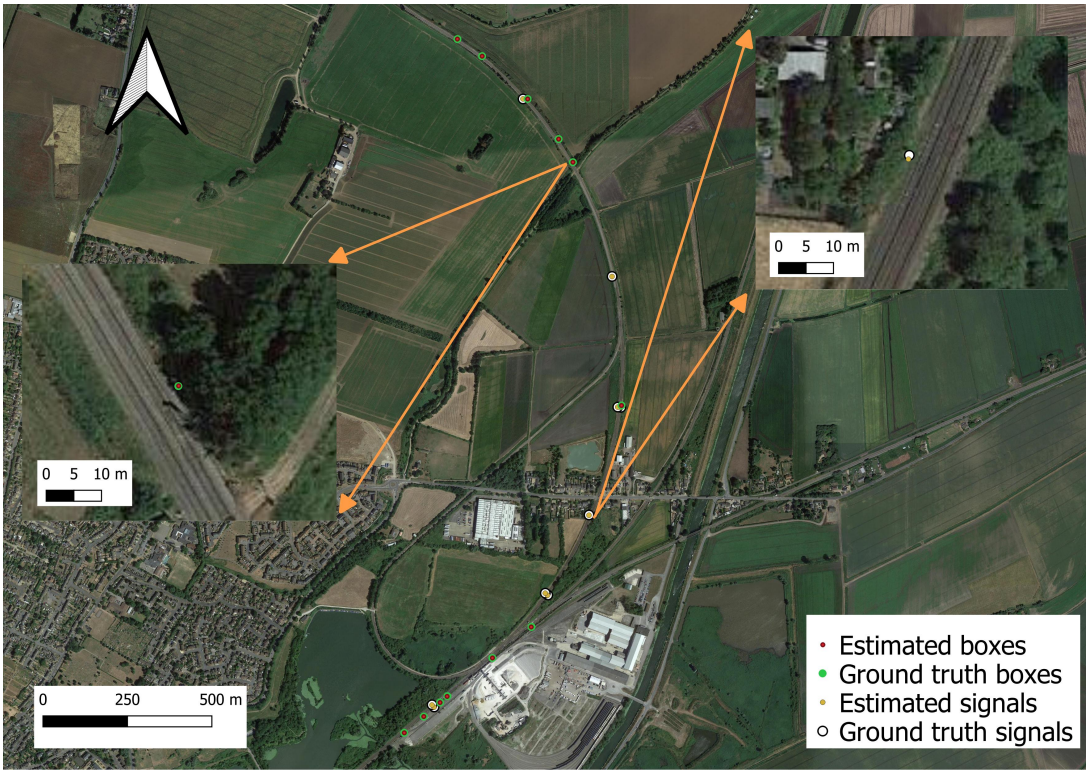
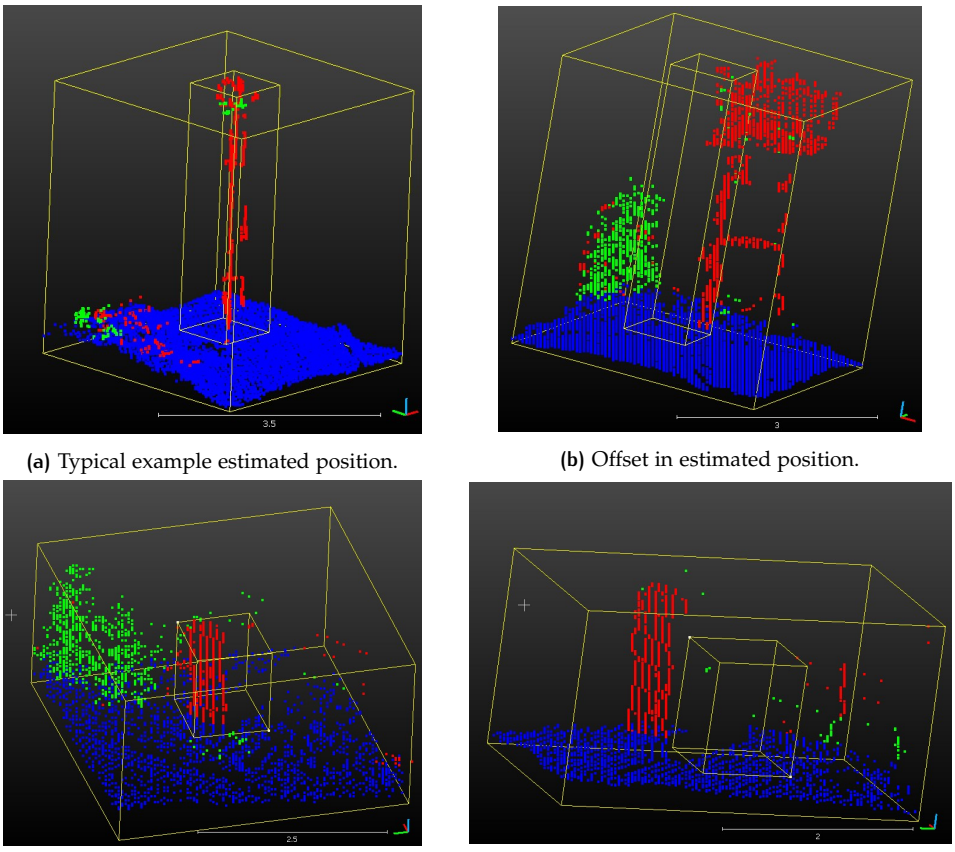


Figure 4.1: The main result of the 2D analysis - a GIS map of 2.2km railroad infrastructure near Ely, United Kingdom. The map illustrates both the positions of the mapped infrastructural objects and the ground truth. The figure contains 12 correctly detected equipment boxes and 9 correctly detected light signals, presenting the 50% and 43% of the ground truth respectively.

For the creation of Ely's map (Figure 4.1), 2744 2D frames taken by the central camera were processed, 471 of them contained at least one of the two considered objects. For this piece of railroad, the numbers of ground truth boxes and signals were 24 and 21 respectively, while the 2D detector identified 17 and 12 respectively. This led to a value (see Table 2.1) of approximately 70% for the class box and 57% for the class signal. Through the inventory process, 12 and 9 boxes and signals were mapped, loosing 29% of the detected boxes and 25% of the detected signals. Hence, through 2D analysis we were able to map 50% of the railway light signals and approximately 43% of the equipment boxes, with an acceptable RMSE in the horizontal plane of $\approx$ 30 cm against the ground truth. Furthermore, while approximately the 75% of the 2D analysis returned reliable positional results in the horizontal plane, there were cases that the estimated positions exceeded the acceptable error and they were not mapped. As it mentioned above, in areas with high density of infrastructural objects, the estimations were less accurate. When the estimations exceeded the 1m difference from ground truth were not mapped. Details are discussed in the limitations chapter in Section 5.3.

To estimate the height of the detected objects, the z value of reconstructed 3*D* position was used as the center of the mass. While 2*D* detector predicts the best fitting bounding box that contains the whole object, the created 2*D* bounding boxes did not always fit correctly the detected objects (see Figure 4.2), hence, the estimated heights were inaccurate. To improve this, point clouds were used (Section 4.3).

Figure 4.2: Not sufficient performance of the 2*D* detector regarding the height borders of the bounding boxes.



(a) Typical example estimated position.

(b) Offset in estimated position.

(c) Typical example of estimated position.

(d) Offset in estimated position.

Figure 4.3: The left figures is a good approximation of the position of the object (interior bounding boxes) while the right figures illustrate an offset in the horizontal plane. The well georeferenced classified point cloud voxels (red: "object", blue: "ground", green: "other"), are used to illustrate the quality of the 3*D* reconstructed estimated position made by 2*D* analysis.

Figure 4.3, illustrates typical positional examples as well as some extremes, in the 3*D* reconstructed position estimated via the 2*D* analysis. The bounding boxes that refer to the infrastructural objects (interior bounding boxes), were obtained

based on the estimated position. Their length is $1m$ while their height borders in the z direction are ± 2.2 and $\pm 0.8[m]$ from the estimated position, for the signals and the boxes respectively. Furthermore, Figure 4.3d reveals also the limitation in approximating the height of the box, as well as the potential of using the classified point clouds to refine it.

Figure 4.4: Cross road in the railway near Ely, UK. The illustrated equipment boxes are different than the boxes used for training. As a consequence, $2D$ detector was not able to capture them.

Lastly, due to the lack of training data from light signals that are present in the crossroads (Figure 4.4), YOLO could not perform well. In more detail, crossroad light signals and equipment boxes, differ from the conventional railroad signals and boxes. In addition, there were conventional differences of what it was counted as equipment box (Figure 4.4) from our labeling and the British rail. It is worth to mentioned that in this part of the railroad, 2 crossroad are present. Hence, the numbers that assess the performance of the $2D$ detection (detectors recall $\approx$ 60% and mapped objects $\approx$ 50%), do not precisely representing the $2D$ detection's performance. Hence, it can be concluded that the $2D$ detection can actually map more than the 50% of the light signals and equipment boxes that are present next to the railway.

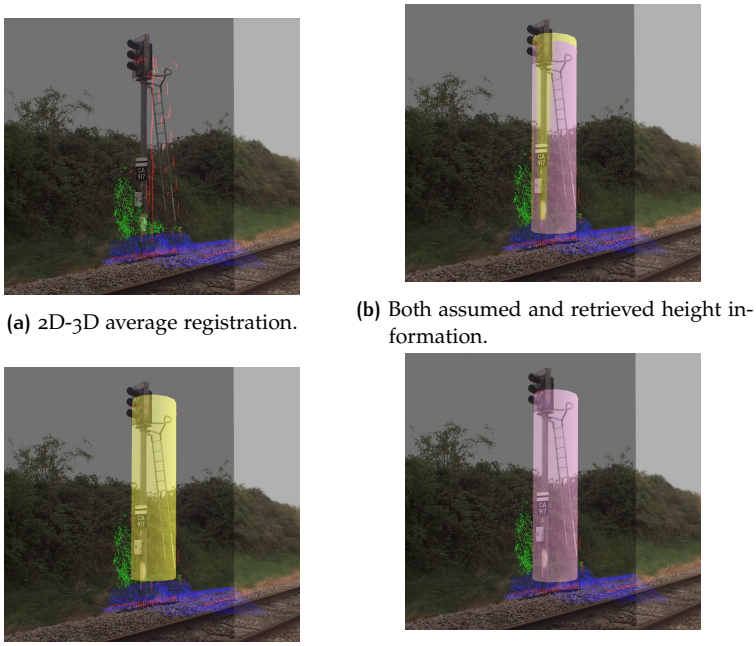
4.3 POINT CLOUD ANALYSIS

As it mentioned in Section 3.9.2 and Section 4.2, the point cloud data were analysed in order to aid the calculation of the vertical position of results retrieved via $2D$ analysis. Hence, considering the lower and higher elevation point of the points classified as *object*, the actual length of the detected objects were estimated.

	Raw height [m]	Outlier removal height [m]	Difference [m]
box01	0.425	0.184	0.241
box02	2.585	2.585	0
box03	2.694	2.694	0
box05	2.06	2.026	0.034
box06	2.416	2.409	0.007
box07	0.648	0.646	0.002
box08	2.196	2.196	0
box09	2.016	2.016	0
box10	2.517	2.02	0.497
box11	2.808	2.808	0
signal00	4.122	4.112	0.01
signal01	0.034	0.031	0.003
signal02	4.087	4.061	0.026
signal03	4.116	4.1	0.016
signal04	4.007	4.007	0
signal05	4.137	4.137	0
signal06	4.112	4.109	0.003
signal08	4.095	4.095	0

Table 4.2: Estimated object's height based on the lower and higher elevation points of the classified points *object*, before and after outlier removal. The median height of the boxes and signals is $\approx$ 2.2 and $\approx$ 4.1 [m] respectively.

After the outlier removal performed as discussed in Section 3.9.3, the height results of the objects were compared. From the Table 4.2, it can be seen that in some cases (e.g. box01, box07 and signal01), the results were not rational. This can be explained by the low point cloud density of these specific voxels and hence, the objective difficulty in defining the height of these object. Another outcome of the table is that after the outlier removal, the heights of the boxes differ on average $\approx 15cm$ and $\approx 1cm$ on signals. Hence, it can be conclude that the height refinement that we wanted when performed outlier removal was insufficient (see Section 5.4).



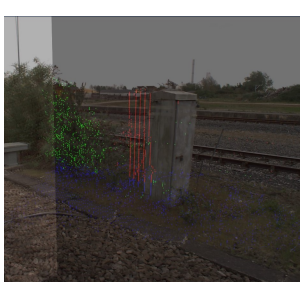
(a) 2D-3D average registration.

(b) Both assumed and retrieved height information.

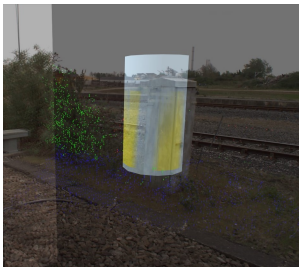
(c) Assumed height.

(d) Height based on the object class points.

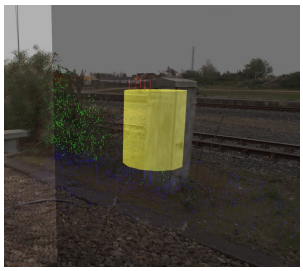
Figure 4.5: Typical example of assumed and retrieved height information of an infrastructural signal.



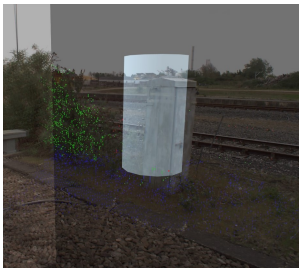
(a) 2D-3D average registration



(b) Both assumed and retrieved height information



(c) Assumed height



(d) Height based on the object class points

Figure 4.6: Typical example of assumed and retrieved height information of an infrastructural box.

Part of the visulation of the results of the point cloud analysis were generated in Fugro's *OnePortal* (see Figure 4.5 and Figure 4.6). From the Figure 4.5 and Figure 4.6d, it can be seen the extra vertical information that the point cloud analysis provided, allowed us to estimate the height of the detected objects instead of assuming it (see Figure 4.3). It is clear that the vertical boundaries of the objects are in-line with the vertical boundaries of the object class (red), where in the Figure 4.5c and Figure 4.6c, the vertical boundaries assumed based on the estimated position. It can be seen in both figures that the offset of the estimated vertical position triggered by the inaccurate classification of the voxels. More detail of the limitations of the point cloud analysis will be follow in the Chapter 5.

4.3.1 Execution time

	Time in [min]
Labeling	960
Training	1320
Cropping & undisotrting 100 frames	8
Object detection in 100 2D frames	1.1
Triangulatting 100 2D frames	0.0
Extracting 22 3D voxels	0.4
Ground filtering of 22 3D voxels via CSF	0.2
Outlier removal of 22 3D voxels	0.7
Feature extraction of 22 DSM	4.1

Table 4.3: The table illustrates the time consumption of the core steps followed in this workflow.

This sub-section illustrates through Table 4.3, the time needed to execute the main steps that followed on this workflow. By far, training and labeling were the most time-consuming processes that took place only once at the begging of the thesis. It is worth to mention that 100 $2D$ frames were used for the calculation of the execution time of cropping, undistorting and object detection processes, because of the limitation in the usage of random access memory (RAM) $\approx 500MB$ in my local machine. The reason that 22 voxels were used to calculate the time in the point cloud analysis was that 22 objects were mapped in the Ely's railway area.

4.4 ERROR BUDGET

This section illustrates the main error sources that contribute to positional offsets from ground truth. The main countable source of error is the accuracy of the GNSS and as a consequence the quality of the georeference of the $2D$ image frames and $3D$ LiDAR point cloud. The creation of the bounding boxes from the $2D$ detector (see Figure 4.2), and the implementation of the methods triangulation (Section 3.7), extrinsic camera calibration (Section 3.6), influence the quality of the measurements as well.

The uncertainties from the implementation of the methods are triggered due to decimal accuracy. For instance, when we triangulate intersected vectors to $3D$ reconstruct a $2D$ position, small variations in the vectors - low decimal accuracy pixels - can lead to remarkable errors when we project the $3D$ estimated position to world coordinates that have higher decimal accuracy. These relatively small variations in the intersected vectors are mainly caused by the imperfect creation of $2D$ bounding boxes. The center of the bounding box is not being representative of the same point in space in other $2D$ video frames in which the object is detected. This 'pixel error' will have results on the triangulated point, especially when the intersection angles are sharp (see higher error in "South-North" direction in Figure 3.16). Hence, a slight shift in vector due to this pixel error can have significant influence on the triangulated point.

To maximize the estimated positional accuracy, Fugro uses a post-processing integrated solution between inertia measurement unit (IMU) and dual frequency DGNSS [74]. After smoothing GNSS estimations of 4 runs and considering $3D$ points of the center-line of the tracks, Fugro states that using a track-distance correction in ideal conditions, can lead to LiDAR accuracy of $8mm$ and $12mm$, in horizontal and vertical plane respectively.

For this thesis, the $2D$ video frames were georeferenced considering one run of the RILA, while the $3D$ point cloud is a merged product of 2 runs. Fugro's RILA specification affirm that under ideal conditions, this can lead to a 2 to $3cm$ accuracy.

The following parameters influence the quality of the positional results.

- GNSS $\rightarrow$ DGNSS.
    1. In urban/mountainous areas or over dense tree canopies, we experience the "urban canyon" phenomenon where there might be smaller number of visible satellites leading to less accurate positioning. In addition, the introduction of multipath effect influence the measurement [32].
    2. Baseline distance. The accuracy of the position depends on the distance between the RILA's GNSS antenna and a reference station [29].
    3. Train velocity influences the accuracy of the position results the GNSS results [32].
    4. Position ilution of precision (PDOP) [75]. Depending on various parameters, we might experience unfavorable satellite geometry that leads to less accurate positioning.

- LiDAR → $RILA4.0$.
  1. Depending on the density of the point cloud. Merging $3D$ point cloud data from multiple scan measurements (runs), increase the density of the point cloud and hence, decrease the uncertainty.
  2. Material nature of the target object by the laser beam (Section 2.1.1). Reflective materials or even non-reflective materials after a rain (equipment boxes), can significantly degrade the measurement quality [76].
  3. Distance from the LiDAR. The uncertainty increases when the distance between the object and the LiDAR increases [77].

5 LIMITATIONS

This chapter illustrates the main limitations of the methodology in every step of the procedure both in 2*D* and 3*D* analysis.

5.1 LABELING – TRAINING

Labeling and training the data is a time-consuming process. In Section 3.3.1, it was mentioned that for the training of the classifier, 995 2*D* frames were labeled. As it was claimed by the developers of the convolutional neural network CNN you only look once YOLO [65], a desirable training dataset should have 2000 labeled images from every class. Although in our case the labeled images usually include more than one object class in every raw frame (see Figure 3.5), the training dataset was below the proposed number. The reason that no more 2*D* frames were labeled was that the manual procedure of labeling consumed time. It is worth to be mentioned here that the labeling cost 16 hours of work (see Table 4.3).

Training the 2*D* detector is a slow procedure as well. Using a small and homogeneous number of input data and reaching sufficient values of the mean Average Precision (mAP) and loss function, the training procedure costed 22 hours in a local machine (see Table 4.3). At the beginning, *Google Colab* was used for the training of the creation of the weights because of the benefit of the provided computational power in its services. Because of the 12 hours limitation that *Google Colab* provides, the training procedure chosen to be in a local machine.

5.2 2*D* DETECTOR'S PERFORMANCE

The object detection can never reach the ideal mAP of 100% and cannot rid of false positives (see Figure 5.1). In addition, as it mentioned in Section 4.1, the detectors localization is imperfect, meaning that the creation of the bounding boxes using the non-maximum suppression is defective (see Figure 4.2).

Figure 5.1: False positive (FP) detection of the 2*D* detector - Confusing a speed limit sign as signal. This lowered the precision of the 2*D* detector (see Table 4.1). Although the sign appeared in multiple 2*D* images/frames, only in one frame the 2*D* detector captured it as a signal.

5.3 OBJECT INVENTORY

As it was mentioned in Section 3.8, the inventory of the detected objects was a crucial and complicated step, involving multiple conditional statements on the back-end development of the software. These conditional statements were not working in all scenarios, so, they failed to map all the detected objects found in 2*D* video frames. In addition, in the Section 4.2, it was indicated that through the inventory process, 29% and 25% of the detected signals and boxes respectively, were not able to be mapped. Bellow, the main limitations are illustrated.

- In case an object appeared in a small number of 2*D* frames, the positional confidence is decreasing.
- When an object appeared in small scale (far from the camera), the triangulation results were not reliable. As it was mentioned in Section 4.2, higher positional variations appeared when the intersection angles from triangulation are sharp Figure 5.2.
- In parts of the railroad, especially in the crossroads, the infrastructural objects appeared closer than the other parts. Depending on how dense they are placed, the window of 4 2*D* frames is not sufficient to map them. Depending on various parameters, an amount of the railroad infrastructural objects was not mapped Figure 5.3.
- When the conditional statements failed to ”track” individual objects and miss-count 2 different objects as one, the product of the triangulation was wrong. When 2 or more objects of the same class are placed next or in front to each other in a corner, the model has the potential to fail. Figure 5.4, is an example of how big can become the offset when the model miss-counts the signals as one.

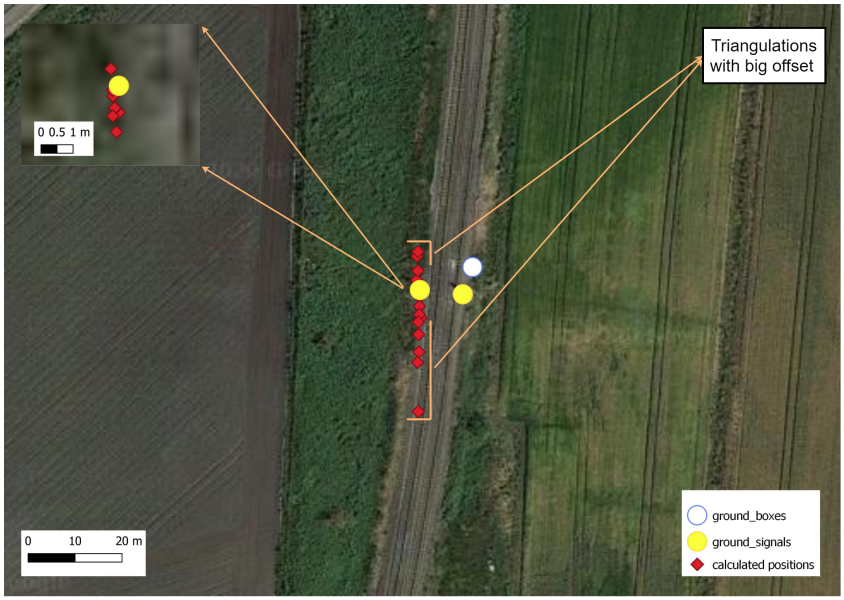


Figure 5.2: The 22 estimated positions of the left signal. The majority of them have 30 cm RMSE. The triangulations that used 2*D* frames including the signal in a small scale have bigger offset.

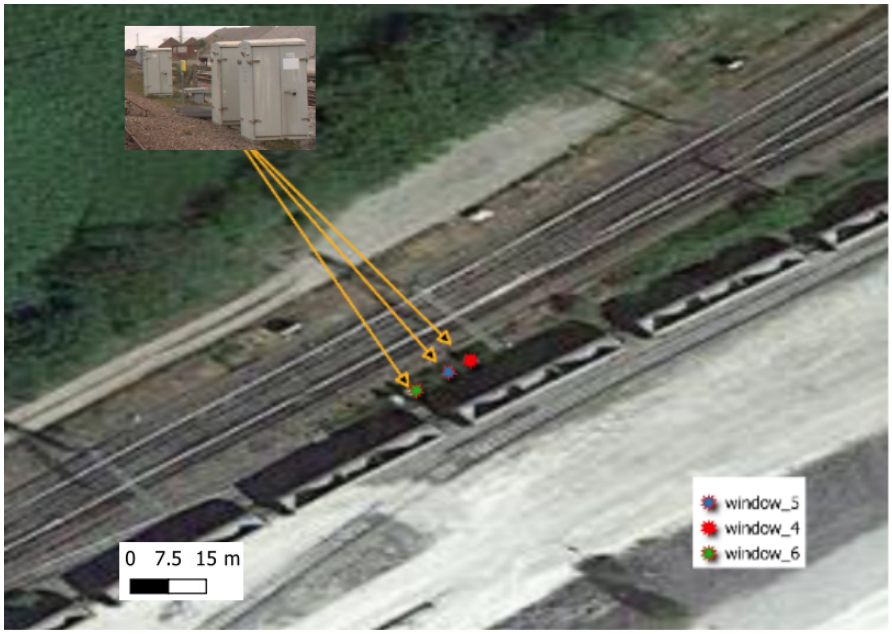


Figure 5.3: Depending on the window frame, different boxes were mapped.

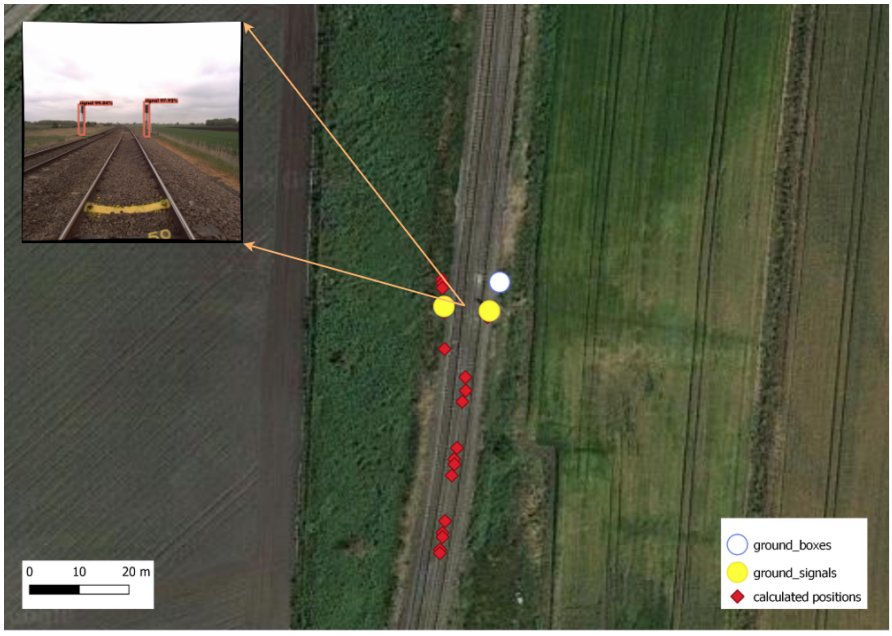


Figure 5.4: Wrong triangulations (red). The model mixed the 2 detected signals as one.

5.4 OUTLIERS

As it mentioned in Section 3.9.3, to accurately calculate the height of the detected objects, satisfactory classification results were needed. Using unsupervised classification (Section 3.9) and ground filtering techniques (Section 3.9.2), does not accurately segment the 3*D* points into ground, object and other points. Although outlier removal was performed to overcome this issue, the results were not good enough Figure 5.5.

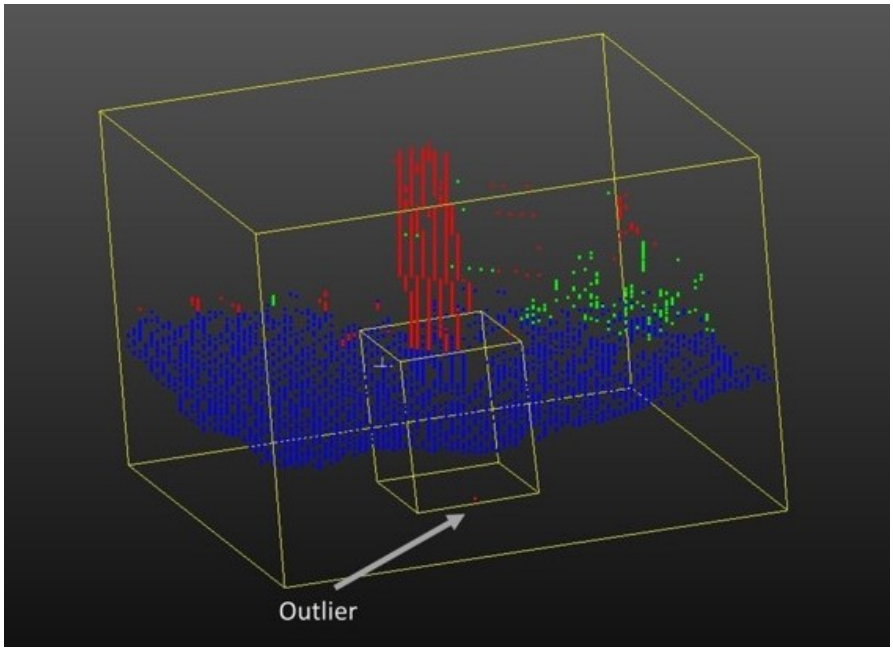


Figure 5.5: Remaining outlier after outlier removal technique performed.

Because of the low density of the 3*D* point cloud, especially the points referring to signals, when removing the vertices linked to long vertices (Figure 3.32), important

information got lost. Hence, the outlier removal algorithm modified to balanced to remove sufficient amount of outliers and to still not lose important information from the 3D point cloud.

6 CONCLUSIONS - RECOMMENDATIONS

Section 6.1, answers the research questions introduced in Section 1.3. In addition, a set of recommendations for future research have been identified and presented in Section 6.2.

6.1 CONCLUSIONS

In summary, 2*D* imagery and 3*D* light detection and ranging LiDAR point cloud needs to be combined to automatically map i. the position in 3*D* world coordinates, ii. the height and iii. the class information of the infrastructural railway objects (i.e. light signals and equipment boxes). The 2*D* analysis provides reliable results in the horizontal plane, while, to refine the vertical position and estimate the height of the objects, the 3*D* analysis is used.

1. **What would be an effective workflow to combine 2*D* and 3*D* data to map the infrastructural railway objects (i.e. light signals and equipment boxes), with an accuracy of a meter?**

   An effective workflow to combine 2*D* and 3*D* data to map the infrastructural railway is mentioned in Section 3.1. The workflow starts with the 2*D* imagery analysis for detecting the infrastructural objects and estimating their position in the horizontal plane. Then, the workflow switches to 3*D* LiDAR analysis to refine the vertical position and estimate the height of the objects.

   For the 2*D* object detection in 2*D* analysis, convolutional neural networks (CNN) were used and approximately 60% of the present infrastructure was detected. Once the pixel information of the central points of the 2*D* bounding boxes of the 2*D* detected objects was retrieved, then, the stereoscopic technique of triangulation was used to reconstruct their 3*D* position in world coordinate system.

   For the inventory procedure, a sophisticated algorithm was created in order to correctly map the detected individual objects. The inventory mapped the position and class information of approximately 50% of the present infrastructural objects, within an accuracy of a meter.

   Having the inventory of the mapped objects, 3*D* point cloud voxels that include points referring to the infrastructural objects, were automatically cropped from the 3*D* LiDAR data. Hence, performing unsupervised machine learning (ML) and sophisticated algorithms to segment and classify 3*D* voxels, we accurately classified the voxels into three classes (Objects, Ground, Other). The classified 3*D* points as "Object" were used to estimate the height and refine the vertical position of the mapped objects.

2. **What are the properties of the input data?**

   The properties of the input data are illustrated expensively in Section 3.2. In short, the input data consist of 2*D* video frames, 3*D* LiDAR point cloud and a .csv file. The 2*D* video frames are mosaics of the three captures of the cameras mounted to Fugro's MMS RILA, with dimensions 4096*x*4096 pixels. Regarding the 3*D* point cloud data, they were acquired via RILA's 360° LiDAR

scanner, having up to 2cm accuracy level in the horizontal and up to 3cm in the vertical plane. Lastly, the .csv file consists of the position, the extrinsic and time information of RILA's IMU system, for every 2D video frame capture, therefore it was used for georeferencing the cameras.

3. **What are the existing techniques for object detection in 2D imagery and what are their pros and cons?**

   Some of the well known existing techniques in object detection in 2D imagery and their advantages and disadvantages are discussed in detail in Section 2.2. In short, there are two main categories of techniques used for object detection in 2D imagery. The ***classical computer vision techniques*** like the temporal/frame differencing, clustering-based techniques and background subtraction, and the ***state-of-the-art deep learning (DL) techniques*** involving CNN.

   The use of CNN has dominated the process of object detection due to its accuracy and ability to be used in real time applications. Another advantage of the use of CNN over the classical object detection techniques is the creation of bounding boxes (i.e. localization of the detected object into 2D pixel space). On the other hand, the classical object detection techniques are better in terms of their independence of the need of the huge amounts of training data and computational power that CNN requires.

   In this thesis project, CNN were used for object detection in 2D imagery because we wanted to localize the objects into 2D space so to use stereoscopic techniques to reconstruct their position in 3D world coordinate space.

4. **How to estimate the 3D position of the detected objects in 2D imagery?**

   The ways to estimate the 3D position of the detected objects in 2D imagery are mentioned in Section 2.3. Advanced computer vision techniques like the creation of the *disparity map/depth map* as well as the oldest stereoscopic technique of *stereo matching*, use the method of triangulation to reconstruct 2D pixels into 3D world coordinate space. All techniques are using georeferenced and calibrated camera(s) and more than one 2D image frame including the same capture from a different view, to triangulate the distances.

   The method of triangulation (Section 3.7) is used to reconstruct the 3D world coordinates of an object. In order to implement the method of triangulation, at least two frames referring to the same object are needed. For this scope, once the central 2D coordinates of the 2D created bounding boxes were retrieved, the method of triangulation was used. Knowing the relative distance of the camera(s) and the difference in pixels of the central point of 2D bounding boxes in pairs of 2D images, we estimate the 3D position of these pixels.

   In this research, to increase the confidence of the estimation of the 3D position of the detected objects in 2D imagery, multiple pairs of triangulated 2D pixels were used when they were available. Therefore, the root mean square error (RMSE) of the estimated positions from the ground truth is approximately 30cm. While this practice is correct, in Section 3.7, Section 4.4 and Section 5.3 it was mentioned that large errors (i.e. small pixel deviations due to the inconsistency of the pixels representing the same point in space in other 2D image - causing big offset) are observed when the objects appeared in small scale (i.e. far away from the camera). This can be explained due to the sharpest intersection angles of the triangulation vectors.

   Lastly, due to the imperfectness in the creation of the 2D bounding boxes (i.e. bounding boxes do not include the object perfectly) obtained from CNN, it is not guaranteed that intersected vectors with less sharp angles return always more accurate results.

5. **What are the requirements of the inspection regarding the accuracy and precision of the position of the signage?**

The inspection accuracy of the mapped infrastructural objects was set at the threshold of $1m$. This was possible to be monitored in the area of Ely because of the given ground truth data. Therefore, estimations exceeding $1m$ from ground truth were not mapped in the inventory (Section 4.2).

To generalize the pipeline in areas where there are no available ground truth data, an iterative outlier filtering algorithm (Algorithm 3.1) was used to improve the precision of the sample (Section 3.7). Therefore, by removing estimations that were labeled as outliers according to the aforementioned algorithm, the variance of the sample decreased, increasing its precision.

6. **To what extent the object detection from $2D$ imagery can be complementary to the $3D$ point cloud classification, and how it might aid in improving the accuracy of the classified point cloud?**

   In Section 3.9 and Section 4.3, it is extensively discussed the reasons of $2D$, $3D$ fusion, and the followed procedure. In short, the resulted object inventory from $2D$ analysis benefited from $3D$ LiDAR point clouds, by automatically estimating the height, and refining the vertical position of the mapped infrastructural objects. On the other hand, the object detection from $2D$ imagery was complementary to the $3D$ point cloud classification, via the automation in the creation of $3D$ training data. Generally the creation of $3D$ training data is a time consuming and expensive procedure. These training data will be used as input to further train Fugro's classifier.

   The automation in extracting $3D$ point cloud voxels that include the points referring to the objects of our interest, aid the classification of $3D$ point cloud. As discussed in Section 3.9, ground filtering techniques perform better in small $3D$ point cloud voxels due to their smaller variations in elevation. Therefore, after extracting the ground points from $3D$ voxels, then the unsupervised classification technique of principal component analysis (PCA) is used. The PCA classifies the non-ground points into two new classes (class: *object*, class: *other*). There are two reasons for the remarkable performance of the PCA classification. The first reason is the sufficient result of the ground filtering. The second reason is that extracting the relatively planar ground, the $3D$ voxels remained with unstructured vegetation (i.e. high scattering properties) and infrastructural objects with planar and omnivariance properties of their geometric features (Section 3.9.2). Therefore, using the positional results from the $2D$ analysis, the discriminative power of the geometric features of the $3D$ objects sufficiently increased, leading to an accurate classification of $3D$ point cloud.

7. **What methods can be used to describe the quality of the results? How can we validate the results?**

   CNN metrics indicating the quality of the $2D$ detector were illustrated in Section 2.2.1. The quality assessment of the training and the detection performance of the $2D$ detector were discussed in Section 3.4 and Section 4.1. The mAP $\approx$ 95% and loss function $\approx$ 0.17 $pixel^2$ were used to describe the quality of the trained weights of the $2D$ detector (see Figure 3.8). The evaluation of the $2D$ detector's performance was made using the mAP $\approx$ 100% and the sensitivity/recall $\approx$ 60% for both classes (see Table 4.1).

   To evaluate the camera calibration results and the forward/backward model, the estimated positions were projected to $2D$ image space (see Figure 3.13). The RMSE $\approx$ 1.575 pixels between the projected estimations and the created by the $2D$ detector bounding boxes, was used to asses the quality of the triangulation method.

   The validation of the $3D$ reconstruction (i.e. positional estimations), was made using the RMSE $\approx 30cm$ against the ground truth (Section 4.2). In addition, the range of the estimated coordinates in horizontal plane was used

to describe the quality of the triangulation (Section 3.7). It was found that depending on the direction of RILA mobile mapping system (MMS), the estimation error showed deviations. More specifically, when the direction of RILA was "South-North" the precision of the estimations in this direction was lower compared to the estimations of the other direction ("East-West") (see Figure 3.15).

To describe the quality of the 3*D* analysis, a PCA metric showing the variance of the created geometric features was used (see Figure 3.23). The choice of the geometric features, the performance of the classification, the ground filtering and the outlier removal algorithms, were visually inspected.

8. **Can this method be linked to Fugro's OnePortal system? Is it robust?**

   It can be concluded that the followed methodology can be integrated to Fugro's asset digital twin named *OnePortal*. In Section 4.3, the results of both 2*D* and 3*D* analysis were illustrated on Fugro's OnePortal visualization system accurately (i.e. within the range of the given threshold of $1m$) . The integration of the method to OnePortal visualization system is generic and robust and as long as ground truth data are provided, it can be used for the visualization of the assets of other case-studies.

6.2 RECOMMENDATIONS

The recommendations section is distinguished in 2 parts. The first part is a discussion for future research based on the 2*D* analysis, while the second part presents potential alternatives in 3*D* analysis.

The recommendations are based on the dominant sources of the errors of the proposed model. The steps of labeling, training and detection in 2*D* analysis (Section 3.1), consist the biggest proportion of the error budget, while the 3*D* analysis portion is smaller. Hence, using alternatives to increase the overall performance of the model, is discussed in this section.

6.2.1 Training Data

Creating training data for deep learning in 2*D* analysis is a time consuming procedure, especially in cases where multiple classes in various non-homogeneous environments are needed. "*Generating large labeled training data is becoming the biggest bottleneck in building and deploying supervised machine learning models*" [78]. In addition, in cases where there are limited data for labeling (e.g. rarely appeared infrastructural railway objects), overfitting might be caused. "*Overfitting is a fundamental issue in supervised machine learning which prevents us from perfectly generalizing the models to well fit observed data on training data, as well as unseen data on testing set*" [79]. To overcome these limitations, augmentation and automatic labeling alternatives are suggested.

- Geometric transformations, color space augmentations, kernel filters, mixing images, random erasing and feature space augmentation, are some augmentation techniques that are used to improve the performance of the model [80].

- Tensorflow library uses image augmentation [81] which improves 2*D* detector's accuracy approximately by 2.3 mean average precision (mAP) and hence it increases the performance, as well as it is more convenient because it allows the use of a smaller number of training data [82].

- Automatic image annotation (AIA), where using machine learning (ML) techniques, extracts semantic features in imagery [83]. Using (AIA) techniques, will reduce the amount of labeling hours dramatically.

6.2.2 Deep Convolutional Neural Networks Alternatives

It was mentioned above that the effectiveness of the 2*D* detector is important for the overall performance of the model. The convolutional neural networks (CNN), can be more efficient detection-wise and on how thoroughly create the borders of a bounding box (localization). This subsection proposes alternatives that may be used to increase the overall performance of the model.

- Gaussian YOLOv3 : a method for predicting the localization uncertainty that indicates the reliability of a bounding box. By using the predicted localization uncertainty during the detection process, the proposed schemes can significantly reduce the false positives and increase the true positives, thereby improving the accuracy [84].

- Using a CNN that combines object detection and localization, with instant segmentation (*Deep Sort*) [85]. In this way, the masks of the detected objects can be used instead of the central coordinate of their bounding boxes as the center of their gravity. In this way, both the mAP (see Table 2.1) and the re-construction of their position in 3D world coordinate system (Section 3.7), may return finer results.

6.2.3 Generalization of the Pipeline

The automation in object inventory (Section 3.8) had some limitations that were illustrated in Section 5.3. This sub-section discusses ways to mitigate those limitation. It is recommended to:

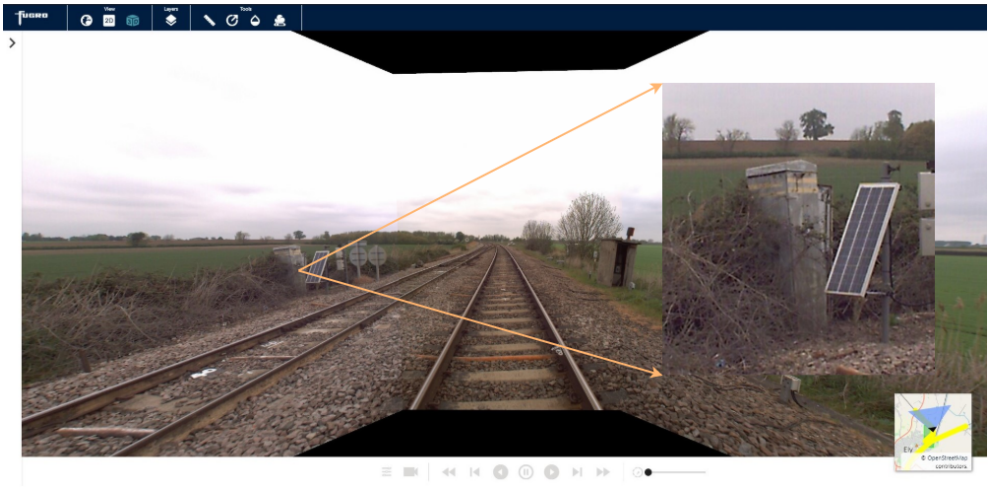


Figure 6.1: The benefit of involving all cameras. This image is taken via Fugro's OnePortal system and illustrates a tessellation of RILA's 3 cameras. In front of the equipment box there is vegetation that blocks the direct and clear view of the infrastructure from the central camera.

- Make use of *Kalman filter* to predict the trajectory of the center of the detected bounding boxes. In this way, a hyper-parameter for the selection of the frame-window and pixel-window, may be introduced. Hence, the choice of the 2*D* frames involving the triangulation may become more robust.

- Involve hypothesis testing and confidence interval in the positional results. In this way, post processing may be introduced to the estimated positions with low confidence.
- Involve all the 3 cameras. Having more $2D$ frames increases the overall model's performance Section 5.3, hence, using the frames taken via all cameras will improve the performance. In addition, when there are other objects or vegetation in front of an object, the central camera is insufficient in providing clear shots of the object Figure 6.1, hence the use of all possible cameras can tackle the "blind spot" issue.

6.2.4 Improve Classification

By improving the classification in $3D$ point clouds, the calculation of the height of the detected objects will be more robust. Bellow, steps for $3D$ point cloud classification refinement are proposed.

- Enhance the $3D$ point cloud data with the R,G,B information from the cameras. Combine the R,G,B with the geometric features (Section 3.9.2) created via the principal component analysis (PCA), will increase the quality of the classification.
- Using supervised classification (see Figure 2.14) after the PCA analysis will return remarkably greater results.
- Optimize the iterative ground filtering method discussed in Section 3.9.1. Apart of the complexity of that algorithm, it can be optimized to perform in voxels including an equipment box as well as a signal. Due to the density differences of these two voxels, the algorithm should tuned-up accordingly.
- The use of more sophisticated algorithms for outlier removal such as a fast cluster statistical outlier removal (FCSOR) [86], may help us further clean the points classified as *object*.

7 APPENDIX

7.1 POINT CLOUD ANALYSIS

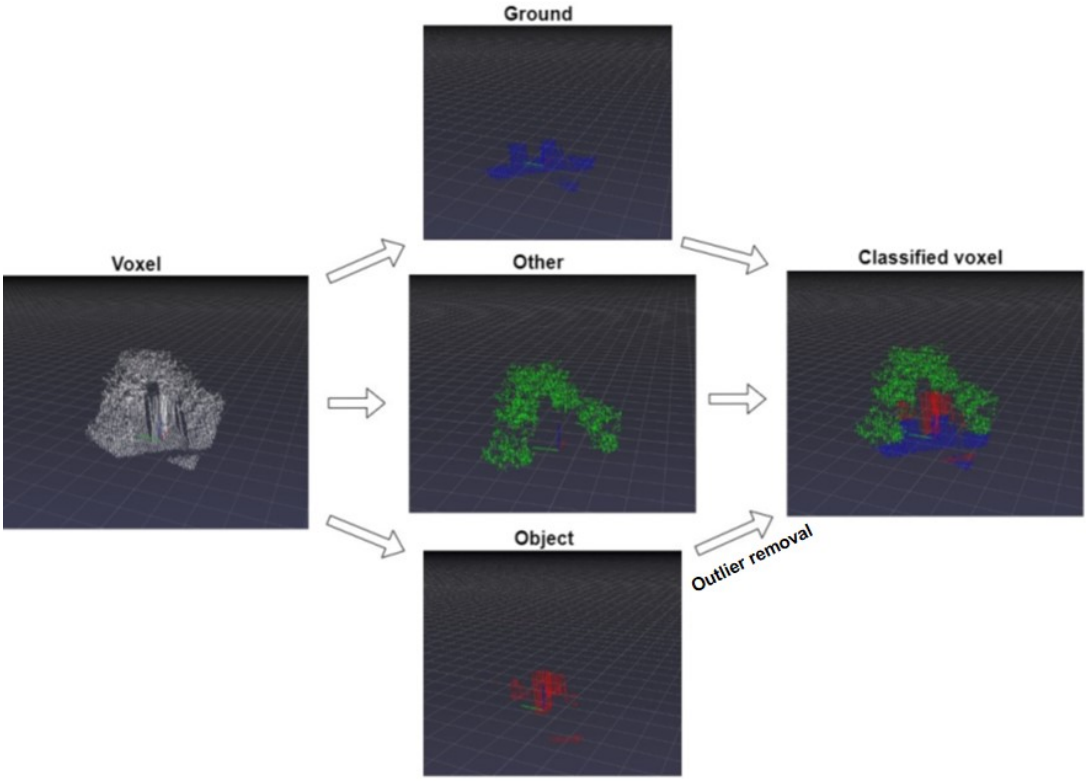
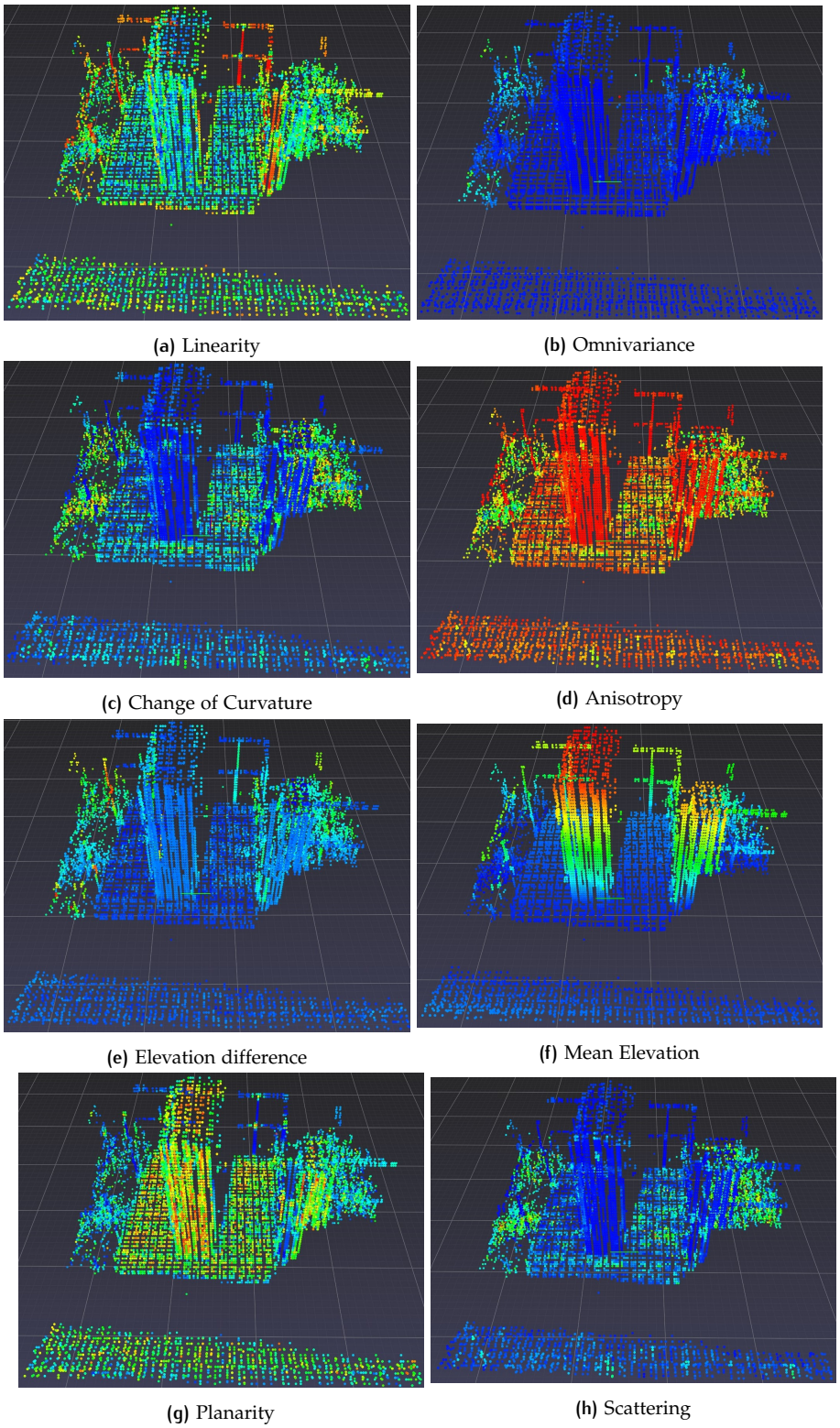


Figure 7.1: Voxel signal classification.

7.1.1 Geometric Features - Principal Component Analysis (PCA)



(a) Linearity (b) Omnivariance

(c) Change of Curvature (d) Anisotropy

(e) Elevation difference (f) Mean Elevation

(g) Planarity (h) Scattering

Figure 7.2: 8 geometric features of a equipment box. The color range follow the same order as the visual color spectrum. $Bluish \approx 0, Reddish \approx 1$.

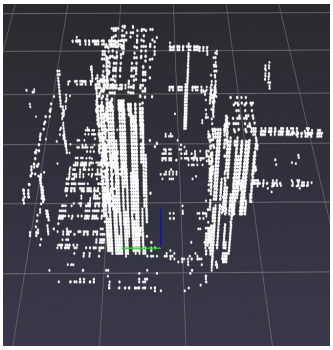


Figure 7.3: Class: Object

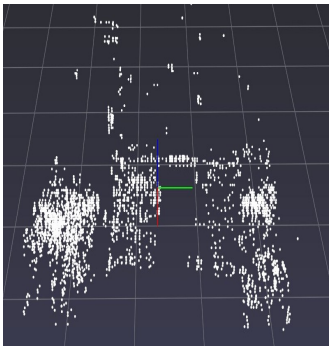


Figure 7.4: Class: Other

Figure 7.5: Box.

Figure 7.6: Classification of the non-ground points of a signal based on the geometric feature *scattering*.

7.1.2 Ground Filtering

The steps and the principal of this method can be seen as a diagram in Figure 7.7 as well as in the pseudo-code Algorithm 7.1.

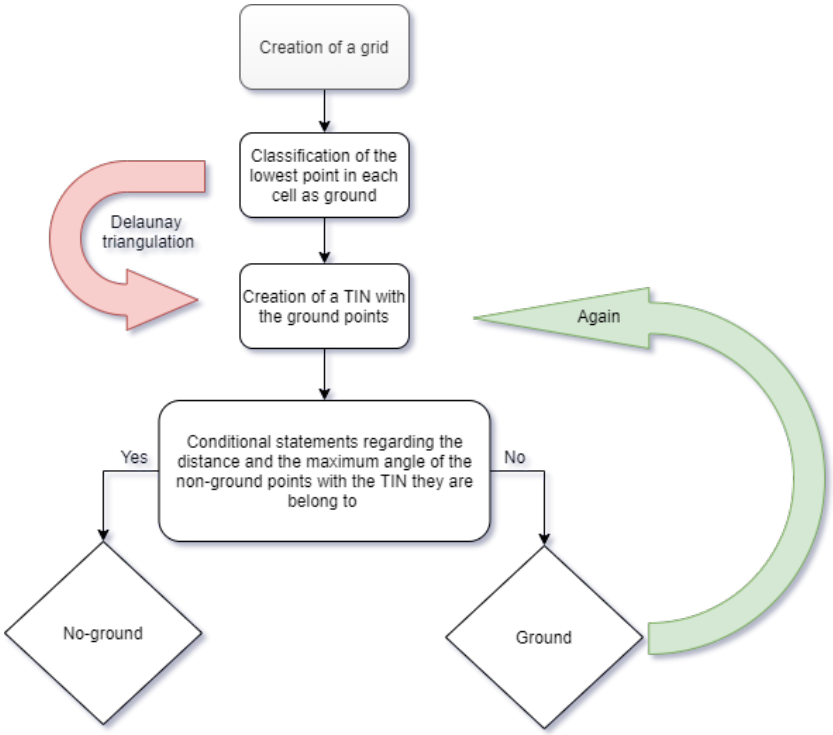


Figure 7.7: Flowchart of the iterative ground filtering method using TIN.

****

Algorithm 7.1: GROUND FILTERING ($\mathcal{P}, grid\_size, \theta, d$)

Input: Unstructured point cloud data $\mathcal{P}$, the size of the grid cell $grid\_size$, the maximum allowable angle value θ, and the maximum allowable distance d

Output: $\mathcal{P}'$: the non-ground points

```
for i ← cell_start to cell_end do
    ground_points ← lowest_point_of_cell
TIN ← ground_points        # Delaunay triangulation
while True do
    for i ← 0 to length(no_ground_points) do
        distance = dist(no_ground_point(i), triangle)
        max_angle = max_angle(no_ground_point(i), triangle_vertices)
        if distance < d & max_angle < θ then
            TIN ← append no_ground_point(i)
```

BIBLIOGRAPHY

[1] Leica Geosystems AG. Improving the railway infrastructure of a capital city, (accessed: 20.01.2021).

[2] H Wang, J Berkers, and Fugro NL Land BV. Absolute and relative track geometry: Closing the gap. 2019.

[3] Technische Univeritat Munchen & Institute of Flight System Dynamics. Differential gnss, (accessed: 06.03.2021).

[4] GueiSian Peng. Performance and accuracy analysis in object detection. 2019.

[5] Ayoosh Kathuria. What's new in yolo v3?, (accessed: 10.08.2020).

[6] Shaunak Halbe. Object detection and instance segmentation: A detailed overview. *medium*, (accesed: 06.09.2020).

[7] OpenCV. Camera calibration and 3*d* reconstruction, (accessed: 01.09.2020).

[8] Benjamin Biström. Comparative analysis of properties of lidar-based point clouds versus camera-based point clouds for 3d reconstruction using slam algorithms. 2019.

[9] Yunting Li, Jun Zhang, Wenwen Hu, and Jinwen Tian. Laboratory calibration of star sensor with installation error using a nonlinear distortion model. *Applied Physics B*, 115(4):561–570, 2014.

[10] Katherine Ellis, Suneeta Godbole, Simon Marshall, Gert Lanckriet, John Staudenmayer, and Jacqueline Kerr. Identifying active travel behaviors in challenging environments using gps, accelerometers, and machine learning algorithms. *Frontiers in public health*, 2:36, 2014.

[11] Florent Poux, Christian Mattes, and Leif Kobbelt. Unsupervised segmentation of indoor 3d point cloud: application to object-based classification. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 44(W1-2020):111–118, 2020.

[12] Martin Weinmann, Boris Jutzi, and Clément Mallet. Geometric features and their relevance for 3d point cloud classification. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4:157, 2017.

[13] Tobias Weis. Triangulate 3*d* points from 3d imagepoints from a moving camera, (accesses: 10.9.2020).

[14] Shangshu Cai, Wuming Zhang, Jianbo Qi, Peng Wan, Jie Shao, and Aojie Shen. Applicability analysis of cloth simulation filtering algorithm for mobile lidar point cloud. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(3), 2018.

[15] Artem Leichter, Martin Werner, and Monika Sester. Feature-extraction from all-scale neighborhoods with applications to semantic segmentation of point clouds. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 43:263–270, 2020.

[16] Nusra Noorudheen, Mitchell McClanachan, Yvonne Toft, et al. Track worker safety: investigating the contributing factors and technology solutions. In *CORE 2012: Global Perspectives; Conference on railway engineering, 10-12 September 2012, Brisbane, Australia*, page 91. Engineers Australia, 2012.

[17] S. Ramos, S. Gehrig, P. Pinggera, U. Franke, and C. Rother. Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1025–1032, 2017.

[18] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):731–747, 2018.

[19] Mario Soilán, Belen Riveiro, Joaquin Martinez-Sanchez, and Pedro Arias. Traffic sign detection in mls acquired point clouds for geometric and image-based semantic inventory. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:92–101, 2016.

[20] Yunus Santur, Mehmet Karaköse, and Erhan Akin. Random forest based diagnosis approach for rail fault inspection in railways. In *2016 National Conference on Electrical, Electronics and Biomedical Engineering (ELECO)*, pages 745–750. IEEE, 2016.

[21] Claudia Gedrange, Reinhard Beger, Marco Neubert, and Robert Hecht. Extraction of railroad infrastructure objects from extremely high resolution aerial imagery and lidar data. In *2nd European LiDAR Mapping Forum (ELMF)*, 2011.

[22] S Neubert, R Hecht, C Gedrange, M Trommler, H Herold, T Kruger, and F Brimmer. Extraction of railroad objects from very high resolution helicopter-borne lidar and ortho-image data, isprs wg iv/4 landscape modeling and visualization. *GEOgraphic Object Based Image Analysis for the 21st Century: GEOBIA*, 2008.

[23] Zhigang Xu, Haigen Min, and Hongkai Yu. Fusion of 3d lidar and camera data for object detection in autonomous vehicle applications. 2019.

[24] C Vincent Tao. Mobile mapping technology for road network data acquisition. *Journal of Geospatial Engineering*, 2(2):1–14, 2000.

[25] Klaus Peter Schwarz and Naser El-Sheimy. Mobile mapping systems–state of the art and future trends. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(Part B):10, 2004.

[26] Fugro raildata. Rila track geometry, (accessed: 03.11.2020).

[27] Jie Shan and Charles K Toth. *Topographic laser ranging and scanning: principles and processing*. CRC press, 2018.

[28] NOAA. What is lidar?, (accessed: 12.02.2021).

[29] Dong-Hyo Sohn, Kwan-Dong Park, and Hyunu Tae. Modeling dgnss pseudorange correction messages by utilizing satellite repeat time. *Sensors*, 17(4):834, 2017.

[30] Enrico Canuto, Carlo Novara, Donato Carlucci, Carlos Perez Montenegro, and Luca Massotti. *Spacecraft Dynamics and Control: The Embedded Model Control Approach*. Butterworth-Heinemann, 2018.

[31] Ahmed El-Rabbany. *Introduction to GPS: the global positioning system*. Artech house, 2002.

[32] Peng Xie and Mark G Petovello. Measuring gnss multipath distributions in urban canyon environments. *IEEE Transactions on Instrumentation and Measurement*, 64(2):366–377, 2014.

[33] V. Malyavej, W. Kumkeaw, and M. Aorpimai. Indoor robot localization by rssi/imu sensor fusion. In *2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pages 1–6, 2013.

[34] L. Sahawneh and M. A. Jarrah. Development and calibration of low cost mems imu for uav applications. In *2008 5th International Symposium on Mechatronics and Its Applications*, pages 1–9, 2008.

[35] Jianming Zhang, Manting Huang, Xiaokang Jin, and Xudong Li. A real-time chinese traffic sign detection algorithm based on modified yolov2. *Algorithms*, 10(4):127, 2017.

[36] Imrankhan Pathan and Chetan Chauhan. A survey on moving object detection and tracking methods. 2015.

[37] M Gomathy Nayagam and K Ramar. A survey on real time object detection and tracking algorithms. *Int. J. Appl. Eng. Res*, 10(9):8290–8297, 2015.

[38] Cong Tang, Yunsong Feng, Xing Yang, Chao Zheng, and Yuanpu Zhou. The object detection based on deep learning. In *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, pages 723–728. IEEE, 2017.

[39] Nishchal K Verma, Teena Sharma, Shreedharkumar D Rajurkar, and Al Salour. Object identification for inventory management using convolutional neural network. In *2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–6. IEEE, 2016.

[40] Yunong Tian, Guodong Yang, Zhe Wang, Hao Wang, En Li, and Zize Liang. Apple detection during different growth stages in orchards using the improved yolo-v3 model. *Computers and electronics in agriculture*, 157:417–426, 2019.

[41] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[42] Rokas Balsys. Yolo v3 theory explained, (accessed: 15.10.2020).

[43] Jacob Solawetz. What is mean average precision (map) in object detection?, (accessed: 03.08.2020).

[44] Yanlai Zhou, Fi-John Chang, Li-Chiu Chang, I-Feng Kao, and Yi-Shin Wang. Explore a deep learning multi-output neural network for regional multi-step-ahead air quality forecasts. *Journal of cleaner production*, 209:134–145, 2019.

[45] Fabio Remondino, Maria Grazia Spera, Erica Nocerino, Fabio Menna, Francesco Nex, and Sara Gonizzi-Barsanti. Dense image matching: comparisons and analyses. In *2013 Digital Heritage International Congress (DigitalHeritage)*, volume 1, pages 47–54. IEEE, 2013.

[46] Grazia Caradonna, Eufemia Tarantino, Marco Scaioni, and Benedetto Figorito. Multi-image 3d reconstruction: a photogrammetric and structure from motion comparative analysis. In *International Conference on Computational Science and Its Applications*, pages 305–316. Springer, 2018.

[47] Ling Zou and Yan Li. A method of stereo vision matching based on opencv. In *2010 International Conference on Audio, Language and Image Processing*, pages 185–190. IEEE, 2010.

[48] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019.

[49] Satyarth Praveen. Efficient depth estimation using sparse stereo-vision with other perception techniques. In *Advanced Image and Video Coding*. IntechOpen, 2019.

[50] Yasir Salih and Aamir S Malik. Depth and geometry from a single 2d image using triangulation. In *2012 IEEE International Conference on Multimedia and Expo Workshops*, pages 511–515. IEEE, 2012.

[51] Richard I Hartley and Peter Sturm. Ge-crd, rm k1-5c39, po box 8, schenectady, ny, 12301 hartley@ bunyip. crd. ge. com y gravir-imag & inria rhône-alpes 655, avenue de l'europe, 38330 montbonnot, france. 1996.

[52] J Tischendorf, C Trautwein, Til Aach, D Truhn, T Stehle, et al. Camera calibration for fish-eye lenses in endoscopywith an application to 3d reconstruction. In *2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1176–1179. IEEE, 2007.

[53] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of ieee computer society conference on computer vision and pattern recognition*, pages 1106–1112. IEEE, 1997.

[54] Fabio Remondino and Clive Fraser. Digital camera calibration methods: considerations and comparisons. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):266–272, 2006.

[55] Jiapeng Liu, Zhengwu Yang, Hong Huo, and Tao Fang. Camera calibration method with checkerboard pattern under complicated illumination. *Journal of Electronic Imaging*, 27(4):043038, 2018.

[56] J. Heikkila. Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1066–1077, 2000.

[57] Yuan Li and Bo Wu. Structural segmentation of point clouds with varying density based on multi-size supervoxels. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 2019.

[58] Florent Poux and Roland Billen. Voxel-based 3d point cloud semantic segmentation: unsupervised geometric and relationship featuring vs deep learning methods. *ISPRS International Journal of Geo-Information*, 8(5):213, 2019.

[59] Chao-Hung Lin, Jyun-Yuan Chen, Po-Lin Su, and Chung-Hao Chen. Eigenfeature analysis of weighted covariance matrices for lidar point cloud classification. *ISPRS journal of photogrammetry and remote sensing*, 94:70–79, 2014.

[60] Stéphane Guinard and Loic Landrieu. Weakly supervised segmentation-aided classification of urban scenes from 3d lidar point clouds. 2017.

[61] Hongbo Gao, Bo Cheng, Jianqiang Wang, Keqiang Li, Jianhui Zhao, and Deyi Li. Object classification using cnn-based fusion of vision and lidar in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics*, 14(9):4224–4231, 2018.

[62] Reinhard Beger, Claudia Gedrange, Robert Hecht, and Marco Neubert. Data fusion of extremely high resolution aerial imagery and lidar data for automated railroad centre line reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6):S40–S51, 2011.

[63] Kaixuan Zhou. Combining lidar and photogrammetry to generate up-to-date 3d city models. 2020.

[64] Hyeok-June Jeong, Kyeong-Sik Park, and Young-Guk Ha. Image preprocessing for efficient training of yolo deep learning networks. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 635–637. IEEE, 2018.

[65] AlexeyAB. Yolo v4, v3 and v2 for windows and linux, (accessed: 20.10.2020).

[66] Tzutalin. Labelimg - git code (2015), (accessed: 02.08.2020).

[67] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.

[68] Todd Kulesza, Denis Charles, Rich Caruana, Saleema Amin Amershi, and Danyel Aharon Fisher. Structured labeling to facilitate concept evolution in machine learning, June 11 2019. US Patent 10,318,572.

[69] José Galvez. Camera calibration, (accessed: 09.02.2021).

[70] Matija Burić, Miran Pobar, and Marina Ivašić-Kos. Adapting yolo network for ball and player detection. In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2019)*, pages 845–851, 2019.

[71] OpenCV. Our train - the highest in the world, (accessed: 09.09.2020).

[72] Peter Axelsson. Dem generation from laser scanner data using adaptive tin models. *International archives of photogrammetry and remote sensing*, 33(4):110–117, 2000.

[73] Chris Lucas, Willem Bouten, Zsófia Koma, W Daniel Kissling, and Arie C Seijmonsbergen. Identification of linear vegetation elements in a rural landscape using lidar point clouds. *Remote Sensing*, 11(3):292, 2019.

[74] P. D. Groves. Principles of gnss, inertial, and multisensor integrated navigation systems, 2nd edition [book review]. *IEEE Aerospace and Electronic Systems Magazine*, 30(2):26–27, 2015.

[75] Kan Wang, Peter JG Teunissen, and Ahmed El-Mowafy. The adop and pdop: Two complementary diagnostics for gnss positioning. *Journal of Surveying Engineering*, 146(2):04020008, 2020.

[76] Xiting Zhao, Zhijie Yang, and Sören Schwertfeger. Mapping with reflection-detection and utilization of reflection in 3d lidar scans. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 27–33. IEEE, 2020.

[77] JL Lovell, DLB Jupp, GJ Newnham, NC Coops, and DS Culvenor. Simulation study for finding optimal lidar acquisition parameters for forest height retrieval. *Forest Ecology and Management*, 214(1-3):398–412, 2005.

[78] Nilaksh Das, Sanya Chaba, Renzhi Wu, Sakshi Gandhi, Duen Horng Chau, and Xu Chu. Goggles: Automatic image labeling with affinity coding. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1717–1732, 2020.

[79] Xue Ying. An overview of overfitting and its solutions. In *Journal of Physics: Conference Series*, volume 1168, page 022022. IOP Publishing, 2019.

[80] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[81] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172*, 2019.

[82] Žiga Emeršič, Dejan Štepec, Vitomir Štruc, and Peter Peer. Training convolutional neural networks with limited training data for ear recognition in the wild. *arXiv preprint arXiv:1711.09952*, 2017.

[83] Dengsheng Zhang, Md Monirul Islam, and Guojun Lu. A review on automatic image annotation techniques. *Pattern Recognition*, 45(1):346–362, 2012.

[84] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 502–511, 2019.

[85] Xinyu Hou, Yi Wang, and Lap-Pui Chau. Vehicle tracking using deep sort with low confidence track filtering. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2019.

[86] Haris Balta, Jasmin Velagic, Walter Bosschaerts, Geert De Cubber, and Bruno Siciliano. Fast statistical outlier removal based method for large 3d point clouds of outdoor environments. *IFAC-PapersOnLine*, 51(22):348–353, 2018.

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the arsclassica package by Lorenzo Pantieri, which is an adaption of the original classicthesis package from André Miede.