Weakly-supervised semantic segmentation of rails point cloud data using label diffusion

Author
P.H. Zwietering

June 12, 2022

First Supervisor
Prof. dr. M.J. van Kreveld

Second Supervisor
Dr. M. Löffler

A thesis presented for the degree of Master of Science

Utrecht University
Faculty of Science
Department of Information and Computing Sciences
Graduate School of Life Sciences

Abstract

Fugro collects point cloud data, a form of unordered spatial data, with LiDAR, a laser scanning system. Using this type of data, detailed maps of the surrounding of railway tracks can be generated. In order to do this, it is necessary to label all the points in these collected point clouds. This can be done in multiple ways. In this thesis, we explore the usefulness of Label Diffusion LiDAR Segmentation (LDLS). We use a panoptic segmentation model on Fugro video data to perform semantic segmentation and verify its performance on RailSem19 data, an open source semantic segmentation dataset of railway image data. LDLS uses the output of this model to diffuse the labels through the point cloud. We show multiple visual results of a few different configurations. We also give quantitative results of the performance for a subset of platform points.

Acknowledgements

First, I would like to thank Aidan, my fellow intern at Fugro. Especially during the pandemic, it was very relieving to have someone go through the same trepidations and have at least some social interaction during the day. The sparring about our theses was also very welcome. I also want to thank Neda and Sudan, our supervisors at Fugro, who helped out a lot with the thesis in general and the data related and domain specific questions we had. Thanks to Marc for all the help on writing the thesis and keeping me on track and motivated, most of all when I had lost faith a bit myself. Last but certainly not least, I want to thank my roommates, without whom I would probably not have finished this thesis. You kept me sane during the pandemic.

Contents

1 Introduction

In this section we will introduce the problem that this thesis is centered around, together with general information about the context of this project, which forms the relevance of this thesis. We will also introduce some basic concepts necessary to understand this thesis. This section is concluded by the research question of this thesis.

1.1 Research context

This research project was started in the interest of Fugro. Fugro is a multinational Dutch company that specializes in collecting and processing all kinds of geographical data. This ranges from geochemical and geophysical investigations to geopositioning and geospatial data aggregation, both on land and in marine situations. Historically, Fugro has mainly been involved in soil investigation for exploring the possibilities of natural resource extraction, but in recent years they have started to transition to applying their expertise on similar problems for renewable energy and infrastructure as well. The main application for this thesis is in the area of geospatial data aggregation of infrastructure as well.

Geospatial data is all data that contains information ascertaining objects, events or other features located near the surface of the earth [13]. In this thesis, the geospatial data that will be investigated is data representing the local spatial environment of railways. This data comes in the form of point clouds, together with georeferenced camera images and calibration files.

1.1.1 Point clouds

A point cloud is a collection of unordered 3-dimensional points. In a geospatial context, a point cloud represents the location of the external surfaces of objects in a certain area. These points can additionally be labelled with all kinds of information, such as for example colour, an object label or any custom label used for different applications. Point clouds in general can be used for many different purposes, such as creating 3D models of single objects and mapping and visualizing the surroundings of certain areas, both inside and outside. Point clouds can be acquired in four different ways [41]: by using image-derivation, Light Detection And Ranging (LiDAR) systems, Red Green Blue - Depth (RGB-D) cameras or Synthetic Aperture Radar (SAR) systems. This thesis only looks at point clouds collected with LiDAR; the results when applying the same methods might vary wildly for point clouds acquired by other means.

More specifically, Fugro collects their point clouds into so called tiles. Tiles are horizontal squares of 25 by 25 square meters, in which any LiDAR points are collected. These tiles are then saved in a LAZ file, which is a compressed file format for storing LiDAR points. Figure 1 shows a top down overview of all tiles that have a corresponding file with some number of LiDAR points.

1.1.2 LiDAR

LiDAR systems, or laser scanners as they are also called, are systems that use emitted laser beams to measure the distance from the system to surfaces surrounding it. It measures the time it takes for the beam to leave the laser to the time it takes for the refracted light to return to the system, together with the direction of the laser beam at the time the beam was sent, to calculate at what exact point a surface was measured. Also, since the speed of light is orders of magnitude higher than the speed of trains or planes, it can be used while moving on those vehicles while retaining a very high accuracy. To calculate the distance d, one uses the light speed c and time t it takes between emitting the laser beam and arrival of the refracted light as follows:

$$d = \frac{c \cdot t}{2} \tag{1}$$

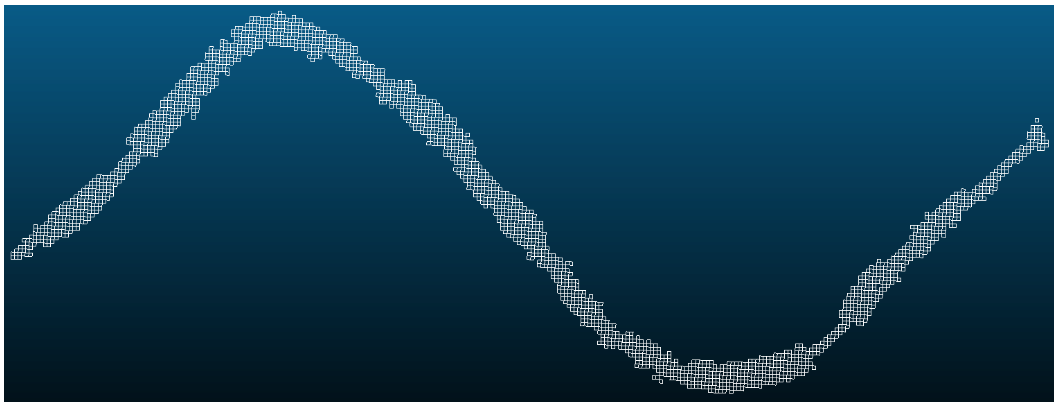


Figure 1: Top overview of all tiles of this specific railway track Fugro collected LiDAR points on.

LiDAR systems can be deployed in multiple ways. The literature distinguishes between four types of scanning [41]: terrestrial laser scanning (TLS), airborne laser scanning (ALS), mobile laser scanning (MLS) and unmanned laser scanning (ULS). TLS is done with a static system, that might only turn around its axes. ALS is operated from an airplane or even satellites. ULS is done using small drones and differs from ALS in that the distance from the system to the surfaces it measures is way smaller; a drone with ULS might fly through and around a factory, whereas a plane equipped with ALS flies a few kilometers high from the surface of the earth that it is surveying. Finally, MLS refers to systems mounted on moving, earthbound vehicles such as trains or cars. The point clouds of these different systems have very different properties, benefits and limitations. The main difference between these methods is the sparsity of the point clouds; ALS will have point clouds that are significantly more sparse than point clouds acquired through TLS.

1.2 Problem description

Fugro thus collects point clouds from railway environments. This is done through MLS, by mounting their own RILA system on the back of trains. This system contains a 360° LiDAR scanner together with 3 cameras and an accurate GPS system, along with other equipment that is not relevant for this research, to create a complete view of the surrounding area. A detailed image of this system can be seen in Figure 2.

The 360° LiDAR scanner scans surfaces in a circular motion along the plane perpendicular to the railway direction. This produces so called scan lines, which are points that are loosely arranged in a linear pattern when looked upon from a top-down view, such as in Figure 4. Scanners that scan in the direction of the movement of the vehicle also exist, but they have the disadvantage that larger objects can obstruct the "view" of the scanning line of smaller objects. On the other hand, thin wayside objects, such as railway signs, might not be completely detected by the RILA system. An example can is shown in Figures 5 and 6. Putting together a multitude of these scan lines forms a point cloud that forms a scene that is moderately recognizable for human eyes, but harder to interpret by computers. See Figure 3 for an example of the views of the three cameras and Figure 4 for an example of point cloud depicting a train station.

Fugro has numerous use cases for point clouds collected in this way. It uses the laser stripers on the bottom of the RILA system to get a very detailed and accurate reading of the height and shape of the railway track, which can be used to determine the need for maintenance. The use case that is especially relevant for this research project is the mapping of the local environment of the railway. This mapping is a dataset of labelled polygonal shapes describing the location of all relevant objects close to the railway track. An example can be seen in Figure 7.

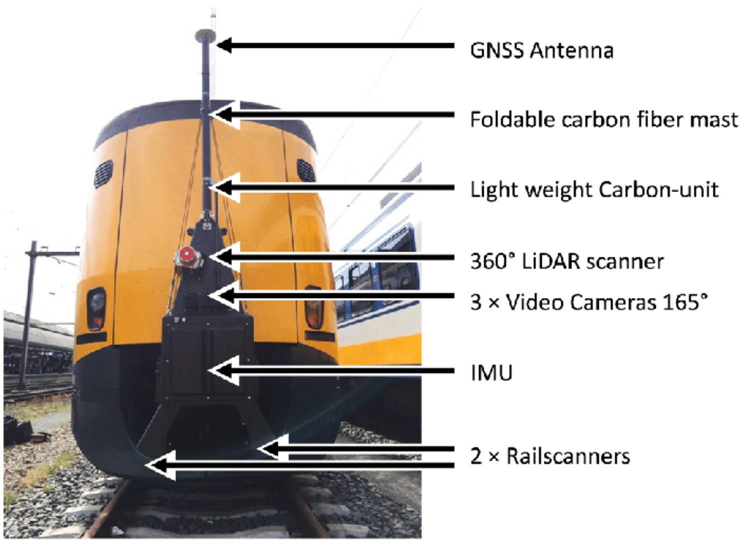
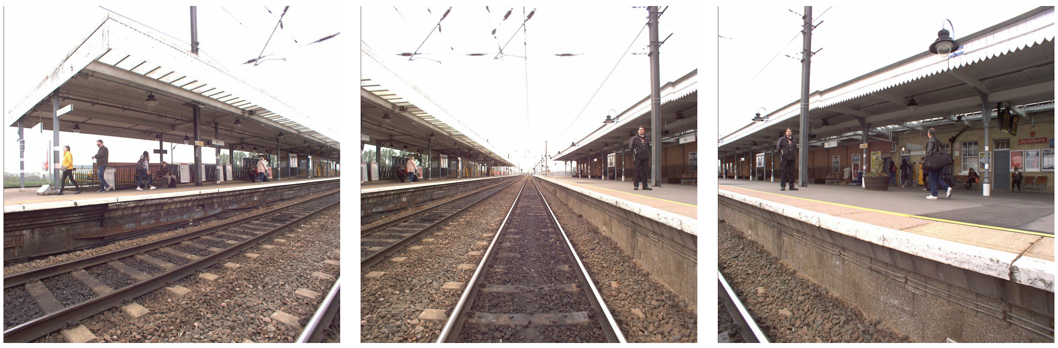


Figure 2: Fugro RILA system [38].



(a) Left facing camera. (b) Front facing camera. (c) Right facing camera.

Figure 3: The three camera viewpoints from the back of the train on the only train station present in the Fugro video data.

In order to deliver a map suitable for sale, a necessary intermediary step is to label all points in the point clouds according to the object they are a part of. This creates a segmentation of the point cloud, where points close to each other that have the same label form segments of points that form objects. Such a segmentation can then be used to create the desired maps. Therefore, it is of great importance that point clouds are labelled accurately and homogeneously. This brings us to the problem that is central in this thesis: semantically labelling point clouds placed in a railway setting.

1.2.1 Current solution

Currently, Fugro uses the following method for labelling the acquired point clouds. First, a smaller, representative subset of the point cloud is labelled by employee with computer assistance. This is done by calculating features of all points, where features are per-point properties such as position, LiDAR intensity or the number of neighbours in a certain radius around the point. These features are then used to group points that have similar features, while visually checking that points are indeed segmented correctly. Hand-labelling points in such a manner takes up a significant amount of time and labour.

Using this piece of reliably labelled point cloud, a neural network is trained on all points that

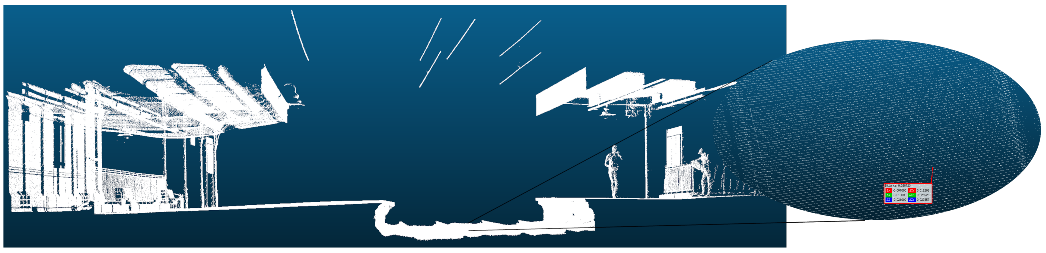


Figure 4: Side view of the station point cloud tile. A top down zoomed view is taken to show the scan lines, roughly 3 centimeters apart here, but this distance is dependent on the speed of the train.

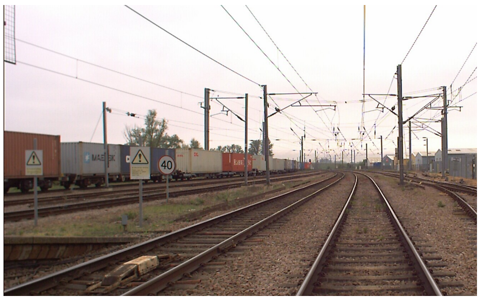


Figure 5: Cropped video frame of a part of the track. On the left the two wayside objects with exclamation marks are clearly visible, without objects obstructing the laser scanner.

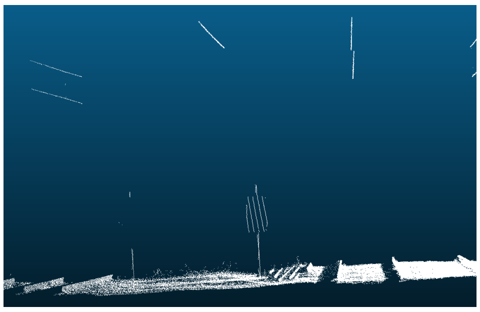


Figure 6: Corresponding point cloud slice. Only one of the signposts is (partly) visible.

uses the features as input and the point labels as output. This is done for every different RILA project, i.e. the same neural network can not be used for all point clouds Fugro needs to label, due to the apparent significance of regional differences in railway environments. Inference on this neural network can take up to an hour per 25 by 25 square meters of point cloud. Since the network considers all points separately, there can be a lot of noise. The accuracy usually lies somewhere between 90% - 95 % on test data, so after inference more computer-assisted manual labelling needs to be done. Accuracy of the labelling is measured simply by checking how many points are labelled correctly compared with the ground truth, divided by the total number of points.

1.3 Research questions

Having such clear drawbacks, Fugro has the wish for a methodology that might (partly) replace or complement their current one in the future. In order to develop such an approach, they asked us to explore available options or come up with a solution better than these options. Fugro has a few constraints for this endeavour.

- The final labelling should be as accurate as possible. Fugro needs to achieve a 100% accuracy in order to deliver correct maps. Of course, such an accuracy will never be feasible, but it does indicate that accuracy is important. For this project however, the accuracy should be at least 80% or higher.

- Preprocessing time notwithstanding, the current method of Fugro takes a pretty long time. The largest point cloud tile that Fugro gave us in this project has just shy of 4.5 million points, which means inference of the neural network per point takes around 0.8

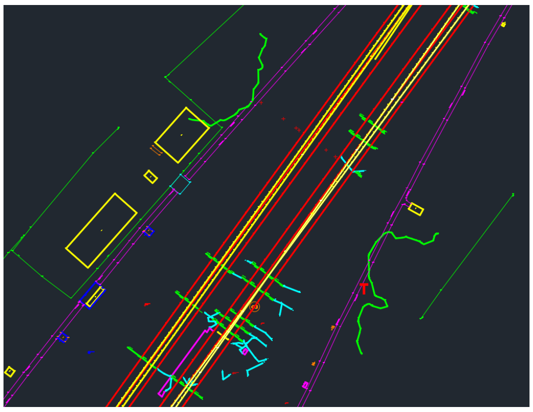


Figure 7: Typical top down view of maps delivered by Fugro. Different types of objects get different colouring and different corresponding geometrical shapes. Object labels are also denoted when zoomed in enough.

milliseconds when assuming the point cloud is not sampled down, which it usually is. This is definitely not quick for neural network standards. Fugro has no hard demands for the execution times, but it would be a success if they could be reduced from a matter of hours to minutes.

- Since creating ground truth, i.e. computer-assisted manual point cloud labelling, is so expensive, our approach should have no need for it. This has the consequence that our approach must also be generally applicable for different geographic locations, as long it is still a railway scene[1]. However, training of models on publicly available training datasets is permitted.

- Semantic segmentation is preferred over non-semantic segmentation. The difference lies in the meaning behind the labels that are given. Because of the previous constraint, the number of methods that can be applied to achieve semantic segmentation is limited.

- The amount of human interaction with the solution should be as low as possible. Tuning parameters and checking the results can take time, more so when there is a great number of parameters.

During the literature research for the research proposal, a lot of methods that could be applied to this research project were tried or immediately rejected if their properties conflicted with the constraints above. These methods will still be discussed in Section 2, in order to give background information and also be able to substantiate why we chose the method we have. This method is called Label Diffusion LiDAR segmentation [37] or LDLS for short, and will be discussed in detail in Section 2.4.1. The relevant information for now are its claimed properties: this method achieves semantic segmentation in a matter of seconds instead of hours, does not need human interaction and claims to be accurate. To get to a semantic segmentation of point clouds, it uses a semantic segmentation of video data instead. In consultation with Fugro it was decided to investigate LDLS. In conjunction with LDLS, a segmentation model of the image classification framework of Detectron2 [39] was chosen.This has resulted in the following research question for this project.

How does LDLS perform in a railway environment?

[1]We will not be able to test this, since only data for one part of track was delivered, located around Ely, England.

In order to check the performance of LDLS, it is necessary to view the operation of its steps separately. This gives rise to the following subquestions:

- How well can Detectron2 be applied to achieve accurate semantic segmentation on railway video data?
- How well does label diffusion perform when applied to point cloud data of Fugros railway data?
- What steps of the LDLS approach are most suitable for augmentation and how?

The rest of this thesis contains a description of relevant related work in Section 2, including the literature research studied for the proposal of this thesis. We describe the methodology used for the experiments in Section 3. The results to these experiments are presented in Section 4, which mainly consist of qualitative results due to a lack of access to ground truth. Topics for discussion and recommendations are given in Section 5. The thesis is concluded in Section 6.

2 Related work

A lot of research is being done with regards to segmentation and classification of point cloud data. So much so that every few years surveys on the topic are written to bundle the published research up until that point [4, 25, 40, 41]. The segmentation methodology is fairly well divided in two main research areas: point cloud segmentation (PCS) and point cloud semantic segmentation (PCSS). The difference between the two is that PCS consists of putting points together in different segments, while PCSS is the process of labelling all points with the object that the point is a part of. In this context, a segment is a subset of locally connected points of the point cloud. So where PCS just creates segments that have no relation with each other, PCSS also groups together these segments into object classes. Each of those classes should contain pairwise disconnected segments. Since Fugro put high emphasis on approach having to be unsupervised, PCS algorithms will be focused upon this thesis and in the rest of this document when talking about segmentation, this refers to PCS, except when specifically talking about semantic segmentation.

The rest of this section is structured as follows. First, we give a short overview of miscellaneous techniques that we tried on actual Fugro data. Then, we will cover the unsupervised segmentation techniques. Then we give a very succinct description of semantic segmentation. Last, we describe methods that aim to fuse camera and lidar data to gain the benefits of both and try to mitigate the shortcomings that come with methods that handle either separate data source.

2.1 Preliminary research

During the preliminary research, numerous techniques and methods were looked at to see if they might fit the constraints given by Fugro. Two of these were tested on actual data, to see if they might prove useful for further research.

The first method was using a supervoxelization algorithm in order to preprocess the data [21]. Code for this technique can be found at
`https://github.com/yblin/Supervoxel-for-3D-point-clouds`. Supervoxelization aims to divide the point cloud into many small areas of points that have very similar features, without the need of any domain-specific knowledge to give the points those labels. Those supervoxels can then be used as a whole, instead of looking at all points individually. However, we quickly found out that such a preprocessing technique, and thus supervoxelization in general, will not be sufficient for this research. We found that at clear boundaries, supervoxelization works wonderful, but not on smaller, but still crucial different objects, such as the distinction between the railway track and the ground. See Figure 8 for an example of this algorithm on a Fugro point cloud tile.

The other approach we tried, was to use surface generation on the point clouds to generate surface meshes, for which there also many different supervised and unsupervised algorithms that can do segmentation. Open3D is an open-source library meant to support many tools useful in 3D data analysis and creation. It has multiple surface generation algorithms [43], the most modern of which is Poisson surface reconstruction [14]. The benefit of Poisson surface reconstruction over other methods in this library, is that it can create non-smooth surface, which are generally present in a railway environment. However, when we applied this algorithm to the Fugro data, we quickly found that it is not useful for this type of data. It does not handle small detached objects well, such as the catenary wires, it shows weird behaviour on sharp edges and is not accurate in general. The way the algorithm generates a surface is by finding an optimal vertex between a number of neighbouring points. This means that surfaces are creating in between points, and not on the actual surface of them. For general purposes, this is not a concern, but customers of Fugro need to have millimeter precision, which this method can not guarantee. For an example of the meshes generated this way, see Figure 9.

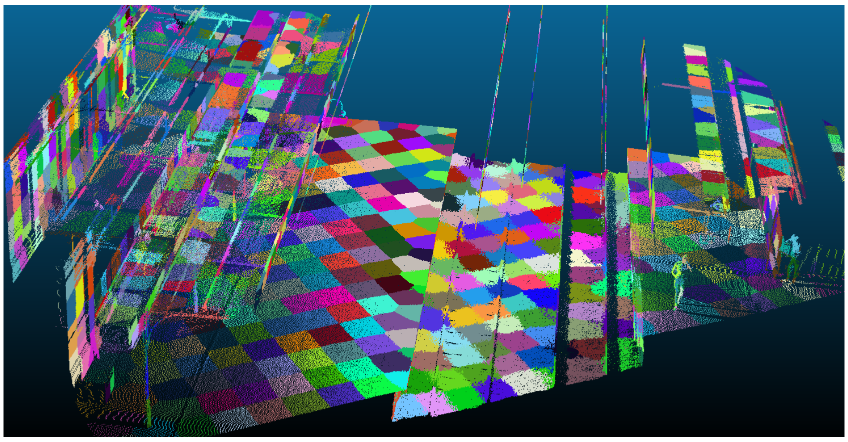


Figure 8: Example of the supervoxelization method presented in [21].

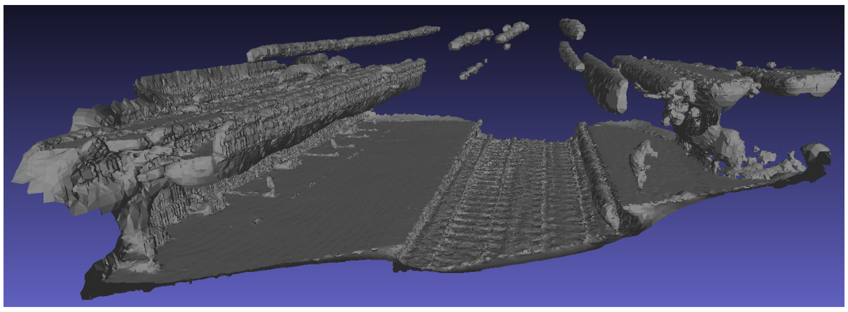


Figure 9: Example of Poisson surface reconstruction applied to Fugro rail data.

2.2 Point cloud segmentation

PCS can be performed in a myriad of ways. Literature often distinguishes the following methods [25, 41], roughly in order of discovery: edge-based, region growing, model-based and clustering-based.

2.2.1 Edge-based PCS

Edge-based PCS is derived from the 2-dimensional edge-based methods that were invented to tackle image segmentation. The shapes of objects can be described by their edges, therefore the idea is that if one can find all edges, one can find all objects. In the case of image segmentation, finding the edges themselves can be done by multiple methods [17]. These usually rely on a core principle to detect a sudden change in intensity between different pixels. This principle still holds for the 3-dimensional case. After finding all edges, the rest of the points are then grouped together into segments based on their similarities.

Earliest methods use e.g. the local surface curvature [8], while newer methods use more sophisticated metrics, such as constructing a binary edge map [30]. All these methods have in common that they can be executed fairly quickly. However, the main drawbacks are that edge-based segmentation algorithms can only reliably segment linear and planar point clouds, because other types of shapes are way harder to describe by either planes or lines. For the same reason, this type of segmentation algorithm also can not deal well with outliers and noise.

2.2.2 Region growing PCS

Region growing is a segmentation strategy which does precisely as the name suggests. Region growing aims to divide the point cloud into multiple regions by grouping together points that have similar properties. Usually, seed points are chosen in some way, either manually or by a heuristic, which then form the basis of the regions that will be reported as segments. All neighbouring points of the initial seed points are then taken into consideration, to see if they are similar enough to be joined into the region. This decision is based on certain criteria like the intensity and geometric properties of the point neighbourhood. This process is repeated iteratively until all points belong to a region. Some methods also allow the joining of regions [9, 26, 36].

Region growing can perform really well, but requires a lot of parameters to be set up front. Tweaking these parameters, i.e. the growing criteria and initial region seeds, can be a tedious process in the form of trial and error. The result can quickly be either over- or undersegmented. This means there are respectively either too many or too little segments. Oversegmentation always has preference, and is sometimes even done as a preprocessing step, in order to treat the many little segments as some sort of supervoxel in the rest of the segmentation process. Undersegmentation does not have practical uses. Also, because the algorithm decides if points belong to a region based on the current aggregated features of a region, it does not deal well with outliers and noise.

2.2.3 Model-based PCS

Model-based segmentation is divided into two approaches: RANSAC (RANdom SAmple Consensus) and HT (Hough Transform). Both of these approaches try to fit certain geometric models on the point clouds and see how well these perform according to pre-determined criteria. The approaches differ in how they fit models and how their performances are evaluated.

RANSAC

In general, RANSAC aims to do two things. First it takes random samples from the point cloud (by picking a set of N random points) and then tries to fit geometric mathematical models on these points [18, 19, 31]. These can be shapes like planes, spheres and toruses: as long as they can be mathematically described by the user. What models specifically are used is a parameter that must be chosen by the user. In the next step, the algorithm tries to pick the best parameters for these models. For example, a spherical model will be represented by $(x-a)^2+(y-b)^2+(z-c)^2 = r^2$. So $(a, b, c, r)^T$ are the parameters that have to be evaluated. RANSAC then chooses the most probable parameter values for our points, which is defined as the minimal loss for all parameter values.

HT

HT consists of three steps. First, all points of the point cloud are transformed into points in a discretized parameter space. Then as the second step, we accumulate the number of points that are contained in a parameter subset by keeping up a score for each eligible set of parameters. In the end the shape parameters are picked that get the highest score. The most simple HT algorithm uses an angle-radius parameterization for plane fitting: $\rho = x\cos(\theta)\sin(\phi) + y\sin(\theta)\sin(\phi) + z\cos(\phi)$, with ρ the distance from the point to the origin and θ and ϕ the polar coordinates of the normal of the plane that goes through the point with coordinates (x, y, z). The major drawback of HT is that even though a discretized parameter space is used, the space itself is unbounded, which can cause very large memory usage and long computation times [2,28].

What both of these methods have in common is that they can detect shapes in places that do not actually exist. For example, imagine that there are a lot of points that all lie on a single line segment, but the line segment itself is interrupted by gaps. These methods might mistakenly detect a straight uninterrupted line segment, because there are multiple point that do support the claim that there is a line segment present. However, the flip side is that these methods can handle outliers and noise relatively well. RANSAC is also non-deterministic, because of the random sampling it does.

2.2.4 Clustering-based PCS

This category of segmentation algorithms is not as strictly defined as the rest. Loosely speaking, these algorithms all group together points based on some features, be they geometric or something else. One might argue that model-based, edge-based and region growing also do this, but for clustering-based algorithms the features that are segmented upon might not even be known beforehand, which make them quite distinguishable. The main directions of clustering-based PCS are K-means clustering, fuzzy clustering, mean-shift and graph-based approaches.

K-means clustering

K-means clustering is a very classic algorithm, which has proven itself to be useful for a plethora of use cases [10]. Roughly speaking, K-means clustering is an iterative algorithm in which K unlabelled groups of points are clustered in the chosen feature space. First, these K centre points are chosen randomly, or spread out uniformly. Then, all feature points get assigned to one of the K centres based on distance (which can be Euclidian or something else based on the use case). Then the K centres of all these groups is recalculated, after which these 2 steps are repeated until the solution converges. In the context of PCS, K-means clustering is performed by clustering based on local features of points and distance to the cluster centre combined [29]. This makes it not very applicable to the segmentation of general scenes, since there different shapes of varying sizes. K-means clustering is very flexible in the sense that it can easily be adapted to handle different features, but it also depends on the regularity of the shapes it considers. It also suffers from the main drawback of K-means clustering: getting the right value for K, which is a recurring problem when using K-means clustering.

Fuzzy clustering

Fuzzy clustering is very similar to K-means clustering. It differs mainly on once aspect; in fuzzy clustering the sample points can belong to multiple centre points in varying degrees, whereas with K-means clustering the sample points either belong completely to a centre or they do not. The rest of the algorithm is in essence the same as for K-means clustering. As for PCS, this method has primarily been tested on planar buildings, which it did quite well [1]. Not a lot of further research has gone into this.

Mean shift

Mean shift is a non-parametric clustering algorithm. Therefore, it is very easy to set up: in contrast with K-means clustering one has no need of knowledge of K. However, because in addition to the shape the number of segments is unknown, mean shift tends to oversegment point clouds. Just as with region growing, this does not need to be a problem however [23].

Graph-based

Graph-based clustering is done in a two-part strategy: graph construction and partitioning. First a graph is constructed on the point cloud, usually based on point features and the local neighbourhood of the k nearest neighbours. This builds a graph on all the points in which the points are the nodes and the edges represent whether or not two points might belong to the same object, just based on proximity. Then, some min-cut algorithm is applied in order to sever the edges that represent a connection between different objects in the point cloud [35]. These kinds of min-cut algorithms are well researched topics and often can be executed in polynomial time, which makes them quite efficient [32]. The intuition behind this approach also makes it an appealing strategy.

2.3 Point cloud semantic segmentation

Since Fugro explicitly wants to have some form unsupervised segmentation and labelling of their point clouds, or in the very least, only use publicly available training data[2], this section will be brief. It is still nice to have some background knowledge of the semantic side of segmentation. The two main research directions are the regular machine learning approaches and the deep learning approaches. The difference between the two is that with deep learning, the techniques used can create their own features that are not immediately present in the training data. Apart from that, the process behind these approaches are very similar. There is some model that is fed training data in the form of global and / or local point features, which the model uses to adjust its internal parameters. This is repeated until the model does not change enough any more that further training is warranted. Then, the model is evaluated on test data instead to see how it performs.

The main challenge on the topic of point cloud segmentation with deep learning is the form of the data[3]. Since point clouds are in essence unstructured data, it is not trivial to insert it directly into learning models, which usually need some regularized input type. This can be dealt with in one of the following ways. The first idea is to use a multiview approach, in which 2-dimensional images are taken from the scene in such a way that all points are visible in at least one of these images [33]. This has major drawbacks, since it is not easy to do this automatically, especially for larger scenes. Also, complex 3-dimensional structures might simply get lost when projected into 2 dimensions. Second, one can regularize the point cloud itself, namely by voxelizing it [22]. This is a fairly straightforward process, but it generates a lot of data in the form of empty space that is now registered[4], while losing information in the form of a reduction loss and it introduces the need of label interpolation. Last, there are deep learning networks that manage to handle point clouds directly. PointNet was the first network that succeeded, which inspired a lot the networks that are still being published [27].

2.4 Camera LiDAR fusion

Camera LiDAR fusion refers to methods that try to combine both visual imagery and 3-dimensional data in order to to gain advantages of using either data source but reduce the disadvantages that come with them as well. These methods can be both supervised and unsupervised and have lots of different purposes, such as depth completion, object detection and semantic segmentation [6]. For this thesis we will focus on one specific research, namely into Label Diffusion LiDAR Segmentation method (LDLS) [37]. In the following sections we will lay out the relevant related work that is connected to LDLS.

2.4.1 LDLS

LDLS aims to perform semantic segmentation of point clouds by using no semantic or a priori knowledge of the point cloud data. Instead, LDLS only needs semantic knowledge of camera data corresponding to the point cloud scenes. Usually there is a substantial lack of annotated point cloud data, which is not the case for annotated image data. These days image datasets like ImageNet [7], MS COCO [20] and Googles Open Images [16] are freely available to the public and offer millions of labelled images that can be used for all kinds of deep learning purposes. For this thesis, a dataset of labelled railway scene images is probably most relevant, called RailSem19 [42]. This dataset consists of 8500 images of both geometrically defined bounding boxes of important railway objects such as switches and rail traffic signs and pixel-based labels on everything that is visible in the images. For this thesis, the geometrically defined “important”

[2]At the time of writing, no labelled dataset for rails point cloud exists.

[3]This seems to not be an issue for regular machine learning, since there the models use each point in the point cloud separately, whereas in deep learning the complete point cloud is used as a sample instead and the idea is to let the network detect patterns.

[4]Which can be partly remedied by using a data structure such as octrees.

labels are not useful, because their labels are too specific for Fugro purposes and their bounding boxes are not tight. Their labels are also included more generally in the pixel-based masks anyway. See the figures in appendix A for more clarity.

The LDLS approach works as follows [37]. As mentioned before, it needs annotated data of relevant images. The method then assumes that a Mask-R-CNN model is trained to learn to draw labelled mask on the test section of that data. Mask-R-CNN is a convolutional neural network approach that can be used to train models to be able to draw object instance segmentation masks on images [11]. Although this method was developed and published in 2017, it does not hold up to contemporary methods, such as models in the frameworks of Detectron2 [39] and MMDetection [5], both of which were first published in 2019. Using an image semantic segmentation model, the approach then infers the models on the desired images. At the same time, an undirected graph is constructed with nodes corresponding to the point cloud points and nodes corresponding to the image pixels. The total graph G consists of 2 types of nodes, namely the 2D pixel nodes and the 3D point cloud point nodes, both of which can be labelled with the semantic labels from the masking model. There are also 2 types of edges: edges between the 3D nodes and edges between a 2D node and a 3D node. Initially, all 3D nodes are unlabelled, and the 2D nodes get the labels that correspond with the inference labels of the masking model on the images. These subgraphs are denoted $G^{2D\to 3D}$ and $G^{3D\to 3D}$ for the 2D to 3D subgraph and 3D to 3D subgraph respectively.

In order to create the 2D to 3D edges, all 3D points are projected to the 2D image, assuming we know the relative position of the image to the point cloud. For details about this projection, see Section 2.4.3. In order to combat calibration errors and errors due to the lack of depth in the 2D segmentation masks, every 3D node is edge-connected with all the pixel nodes that lie in a 5 pixel by 5 pixel box around its 2D projection. For the 3D to 3D edges, simply a nearest-neighbour graph is constructed. All the weights of both types of existing edges are initialized in a certain way, and the weighted adjacency matrices of both parts of the graph are put together in one, with G represented by Equation 2. This makes the label diffusion easier later on.

$$G = \begin{bmatrix} G^{3D\to 3D} & G^{2D\to 3D} \\ 0 & I \end{bmatrix} \tag{2}$$

Now the initial graph is set up, the labels can be diffused from the 2D nodes to the 3D nodes. The diffusion itself is actually quite elegant and simple. We define $z^{(m)} \in \mathbb{R}$ as the label vector for label m for all points and pixels, so it has the same length as the width of the complete matrix for G. As $z^{(i)}$ is defined to hold the information for one label i for all points, it is initialized for all 3D points as 0 and a 1 if a 2D pixel holds that class label. Then, the only operation needed to diffuse the class labels of the 2D pixels to the 3D points is given in Equation 3. This is done consecutively until all vectors z converge. It can be proven that they do converge [37]. The way that G is constructed ensures that only the 3D node labels are changed after each multiplication.

$$z^{(m)} \leftarrow G \times z^{(m)} \tag{3}$$

Since the algorithm so far is still prone to some errors due to the projection of background 3D point cloud points to the front in a segmentation mask, an outlier removal substep is executed in the end. For all class labels the largest connected component is taken, and all points that have the same class label but are not part of the connected component have their class label set to the default 0 again.

2.4.2 Semantic image segmentation with Detectron2

In the area of image segmentation, Facebooks Detectron2 claims to be a state-of-the-art next generation library [39]. It provides various types of image segmentation and detection networks.

For this thesis in particular, the image segmentation models are useful. There is a distinction between 3 types of semantic image segmentation: instance segmentation, semantic segmentation[5] and very recently also panoptic segmentation [15].

Instance segmentation is the segmentation task in which countable, distinguishable *things*, such as people, cars or road signs are segmented. In particular, this segmentation task seeks to detect each of these objects and label them separately. Labels will typically consist of a set (l_i, z_i) denoting both the semantic label $l_i \in \mathcal{L}^{th}$ and the instance $z_i \in \mathbb{N}$, where $\mathcal{L}^{th}$ contains all possible thing labels, including a void label for pixels that do not belong to a thing. For pixels belonging to the same object, both l_i and z_i need to be identical. Note that instance segmentation only looks at a select number of semantic labels, thus generally labelling a tiny subset of pixels in an image, since most images will not completely consist of things.

Semantic segmentation on the other hand is a segmentation task in which all *stuff* is distinguished; stuff being all different, amorphous entities visible in an image, e.g. sky, road, railway and vegetation. As such, semantic segmentation seeks to assign a label to each pixel in an image, without distinguishing between different objects of the same label. Semantic segmentation produces only one value per pixel, a label $l_i \in \mathcal{L}^{st}$. $\mathcal{L}^{st}$ denotes all defined stuff labels.

In 2019 panoptic segmentation was introduced to describe the segmentation task that attempts to combine both instance segmentation and semantic segmentation [15]. This segmentation task both labels every pixel of an image and distinguishes between different, countable objects. Pixels get assigned a pair of values (l_i, z_i) similar to instance segmentation. However, if $l_i\ in \mathcal{L}^{st}$, the value of z_i is simply disregarded. The sets of labels $\mathcal{L}^{st}$ and $\mathcal{L}^{th}$ must thus be mutually exclusive and collectively exhaustive.

Instance segmentation is not sufficient for this thesis, as the goal is to classify the complete point cloud, without the need of distinguishing between different object instances.

The Detectron2 framework used in this thesis does not provide a baseline semantic segmentation network, so in order to still do semantic segmentation on the video images, a panoptic segmentation model is used instead. The chosen model is the COCO Panoptic Segmentation Baseline R101-FPN model, since it achieves the highest accuracies and the difference in inference time with other models is negligible. This model has a ResNet backbone [12], which means that the first 101 layers have a ResNet architecture. ResNet is a very successful neural network architecture that manages to create deeper neural networks without degradation of the backpropagation gradient.

2.4.3 3D to 2D point projection

LDLs uses a projection of the 3D point cloud points onto the 2D images to create an adjacency matrix that is used for further processing. Creating this projection is done using intrinsic and extrinsic calibration parameters of the cameras. Intrinsic parameters are internal camera parameters such as camera focal length and distortion, external parameters include the translation and rotation of the camera compared to the calibration value [34].

Transforming 3-dimensional spatial coordinates to 2-dimensional pixel coordinates is done as follows [34]. The image position (x, y) of the 3-dimensional point (X, Y, Z) is given by the transformation in Equation 4. f denotes the focal length of the point, which is the distance from the point to the camera plane. This equation only holds in an ideal case, i.e. a camera without any internal or external distortions.

$$\begin{pmatrix} x \\ y \\ f \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{4}$$

In order to truly transform from the 3D point to image, we define three coordinate frames: the world coordinate frame $\vec{X}_w$, the camera coordinate frame $\vec{X}_c$ and the image coordinate frame

[5]The nomenclature of this type in particular is very confusing.

$\vec{p}$. The first two use meters as unit, the image coordinate frame uses pixels. The transformation from $\vec{X}_w$ to $\vec{X}_c$ is calculated as in Equation 5. M_{ex} is the 3 by 4 extrinsic calibration matrix $M_{ex} = (R \quad -R\vec{d}_w)$, where R is the 3 by 3 rotation matrix. $\vec{d}_w$ is the location of the center of the camera in world coordinates.

$$\vec{X}_c = M_{ex}[\vec{X}_w^T, 1]^T \tag{5}$$

Then, X_c, which is still in world coordinates, is transformed to camera coordinates as in Equation 6.

$$\vec{x}_c = \frac{f}{X_{3,c}}\vec{X}_c \tag{6}$$

Finally, the perspective corrected camera coordinates $\vec{x}_c$ are transformed to pixel coordinates $\vec{p}$ by Equation 7.

$$\vec{p} = \frac{1}{f}M_{in}\vec{x}_c \tag{7}$$

Here M_{in} is the intrinsic calibration matrix defined as in Equation 8. These parameters are camera specific. f_{s_x} and f_{x_y} denote the pixel focal length. o_x and o_y denote the offset from the top-left corner of the pixel coordinates compared to the camera coordinate center.

$$M_{in} = \begin{pmatrix} f_{s_x} & 0 & o_x \\ 0 & f_{s_y} & o_y \\ 0 & 0 & 1 \end{pmatrix} \tag{8}$$

Using all three transformations in series, starting with the 3D points, yields the pixel coordinates. Slight variations of these formulas exist, to accommodate a linear transformation, whereas the equations given above give a nonlinear transformation.

3 Methodology

In this section we will discuss the approach to the experiments that is needed to reproduce the results that are presented in Section 4. This approach mainly consists of two steps: using Detectron2 to semantically segment the video frames and using LDLS to diffuse the labels generated by Detectron2 to more accurately label points. Additionally we describe how some measure of ground truth is generated from the Fugro maps, in order to be able to do a limited quantitative analysis.

Unless otherwise mentioned, all code used and written for this thesis can be found at `https://git.science.uu.nl/p.h.zwietering/thesis-code`.

3.1 Video segmentation

Completing the first step of the LDLS algorithm means segmenting the video data corresponding to the point cloud. While technically it does not matter what algorithm is deployed to achieve a segmentation, it is highly recommended to use a trainable neural network, since their results outclass all traditional non-learning approaches [24]. The authors of LDLS themselves use Mask-R-CNN to segment their images. This network is already outdated as of this thesis; among others, Detectron2, MMDetection and certain models present in PyTorch report better results.

At first we tried to transfer learn the relevant, differing labels of the RailSem19 dataset onto pre-trained Deeplabv3 models, both using the Detectron2 framework and PyTorch. Both of these attempts were futile, the former because of the complexity of the framework, the latter because of failing code and completely different labelling structure. The model also had very high hardware demands on both frameworks, which meant cloud computing was used in the form of Google Colab. After failing to get this to work for 2 months, we decided it would be better to use a pre-trained model instead.

For this purpose we used the pre-trained COCO panoptic R101-FPN model available on Detectron2. This model is trained on panoptic COCO data for around 37 epochs. The R101-FPN part indicates that this model uses a ResNet-101 + FPN backbone and is said to have the best trade-off between accuracy and speed [39]. More information on specific training parameters and model architecture can be found at `https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md`.

To achieve segmented images, inference is run on this model immediately. Only the semantic segmentation part is selected as output, as instance segmentation is not relevant for this research. This creates a bitmask for the whole image, which is transformed into decimal label values for all pixels. Results for just this step can be produced and are shown in Section 4.1.

3.2 Point cloud projection

To compute the point projection, we apply the theory from Section 2.4.3. The intrinsic and extrinsic parameters are extracted from the JSON calibration files. The base position of the camera per video frame can also be extracted. With this information, all point cloud tiles and video frames can be aligned and each point projected on the image. For this we use the `projectPoints` method from the OpenCV library in Python [3].

Of course, the combined point cloud for the complete railway track is not regarded for each single video frame. This would give very skewed results. Thus, only a subset of points is taken into account. What points are selected depends on which video frame is taken, not the other way around. Because of this, the number of video frames necessary to create a contiguous view of the environment depends on how many points are clipped.

The first clipping step is to exclusively take into account all nine point cloud tiles around the tile the point lies in and the tile the point lies in itself. So the maximum viewing distance as seen from the camera viewpoint will be $2\sqrt{2 \cdot 25^2} \approx 70$ meters if the tiles are 25 x 25 square meters. Then, all points that are behind the plane of the camera viewpoint are clipped, by calculating for every point at which side of that plane they. If the points in the viewpoint plane are (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3), the plane through the camera viewpoint is calculated by solving the system of equations in Equation 9 for a, b and c. d can be chosen freely.

$$\begin{aligned} x_1 \cdot a + y_1 \cdot b + z_1 \cdot c = d \\ x_2 \cdot a + y_2 \cdot b + z_2 \cdot c = d \\ x_3 \cdot a + y_3 \cdot b + z_3 \cdot c = d \end{aligned} \tag{9}$$

Given a plane defined by $(a, b, c, d)^T$, we calculate for each point $(x, y, z)^T$ if $a \cdot x + b \cdot y + c \cdot z + d > 0$. All points that do not satisfy this condition, are thrown away.

The last clipping step is optional and not initially used for creating results. During the research, we discovered that reducing the viewing distance to a much smaller radius is quite influential on the accuracy of the results. This will be discussed in more depth in later sections. Clipping points based on distance from the camera viewpoint can be done by calculating the Euclidean distance between the camera viewpoint and all point clouds points and checking if the distance is lower than some certain threshold.

As an intermediary result, giving the point cloud points the label that belongs to the image pixel that has the same location as the projection of the point is already useful. Results for this will be shown compared to results that were created using LDLS afterwards as well.

3.3 LDLS methodology

LDLS uses the projection method from Section 3.2 as input, in the form of a per pixel bitmask of the labels that are assigned to that pixel by the image classification model, together with the point cloud. We then simply run the LDLS algorithm, i.e. matrix multiplication, until the bitmasks for the point cloud nodes converges. The code for LDLS can be found at `https://github.com/brian-h-wang/LDLS`. Only minor modifications were made in order to smoothly transition that code from Mask-R-CNN to Detectron2 and the OpenCV projection method.

Because LDLS performs a lot of sparse matrix multiplications, we advise to use GPU parallelization in the form of CUDA to be able to do these computations very quickly. We performed these experiments on a Nvidia RTX 3070 GPU. Although this is quite a modern GPU with 8 GB of VRAM, the matrix multiplications were sometimes too big to fit in its memory. Through some empirical testing, we found that approximately 300,000 points fit without crashing the GPU. Consequently, we had to pre-segment the point cloud tiles in order to do the LDLS segmentation. This was done by splitting the point clouds into pieces of at most 300,000 points, in the order that they appeared. Because of the way the points are saved, these segments will have the form of axis-aligned rectangular strips. See Figure 10 for clarification.

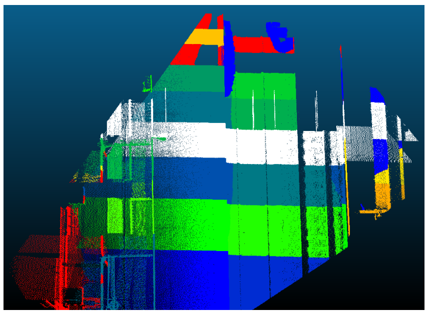


Figure 10: Top overview of point cloud strips for a certain piece of point cloud.

3.4 Ground truth extraction

For quantitative analysis of point cloud segmentation, it is common to perform a per-point comparison of semantic labels of the ground truth with the output of the methodology, resulting in a $k \times k$ confusion matrix for all k labels. Because of client contracts, Fugro was not allowed to share their completely checked, high quality semantically labelled point clouds. Instead, we had to create ground truth from their maps. An example of what those look like can be seen in Figure 7. These maps consist of geometrical shapes that indicate the outlines of objects on and around the railway track. The outline of some objects are only given for the part closest to the track, such as for station platforms.

Since the panoptic image segmentation model was not specifically trained on railway data and labels, it has limited labels that are interesting to analyze for a railway context. The only two railway specific labels are "railway" and "platform". Because the railway label encompasses multiple objects that are separately indicated in the map, we are unable to do an accurate analysis on that label. The platform label however is distinct for both the map and the segmentation model.

In the map, platforms are indicated by sets of pairwise connected line segments. Only the part of the platform closest to the railway track is indicated. So, we can only extract a part of the points that belong to the platform, since we do not know how far the platform extend away from the track. In order to be sure that we collect the right points, we will assign points to be platform points in the following way. For every line segment l of platform, we define a cylinder with its axis equal to l and a radius of 0.8 meters. This is because in England, platforms should always have a height of 0.915 meters. This avoids collecting points that belong to the track or ballast. The downside is that platforms usually extend more than this distance away from the track, and we do not collect those points. However, in order to recreate the platform line segments in the map, we just need this outlined part anyway.

For all line segments l with start point c_1 and end point c_2, checking if a point p lies in the cylinder defined as a cylinder with its axis equal to l and radius r goes like this. First we check if p lies between the two parallel planes that go through the disks of the top and bottom of the cylinder, in a way that is similar to the plane checking necessary for projecting points in Section 3.2. Then p is checked on the distance from the line through l; if it is at most equal to r, p lies inside of the cylinder. This is calculated as in Equation 10.

$$|(\vec{p} - \vec{c_1}) \times (\vec{c_2} - \vec{c_1})| \leq r \cdot |\vec{c_2} - \vec{c_1}| \tag{10}$$

3.5 Time measurements

In order to be able to compare the LDLS method with just projection, we have used the Python built-in `time` module. For each point cloud tile, we start a timer after the loading of the points and stop the timer before the result is written to disk again. This ensures we only measure the execution times of just the algorithmic parts of these methods.

4 Results and evaluation

In this section we will show qualitative results for the inference of Detectron2 on video frames of various railway scenes. We will also show qualitative results of both the projection method and LDLS. Then we will show the limited quantitative analysis we have been able to do, together with running times of different methods on various sizes of point cloud tiles. For all quantitative results, we will show the interesting results in this section. Additional images, giving a more general overview of the results, will be shown in the appendices.

4.1 Detectron2 panoptic inference

First, we will show just the inference output of the chosen panoptic segmentation model on Fugro camera data, together with the corresponding input images. After that, inference results will be shown of the model on RailSem19 images, together with the ground truth labelling of these images and the input images. The labelling of the panoptic segmentation model and the labelling used by RailSem19 does not have a 1-1 mapping, but it does give an impression of the usefulness and accuracy of the panoptic segmentation model.

4.1.1 Panoptic segmentation inference on Fugro video data

The output of Detectron2 gives coloured planes on top of the input images, where each colour corresponds to a certain label. The label is shown at least once per present label in the image. Because of the interest in semantic segmentation, all instance segmentation labels are disregarded and shown as a "things" label instead.

The images in Figures 11, 12 and 13, show the inference of the panoptic segmentation model on all 3 camera views in the busiest part along the track: the only train station available in the Fugro dataset. A lot of different objects and structures are visible in these images, but the model still manages to give a fairly nice rough segmentation.

There are two main remarks regarding these images. First, for the inference of the third camera viewpoint, the platform is not labelled as platform, but instead it is described as both pavement and wall. While technically this is correct, this is not desirable. For the inference of other images before and after this particular viewpoint, this outcome is similar. We think this is because of the short distance between the camera and the platform and the lack of railway in the image, which the model might need in the picture to understand that the image is taken in a railway environment and thus assign platform labels more easily.

Second, and this is behaviour that holds for all inference images, the quality of the segmentation seems to degrade the further an object or point is placed away from the camera viewpoint. This is most visible in the second camera viewpoint, where we can see far in the distance along the track. We can see that a lot of the area above the railway tracks is erroneously labelled as railway, while it is usually something else, such as wiring or wayside objects.

We have also included a few pairs of images to illustrate the kinds of situations where the segmentation model does seem to have a more significant problem to generate a proper labelling. These are shown in Figures 14 and 15. Figure 14 shows the view of the second camera on a railway crossing, with on the side of the track a big electrical box. Because of the ambiguity of what this electrical box looks like and the presence of a warning sign on the box, it gets a very inaccurate combination of a lot of small sized labelled areas. While some of these labels would not be very incorrect, the fact that the model designates all of those in small quantities makes this behaviour undesirable.

Figure 15 shows the view of camera 1 on a fence in front of the ramp of a bridge that has just been passed. While the segmentation model correctly recognizes the presence of the fence, it is not segmented completely as we would expect. In addition, the model incorrectly classifies the ramp of the bridge as a mountain.

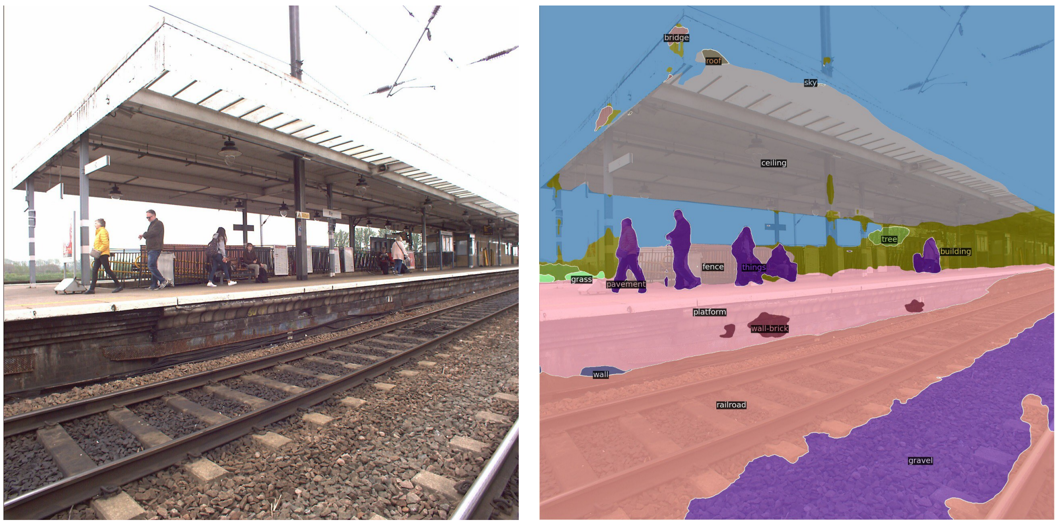


Figure 11: View of camera 1 of part of the station with Detectron2 inference.

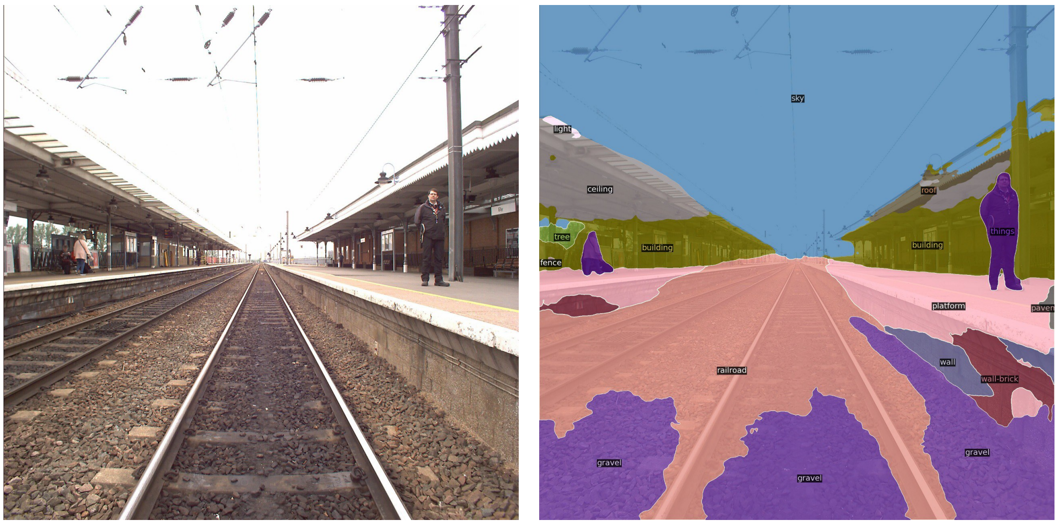


Figure 12: View of camera 2 of part of the station with Detectron2 inference.

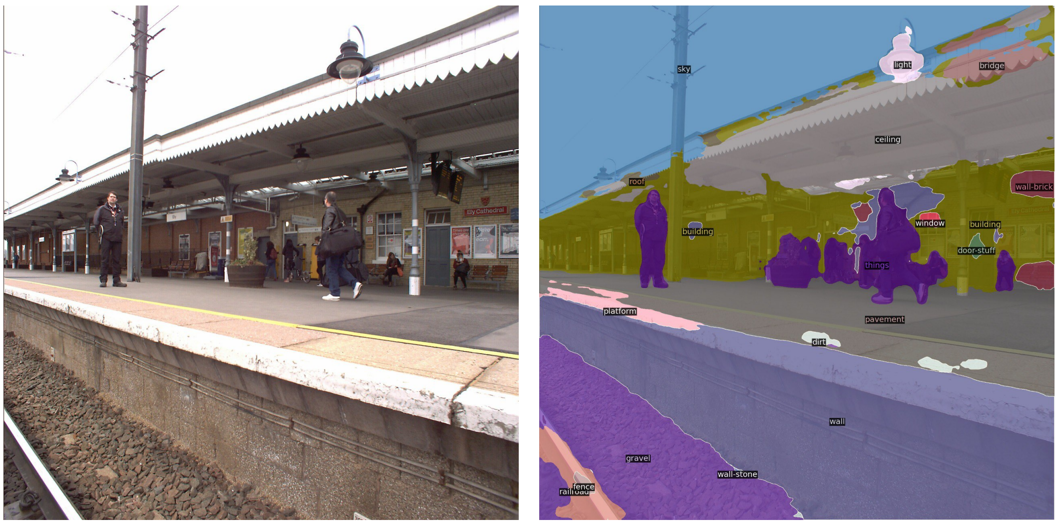


Figure 13: View of camera 3 of part of the station with Detectron2 inference.

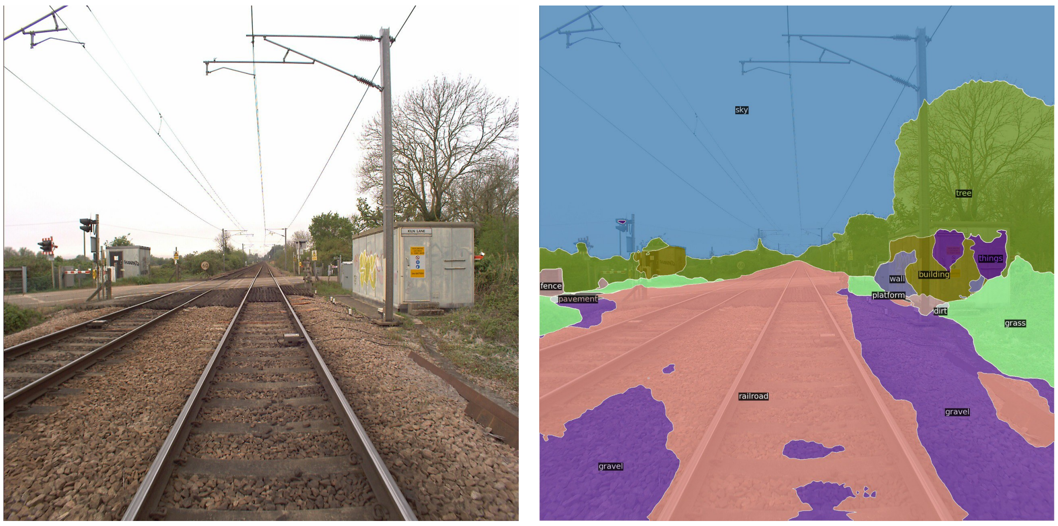


Figure 14: Camera image and inference of part of the track with a big electrical box.

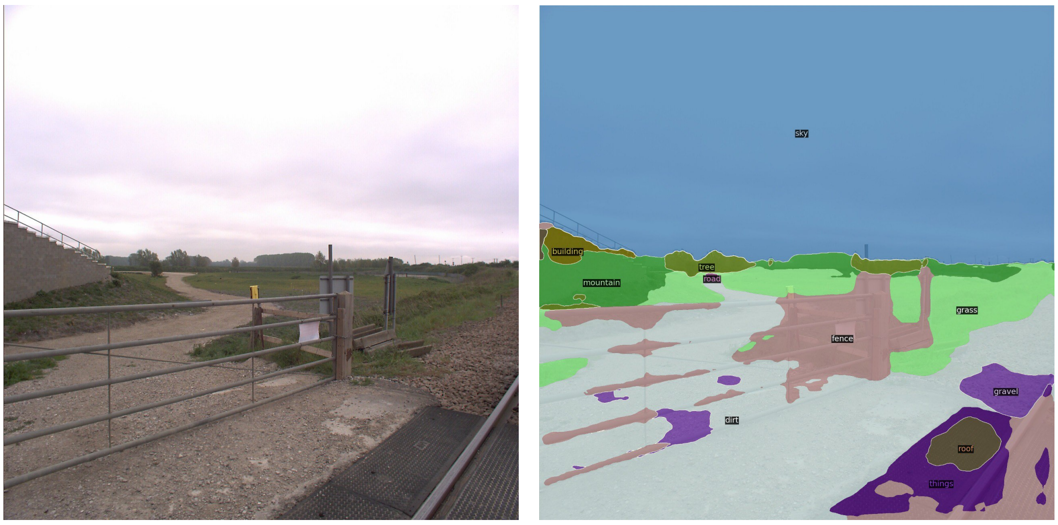
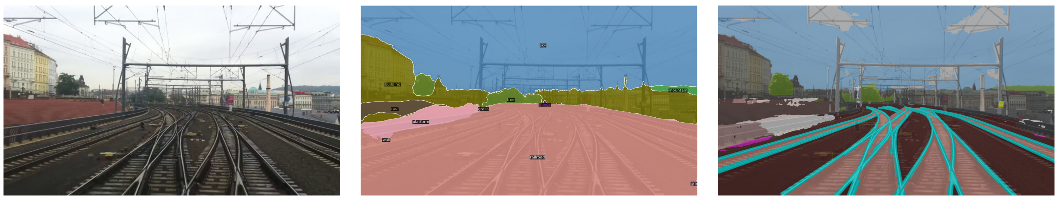


Figure 15: Camera image and inference of part of the track with a fence and somewhat irregular background.

4.1.2 Panoptic segmentation inference on RailSem19 data

The results of the panoptic segmentation of the model on RailSem19 image data has the same format as for the Fugro data. For reference to its ground truth, we also included the ground truth semantic segmentation as labelled by the creators of RailSem19. The semantic labels differ between these two data sources, but the outlines of the segments can be roughly qualitatively compared to get an idea what the strengths and weaknesses of the panoptic segmentation model are. For each result, we include the original photo taken, coupled with the ground truth of RailSem19 and the panoptic segmentation inferred by the Detectron2 model.

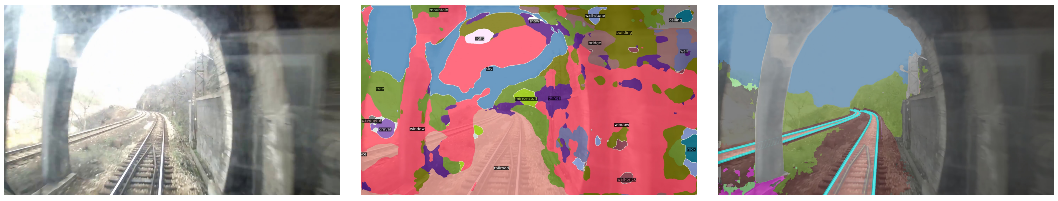
Figure 16 shows the inference output of a very wide part of railway with multiple tracks. In the RailSem19 dataset, all those tracks are neatly outlined and distinguished from the gravel and sleepers. This is not the case for the panoptic segmentation model, where only the complete area of the railway is detected and labelled as such. Compared with the ground truth, the model does seem to separate the different entities in the background fairly well.



(a) Input image. (b) Inference by model. (c) Ground truth.

Figure 16: Comparison of RailSem19 labelling and panoptic segmentation model of a clear railway scene with multiple of tracks.

RailSem19 has the advantage of having a varied number of railway scenarios, such as in Figures 16, 17 and 18. In Figure 17, we can see that the panoptic segmentation completely breaks down. The glare of the light in the camera probably makes the problem more severe. But this is a clear failure of the model, which might suggest it is overfitted or not trained on enough different railway scenes. It is especially strange that the model does not even recognize sky or the railway itself.

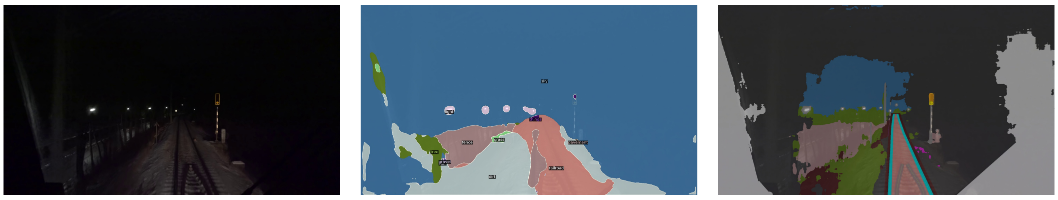


(a) Input image. (b) Inference by model. (c) Ground truth.

Figure 17: Comparison of RailSem19 labelling and panoptic segmentation model of a tunnel railway scene.

Another important variation are the weather conditions and time of day. In Figure 18, a nightly railway scene is tested. The segmentation model recognizes the darkness as sky and still manages to recognize a part of the railway. In contrast with the previous highlighted images, where a tunnel completely broke the segmentation, this was a surprising result.

Further results can be seen in appendix B for the inference on Fugro video data and appendix A for the inference on RailSem19 images. The Fugro video images are put in as reference and to show that the model mostly seems to hold up fairly well on this specific type of railway scenes. For the RailSem19 data, we have tried to put in more unique railway scenarios, to try and show whether the panoptic segmentation model breaks down or not in those situations.



(a) Input image. (b) Inference by model. (c) Ground truth.

Figure 18: Comparison of RailSem19 labelling and panoptic segmentation model of a railway scene at night.

4.2 Point projection

In order to demonstrate at least to some extent the correctness of the point projection step, we show multiple images of the point projection of the Fugro railway scenes. All points in a circle with a radius of 20 meters and a center at the camera viewpoint are taken and processed such as explained in the methodology in Section 3. These are then projected onto the image as a very small circle with OpenCV. One of these results can be see in Figure 19. This image shows a Fugro railway scene from the front facing camera. The lack of points at the white sign is also very noticable, except for the pole that it is attached to, which is probably thicker than the sign itself and therefore registered better. When looking at the catenary wires, we can see that on the right side of the image the points do not seem to lie exactly on the right place in the image. This highlights the importance for LDLS to not just take the pixels the points are projected on, but a square area of pixels around the projected point instead.

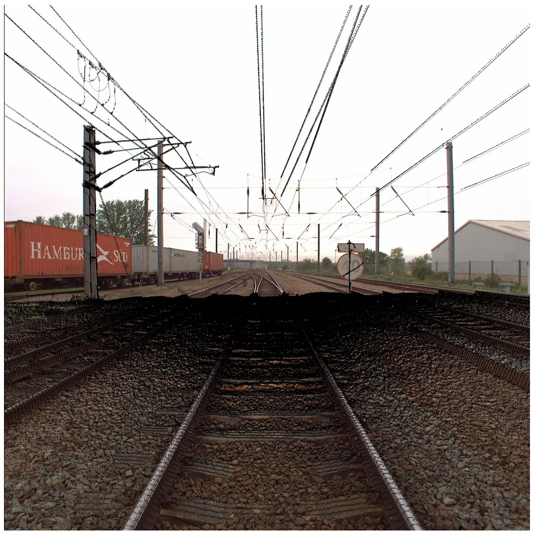


Figure 19: Projected point cloud points on input image.

4.3 LDLS versus labelled point projection

We will show multiple views of parts of the processed point cloud, and, if applicable, give a corresponding camera view as well. The colouring is based on the Detectron2 default colouring. A label legend is included in Figure 20. In order to give the exact same views for all different experiments, we use viewpoints in the free point cloud program CloudCompare, which gives us the ability to save the point of view and use it to view different point clouds from the same place. The results with short viewing distance of 11 meters are based on the combined point clouds outputs of all 3 camera views, whereas the results with a long viewing distance of upwards to 70 meters are based on only the point cloud output of the second, front facing camera.

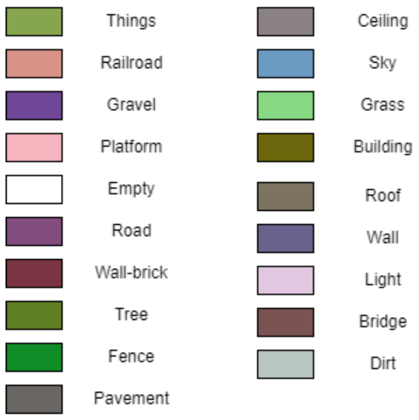


Figure 20: Point cloud label legend

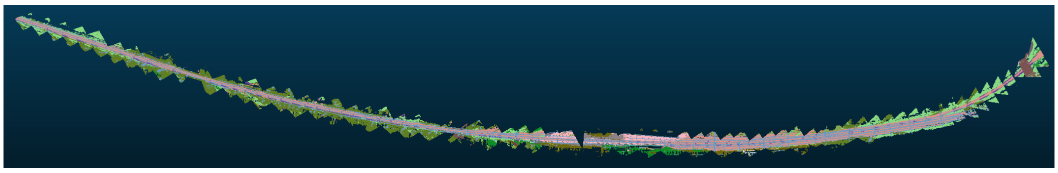
First, we give an top down overview of a stitched together set of point clouds, together forming a complete overview of the second run of the railway track in Figure 21. It is impossible to make out any details from this point of view, but it gives a rough overview of what labels are dominating in each experimental setup. What is very striking is that the LDLS methodology gives a lot of empty labelled points, regardless of viewing distance. We can also see that the experiments with high viewing distance give a broader view of the environment around the railway track, which is expected. The labels appear to match pretty well for the two projection experiments, apart from maybe a bit more gravel labels in the short range experiment.

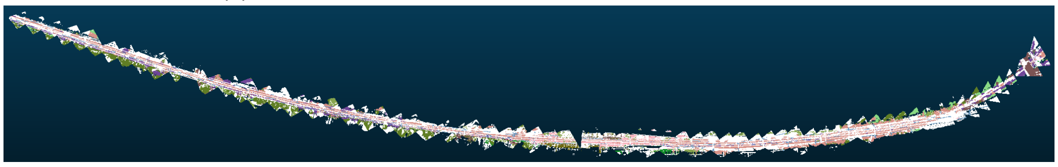
In Figure 22 we included a view of the train station from the viewpoint of the camera such as shown in the first image. We can see that the difference between both projection based results is small compared to the difference between the LDLS based results. The biggest visible difference between the projection results is the number of ground points classified as gravel, the number of platform points classified as wall-brick and the difference in classification of the station buildings. For the LDLS method it appears that the viewing distance made a higher visible difference. From this point of view, the high range experiment has yielded a significantly higher number of labelled points than the low range experiment. Also, less ground points are labelled as gravel in the high range experiment.

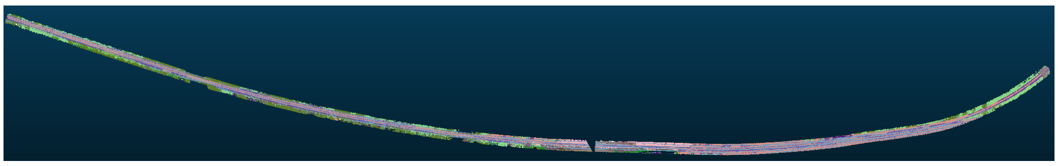
In Figure 23, we show point clouds of a rural part along the track. The differences between the two projection methods are again minimal, except for the number of points labelled as gravel. We can also see that the vegetation appears to be well segmented. The LDLS point clouds are segmented really poorly however. A lot of points are labelled as empty, and the point cloud made with low viewing distance has a lot of the vegetation labelled as gravel and only a small part of the railway actually labelled as railroad. The long range LDLS point cloud does appear to keep more of the vegetation point cloud points in and labelled as such.

In Figure 24, we show the same fence again as in Figure 15, but now with corresponding point clouds. We only ran the shorter range experiments on the side cameras, so there are no long range experiments we can show for this camera image. We can see that the errors of the segmentation model are propagated into the point cloud obtained with the projection method. The fence in the point cloud made with LDLS is labelled as empty.

In general, LDLS does not function well in large connected areas of point cloud. Either the algorithm gives a lot of points an empty label or it pushes one label to many points. We do not know why it gives so many empty points, since the initial state of the LDLS algorithm should be the same as the projection outcome. The only reason we can think of, is that since there can be many points with a lot of labels assigned to them, with each a percentage that is too low for a certain threshold, none of the labels are returned as output in the end. This could also


(a) Projection method, viewing distance of up to 70 meters.


(b) LDLS method, viewing distance of up to 70 meters.


(c) Projection method, viewing distance of 11 meters.

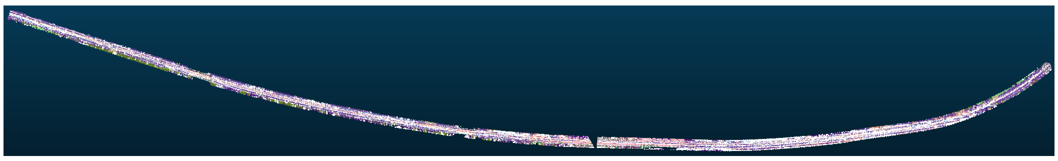
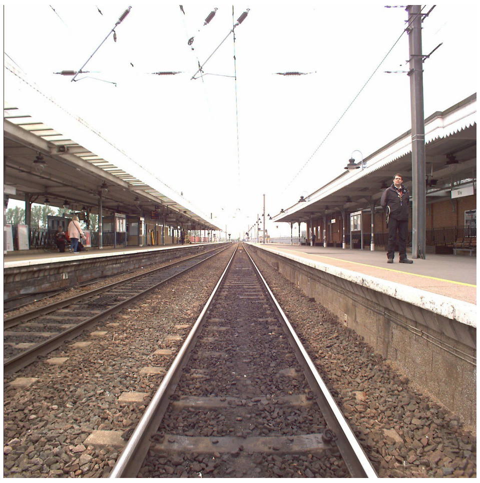
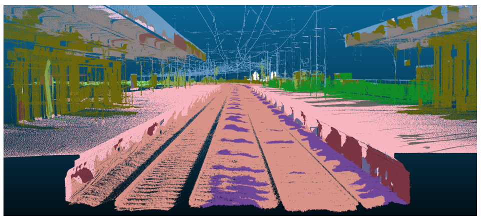
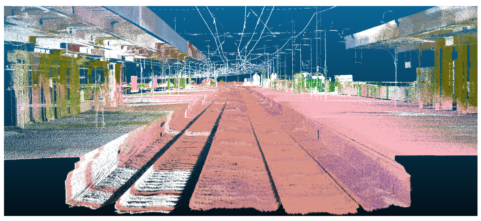
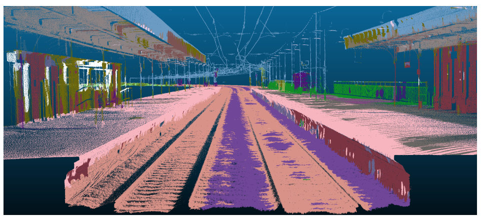

(d) LDLS method, viewing distance of 11 meters.

Figure 21: Top down overview shots of the different experiments. These were created by simultaneously showing the point cloud results of many processed images.


(a) Corresponding front facing camera image.


(b) Projection method, viewing distance of up to 70 meters.


(c) LDLS method, viewing distance of up to 70 meters.


(d) Projection method, viewing distance of 11 meters.

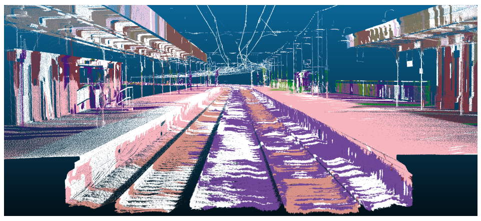
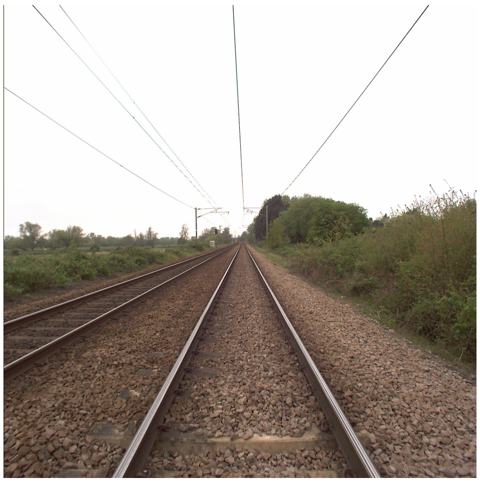
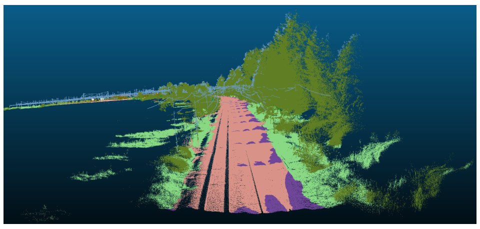
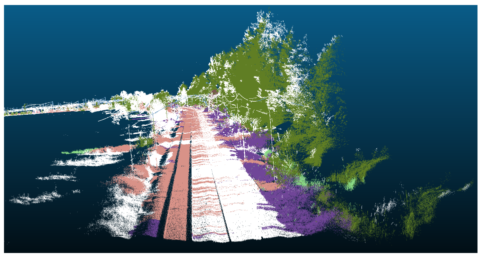
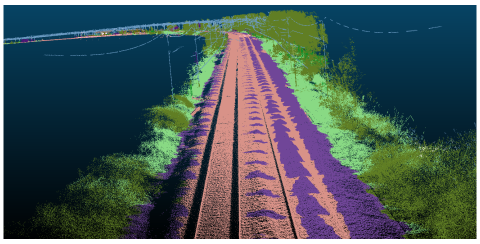

(e) LDLS method, viewing distance of 11 meters.

Figure 22: Labelled point clouds and corresponding camera image of the train station along the track, obtained through various methods.


(a) Corresponding front facing camera image.


(b) Projection method, viewing distance of up to 70 meters.


(c) LDLS method, viewing distance of up to 70 meters.


(d) Projection method, viewing distance of 11 meters.

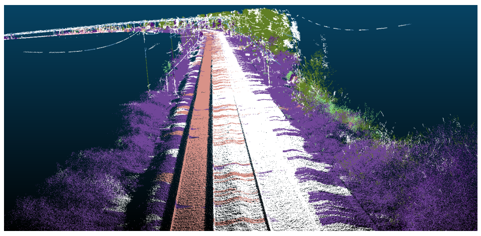
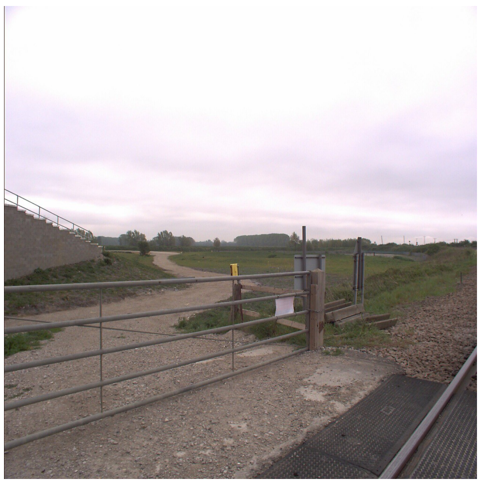
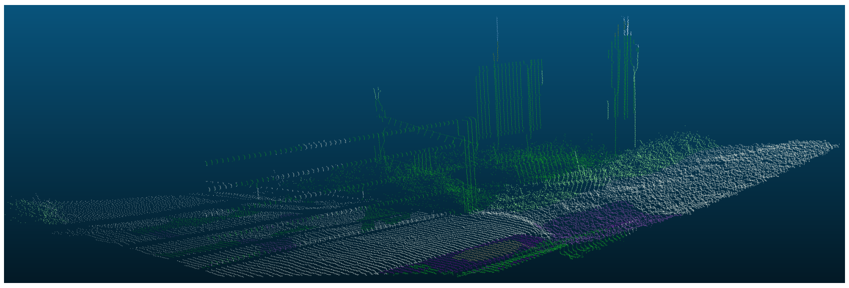

(e) LDLS method, viewing distance of 11 meters.

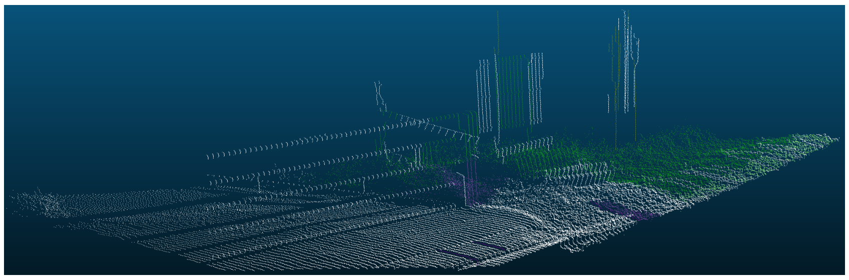
Figure 23: Labelled point clouds and corresponding camera image of a rural part of the track, obtained through various methods.



(a) Corresponding left facing camera image.



(b) Projection method, viewing distance of 11 meters.



(c) LDLS method, viewing distance of 11 meters.

Figure 24: Point cloud points belonging to the fence picture from Figure 15.

explain why disconnected areas with singular labels remain unchanged by the LDLS algorithm. For example, in Figure 23, the left track bed, which is disconnected from the rest of the point cloud, is still completely classified as railway, whereas the right track is not. This holds for both the short and long range experiments. However, this does not explain why the gravel label has such a high prevalence compared to all other labels.

We also think that the way the strips of point cloud are selected in order to fit them in VRAM for the LDLS algorithm might affect the outcome of the algorithm. This is most notable in big point clouds, when the point clouds need to be reduced in size the most. For example in Figure 38 in the appendix, we can see a few horizontal strips perpendicular to the track, which can indicate that the way we choose these strips is an issue.

More results can be seen in appendix B and videos of the results on `https://git.science.uu.nl/p.h.zwietering/thesis-code`, which are a much more suitable medium than images to show generated point clouds in our experience.

4.4 Label accuracy for platform points

The platform label is one of the two most relevant semantic labels that the panoptic segmentation model can distinguish for a railway environment. The other label, railway, is not as easily extracted from the Fugro ground truth maps as platform is, which is why the quantitative analysis was only performed for the platform label, as described in Section 3.4. When only those points are selected, we get two long strips of points in the shape of a right angle, opposite of each other and mirrored in the track. A top down overview of these points is shown in Figure 25.

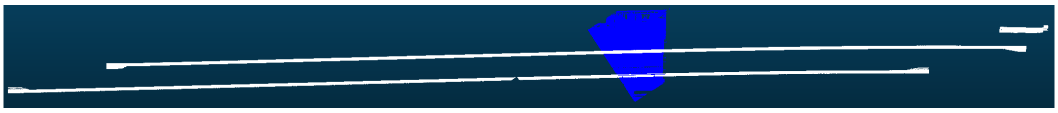


Figure 25: Top down overview of platform points in white, extracted by checking cylinder inclusion along line segments defined in the ground truth Fugro map. In blue, one tile of train station with the most points of the track is included as reference to the length of the platforms.

Table 1 contains percentages of the occurrence of each of the mentioned labels per type of experiment for the extracted platform points. We did four experiments, in which we tested projection and LDLS, both for a range of up to two tiles (radius of around 70 meters) and a radius of 11 meters around the camera viewpoint. From the table, we can see that just using the projection with a range of 11 meters has the highest accuracy of these four experiments, with an accuracy of 84 percent. Using projection with the extended range has the worst performance, with an accuracy of 36 percent. We can see that for both LDLS and the projection method with viewing distance, the railroad label occurs very often. Apart from the empty label, the most occurring labels are wall-brick and gravel. Labels with fewer than 0.1 % occurrence in all experiments are excluded from these results.

4.5 Algorithm execution times

The execution times of the different experiments are shown in Table 2. These are measured per image, excluding reading and writing times of point cloud tiles. This was done for 122 images in total, spread over all runs and cameras. It is important to note that a smaller viewing distance also means less points are projected and put through LDLS. In order to properly compare the two different viewing ranges, we also included the average per-point processing times. These experiments were run on an AMD Ryzen 5600G CPU and NVIDIA GeForce RTX 3070 GPU with 32 GB of RAM. Unsurprisingly, the projection method with the lowest viewing distance has the lowest execution time per image, whereas the LDLS method for the larger point clouds is the slowest. Both methods have a high relative standard deviation, which means there is a

Label Occurrence (%)	Platform	Wall-Brick	Railroad	Gravel	Empty (LDLS)	Wall	Pavement
Projection, radius max. 70 meters	35.89	1.70	61.36	0.062	0.0	0.74	0.11
LDLS, radius max. 70 meters	49.43	0.077	34.97	4.98	8.37	0.001	0.0
Projection, radius 11 meters	84.36	8.58	2.44	0.31	0.0	3.82	0.48
LDLS, radius 11 meters	65.43	0.33	0.39	23.59	10.06	0.005	0.0

Table 1: Label occurrence of points with ground truth platform.

Experiment	Av. ex. time (s)	Av. ex. time per point (s)
Projection, radius max. 70 meters	10.52 ± 5.04	$6.77 \cdot 10^{-6} \pm 1.47 \cdot 10^{-6}$
LDLS, radius max. 70 meters	56.41 ± 39.08	$3.35 \cdot 10^{-5} \pm 3.69 \cdot 10^{-6}$
Projection, radius 11 meters	13.07 ± 7.32	$3.86 \cdot 10^{-5} \pm 8.94 \cdot 10^{-6}$
LDLS, radius 11 meters	25.61 ± 14.7	$7.53 \cdot 10^{-5} \pm 1.46 \cdot 10^{-5}$

Table 2: Average execution times per image for different experiments, spread over 122 images.

lot of variance in the actual run times. This is probably caused by the variance in point cloud sizes, since the relative variance in the per-point processing times is significantly lower.

5 Discussion

This section will discuss the general picture that rises from the current results and point out remarkable results and try to interpret them. We give a reflection of the process on this thesis. We will also give ideas that we think could be beneficial in future research or ideas that can be used to improve the current methodology.

5.1 Discussion on image segmentation methodology and results

Originally, we planned to use transfer learning on a Detectron2 or PyTorch pre-trained semantic segmentation model. This means we would take such a model and replace only the last layer with an untrained one, to then train that layer on new data that is more suitable for railway scenarios. The original plan was to train a network on RailSem19 data. Unfortunately, we repeatedly ran into problems while trying to get this transfer learning step to produce any meaningful results. One of the reasons for this was the lack of professional grade hardware that is necessary to train these large networks, but we also did not have enough experience or expertise in using these kinds of neural network libraries. It is important to point out that the Detectron2 framework is still under heavy development, so better support for transfer learning might be added in the future.

The results we did manage to get must therefore be taken into the context of not being the results of a segmentation model exclusively trained on railway related scenes. Considering that, the model actually manages to semantically segment the images fairly well, especially on short range. The effectiveness of the model is most noticeable when looking at the results of inference on the RailSem19 dataset, since we can actually compare the ground truth of the RailSem19 dataset qualitatively to these results. We can not do this quantitatively, because the labels of the different data sources do not match. We noticed the following things when looking through the inferred results.

First, vegetation and sky is generally segmented correctly. This probably has to do with the fact that nearly every picture taken outside has both vegetation and sky in it, and thus a large number of training images probably had these classes in it. Second, the railways are also recognized most of the time, with some exceptions being when the tracks are covered in snow, when the picture is taken inside of a tunnel or when the picture is taken at night time. For a model that has only two railways specific labels, we found this to be an encouraging result. It gives the confidence to state that were a more specialized segmentation model used, the results would have been significantly better.

The panoptic segmentation model also had some clear drawbacks. As mentioned briefly in the results section, the quality of the segmentation seems to drop drastically as the distance of objects to the camera viewpoint increases. When looking at all images, this is most noticeable when looking at the railway label. Usually this distance proportional mislabelling means all objects slightly above and around the track are labelled as railroad instead, which is also reflected in Table 1. Another issue with the segmentation model is that a more general issue with any kind of object detection or recognition model: it can not recognize that which it does not have any knowledge of. This is most apparent in Figure 14, where it fails to figure out what an electricity box with paintings all over it is.

5.2 Discussion on point cloud labelling results

Because of the nature of our chosen approaches, the strengths and weaknesses of the segmentation model will propagate to the labelling of the point clouds, most notably in the results created with just the point projection labelling. The LDLS algorithm skews the labelling so much, that these pros and cons are not directly visible any more.

In general, the projection labelling method performs well enough that most big objects at ground level appear to be segmented correctly from a visual standpoint. This is also seen when examining the quantitative results for the platform label, where the experiment with projection based labelling with a viewing distance of 11 meters has an accuracy of 84%, which is slightly higher than the minimum desired accuracy set by Fugro. Also from Table 1, it is evident that it is highly important to trim points based on their distance from the camera viewpoint. 61% of platform points are misclassified as railroad points when using a high clipping radius with the projection labelling, which has everything to do with the degradation of segmentation quality for points that are far away from the camera. LDLS performs too poorly for both clipping radii we tested by the standards set by Fugro. Both the short and long range experiment give an empty label percentage of around 10%, which is very disappointing. From the side view in Figures 22 and 23, we can see that mainly the ground points are labelled as empty by LDLS, which means that if we had picked the rail track points as our basis for quantitative analysis, LDLS would probably have performed much worse.

The point clouds that are created with LDLS look disappointing. Looking from above and from the side, a large number of points does not seem to be labelled. This makes it impossible to give a fair comparison between the projection labelling and LDLS labelling. We first thought this might be the result of too few iterations of the LDLS algorithm, but letting it run until convergence did not make any discerning difference in execution time or visual result, so these results are in fact achieved by the algorithm running until convergence already. The empty labels notwithstanding, visually the resulting point clouds look significantly worse than the projection ones, which was a very surprising result. We assumed at the start of this research that the LDLS methodology might improve the quality of the segmentation, but in fact it the segmentation became worse. What is also very interesting is that visually the LDLS with short range seems to be performing worse than with long range, which is in juxtaposition with the projection method. When looking in Figure 23, a lot of railway and vegetation is labelled as gravel, which we have no explanation for. It appears that the LDLS algorithm pushes out correct labels and replaces them with bad labels.

A drawback that is unique for the projection method (and not testable for LDLS), is the obstruction of view of objects behind other objects. As mentioned in the introduction, because of the used LiDAR system that emits laser beams in a direction perpendicular to the camera view, objects obstructed from view by the cameras will still be detected by the LiDAR system. In this research, we have not taken this into account with any countermeasure. Since the platform is never obstructed from view, this is not reflected in the limited quantitative analysis. We could not find any completely obstructed objects in the point clouds that are completely visible in the video footage, in order to visually check and see what our methods would result in.

5.3 Research recommendations

For future research, we recommend to have access to reliable and relevant ground truth, both for training a segmentation model and verifying the quality of a chosen methodology. Having no access to (relevant and usable) ground truth in this research caused that we could only make some educated guesses as to whether our approaches worked. In related work, numerous statistical analyses are commonly performed to give a solid understanding of the actual outcome of these segmentation algorithms. As such, the scientific value of this work is limited, since proof by picture is no proof at all. We hope this work can still inspire others to delve deeper into this topic and come up with new ways to improve segmentation techniques, since semantic point cloud segmentation is very much still an open research question.

Apart from that, if people are still interested in using LDLS, we would advise to try and vary other parameters of the algorithm, such as the diffusion parameter and the size of the box of pixels that is connected to each projected point cloud point. Another reason LDLS fails is the number of labels that we tried to segment all at once; it might be that LDLS is better suited for segmentation of one or two labels at a time, like the authors do. Because of time constraints, we were not able to perform analyses for these topics.

An issue that has been purposely overlooked so far, is the issue of multiple labels per point. Because these methods uses the video data as central starting point and point cloud points are selected based on their position relative to the camera position, many points are classified multiple times. For this thesis and these evaluation methods, this was not problematic, since we only look at how many points are classified correctly. Points that are classified multiple times are counted separately for every time they are classified. However, in the case of using one of these methods to create point clouds that are then used as a basis to make an end product, all points must have a single label. The label confidence in LDLS did not work, so even for that method some heuristic must be added to determine what labels to pick, especially if multiple labels have equal occurrences on a certain point.

6 Conclusion

This section will wrap up this thesis, by giving answers to the research questions and giving recommendations to Fugro about which direction they should go with this work.

6.1 Concluding research questions

This research has not proceeded as successful as we would have wished. We had to deviate from our original plan on some crucial points, and in addition we lacked access to ground truth, which has made it harder to give hard conclusions about the research questions or draw a conclusion in general. The main research question for this thesis is: How does LDLS perform in a railway environment? In order to split up the problem, we drafted a few subquestions as well. We will answer these now.

How well can Detectron2 be applied to achieve accurate semantic segmentation on railway video data?
For this question, we reviewed the results of a panoptic segmentation model on both RailSem19 data and Fugro video data. As discussed before, we can visually see that this particular model performs reasonably well on this data. It has clear drawbacks, mainly caused by environmental factors, a lack of domain-specific training and degradation of the segmentation quality for objects that have a higher distance away from the camera. We think that Detectron2 as a image recognition platform is definitely suitable to achieve accurate semantic segmentation on this type of railway video data, provided that domain-specific transfer learning is done.

How well does label diffusion perform when applied to point cloud data of Fugro railway data?
We qualitatively reviewed point cloud labelling for both the projection method and the LDLS method. This shows that LDLS drastically worsens the classification of points. This is reflected in the quantitative results for the platform points. LDLS mislabels 20 percent point more points than just projection in the short range experiments. The long range experiments are not accurate for both projection and LDLS. We therefore conclude that LDLS is not useful for application on the Fugro railway point cloud data.

What steps of the LDLS approach are most suitable for augmentation and how?
Due to time constraints, this research question has not been focused on as much as we would have liked. We have not been able to try and see if some internal functions of LDLS could be programmed better or done in a different way, to properly research their effects. However, the results of LDLS programmed in this way are significantly worse than without, in the best settings we could figure out, so the only answer we can give to this question, is that in our opinion LDLS as a whole is obsolete in this use case. We recommend that if there is still interest in using LDLS, to do more research in different strategies to subdivide point clouds for them to fit in memory and choosing the best label from the one LDLS suggests.

How does LDLS perform in a railway environment?
The answer to the main research question can be short. We think LDLS as a tool to classify or semantically segment point clouds is not useful. There can be multiple reasons why in our experimental setup LDLS failed, which are discussed in the previous section. We think that using just a projection of the labels obtained from a semantic segmentation neural network works good enough, is quicker and easier to use and maintain. The hardware requirements needed to run a segmentation model are minimal compared to LDLS, which needs a modern graphics processor to yield substantial results.

6.2 Recommendations for Fugro

In the introduction of this thesis, we listed the constraints Fugro gave for the desired functionality of the methodology. Considering these constraints and the current methodology in use at Fugro, we think that the use of a projection based approach can certainly be of service. The accuracy is not as high as the accuracy of the current Fugro methodology, but we think the projection method can be useful in classifying large, rural parts of the point cloud. There are fewer ill-defined objects and this way a large portion of the points in the point cloud can already be labelled, which also helps to generate a training subset for their feature based neural network. The cost of this method is very low, its speed is high and it is quite easy to work with.

We also recommend that if Fugro were to adopt such a projection method, either for preprocessing for their current methodology or as a complete replacement, it is necessary to transfer learn a semantic segmentation network on railway specific image or video data. This would probably significantly increase the accuracy of the semantic segmentation and thus the accuracy of the point cloud semantic segmentation. In order to do this, we also recommend that Fugro maintains a database of past point cloud classification results as ground truth, which can then be used to train different models in the future. We definitely believe that investing in well labelled training data can yield great benefits when wanting to pivot to semi-supervised approaches such as this one.

7 References

[1] Biosca, J. M., and Lerma, J. L. Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods. *ISPRS Journal of Photogrammetry and Remote Sensing 63*, 1 (2008), 84–98.

[2] Borrmann, D., Elseberg, J., Lingemann, K., and Nüchter, A. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research 2*, 2 (2011), 1–13.

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

[4] Che, E., Jung, J., and Olsen, M. J. Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review. *Sensors 19*, 4 (2019).

[5] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155* (2019).

[6] Cui, Y., Chen, R., Chu, W., Chen, L., Tian, D., Li, Y., and Cao, D. Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems* (2021).

[7] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255.

[8] Fan, T.-J., Medioni, G., and Nevatia, R. Segmented descriptions of 3-d surfaces. *IEEE Journal on Robotics and Automation 3*, 6 (1987), 527–538.

[9] Hähnel, D., Burgard, W., and Thrun, S. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems 44*, 1 (2003), 15–27.

[10] Hartigan, J. A., and Wong, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics) 28*, 1 (1979), 100–108.

[11] He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2961–2969.

[12] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* (2015).

[13] IBM. What is geospatial data? `https://www.ibm.com/topics/geospatial-data`, 2022.

[14] Kazhdan, M., Bolitho, M., and Hoppe, H. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (2006), vol. 7.

[15] Kirillov, A., He, K., Girshick, R., Rother, C., and Dollár, P. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 9404–9413.

[16] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., and Ferrari, V. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV* (2020).

[17] Lakshmi, S., Sankaranarayanan, D. V., et al. A study of edge detection techniques for segmentation computing approaches. *IJCA Special Issue on "Computer Aided Soft Computing Techniques for Imaging and Biomedical Applications" CASCT* (2010), 35–40.

[18] Li, J., Hu, Q., and Ai, M. Point cloud registration based on one-point ransac and scale-annealing biweight estimation. *IEEE Transactions on Geoscience and Remote Sensing* (2021).

[19] Li, L., Yang, F., Zhu, H., Li, D., Li, Y., and Tang, L. An improved ransac for 3d point cloud plane segmentation based on normal distribution transformation cells. *Remote Sensing 9*, 5 (2017), 433.

[20] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. Microsoft coco: Common objects in context, 2015.

[21] Lin, Y., Wang, C., Zhai, D., Li, W., and Li, J. Toward better boundary preserved supervoxel segmentation for 3d point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing 143* (2018), 39 – 47. ISPRS Journal of Photogrammetry and Remote Sensing Theme Issue "Point Cloud Processing".

[22] Maturana, D., and Scherer, S. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015), IEEE, pp. 922–928.

[23] Melzer, T. Non-parametric segmentation of als point clouds using mean shift.

[24] Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., and Terzopoulos, D. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence* (2021).

[25] Nguyen, A., and Le, B. 3d point cloud segmentation: A survey. In *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)* (2013), IEEE, pp. 225–230.

[26] Poppinga, J., Vaskevicius, N., Birk, A., and Pathak, K. Fast plane detection and polygonalization in noisy 3d range images. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2008), IEEE, pp. 3378–3383.

[27] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 652–660.

[28] Rabbani, T., and Van Den Heuvel, F. Efficient hough transform for automatic detection of cylinders in point clouds. *Isprs Wg Iii/3, Iii/4 3* (2005), 60–65.

[29] Saglam, A., Makineci, H. B., Baykan, Ö. K., and Baykan, N. A. Clustering-based plane refitting of non-planar patches for voxel-based 3d point cloud segmentation using k-means clustering. *Traitement du Signal 37*, 6 (2020).

[30] Sappa, A. D., and Devy, M. Fast range image segmentation by an edge detection strategy. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling* (2001), IEEE, pp. 292–299.

[31] Schnabel, R., Wahl, R., and Klein, R. Efficient ransac for point-cloud shape detection. In *Computer graphics forum* (2007), vol. 26, Wiley Online Library, pp. 214–226.

[32] Strom, J., Richardson, A., and Olson, E. Graph-based segmentation for colored 3d laser point clouds. In *2010 IEEE/RSJ international conference on intelligent robots and systems* (2010), IEEE, pp. 2131–2136.

[33] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 945–953.

[34] Szeliski, R. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[35] Ural, S., and Shan, J. Min-cut based segmentation of airborne lidar point clouds. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 39* (2012), B3.

[36] Vo, A.-V., Truong-Hong, L., Laefer, D. F., and Bertolotto, M. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing 104* (2015), 88–100.

[37] Wang, B. H., Chao, W.-L., Wang, Y., Hariharan, B., Weinberger, K. Q., and Campbell, M. Ldls: 3-d object segmentation through label diffusion from 2-d images. *IEEE Robotics and Automation Letters 4*, 3 (Jul 2019), 2902–2909.

[38] Wang, H., Berkers, J., Hurk, N., and Farsad Layegh, N. Study of loaded versus unloaded measurements in railway track inspection. *Measurement 169* (10 2020).

[39] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[40] Xia, S., Chen, D., Wang, R., Li, J., and Zhang, X. Geometric primitives in lidar point clouds: A review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 13* (2020), 685–707.

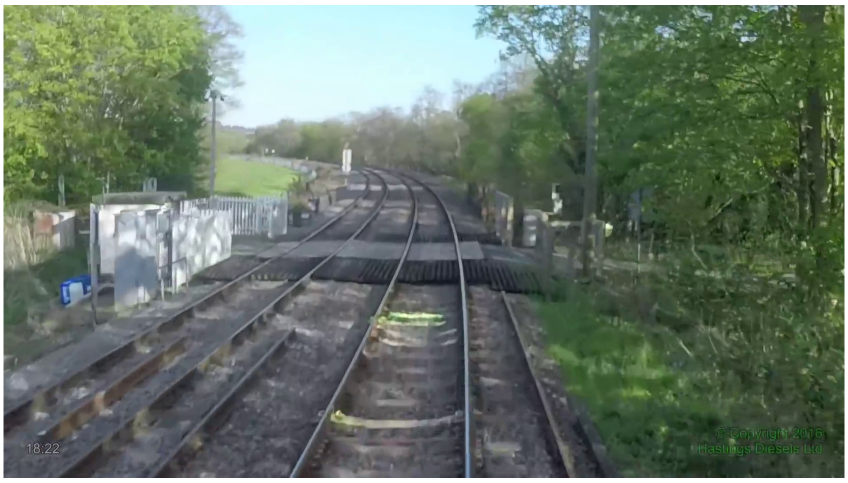
[41] Xie, Y., Tian, J., and Zhu, X. X. Linking points with labels in 3d: A review of point cloud semantic segmentation. *IEEE Geoscience and Remote Sensing Magazine* (2020).

[42] Zendel, O., Murschitz, M., Zeilinger, M., Steininger, D., Abbasi, S., and Beleznai, C. Railsem19: A dataset for semantic rail scene understanding. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2019), pp. 1221–1229.

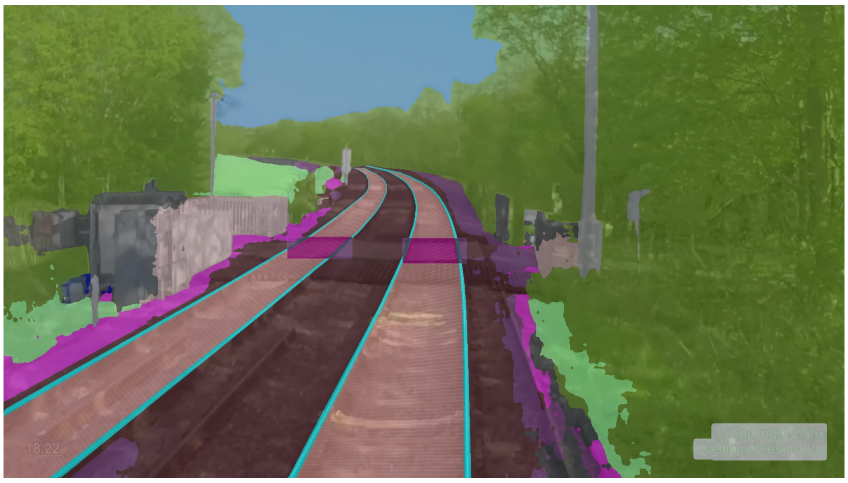
[43] Zhou, Q.-Y., Park, J., and Koltun, V. Open3D: A modern library for 3D data processing. *arXiv:1801.09847* (2018).

A Detectron2 inference on RailSem19 data

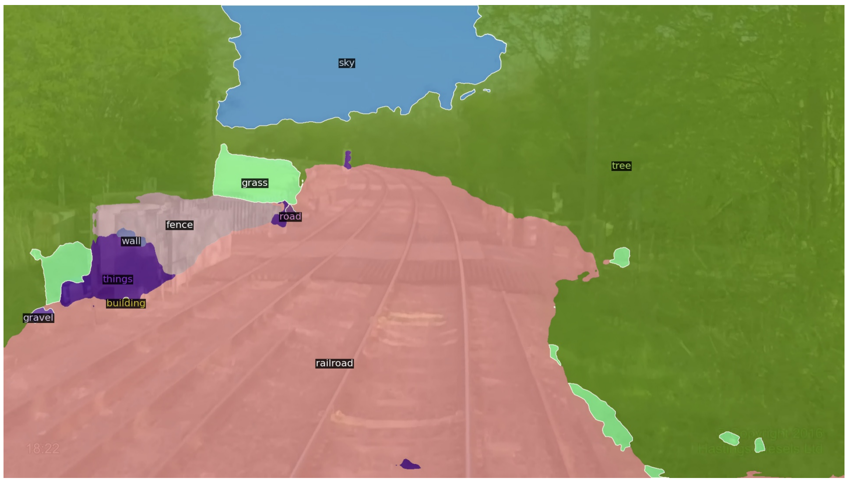
This appendix contains multiple examples of the inference of the semantic half of the panoptic segmentation model on the RailSem19 dataset.



(a) Input image

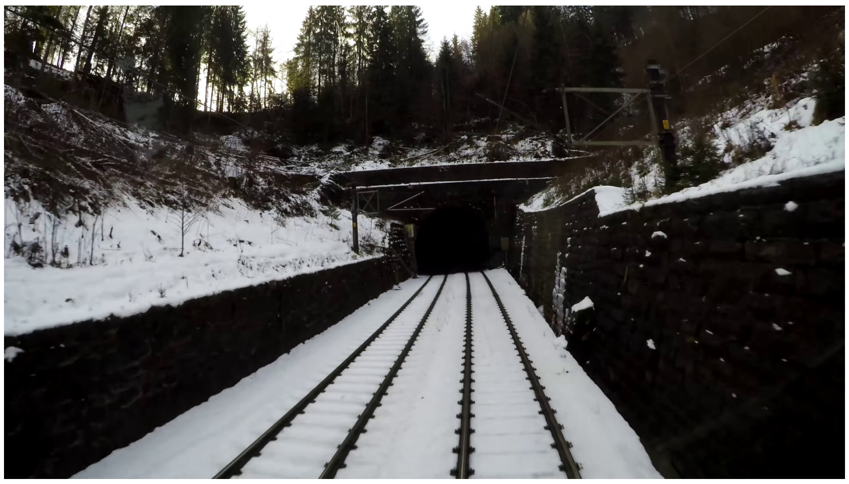


(b) Ground truth

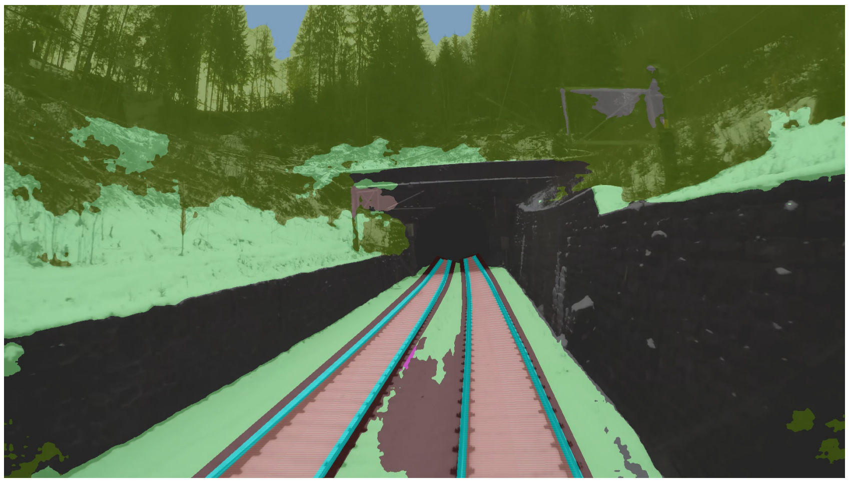


(c) Model output

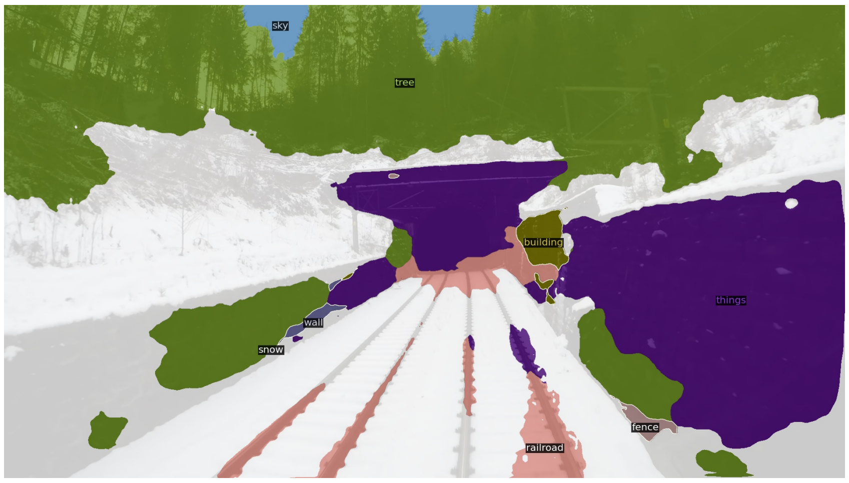
Figure 26: Semantic segmentation inference on RailSem19.



(a) Input image

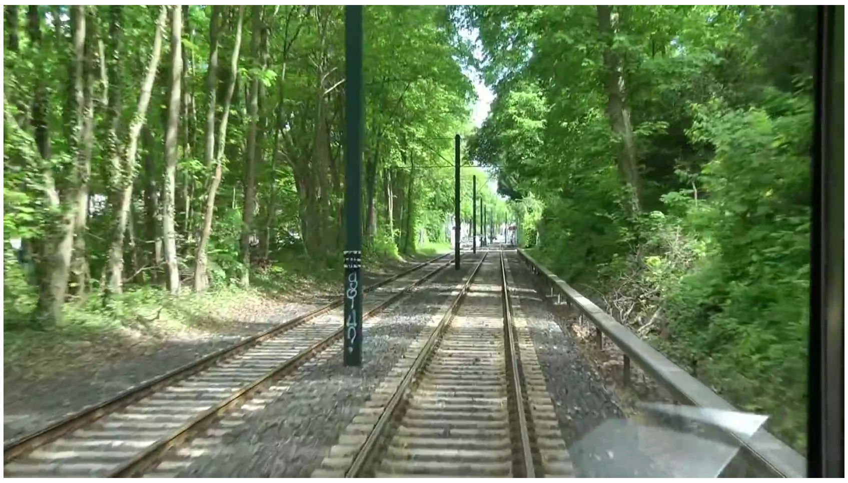


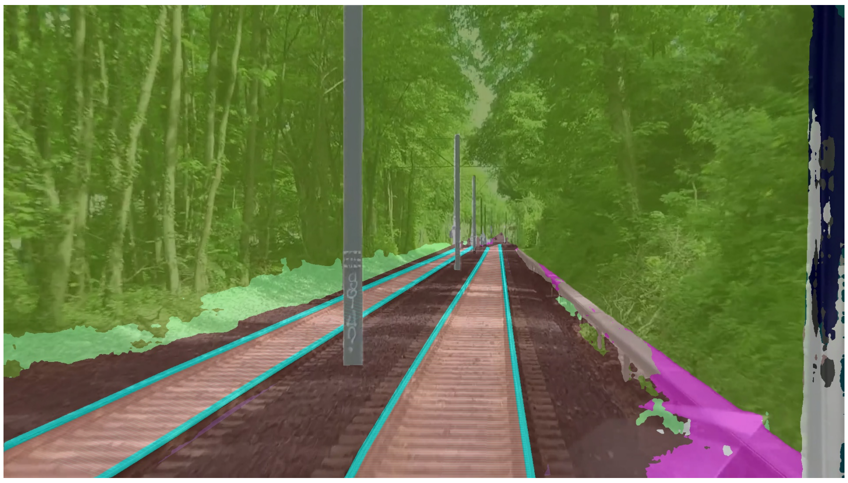
(b) Ground truth

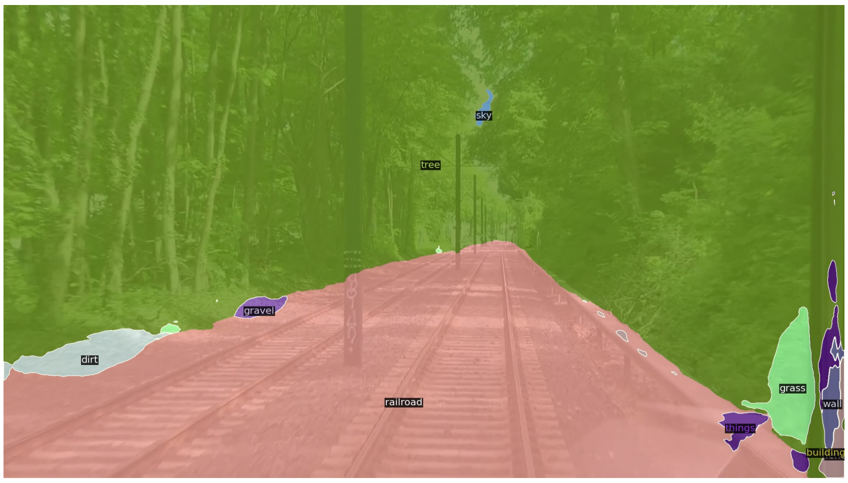


(c) Model output

Figure 27: Semantic segmentation inference on RailSem19.


(a) Input image

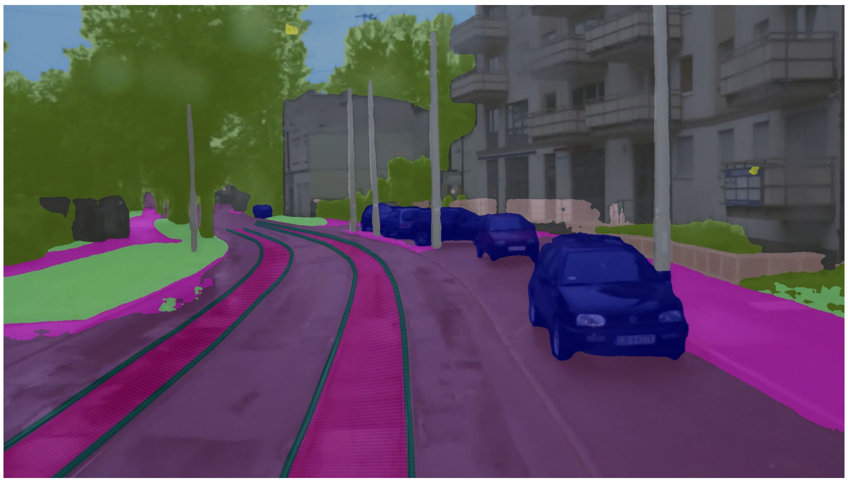

(b) Ground truth

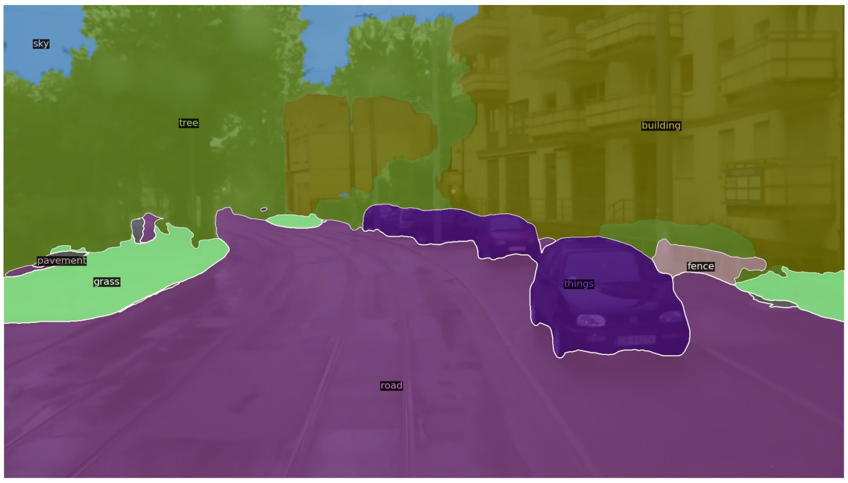


(c) Model output

Figure 28: Semantic segmentation inference on RailSem19.

(a) Input image

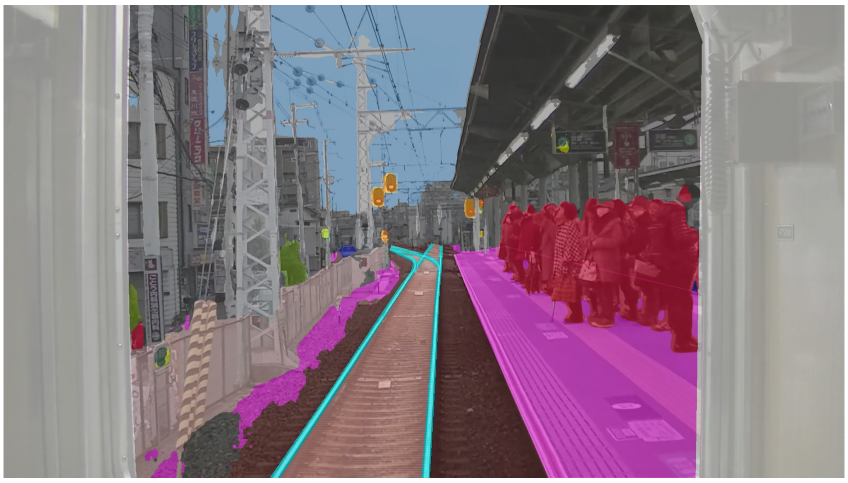

(b) Ground truth



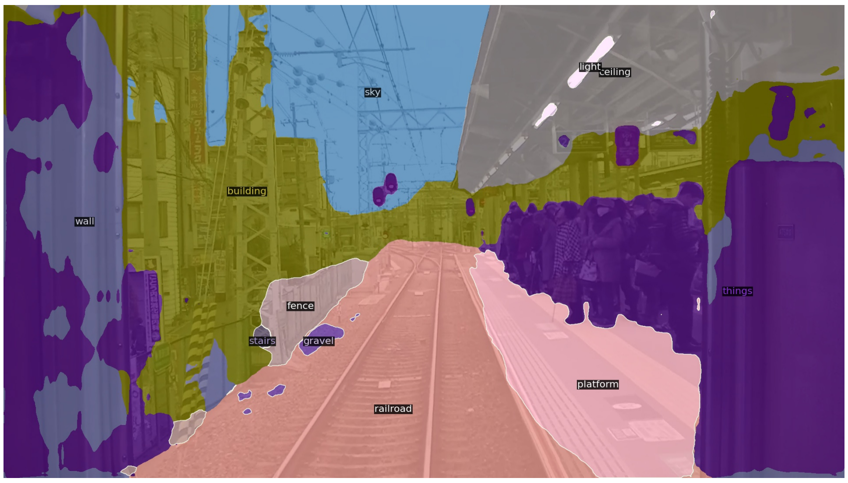
(c) Model output

Figure 29: Semantic segmentation inference on RailSem19.

(a) Input image



(b) Ground truth

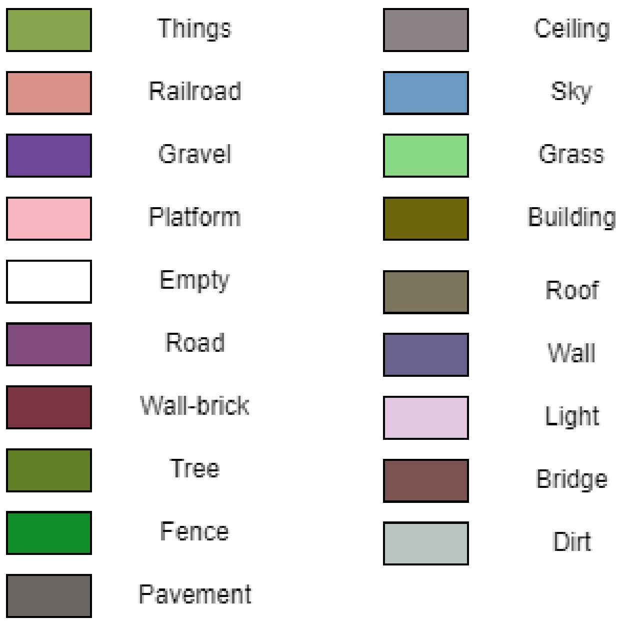


(c) Model output

Figure 30: Semantic segmentation inference on RailSem19.

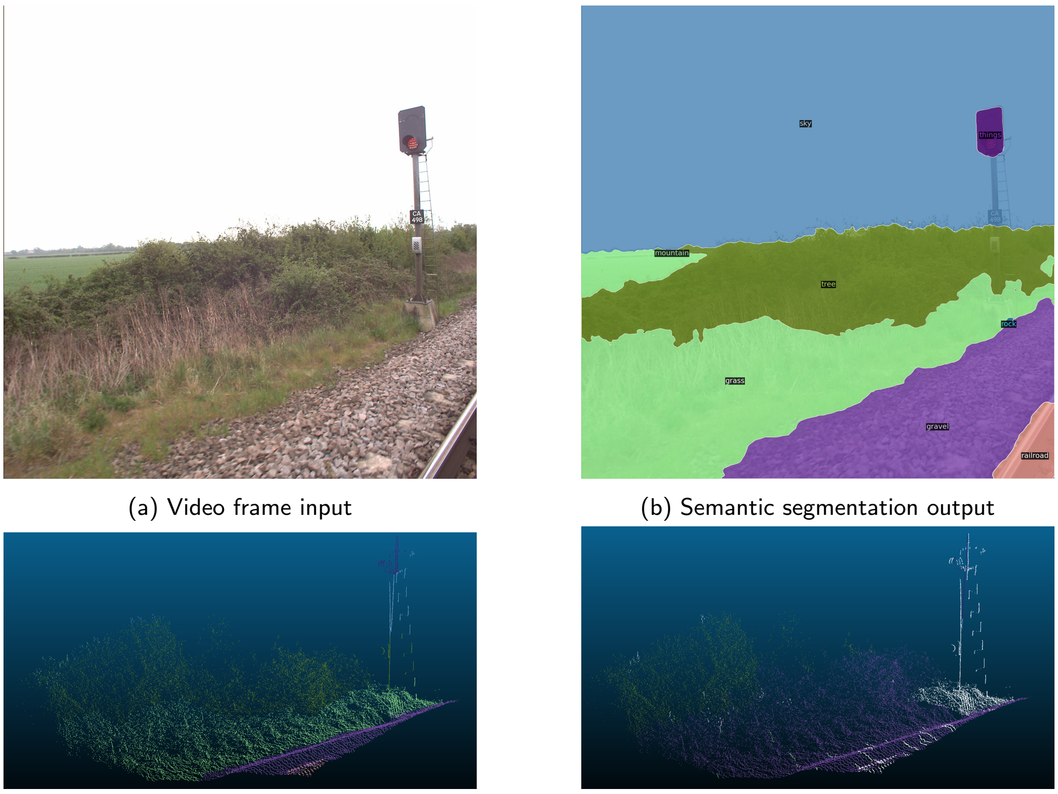
B Fugro data complete methodology examples

This appendix contains multiple examples of the complete methodologies, both for a reduced viewing distance of 11 meters and a full range of up to 50 meters, for different areas along the track. For each of those points, we included all 3 camera views for the reduced radius experiments. For the full range experiments, we only included the middle camera view. The label legend is included in Figure 31.



🟩	Things	⬛	Ceiling
🟥	Railroad	🟦	Sky
🟪	Gravel	🟩	Grass
🟥	Platform	🟫	Building
⬜	Empty	🟫	Roof
🟪	Road	🟪	Wall
🟥	Wall-brick	🟪	Light
🟩	Tree	🟫	Bridge
🟩	Fence	⬜	Dirt
⬛	Pavement		

Figure 31: Point cloud label legend



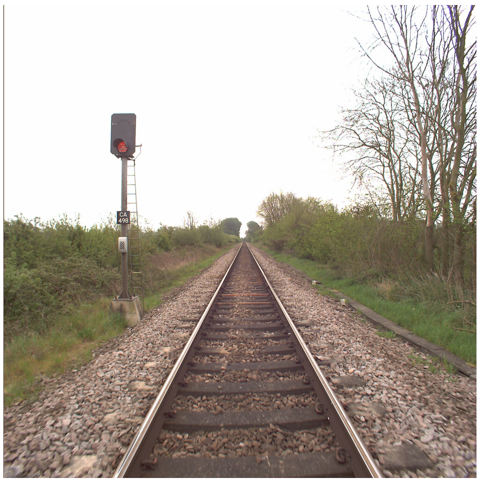
(a) Video frame input

(b) Semantic segmentation output

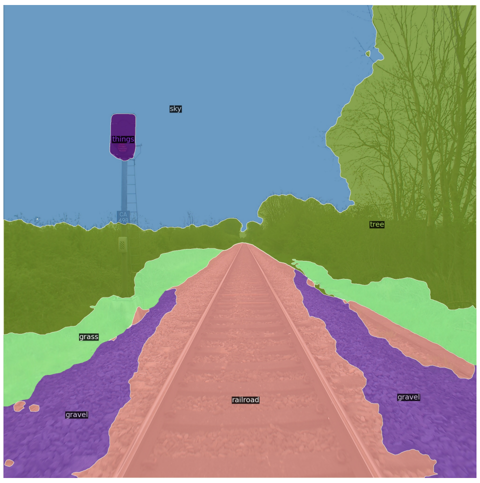
(c) Point cloud generated with projection

(d) Point cloud generated with LDLS

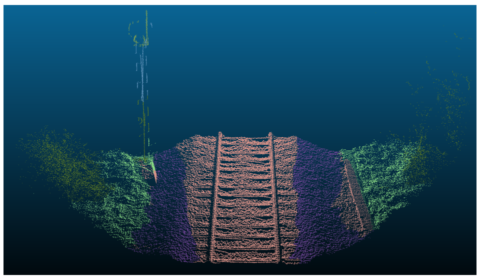
Figure 32: Methodology overview on Fugro data, left camera.



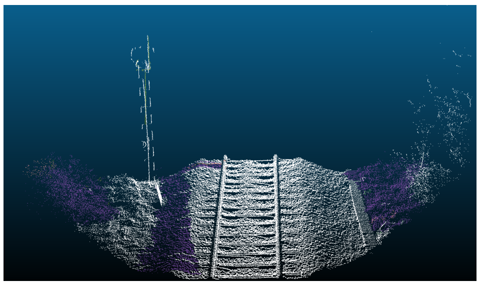
(a) Video frame input



(b) Semantic segmentation output

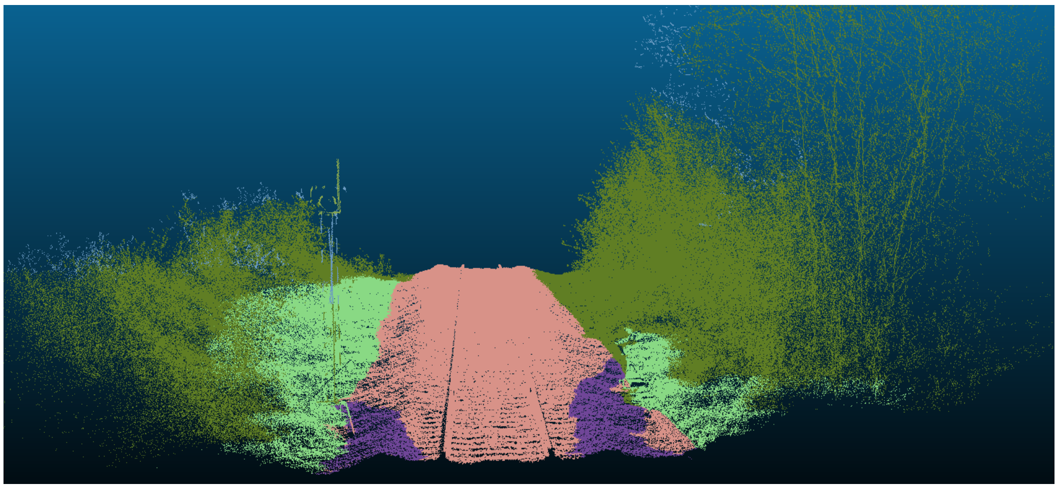


(c) Point cloud generated with projection

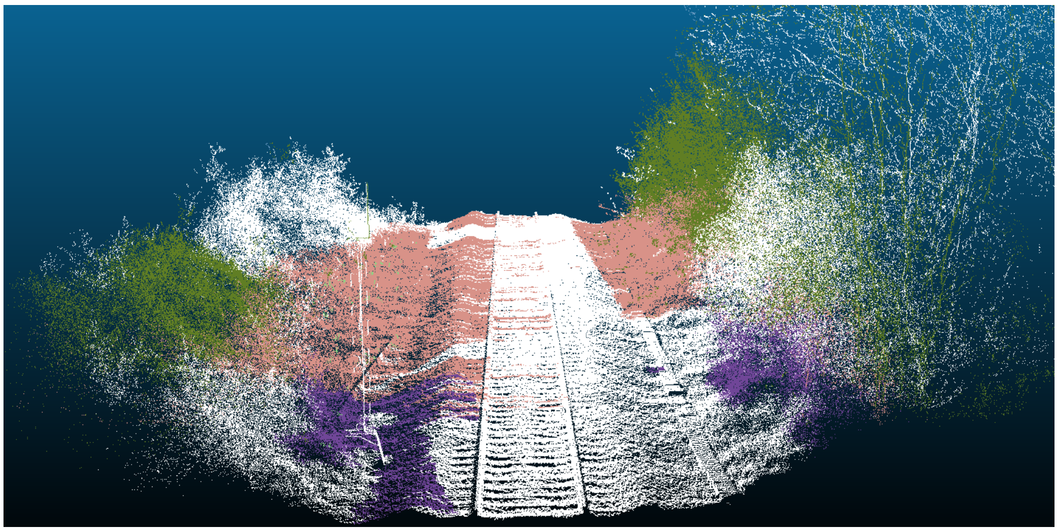


(d) Point cloud generated with LDLS

Figure 33: Methodology overview on Fugro data, middle camera.

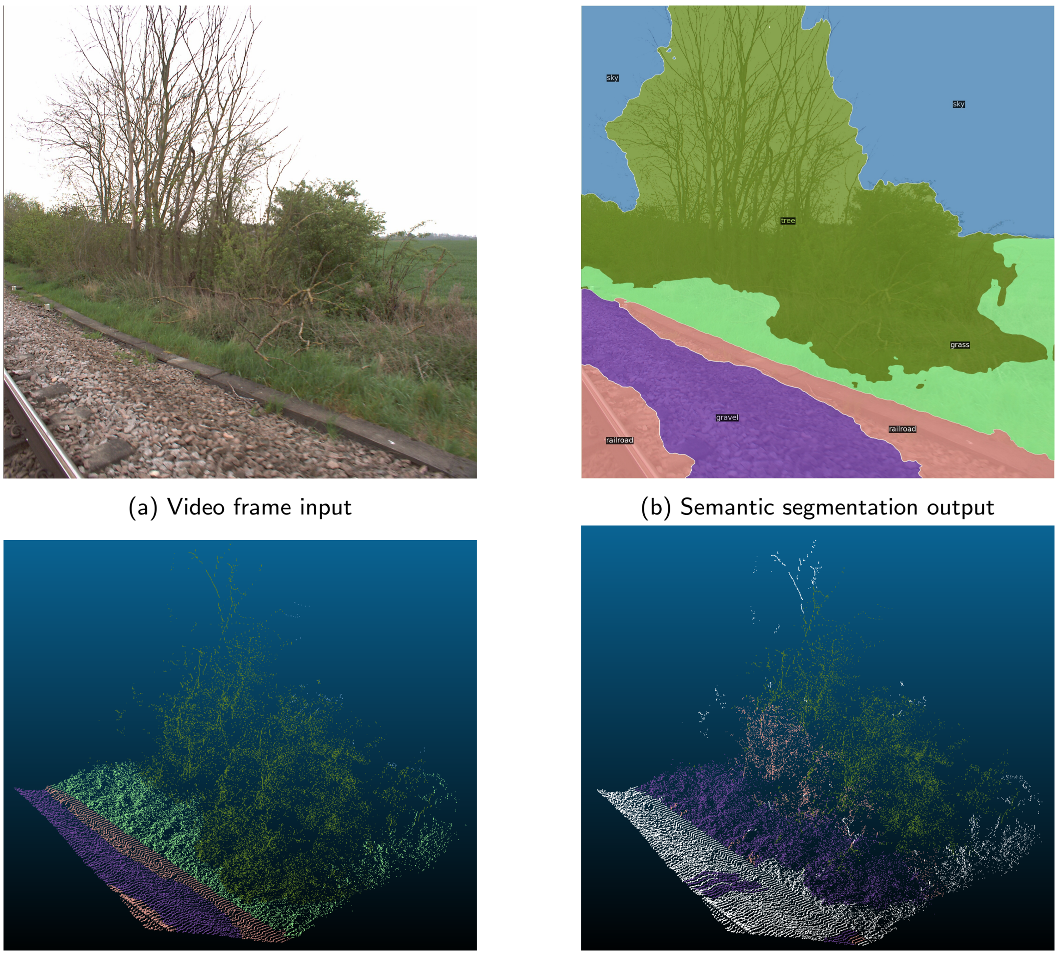


(a) Point cloud generated with projection



(b) Point cloud generated with LDLS

Figure 34: Full range methodology overview on Fugro data, middle camera.



(a) Video frame input

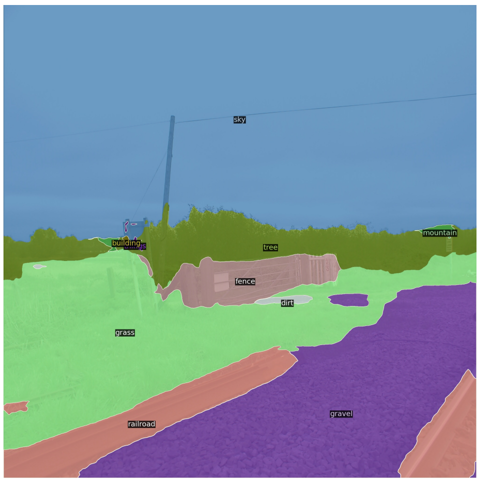
(b) Semantic segmentation output

(c) Point cloud generated with projection

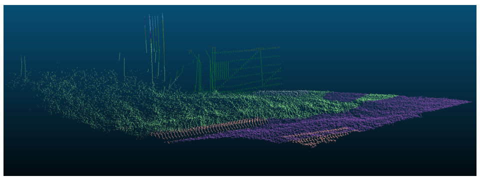
(d) Point cloud generated with LDLS

Figure 35: Methodology overview on Fugro data, right camera.

(a) Video frame input



(b) Semantic segmentation output


(c) Point cloud generated with projection

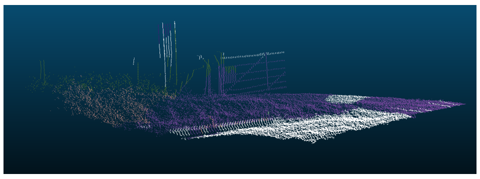
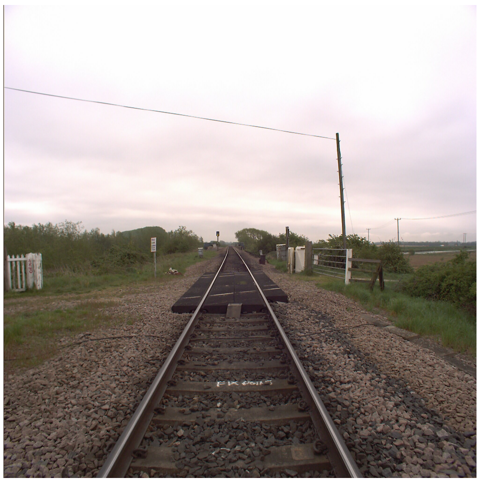
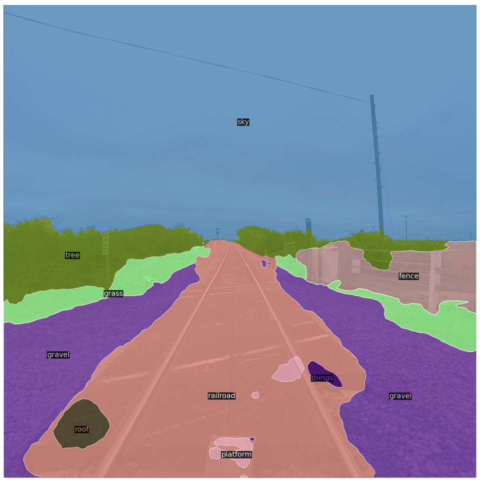
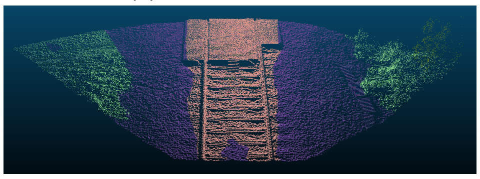

(d) Point cloud generated with LDLS

Figure 36: Methodology overview on Fugro data, left camera.


(a) Video frame input



(b) Semantic segmentation output


(c) Point cloud generated with projection

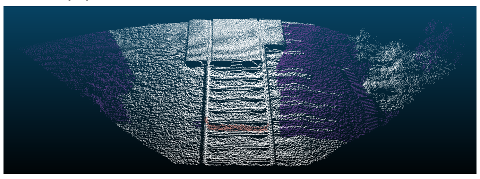
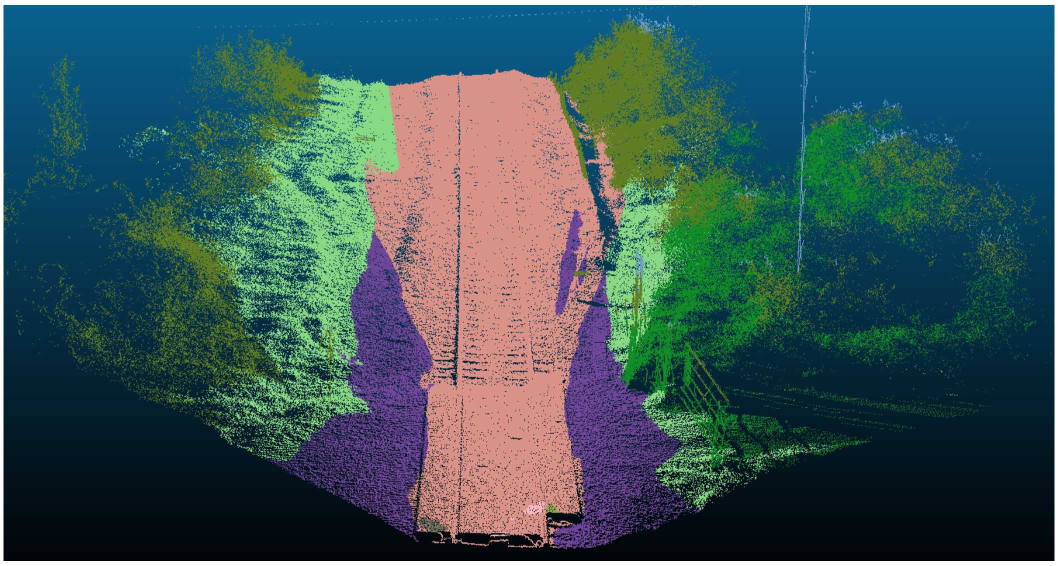

(d) Point cloud generated with LDLS

Figure 37: Methodology overview on Fugro data, middle camera.


(a) Point cloud generated with projection

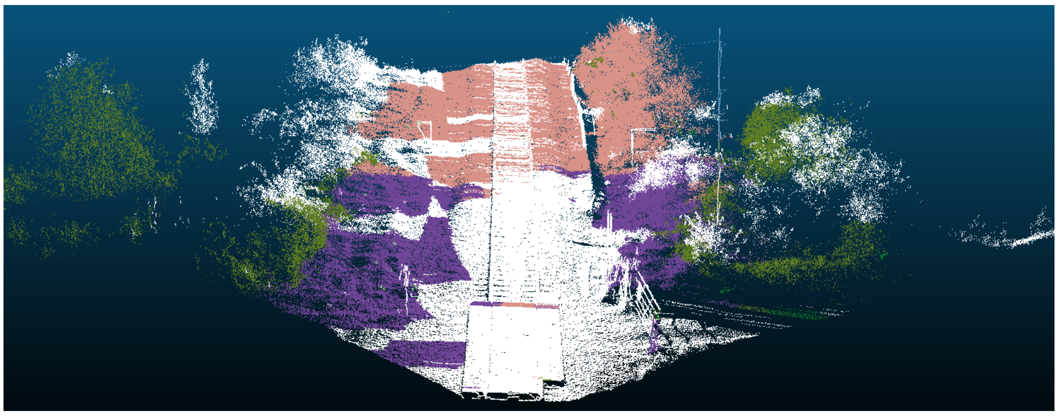
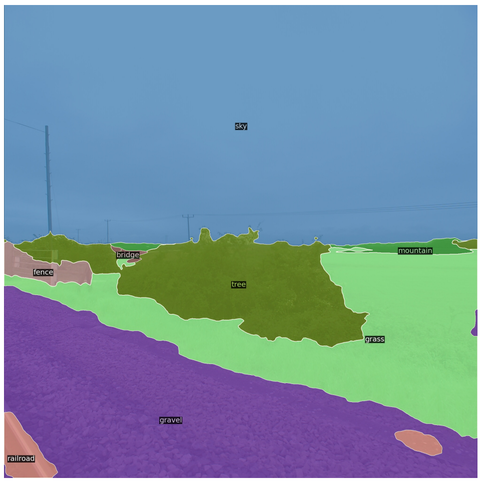
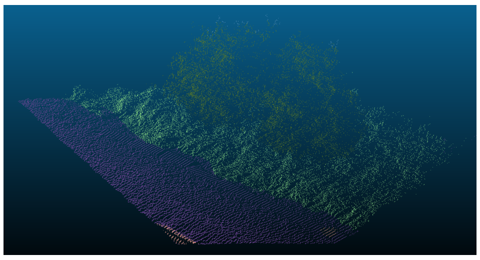

(b) Point cloud generated with LDLS

Figure 38: Full range methodology overview on Fugro data, middle camera.

(a) Video frame input



(b) Semantic segmentation output


(c) Point cloud generated with projection

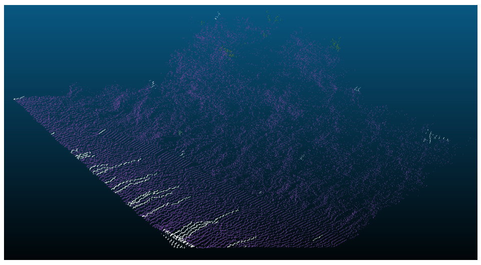
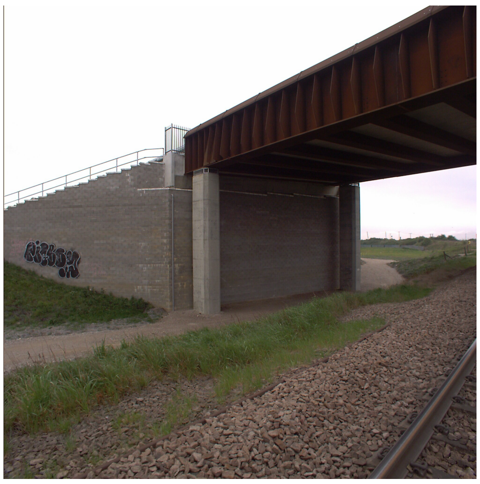
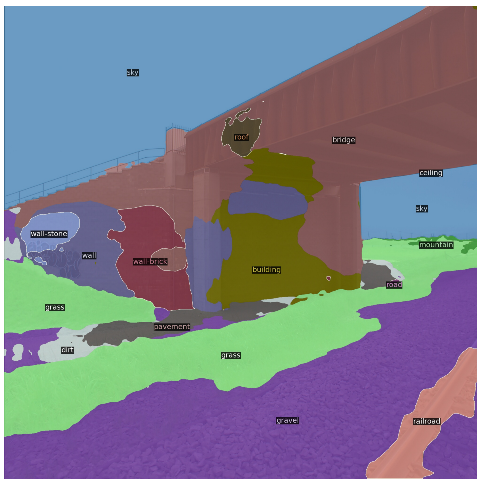
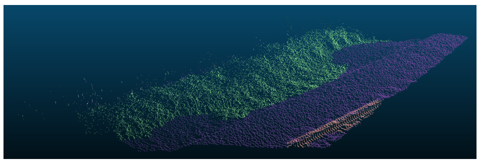

(d) Point cloud generated with LDLS

Figure 39: Methodology overview on Fugro data, right camera.


(a) Video frame input



(b) Semantic segmentation output


(c) Point cloud generated with projection

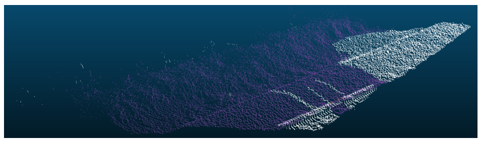
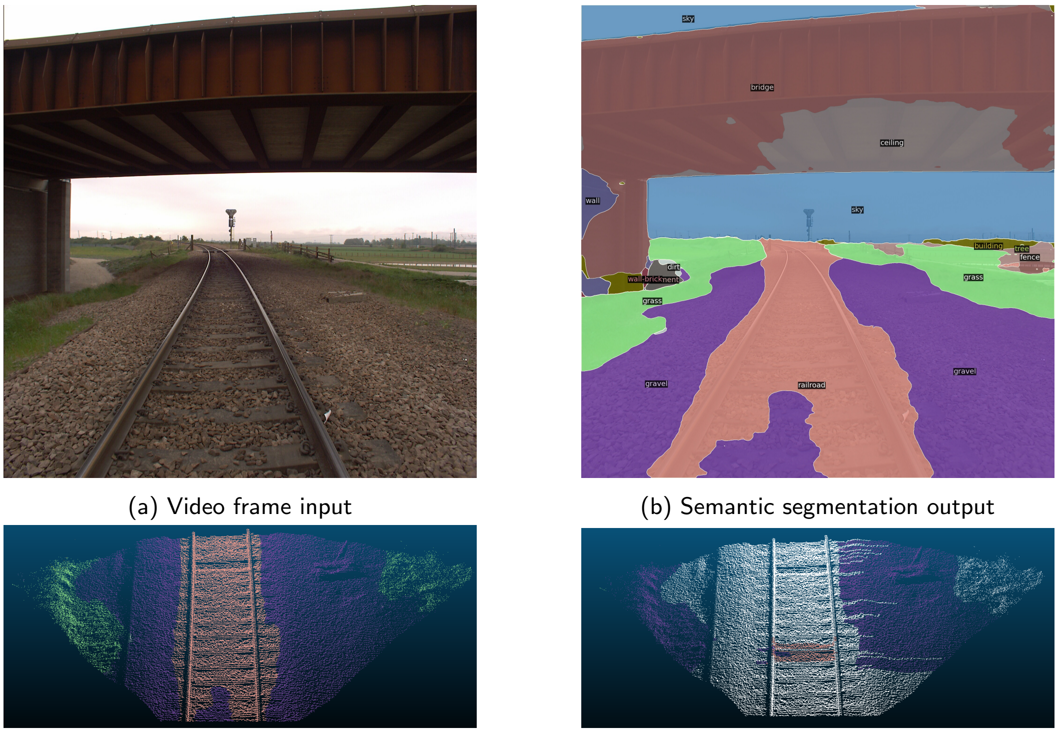

(d) Point cloud generated with LDLS

Figure 40: Methodology overview on Fugro data, left camera.



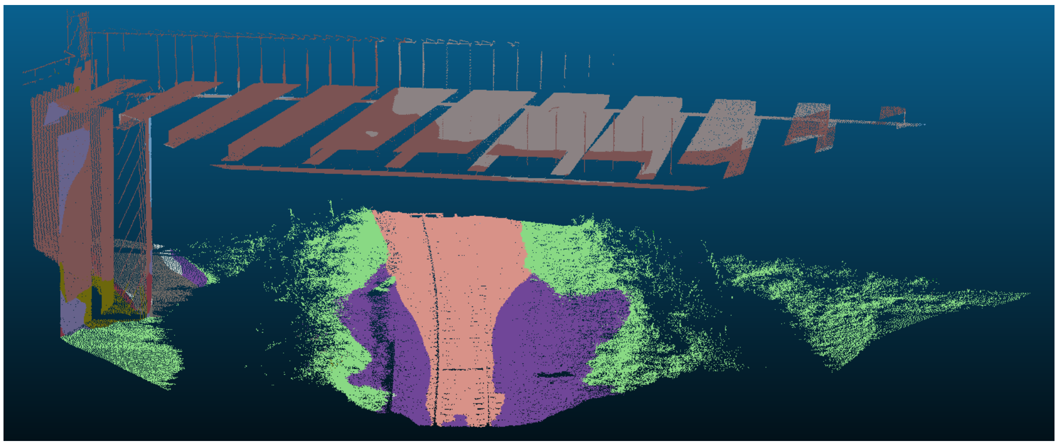
(a) Video frame input

(b) Semantic segmentation output

(c) Point cloud generated with projection

(d) Point cloud generated with LDLS

Figure 41: Methodology overview on Fugro data, middle camera.


(a) Point cloud generated with projection

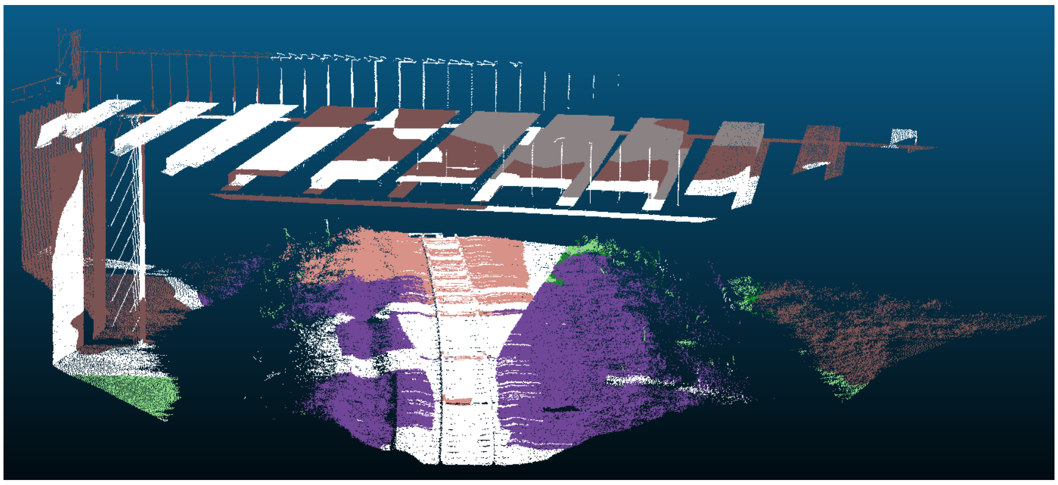
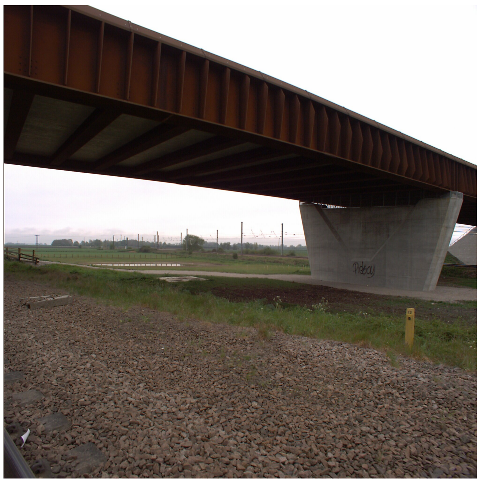
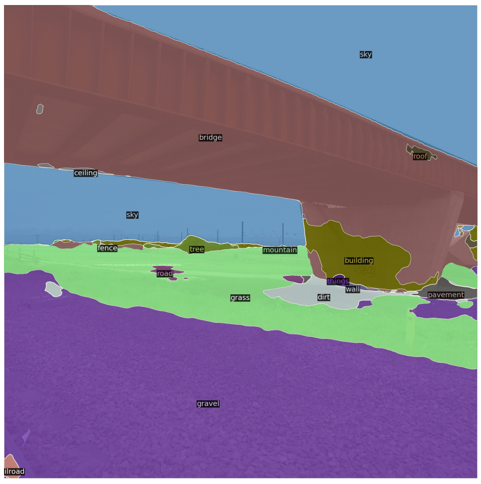

(b) Point cloud generated with LDLS

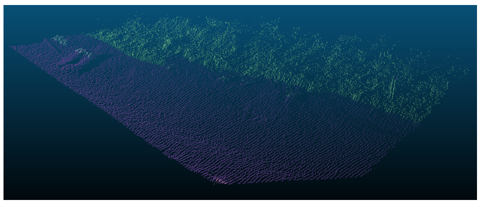
Figure 42: Full range methodology overview on Fugro data, middle camera.



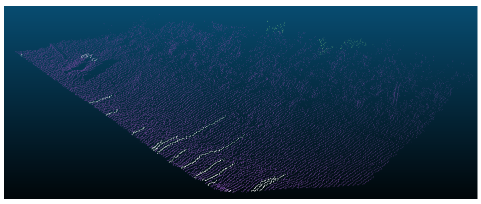
(a) Video frame input



(b) Semantic segmentation output

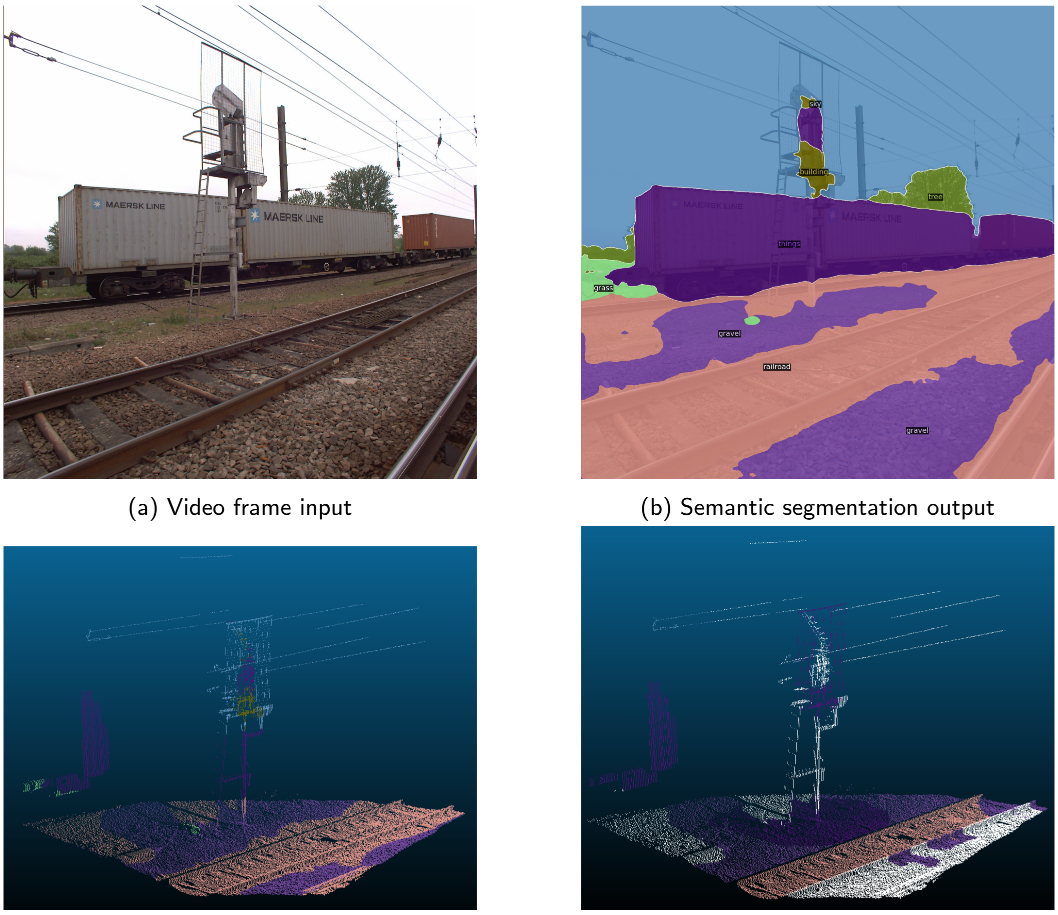


(c) Point cloud generated with projection



(d) Point cloud generated with LDLS

Figure 43: Methodology overview on Fugro data, right camera.



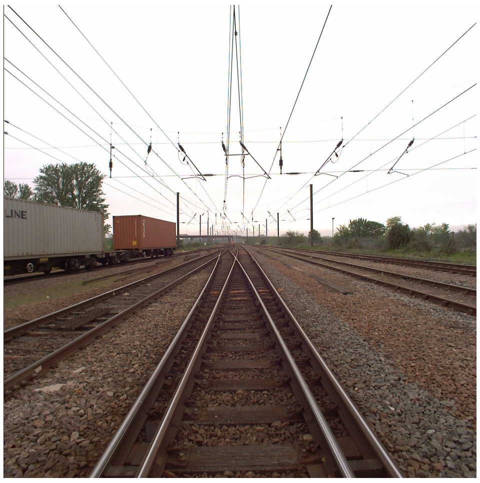
(a) Video frame input

(b) Semantic segmentation output

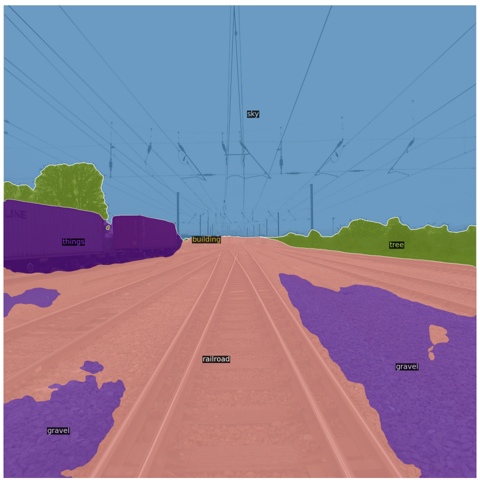
(c) Point cloud generated with projection

(d) Point cloud generated with LDLS

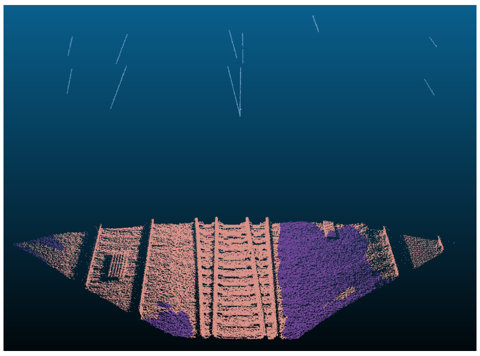
Figure 44: Methodology overview on Fugro data, left camera.



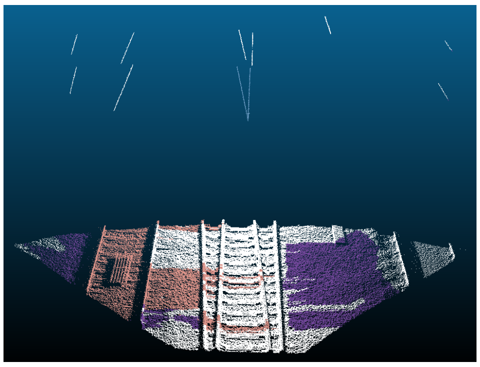
(a) Video frame input



(b) Semantic segmentation output

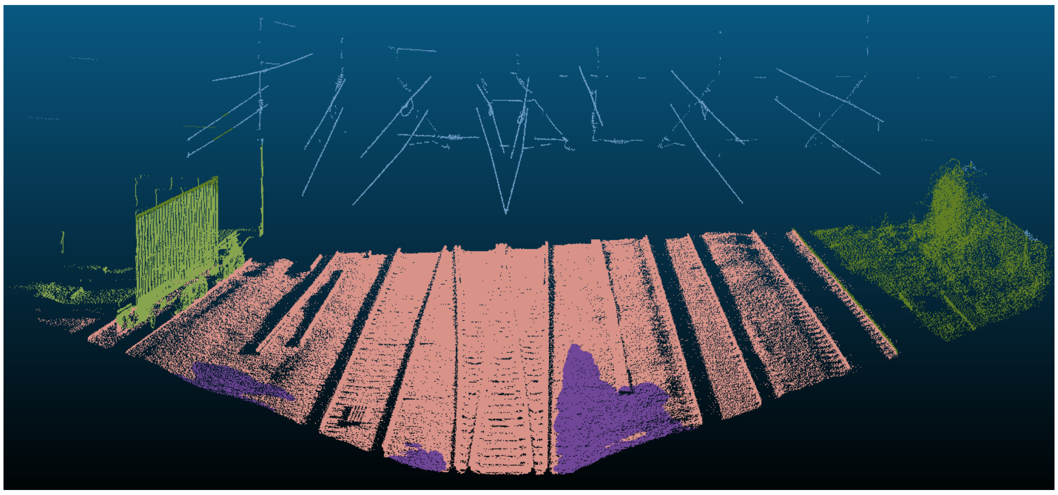


(c) Point cloud generated with projection

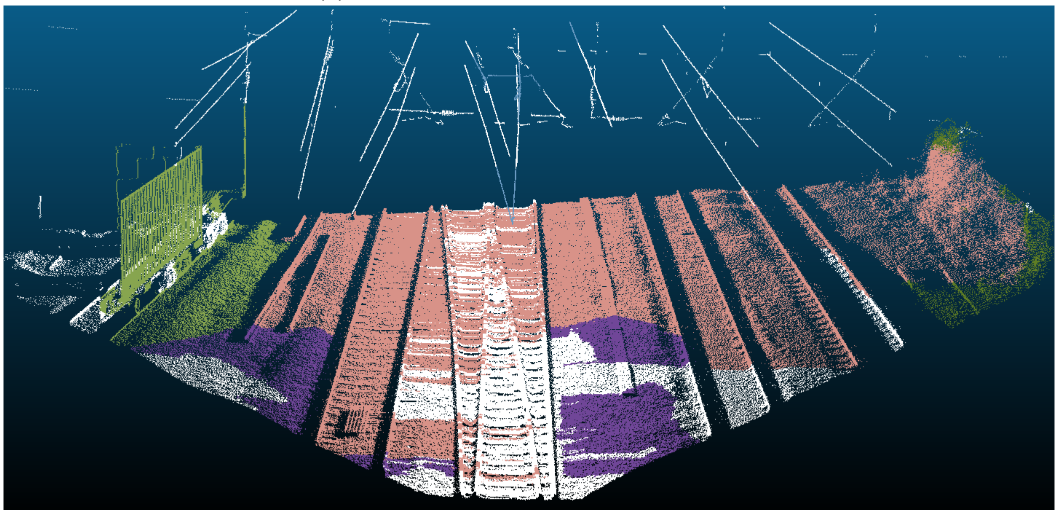


(d) Point cloud generated with LDLS

Figure 45: Methodology overview on Fugro data, middle camera.

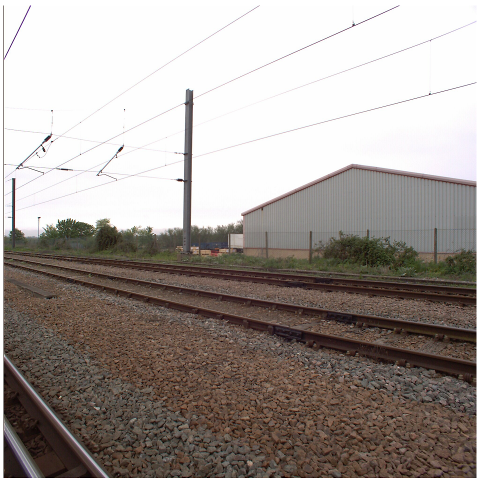


(a) Point cloud generated with projection

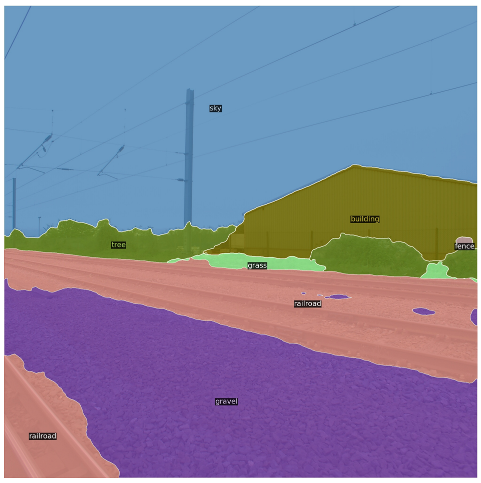


(b) Point cloud generated with LDLS

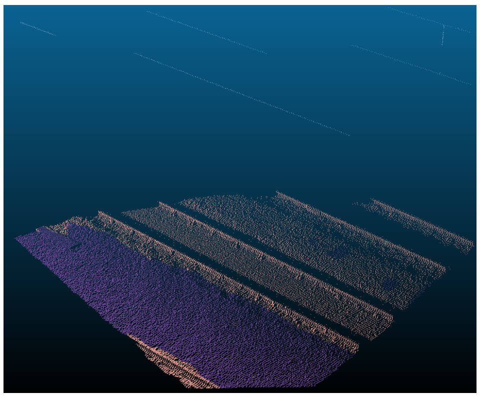
Figure 46: Full range methodology overview on Fugro data, middle camera.



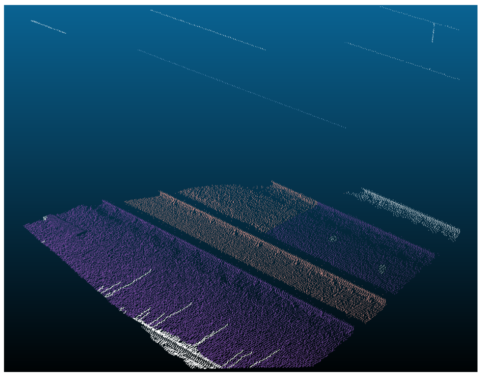
(a) Video frame input



(b) Semantic segmentation output

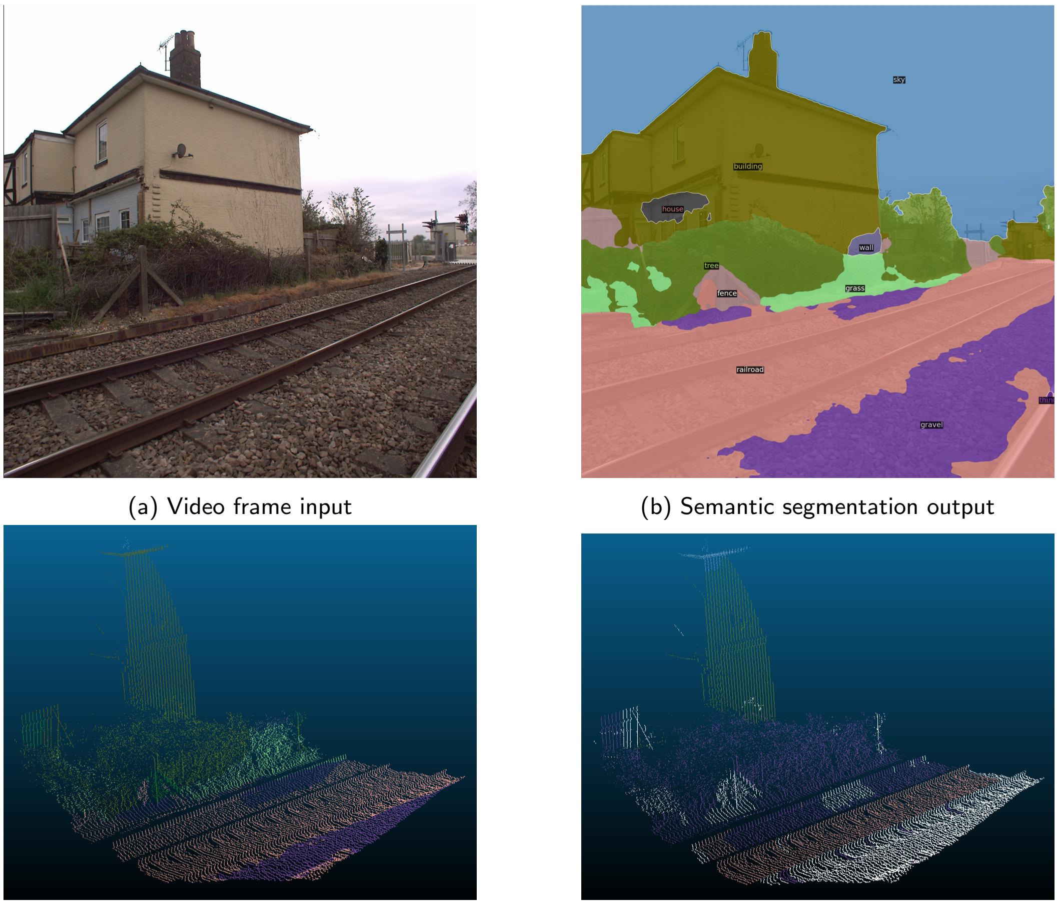


(c) Point cloud generated with projection



(d) Point cloud generated with LDLS

Figure 47: Methodology overview on Fugro data, right camera.



(a) Video frame input

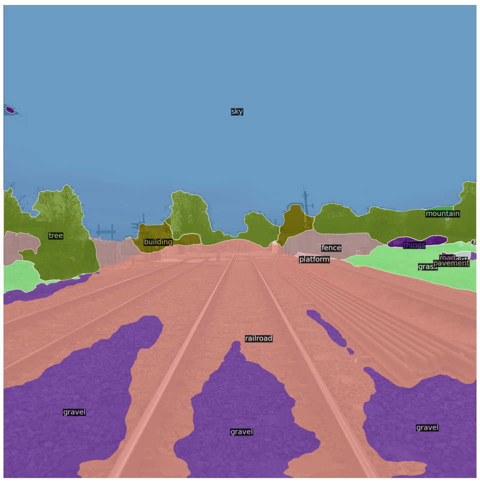
(b) Semantic segmentation output

(c) Point cloud generated with projection

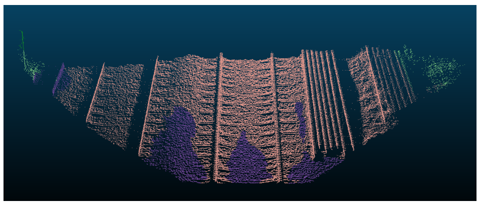
(d) Point cloud generated with LDLS

Figure 48: Methodology overview on Fugro data, left camera.

(a) Video frame input



(b) Semantic segmentation output


(c) Point cloud generated with projection

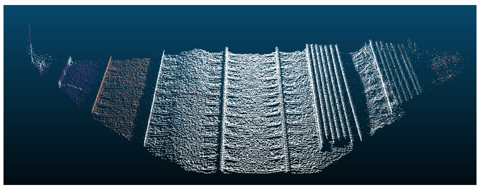
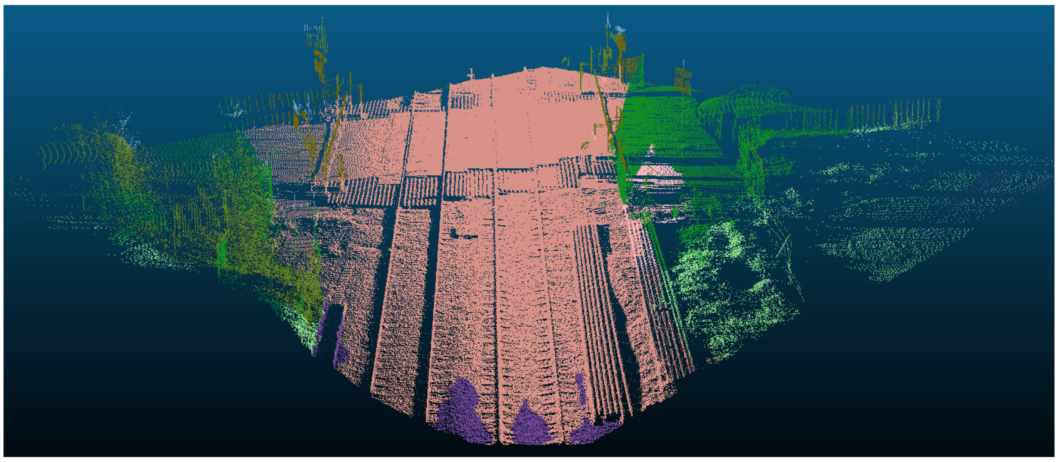
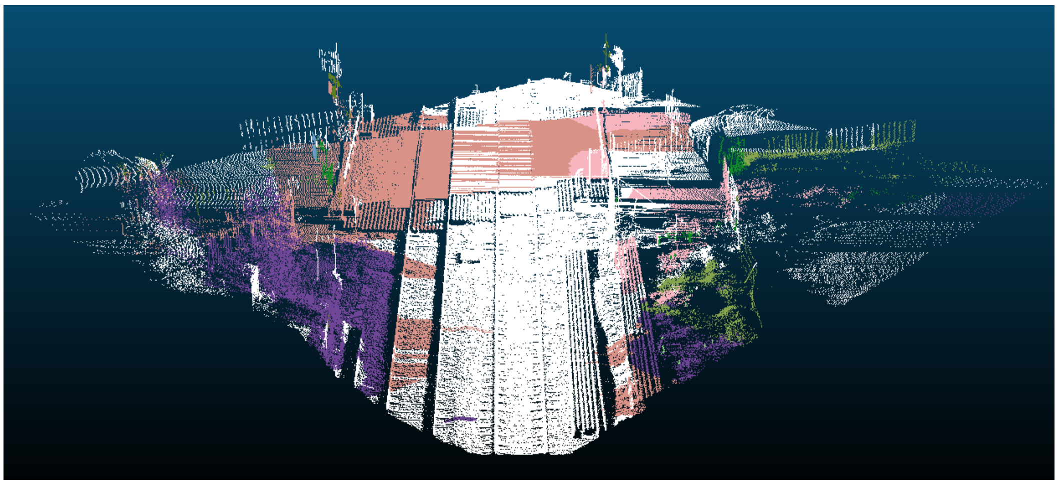

(d) Point cloud generated with LDLS

Figure 49: Methodology overview on Fugro data, middle camera.



(a) Point cloud generated with projection



(b) Point cloud generated with LDLS

Figure 50: Full range methodology overview on Fugro data, middle camera.