Utrecht University

Reconstructing 3D object information from monocular camera data in a railway environment

Master's Thesis

Aidan Bundel

Supervisors:
Marc van Kreveld, Maarten Löffler

External supervisors:
Neda Sepasian, Sudan Xu

Faculty of Science
Department of Information and Computing Sciences
Graduate School of Natural Sciences

March, 2022

Acknowledgements

I would like to thank my university supervisor Marc, for his insights, quick responses and flexibility, all of this has been truly helpful throughout this research. I would like to thank my second supervisor Maarten, for his flexibility in the last phase and taking the time to read my work. I would also like to thank my external supervisors, Neda and Sudan. During our weekly meetings both of you always helped me with your insights to continue the research.

Furthermore, I want to thank my fellow student Philippe. Especially in times with lockdowns, work from home days and getting acquainted at a company, it was very valuable to work with someone going through the same process.

Finally, I want to thank to my family and my friends, who are always there for me and supported me throughout this research.

Abstract

As humans we are able to deconstruct a scene into different (primitive) shapes and objects and understand the structures in 3D from just a 2D image, even if the scene contains occluded regions. Nevertheless, reconstructing a scene from images is a non-trivial problem to automate, as 2D representation of geometry leaves many ambiguities when projecting to 3D. This requires a solid depth perception, which is a challenging task for vision-based algorithms. Yet, many advancements have been made in the last few years on estimating the depth from camera input.

In this thesis the retrieval of 3D object information from a single moving camera in a railway environment is investigated. We explore three different methods that involve estimating depth from sequences of images and compare it with point clouds and annotated 3D data: pseudo-LiDAR, Structure from motion (SfM) and 3D object detection. We use qualitative and quantitative metrics to compare pseudo-LiDAR and SfM to LiDAR. 3D object detection is evaluated using LiDAR-aligned annotations.

We show that we can recover certain 3D information from the camera data. First, we find that the general depth estimation method pseudo-LiDAR yields unsatisfying results and that SfM creates qualitatively promising point clouds, although not accurate in dimensions and GPS alignment. 3D object detection shows promising results in both classifying and predicting the 3D locations of objects of interest.

Contents

Chapter 1

List of Terms and Acronyms

Glossary

Epochs and iteration and batches During training, the training dataset is split into batches of a given size (typically a multiple of 2). In one iteration (also called pass), a single batch of data is processed. During one epoch, the model sees the complete training dataset, i.e. all the batches. 28

Instance segmentation This type of segmentation classifies each object separately. This is useful for tasks such as counting the number of objects. Here the goal is to classify and segment each object individually. 10

IoU Intersection over Union, an evaluation metric used for 2D or 3D object detection models that measures the amount of overlap between two regions. In the context of this research, the relevant shapes are bounding boxes. 26

Semantic segmentation This method groups points by semantic meaning. For example, points belonging to a person are grouped into one class. 10

Train-Test-Val split Training a machine learning model requires the dataset to be split in disjoint subsets, a training, validation and test set. Training and validation subset are used during training. The validation split provides an unbiased evaluation while tuning the model's hyperparameters (weights). The test set is only seen by the model after training and does not influence the model's weights. The performance on the validation set is typically higher than the testing split, because of the model fitting.. 18, 25

Acronyms

C2C Cloud to cloud.. 29, 31, 49

LiDAR Light Detection and Ranging. 13, 20, 23, 28–31, 37–39, 41, 49, 54

MVS Multi-view Stereo. 14, 17

SfM Structure from motion. 2, 8, 13, 14, 17, 18, 20, 22, 24, 28–31, 37–39, 41, 54

Chapter 2

Introduction

Data intensive technologies are motivating companies in virtually every industry to find new solutions for processing and storing data about their enterprise operations. One such operation is analysing real-world objects and environments, which is important in several industries, such as construction, robotics, augmented reality and automotive. For example, self-driving cars need to map and understand their surroundings in real-time under different situations and weather conditions. For the rail industry, three-dimensional (3D) scanning technologies and cameras quickly capture track information, in some applications even without interrupting or affecting train operations.

These innovations involve different data processing methods depending on the application. One such application is 3D reconstruction, which is the topic of this thesis. As humans we are able to deconstruct a scene into different (primitive) shapes and objects and understand the structures in 3D from just a 2D image, even if the scene contains occluded regions. For computational models, however, reconstructing a scene from images is a challenging problem that requires a solid depth perception. Point cloud detection and segmentation techniques have become better and faster at analysing the point cloud data coming from laser scanner sensors. In computer vision and photogrammetry, 3D reconstruction is a major field describing the extraction of 3D geometry of scenes or objects from images, also referred to as image based reconstruction. Extracted 3D geometry is typically assembled in the form of a point cloud or mesh. 3D reconstruction has overlap with 3D object detection, where the latter also involves determining 3D information from images. Instead of recovery of scene geometry and object shapes, 3D bounding boxes are predicted based on a single image. All of these methods, at some point in their pipeline, need to infer depth in an image to add a dimension. In the case of 3D reconstruction, this is generally done for every pixel in an image. In 3D object detection, depth estimation is only necessary for predicting the eight vertices of a bounding box. Creating input data can be done with monocular or stereo camera input, where the former means a single camera was involved and the latter is an aligned setup of two cameras taking pictures at the same time. We discuss the different camera systems in further detail in Chapter 3.

In this research, we focus on processing camera data from a railway environment to capture objects of interest surrounding the train, referred to as *wayside objects* in this document. This project was done in collaboration with *Fugro*, a company specialising in geographical data.

2.1 Project motivation

In the case of Fugro, analysing the surroundings of railway tracks is useful for business operations. Their clients require accurate 3D information of the track and its surroundings as part of their safety inspections, design, measuring of track geometry, or asset management operations. This information is stored in the form of point clouds and an annotated 3D map (see Figure 2.3).

The data that will be used in this project originates from the RILA (Rail Infrastructure aLignment Acquisition) system as seen in Figure 2.1. This system is mounted in the front or at the back of a train. As the train moves, a laser scanner rotates laser beams 360° perpendicular to the track at every interval. Three cameras capture the left, middle and right part of the track (see Figure A.1).

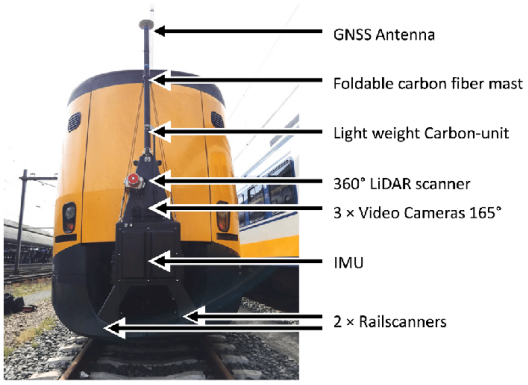


Figure 2.1: The mobile track inspection system RILA, which generates the data used in this thesis. Image from ResearchGate.

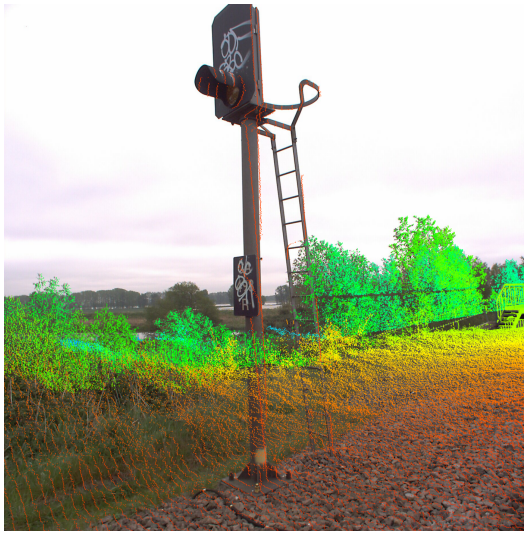


Figure 2.2: Points from laser scanner projected on camera data, showing an object only partially present in the point cloud.

One problem is that the current laser scanner is not able to scan all wayside objects along the track, especially smaller and thin objects such as signs. We discuss possible causes for incomplete scans in more detail in Chapter 3. As a result, wayside objects are not present or only partially captured in a point cloud, as shown in Figure 2.2 and 2.4. Hence there is no ground truth point cloud available, where all wayside objects are adequately present.

A train could also perform multiple runs on the same track to capture a set of point clouds that would then be merged into one point cloud with less sparsity. Despite the final point cloud containing more objects than every point cloud individually, (parts of) objects could still remain occluded. In addition, it is expensive and time-consuming to perform multiple runs. Thus, one run to scan the track surroundings is preferred.

The company's current method of producing the annotated data consist of manually marking missing objects in the camera and then adding them to laser scanner GPS coordinates. However, while very accurate, this process is labour-intensive and time-consuming for production purposes. Moreover, the necessary manual calibration and alignment of all the sensors also takes time. With less sensors, the setup duration could be reduced. Finally, the ambition is to reduce expensive equipment, minimise calibration time and replace post-processing tasks with more economical and automated solutions in the near future.

To summarise the key obstacles that are currently being faced in production:

- Wayside objects are missing or only partially present in the point cloud created from laser scanners.
- More accurate equipment or multiple scanning runs is less cost-effective.
- Manually marking wayside objects and map them in 3D is labour intensive.

2.2 Research Question

As mentioned in the previous section, missing information in the laser scanner data is only part of the problem. Combined with the other issues it calls for an offline essentially automatic framework that retrieves 3D object information from camera sensor data. We thus define this as the goal of this project, which leads to the main research question:

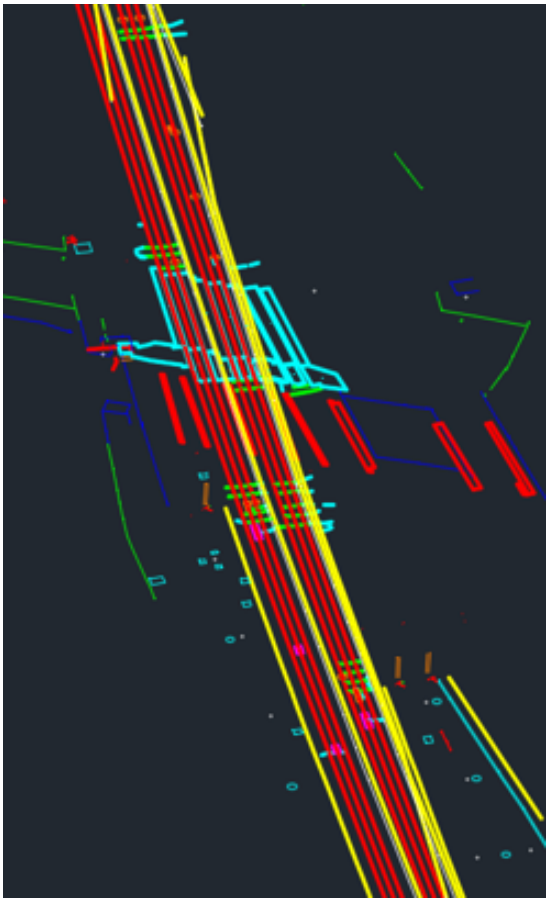
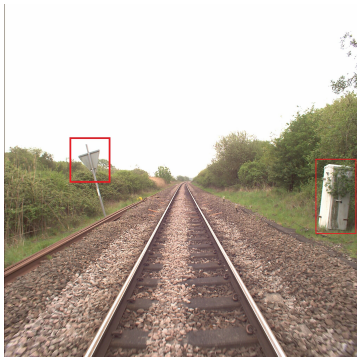
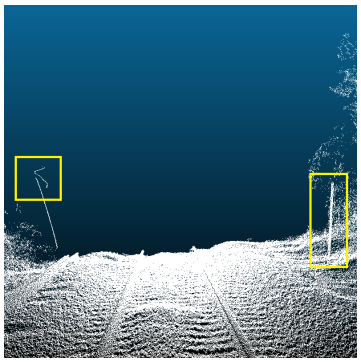


Figure 2.3: Point cloud aligned annotated map example. Every object has a category assigned with a different colour.



(a) Camera image.



(b) Corresponding point cloud.

Figure 2.4: Examples of rail road sign on the left and an electrical cabinet on the right side of the track, where parts of its structures in the point cloud are missing 2.4b when looking at the camera image 2.4a.

How effective is depth estimation at reconstructing 3D geometry, location and orientation of wayside objects from monocular video data in a railway environment?

We formulate the following three subquestions to answer the research question:

1. How can depth estimation improve the point cloud quality and localisation of wayside objects?
2. How can we measure the effectiveness of object retrieval?
3. How does depth estimation compare to laser scanner and annotated data?

2.3 Contributions

There are multiple benefits of generating 3D data from video input. Even though a laser scanner is highly precise, the scanning technology will not reliably scan all objects, such as small, thin structures as seen in Section 2.1. Another benefit is the reduced hardware cost of a camera compared to laser scanner. The contributions of this work are:

- An implementation of three depth estimation methods applied to railway data, using camera data as input: pseudo-LiDAR, SfM and 3D object detection.
- A comparison in quality between pseudo-LiDAR, SfM and point clouds from a laser scanner sensor in a railway environment.

- An evaluation of an existing 3D object detection model using LiDAR-aligned annotations.

To the best of our knowledge, pseudo-LiDAR and 3D object detection methods have not yet been applied to railway data.

Chapter 3

Background

Inferring shapes of objects and understanding scenes in 3D from 2D input is an ill-posed problem [49]. Inherently, 3D information is lost in camera data after projection to an image. Inverting this 2D representation of geometry is not a unique one-to-one mapping when projecting to 3D, because there exists many 3D scenes for a single image. Optical illusions in street art are a simple example where 3D information is ambiguous .

Below we discuss the background information relevant for reconstructing a scene from images, namely the data structures to represent a scene, how to mathematically represent the camera and what common 3D reconstruction technologies exist.

3.1 Point cloud fundamentals

One purpose of collecting data from laser scanners is to construct digital 3D models. In our case, we want to store the reconstruction as point clouds. A point cloud is an unordered and continuous set of points in 3D space, also referred to as a point set. Each point represents a single measurement of a laser scan. Point clouds from laser scans can also encode per-point information, such as reflective intensities and colour. Point clouds are useful data structures in applications such as civil engineering, robotics and architecture, where they can be used for structure analysis, digital twins or surveying operations [53].

An important process in handling point clouds is registration, which means aligning two point sets, either using manual or algorithmic transformations. A common registration algorithm is Iterative Closest Point (ICP) [7], which in short aligns two models (point clouds or meshes) by minimising the transformation needed between the two instances. A rough estimate of the alignment should be known a priori, otherwise the method is likely to fail because the algorithm stays at a local minimum [58]. More algorithms exist, but it is out of the scope of this research to discuss them all. Registration can be used when merging or comparing point clouds.

All points initially belong to one class in a point cloud. Instance segmentation or Semantic segmentation methods divide each point from a point cloud into separate classes.

3.2 Camera model fundamentals

A camera's main attributes are divided into the camera extrinsics and intrinsics, where the former refers to attributes describing the camera in world coordinates: translation and rotation. The latter depict the internals of a camera: focal length, optical center or principal point, skew coefficient, distortion coefficients. The latter denotes how much the camera lens warps incoming light before it hits the sensor.

The camera intrinsic matrix transforms 3D camera coordinates to 2D homogeneous image coordinates and the extrinsic matrix describes the location and orientation (pitch, yaw, roll) of the camera in world coordinates.

3.2.1 Inverse projection

While camera data is inherently two-dimensional, three-dimensional information, such as depth, in a scene can still be obtained. Combined with a depth value (z-value), the world to camera projection process can be reversed. This is called inverse projection.

The pinhole camera model is the simplest mathematical representation of a camera. It describes the projection of points in 3D coordinates onto an image plane in ideal circumstances, meaning little to no distortion, noise and calibration errors (see Figure 3.1 and Figure 3.2). By default, points behind the camera will also be projected to the image plane, so clipping of points behind the camera is necessary.

This mathematical model is reversible given a perfect depth map for the image plane; a format that stores a single depth value (relative or absolute) in every pixel of a frame or a video (also referred to as dense reconstruction or volumetric video). Every pixel of the depth map is then projected back to 3D using the pinhole camera model parametrised with the camera intrinsics and extrinsics. The result is a new point cloud.

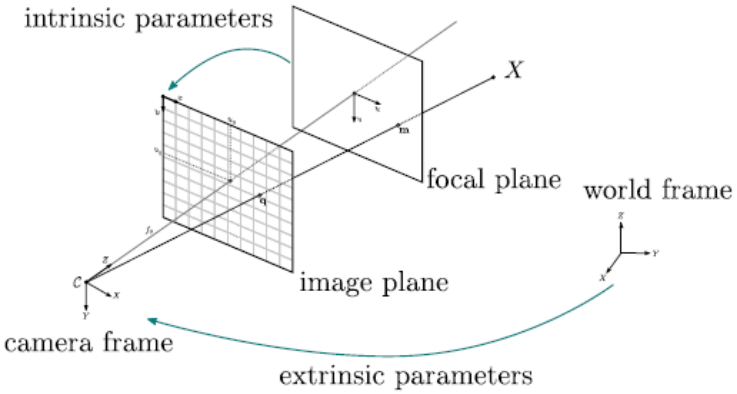


Figure 3.1: Pinhole camera model. Image from OpenMVG [1].

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic matrix}} \times \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{\text{Extrinsic matrix}} \times \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\text{World coordinates}}$$

Figure 3.2: Camera projection function. From OpenCV [2].

The radial and tangential are the two types of distortion coefficients. The former is the amount of light that is bent around the center of the camera, the latter denotes the alignment between the lens and the sensor. Radial distortion is mathematically defined as

$$u_{\text{distorted}} = u(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \tag{3.1}$$

$$v_{\text{distorted}} = v(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \tag{3.2}$$

and tangential as

[1] OpenMVG `https://github.com/openMVG/openMVG`
[2] OpenCV `https://opencv.org/`

$$u_{\text{distorted}} = u + (2p_1 * uv + p_2(r^2 + 2u^2)) \tag{3.3}$$

$$v_{\text{distorted}} = v + (p_1(r^2 + 2v^2) + 2p_2 * uv) \tag{3.4}$$

where $r^i = x^i + y^i$, k_i are the radial and p_i are the tangential coefficients respectively. The parameter k_3 is only necessary in fisheye cameras or when there is much radial distortion present. Distortion can be (largely) removed by refining the camera parameters or a remapping from the distorted to the undistorted image.

3.2.2 Camera parameters estimation

In situations where the camera's intrinsic or extrinsics are unknown, the parameters can be estimated [39]. The former can be approximated when the focal length, the resolution and the distance to an object in an image is known. The camera's optical center is then approximated by taking half of the height and width of the resolution. The camera's extrinsics can be computed using a Perspective-n-Point (PnP) algorithm, which, given a set of object points, finds the camera transformation matrix in the function 3.2 that minimises the reprojection error going from 3D to 2D coordinates (see subsection 3.3.3).

3.2.3 Camera systems

The two main types of camera systems are stereo and monocular vision.

Stereo. Images from two or more viewpoints of an object can be compared to find corresponding points between frames in order to estimate depth, similar to how humans perceive depth. The distance between two corresponding points, called the disparity, can be stored in a disparity map. Each value in such a map is inversely proportional to the depth at the corresponding pixel location in an image, also called inverse depth. This method is called stereo vision. The cameras need to be calibrated such that their external and intrinsic parameters are known. One advantage is that these cameras remove the need for training a model. With depth information, objects in a scene are easier to distinguish, making segmentation tasks more straightforward.

Monocular. Images can also be taken at different times from a single moving camera, or at the same time from many cameras, where the camera parameters are known or unknown.

From both of these configurations, the optical flow can be extracted. This refers to amount of visual motion in a scene relative to the camera,s which can be used to estimate both the camera's velocity and where objects are from the camera.

3.3 3D reconstruction technology

Capturing information in a physical scene can be done using 3D scanning technology. Naturally, objects are not (entirely) observed if they are behind another object. This occlusion is a limitation that occurs with every laser scanning and image capturing method. Unobserved objects could also be due to the speed the sensor or object is moving. The quality of the calibration and resolution of the sensor is also an important factor. Low sensor resolution can of course be remedied by installing more advanced equipment, but the cost of these sensors is very high. As discussed in Section 2.1, in the case of the RILA system, the scan line perpendicular to the track also makes it difficult to fully record objects that are also perpendicular to the track, such as railway signs or poles (see Table C.3, where the middle row shows laser scanned point clouds of signs).

Other problems for estimating the depth of a scene robustly include but are not limited to: noise, camera shake, dynamic objects, poorly textured image regions, reflections, repetitive patterns and occlusions [8]. Moreover, as far away regions naturally appear smaller, that information is grouped

together in pixels. This dense representation makes it difficult to estimate the depth of far-away objects more precisely compared to close objects in an image.

Different techniques can be used to collect the surroundings in 3D, with options ranging from sensing methods such as radar, laser scanning to optical methods such as SLAM, photogrammetry and computer vision.

All these methods try to reverse the projection process using depth estimation, i.e. finding the optimal z-value, with the help of geometric and/or learning methods. In some cases, the camera intrinsic and extrinsic parameters can also be recovered if these are unknown. We discuss these techniques in more detail below.

For improving alignment and scaling, a technique that can be used in the methods discussed below are Ground control points (GCP). GCPs are landmarks visible on the images for which the GPS position is known. Preferably, a single GCP should be observable in two or more images.

3.3.1 Laser scanning

Laser scanning works by a sensor sweeping a laser beam over its surroundings. Light pulses bounce off surrounding objects and return to the sensor. The time it takes for each pulse to return to the sensor, together with the position of the sensor, is used to compute a position of one point in 3D coordinates. Through this process, the sensor gathers many points to create a 3D map of the environment. LiDAR[3] is a very accurate form of laser scanning. The name stands for Light Detection and Ranging or Laser Imaging, Detection, and Ranging. The different types of applications for LiDAR are mobile (e.g. on the roof of a moving vehicle), airborne (e.g. from a drone) and terrestrial (e.g. from a static tripod) [48]. Points in a point cloud from a laser scanner do not include a colour property, but they usually have attributes such as reflection or intensity.

3.3.2 SLAM

SLAM stands for Simultaneous Localization And Mapping, a technique which reconstructs a map while a system of sensors is moving through an environment in real-time. While SLAM is a general notion and sensor agnostic, sensors are typically a monocular camera combined with a laser scanner. In that case, the process is also referred to as visual SLAM or vSLAM. Over the years, there has been more research into visual SLAM that uses mainly cameras, either monocular or stereo. The real-time performance requirement can also make the use of SLAM unsuitable in some situations, because of the high cost, heavy computational load and high quality of calibration it requires. It uses Structure from motion to estimate the 3D structure of a scene [36].

3.3.3 Photogrammetry and Structure from Motion

In contrast to SLAM, photogrammetry and Structure from motion SfM are not designed to work in a dynamic setting, as an online algorithm, with a map being built incrementally with every newly acquired image. Instead, an offline approach is taken.

Traditional photogrammetry approaches are designed to acquire 3D measurements of an object or scene, using (a combination of) different types of sensors. SfM is related to photogrammetry. It estimates intrinsics and extrinsics of the camera if these parameters are not available. As a result, the camera positions lack scale and orientation and the resulting point clouds are in relative coordinates. The transformation to world coordinates can be approximated using GCPs, making SfM a subcategory of photogrammetry.

In short, the default SfM pipeline starts by recognising and finding distinct features in different image pairs [46]. This pairing can be time-based, location-based, a custom pairing or a mixture. Subsequently, points corresponding between frame pairs are then matched as so-called tracks and outliers are removed. An initial triangulation is then computed between one pair of images that best fit the *cheirality* constraint, which implies that point correspondences in front of the camera are at finite distance. Finding a matching of image pairs with the maximum number of reconstructed points

[3]Laser scanner and LiDAR are used interchangeably in this thesis.

that satisfy the constraint means the correct unique 3D camera pose is found. If camera external and internal calibration parameters are known, points can be directly triangulated without this intermediate matching step. After the initial triangulation of one image pair, the 3D reconstruction is iteratively refined using bundle adjustment that minimises the projection error while adding more camera poses and 3D points. The reprojection error is defined as the sum of squared distances between all feature points in an image coordinates and a reconstructed 3D point projected onto the same image [35]. The depth information of the 3D points is stored in sparse depth maps. A dense MVS matching algorithm can then be used to generate denser depth maps for every image. Finally, a point cloud is created after merging the depth maps and (estimated) camera parameters. Different versions of algorithms exist for each step and can also be combined with machine learning (see subsection 3.3.3).

The complexity of SfM is in theory $O(n^4)$ with respect to the number of images, but an approximate linear time implementation that uses parallelization optimisations exists [56].

A linear forward moving motion can be a difficult scenario for SfM than rotating around an object. Features around the center of the image, i.e. the center of the horizon, are essentially at infinite distance and are estimated incorrectly (see Figure C.7). In addition, features at the sides move out of view faster than rotating around a center, resulting in relatively short track lengths that are difficult to triangulate. We can observe this behaviour in Figure C.7. Another challenge for SfM in the feature selection process is the presence of reflective and repetitive surfaces. In these areas, points in pairs of image frames are difficult to distinguish. Consequently, these parts are filtered out or incorrectly computed, leaving holes or creating noise in the resulting depth maps and point cloud.

3.3.4 Computer vision

Computer vision methods have demonstrated impressive results relating to object recognition and detection, classification, tracking from camera input. In addition to 3D reconstruction, estimating depth from camera data has applications in a variety of other fields. For example, video-based applications benefit from knowing the corresponding depth of every pixel in a camera frame, such as 3D video stabilisation, virtual reality, augmented reality and special effects.

While techniques such as SLAM and SfM perform better for stationary scenes with a calibrated camera system, recent developments in the computer vision field have achieved good results with hand-held smartphone cameras without a priori knowledge of camera parameters [31]. By training networks on ground truth depth information from (a mixture of) laser scanner and camera data, they can learn to infer the relative depth from an image or a video. More about current research will be discussed in the next chapter.

Chapter 4

Literary Review

Scanning real-world objects and automatically processing them into digital assets has been studied extensively in the last few years. For example, applications range from augmented reality, special effects to robotics and self-driving cars. All require mapping and detecting objects in their surroundings in real-time.

In this chapter, we explore literature that studies the problem of accurately reconstructing surroundings using either point clouds, camera data or a fusion of both.

4.1 Point cloud processing methods

Although point clouds are already a representation of a scene in three dimensions, some processing might still be required to create a useful reconstruction. The unordered and continuous characteristics of point cloud makes extraction of features, which are necessary for performing further processing and reconstruction, challenging for existing deep learning models. This required modifications of existing models and developing new methods to process this data format. Later in this section, we discuss some approaches. These approaches require different ways to represent a point cloud.

Voxel representation. Voxelisation of a point cloud discretises the continuous coordinates of the unordered set of points into a 3D uniform grid. 3D convolutional neural networks (CNN) can then be applied to this regularised data [64, 13]. While voxel-based features enable the application of CNNs, voxels have a cubically growing memory cost when scaled in terms of resolution. In addition, geometric (and semantic information in case of classified points) is naturally lost due to the discretisation of the points.

Point representation. Point representation based methods directly operate on the point cloud without transforming it into an intermediate data structure. The classification and segmentation network PointNet [40] and the completion network PCN [61] have laid important groundwork for deep learning in point clouds without voxelisation as a preprocessing step, but instead employ a point-based method. In contrast to voxel-based features, point-based features provide an accurate representation of 3D structures.

Hybrid representation. Although accurate, extraction of features is computationally expensive in point-based methods due to a larger number of points and random memory access patterns compared to voxels. VoxelNet pioneered the use of voxels as features, but becomes slow when dealing with a large amount of voxels [20]. Some newer methods therefore adopt a hybrid, point-voxel representation of point clouds to solve 3D computer vision problems such as object detection or shape completion [59, 62]. Hybrid point-voxel represents '3D input data as point clouds to take advantage of the sparsity to reduce the memory footprint, and leverages the voxel-based convolution to obtain the contiguous memory access pattern' [30]. [59] also leverages voxel-based feature encoders but represents each voxel using 'pseudo-image feature maps', on which 2D CNN operations can be

applied.

4.1.1 Object detection and segmentation in 3D

Traditional feature-based methods made use of statistical learning models (edge detection, corner detection or threshold segmentation) for detecting and segmenting objects [33]. Over the past few years however, learning-based approaches became the standard for detection and segmentation in 3D [12]. Some state-of-the-art networks are PointNet, VoxelNet and PointConv [57]. These networks are able to classify sparse or incomplete point clouds, if the input point cloud contains features the models have been trained on. The detection of objects is commonly used in the field of autonomous driving, where vehicles, pedestrians and cyclists are detected from front-facing LiDAR input. The methods mentioned previously take a point-based approach, i.e. they train directly on features from point clouds.

4.1.2 Shape completion

As discussed in Chapter 3, sparsity in point cloud data points from the real-world is often a result of occlusion and limited sensor resolution. This causes loss in geometric and semantic information. Some processing might then still be desired to generate workable data. 3D shape completion methods are designed to recover structures from partial or sparse input. Shape completion can be categorised in three main approaches.

Geometry-based. This approach uses geometric information from the partial input only to complete shapes. To reconstruct surfaces of a shape, interpolation can fill holes in local regions and symmetry axes can be identified such that structures can be mirrored [6, 38]. While this approach is relatively straightforward, since no additional data is required, one major drawback is that the method requires near-complete inputs.

Alignment-based. Similar shapes can be aligned with the input for comparison and completion of the final output, where the shape data comes from databases. This data-driven technique is not suitable for online applications, considering it takes much time to compare shapes [61, 11].

Learning-based. In a learning-based approach, the completion of shapes is learned by a deep neural network. The advancements in deep learning have contributed to significant performance improvements for learning-based approaches compared to the previous two [19]. Despite the success of deep learning, traditional methods are still better interpretable and consume less computational resources [12].

When points belonging to different objects are within a bounding box of another object, a segmentation step might be required before applying a shape completion approach [61].

In some cases, these processing methods may not work properly. If a point cloud is very sparse or significant parts of objects are missing, object detection methods will not identify objects, segmentation methods may fail to classify points correctly and completion methods cannot reconstruct shapes accurately.

4.2 Camera processing methods

Deep learning improved the processing of point clouds, while also revolutionising the field of 2D computer vision. Results of non-trivial tasks such as object detection, recognition, classification and segmentation have reached, and in some cases exceeded, levels of human vision [23].

4.2.1 Depth estimation networks

To estimate depth, models often exploit geometric and observational cues, such as the horizon line, object shapes, occlusion, textures and poses. Object detection and segmentation networks such as

Faster R-CNN [44] and Mask R-CNN [22] can support this task by masking out dynamic objects that would otherwise make the depth estimation more difficult.

Creating a depth map or video recently enjoyed success with monocular image [41, 34, 15, 26, 51] or video [31, 26] input. Some computer vision networks [31, 26] are also able to estimate depth without prior knowledge of the camera intrinsics and external calibration parameters.

Recent research from [47] proposes that stereo image pairs as input currently still yields better results than monocular depth estimation. Incorporating different inputs from stereo, laser scanners or synthetic scenes when training a network model increases the ability of a depth estimation network to generalise well on different scenes[41].

One main disadvantage of accurate depth information from stereo of laser scanners is the acquisition, as it is very time consuming and costly to collect. Moreover, dense depth maps can be created from sparse LiDAR input [1], but the result will still contain holes an inaccuracies where the input is very sparse. However, a depth estimation model trained on the more accessible synthetic data might not generalise well to real-world data, since the input does not contain any natural artefacts. According to [27, 4], this approach is not directly suitable for real-world applications, unless the synthetic data is either mixed with multiple sources of data or a style transfer is done on the input before inference. With style transfer, colours and textures of a reference image are used to merge it with the contents of another image.

Considering these issues around acquiring data for supervised training, researchers have also developed self-supervised or unsupervised pipelines to circumvent them [9, 18, 63]. Only monocular or stereo image sequences are used for training in this case. Based on the estimated camera poses and synthesizing novel views using SfM as supervision signal, the pipeline becomes similar to a MVS system. Depending on the implementation of the network, training on cues from solely monocular video input in an unsupervised manner can achieve good results, but are still lacking in accuracy compared to supervised methods [43].

Pseudo-LiDAR

Depth or disparity maps can be fused with 3D laser scanning data to improve 3D object detection, for example as studied in [24, 28]. The idea is training a model with multiple data sources in 2D and 3D, so camera-LiDAR or multi-sensor fusion if more sensors are involved, to combine the advantages of both data types.

Pseudo-LiDAR mimics the output of an actual LiDAR sensor by taking a general depth estimation model to predict a sequence of depth maps and projecting the depth maps to 3D coordinates using the pinhole camera model. With camera-LiDAR fusion that creates a pseudo-LiDAR point cloud, results for object detection and segmentation in the KITTI dataset were improved [12]. Works such as [54] created pipelines for generating reliable pseudo-LiDAR point clouds for 3D object detection from images captured in traffic. However, pseudo-LiDAR implementations also suffer from a few issues. The main problem arises from errors in estimating the z-value, since these are very noticeable in 3D. In addition, incorrect camera intrinsics can result in a point cloud skewed towards the camera.

In [60], the researchers therefore focused on using stereo images for their depth estimation training and improving the initial depth estimation. The depth of the output point cloud was corrected with a sparse LiDAR input originating from a less expensive laser scanner.

4.2.2 Structure from Motion

From Chapter 3, we discussed that Structure from motion geometrically computes the depth of every detected feature in matched image pairs to reconstruct a scene. According to [26], SfM is error-prone for scenes with dynamic objects if these are not filtered out and reconstructions typically contain holes, i.e. parts that lack depth information due to failure of the reconstruction algorithm. In another paper where SfM was specifically implemented for railway environment, it was concluded that while the general shape can be reconstructed well by SfM, details are left out [16]. Methods have also combined SfM with deep learning in order to reduce errors in depth and pose estimation.

In [55], the bundle adjustment stage is replaced by two networks, one for depth and one for pose estimation. These are iteratively run for multi-objective optimisation.

4.2.3 3D object detection

3D object detection, which is also known to as 6D (location and orientation) pose estimation or 3D object recognition[1], infers both the class and pose of an object. Estimating the pose of an object and drawing a 3D bounding box around can be seen as a form of depth estimation, since the spatial transformations of an object need to be approximated from 2D features (keypoints).

Networks such as pseudo-LiDAR can support the task of 3D object detection for autonomous driving, for example in detecting other vehicles. The depth information make 3D bounding box predictions more accurate. Nevertheless, depth predictions are used directly as input for the 3D object detection stage. This causes networks to be overconfident in their depth estimates, which degrades localisation performance [42].

Alternatively, there exist networks that are directly trained on annotated bounding boxes in images. Typically, these networks consist of a two-stage pipeline, where the first stage detects an object and finds the 2D bounding box around it. The second stage then estimates the 3D bounding box from the 2D bounding box. In some cases this is done for every frame, but [3] also computes the 2D crop of the object for the next frame, such that the object detector does not need to run every frame. Their single-stage model jointly predicts an object's 2D bounding box with detection to find the 2D centroid and 3D bounding box with regression. The regression task approximates the 2D coordinates of the eight box points. The 3D coordinates of the bounding box are acquired by a PnP pose estimation algorithm (Efficient PnP).

SMOKE [29] adapts a similar framework, except it both classifies and predicts the 3D bounding box directly without a 2D detection stage. It is done by pairing each object with a single keypoint in 2D coordinates, from which the centroid and the 3D bounding box itself can be estimated using similar methods to Objectron. The advantage of 3D object detection is the focus on objects and semantic information, instead of inferring the depth of every pixel in an image.

Research shows that samples with distant annotated objects worsens the performance of the model and that 2D detection can still be an important intermediate step [32].

Related to 3D object detection and SfM, is the generation of 3D geometry of individual objects from monocular images. This is done by supervised training on datasets with ground truth meshes or point clouds [10, 52]. The main disadvantage of this method is the sparse availability of these datasets, and the large effort it takes to create this data. The method from [25] use segmentation masks together with point clouds to detect bounding boxes of objects by predicting a complete mesh, instead of relying on 3D annotated bounding boxes.

4.3 Datasets

Ground truth depth maps are often not available in real-world datasets, such as KITTI or nuScenes. KITTI [37] is a dataset created for autonomous driving purposes and it is widely used throughout the computer vision research field to evaluate learning-based models.

However, research suggests a performance bias present in the validation scores of the KITTI benchmark for 3D object detection. In [45], researchers concluded that there exists an overlap between the training and validation set, and that the benchmark for the 3D object detection compares models based on the performance of the validation split instead of the test split. This overlap and way of evaluating results in higher reported performances than if the sets were disjoint and results were reported of the testing split (also see Train-Test-Val split).

The nuScenes dataset [17] is also a large dataset for autonomous driving. Unlike KITTI, difficulty of detection is not used as an evaluation metric, and there are different measures for individual properties of a 3D bounding box instead.

[1]The concepts of detection and recognition are used interchangeably in this thesis

Synthetic datasets generated by games or specialised software are a cost-effective alternative that provide very accurate ground truths. Depth maps have to be computed to render a scene, thus they are readily available and therefore straightforward to extract. The game GTA V and the CARLA simulator [14] are examples of such sources often used for generating autonomous driving datasets.

Chapter 5

Methodology

Subquestion 1. *How can depth estimation improve the point cloud quality and localisation of wayside objects?*

In this chapter, we describe the approach to address the research question.

The laser scanner point clouds, the video frames of the middle camera and the internal and external calibration parameters for the camera and every frame are used as input data for conducting the experiments. Unlike hand-held footage, the calibration quality of our data is very high, which is an advantage for any depth estimation task. In addition, the environment is quite static, i.e. there are very little dynamic objects present in the data.

Even though the literary research suggests stereo cameras to be very useful for depth estimation, there is too little overlap ($< 50\%$, see Figure A.1) in the outward facing sensor system to function as a stereo camera. Therefore we only use the monocular camera input. We consider the data points in our LiDAR generated point clouds and annotated map to be the ground truth measurements and classifications of our railway environment, taking small calibration and rounding errors into account. Considering we do not have a dense complete ground-truth point cloud available of the environment and wayside objects, approaches such as shape completion are unsuitable. Moreover, our data does not contain segmentation masks, to be used in methods such as [25].

As discussed in Chapter 4, unsupervised approaches are not relying on perfect ground-truths, but rather create supervision signals from merely the camera data. However, the state-of-the-art performance of these networks currently tends to lack behind supervised methods.

Furthermore, we cannot use the laser scanner point cloud to train an object detection model or create ground-truth depth maps for a depth estimation network, because of the sparseness of the data (see Table 7.2c) and different projection direction (see Figure 2.2). Finally, it is also not possible to train a monocular depth estimation network on generated depth maps from SfM. Initial tests showed that these depth maps are generally too sparse (Table 7.2c).

Three different methods were investigated: pseudo-LiDAR using a depth estimation network, Structure from motion and 3D object recognition. Respectively, we can loosely categorise their approaches into *general depth estimation*, *geometric depth estimation* and *object depth estimation*. We define object retrieval as retrieving either of the following 3D information of a wayside object:

1. The object's shape in the form of a point cloud.
2. The object's class and its 3D pose (location and orientation).

Furthermore we define three major types of objects that are of interest for comparison of the three methods:

- Signal lights.
- Electrical cabinets.
- Markers (signs).

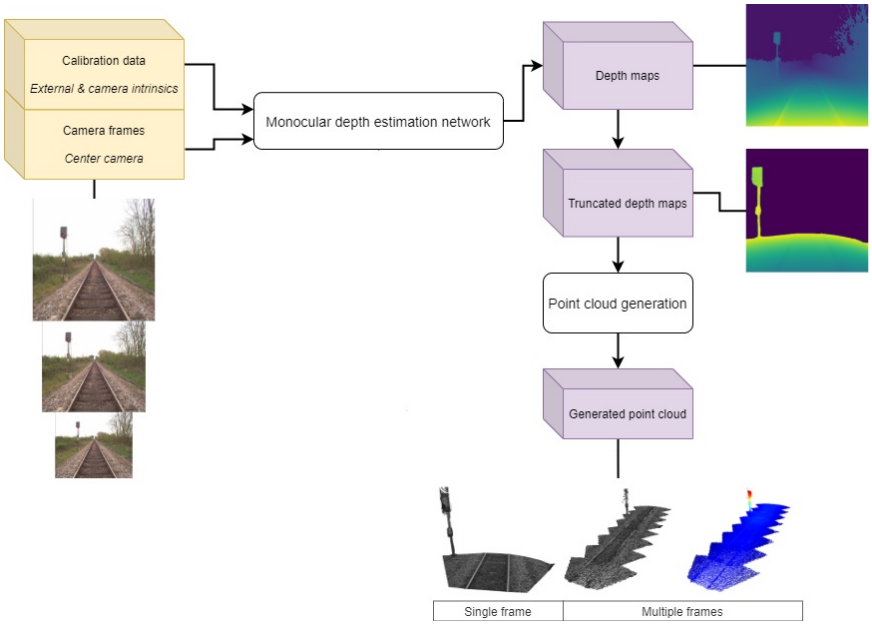


Figure 5.1: Pseudo-LiDAR pipeline.

5.1 Pseudo-LiDAR: MiDaS

The first method is based on the pseudo-LiDAR pipeline from [54]. Their approach generates a pseudo-LiDAR from either stereo or monocular images and is agnostic to different depth estimation networks. We use an existing monocular depth estimation model to create a depth map for every frame. After initial tests on the video data with different models Figure B.1, we picked the MiDaS model for our pipeline. This model is a general depth estimation network, as it was trained on 10 datasets with multi-objective optimization. Initially, it gave the most promising results out of the box on our data. For example, sky was completely removed and objects were present in the resulting point cloud. The complete training pipeline is not made public, but the loss function is readily available.

In short, the MiDaS model proposes a combines a gradient regularization term which discourages learning a complex model, with a scale- and shift-invariant trimmed loss (SSITrim) that trims the largest values in every image. These losses can handle data from different sources of datasets with different scaling and shift stereo disparities. Both the prediction and the ground truth are mapped to a unit scale. The output of the model is the relative inverse depth, which can be inverted again using the focal length to get the relative depth.

First, the network infers the depth of a certain number of frames. We then truncate every depth map by a fixed distance of 3 meters in front of the camera to keep reliable depth information before projecting it into a point cloud in the GPS coordinate system. The projection is done with the pinhole camera model using the provided intrinsic and extrinsic camera parameters. All distortion coefficients except the k_3 parameter are used, because our camera is not very distorted.

Afterwards the generated point cloud, which is in relative coordinates, is scaled with fixed constant to match its size with the real-world. This scaling maps the relative depth value on a unit scale to the world scale. We acquire the scaling value by measuring the actual distance (in meters) of at least two pixels to the camera and averaging it.

Even though the goal is to reconstruct wayside objects, we use all depth information in the truncated frame (including vegetation, the track, etc.). The reason is that segmenting wayside objects from an image results in essential information loss. Important features for estimating depth are removed from the input, complicating accurate predictions. Initial tests confirmed that only performing inference on a part of an image resulted in a worse depth map than without cropping.

5.1.1 Synthetic data

Since the MiDaS model is general-purpose and not trained on rail data specifically, we try to transfer learn the model. However, as explained in the beginning of this chapter, ground-truth depth maps cannot be generated from our data. Instead, synthetic data from the railway simulator game Train Sim World is used as training data. We test the pipeline for both the regular MiDaS model and the transfer learned MiDaS.

5.2 Structure from Motion: OpenSfM

For the SfM pipeline, we apply the open-source program OpenSfM[1] [2]. This framework provides an incremental Structure from motion algorithm that can be configured with different settings for every stage of the reconstruction. We use most of the standard settings and match every image based on GPS coordinates. All settings can be found in our documentation and code.

5.3 3D object detection: SMOKE

Thirdly, the data was fitted into an existing 3D object detection model called SMOKE. Originally developed for the KITTI dataset, this single-stage model predicts 3D bounding boxes using keypoint estimation directly from an image. In this model, each object is represented by a keypoint, from which the eight corners of a bounding box can be computed. The keypoint is defined as the 3D centroid of an object in 2D projected image coordinates. The loss function contains two components that jointly work together: the keypoint classification and the bounding box regression losses. The former predicts a keypoint with the class to which the object belongs and the latter estimates the geometry of the 3D bounding box from the keypoint. For the regression loss, the average dimensions of a class are used as baseline for prediction. In terms of accuracy, a value reported above 80% on the validation split is acceptable as required by the company. subsection 6.3.3 describes in more detail how the accuracy of the model is evaluated.

In our case, we do not directly have 2D bounding boxes. The 3D bounding boxes are extruded from flat annotated data to an average fixed height of that certain category, since manually adding a height to every object would be too tedious. The resulting bounding box is therefore not entirely correct for some wayside objects. 2D bounding boxes could be roughly acquired from ground-truth 3D bounding boxes, but similar to the 3D bounding boxes, this does not reach the level of human-made annotations.

One advantage of this model is that it does not rely on learning with ground-truth 2D bounding boxes. The authors of [29] argue that since 2D information can be naturally obtained if the 3D variables and camera intrinsic matrix are already known, the SMOKE model predicts 3D bounding boxes as well as 2D bounding boxes (see Figure D.3). The original SMOKE model was trained on a single object class, namely vehicles, but can be adapted to perform classification on multiple classes.

[1] `https://opensfm.org/`

Chapter 6

Experiments

Subquestion 2. *How can we measure the effectiveness of object retrieval?*

In this chapter, we describe the details of the experiments and address the second research question: *how can we measure the effectiveness of object retrieval?*. For the first two methods, the resulting point clouds are qualitatively compared to each other and the LiDAR data, as well as quantitatively, by doing a point cloud to point cloud (C2C) distance analysis using the open-source software CloudCompare [1]. The distance analysis works by computing for each point of the first point cloud the distance to its nearest neighbour in the reference point cloud (i.e. the point cloud that does not move). According to the authors of CloudCompare, this type of metric works well when the overlap between the point clouds is high, the reference point cloud has a high density and the reference point cloud has a larger footprint than the compared point cloud. Other common metrics, such as Earth's Mover Distance (EMD) and Chamfer Distance (CD) are limited to point clouds with the same point count and susceptible to noise as the former relies on one-to-one point correspondences and the latter is high when large portions between point clouds differ [61, 50]. By requirements of the company, distance errors up to 3cm are acceptable.

Since our LiDAR data misses points in the wayside objects, points that are correctly placed by depth estimation methods in areas with lack of data, these points would be incorrectly considered as outliers. Therefore we measure the overall registration quality to balance out the differences in density in the LiDAR data, instead of analysing a selection of wayside objects.

6.1 Pseudo-LiDAR

As the training code was not provided, a standard training pipeline was used to transfer learn MiDaS on the synthetic data. We train MiDaS for 100 epochs on 600 images, with 70% in the training set and 30% in the validation set with a batch size of 2 and learning rate $1e-05$. See Figure 6.2 for the training graph. Experiments for MiDaS were run on a Intel Xeon 2.20GHz CPU and 32GB RAM system running Ubuntu. MiDaS v2.1 and Python version 3.8 was used. Synthetic data was extracted using RenderDoc v1.17.

Performing segmentation after inference on the complete image to extract only wayside object information is still possible, given the great performance of current detection and segmentation networks. However, in Chapter 8 we explain why this was not beneficial for pseudo-LiDAR.

[1] `https://www.danielgm.net/cc/`

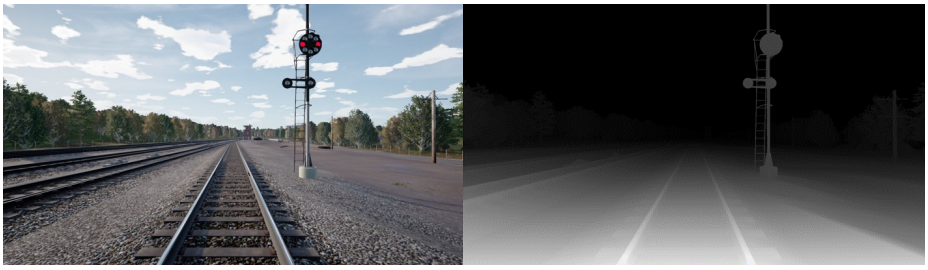


Figure 6.1: RGB image and extracted depth map.

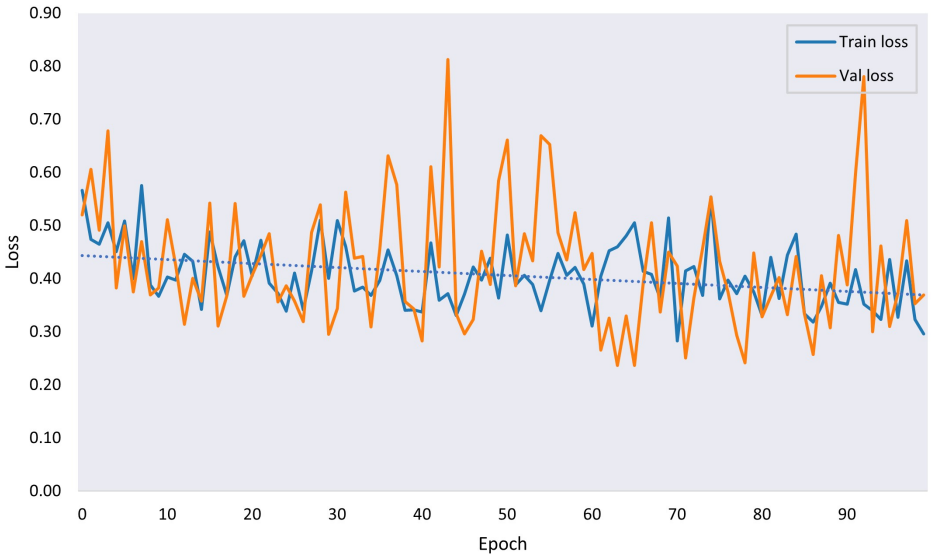


Figure 6.2: MiDaS training graph of the training with trend line (dotted) and validation loss.

6.2 Structure from Motion

Experiments for SfM were run on a Intel Core i7-7820HQ 2.90GHz CPU with an NVIDIA Quadro M2200 GPU running Windows 10. OpenSfM v0.5.1 and Python version 3.8 was used.

As discussed in subsection 3.3.3, transforming the point cloud from the SfM pipeline is in relative coordinates, making a transformation to world coordinates necessary. Since we have the accurate GPS calibration of the camera, we add the location coordinates and GCP settings to approximate this.

In Chapter 8 we elaborate why adding an automatic segmentation step after creating point clouds to extract wayside objects automatically for measurements using SfM was not beneficial.

6.3 3D object detection

We use an open source toolbox called MMDetection3D[2] for general 3D detection, which contains several object detection models including the SMOKE model. We first test the SMOKE for 25 epochs (3 hours), merely to confirm that the training losses would indeed decrease. Then we trained it for 72 epochs (9 hours), because both the performance and errors stayed stable after around 55

[2] `https://mmdetection3d.readthedocs.io`

epochs and the performance was over 80% on the validation split. Apart from adjusting the codebase to use our custom classes, most of the settings for training were kept the same.

See Figure D.1 and Figure D.5 in the appendix for the training results on the validation set with 25 epochs.

Object class	**Total**	**Occurrence in image**			**Mean dimensions in meters (h/w/l)**
		Train	Val	Test	
Signal Light	65	299	77	66	5.0 (0.0)/1.99 (0.47)/1.99 (0.47)
Cabinet	190	582	127	114	2.5 (0.0)/0.85 (0.27)/0.85 (0.27)
Marker	143	760	154	153	2.01 (0.04)/1.06 (1.21)/1.06 (1.29)

Table 6.1: Total number of objects, total number of objects seen in images and average dimensions of 3D bounding boxes per category. Number between brackets denote the standard deviation.

6.3.1 Data preparation

Training the SMOKE model requires the data to be in the KITTI format. This format dictates that for every image frame, a calibration file with camera intrinsics and a file with annotations of bounding boxes (which is empty if there are none) and a point cloud (if available) is present. An annotation stores the type and pose in camera coordinates of the bounding boxes. By applying a transformation from camera to world coordinates using the camera extrinsics, the bounding boxes can be accurately aligned with the LiDAR data. In the KITTI dataset format, every object has an angle of observation. This is illustrated in Figure B.2, where in the current configuration, every bounding box is oriented towards the camera at a perpendicular angle, similar to the vehicle on the left.

In total there are 6791 frames, consisting of roughly 70% empty (negative) camera images. However, fitting a model on large amounts of negative data negatively impacts the performance, since the data would not be representative enough to learn important features. Thus the model would learn to predict fewer bounding boxes or none at all. To remedy this, we first project all objects within 30 meters in front of the camera and only include objects with all 8 corners present in the frame. Every empty data sample that does not contain an object is filtered out.

The remaining non-empty data of 1528 frames is randomly split into a standard Train-Test-Val split with 70% train, 15% validation and 15% test data. Every empty data sample is then added to the test subset, making the test set much larger than the train and validation set. Still, the number of positive instances containing an object is still 15%. So, having a large quantity of negative samples in the testing set as well also demonstrates the real-world potential of the model than without negative samples.

The training split contains 1068 images, the validation split 231 images and the test split 5492. Only the test data is not seen during training.

We remove any degenerate cases of labels where bounding boxes are present in an image but the object is occluded from the camera. An example of such a case is shown in Figure D.2. We also refine the classes to clean up the dataset: we remove signal lights not meant for railway, very small markers or markers that are not for railway. Since a fixed height is used for all objects within a class, markers that are very short will have a very tall bounding box. For the cabinets, we remove ones that are in reality small buildings instead of a small box next to the track. While taking out samples from the complete dataset seems counter-intuitive, including these instances in the three classes would result in a noisy dataset with incorrect samples. Instead, for future reference, these objects should have classes or class attribute of their own. The removed objects are documented with the reason of removal in the code. Further details of this conversion can be found in the code.

6.3.2 Training

For training, the default settings with the Adam optimizer and a learning rate of 0.00025 were used and the batch size is 8. We divide the original image resolution by half to 1008×1008 and padded by 32. Evaluation is done every 5 epochs and images are randomly flipped or shifted and rescaled with a probability of 50% and 30% respectively. The training experiments for SMOKE were run on a Intel Xeon 2.20GHz CPU with 32GB RAM and an NVIDIA Tesla P100 GPU system running Ubuntu. Python version 3.7 was used.

6.3.3 Data evaluation

The standard nuScenes evaluation metrics were used instead of the KITTI benchmark. The reason for this is the detection difficulty score of KITTI (easy, moderate, hard), a measure based on certain attributes of an object such as the level of occlusion, truncation of the object leaving the image boundaries and the height of the 2D bounding box. For our case, the occlusion and truncation level of all objects were set to 0, i.e. fully visible and non-truncated respectively. Annotating the occlusion manually would take too much time. As mentioned in Section 5.3, we do not have annotated 2D bounding boxes in the data. For these reasons, the KITTI difficulty score would not give representative results.

The following 3D bounding box attributes for true positives matches are computed:

- **AP**: Average Precision, where predictions are matched with the ground truth objects that have the smallest 2D center-distance on the ground plane. If this distance is within a fixed threshold and the class is correct, it is considered a true positive (TP) match. The default thresholds are $0.5, 1, 2, 4$ as defined by nuScenes.
- **ATE**: Average Translation Error, Euclidean center distance in 2D in meters on the ground plane.
- **ASE**: Average Scaling Error, computed as 1 - `IoU` after aligning centers and orientation.
- **AOE**: Average Orientation Error, which is the smallest yaw angle difference between prediction and ground-truth in radians, which is at most $\frac{1}{2}$ rad.
- **AVE**: Average Velocity Error, absolute velocity error in `m/s`.
- **AAE**: Average Attribute Error, calculated as 1 - attribute classification accuracy. Attributes are attached to classes and provide extra information about an instance. For example, vehicles can have the attribute moving, parked or stopped.
- **NDS**: NuScenes Detection Score, the weighted sum over all mean scores, on a scale of 0 to 1.

Predictions are matched once, it is thus not possible to match one inferred bounding box with multiple ground truth bounding boxes. The matching starts with the prediction that has the highest confidence value, which the SMOKE model scores to each bounding box.

We use the default $0.5, 1, 2, 4$ meter thresholds for matching 2D center-distances between two bounding boxes for the AP scores. We compute the precision and recall curves for the three classes on both the testing and validation split. Precision is the rate between TP predictions and both TP and FP (false positive). Recall is the rate between TP predictions and both TP and FN (false negative), indicating how accurately a model is able to recall relevant data. Every classifier model outputs a confidence along with the predicted class. By setting a threshold to this confidence value for rejecting or accepting a prediction when lower and higher than the threshold respectively, the recall and precision values will be affected. The recall and precision curve visualises these changes by plotting for each threshold from > 0.1 the recall and precision of the bounding box classifier. For every matching $0.5, 1, 2, 4$ threshold, the AP is computed by the integral of the recall and precision curve. Then the mean over all these integrals is taken to get the mean AP over one class.

As our objects are stationary and do not contain extra attributes, the code was changed to run the nuScenes evaluation on the KITTI format with only the ATE, ASE and AOE errors and the

NDS score based on the mean of these values. We run the evaluation method on both the validation and test split.

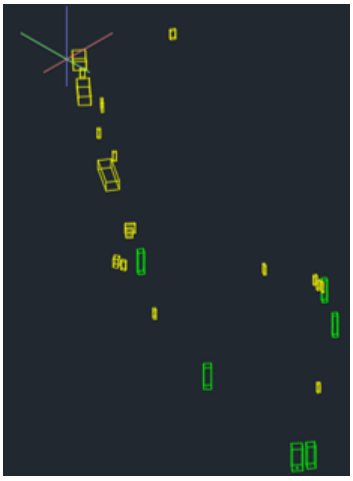


Figure 6.3: Example area of bounding boxes extracted and extruded from the ground-truth annotated map.

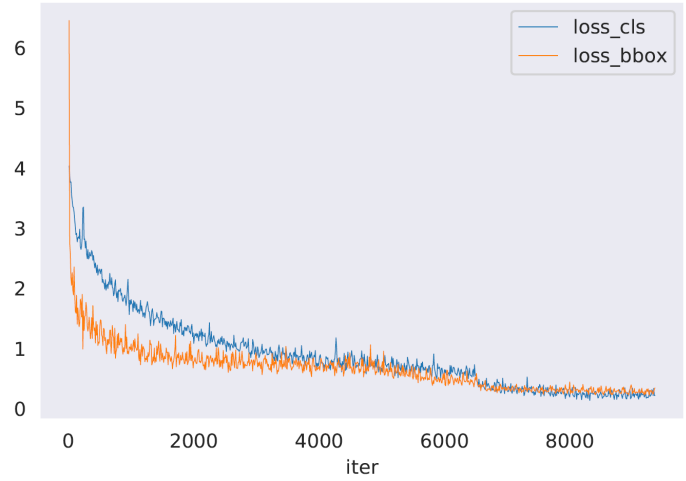


Figure 6.4: SMOKE average training losses of object classification and bounding box estimation for every iteration. One epoch took 231 iterations.

Chapter 7

Results

Subquestion 3. *How does depth estimation compare to laser scanner and annotated data?*

7.1 Pseudo-LiDAR

From the training graph of MiDaS on the synthetic data in Figure 6.1, a few observations can be made. Usually, the training loss should decrease significantly (multiples of 0.1) after a few epochs [5] Epochs and iteration and batches, as seen in Figure 6.4. However, the spikes and a very slow decreasing average indicate that the validation and testing errors have only dropped about 0.05 in loss, even after 100 epochs.

Since the transfer learning of MiDaS on the synthetic depth maps did not improve the model, we only show the pseudo-LiDAR visualisations of the standard MiDaS network. These pseudo-LiDAR point clouds show many distortions in the wayside objects, as seen in the example in Figure 7.1. In Figure B.3a and Figure B.3b we show the full segment with the corresponding histogram of the distances. While the tracks looks acceptable in this example, a cross-section of this area as seen in Figure B.4 reveals it is also warped with inaccuracies up to around 9cm.

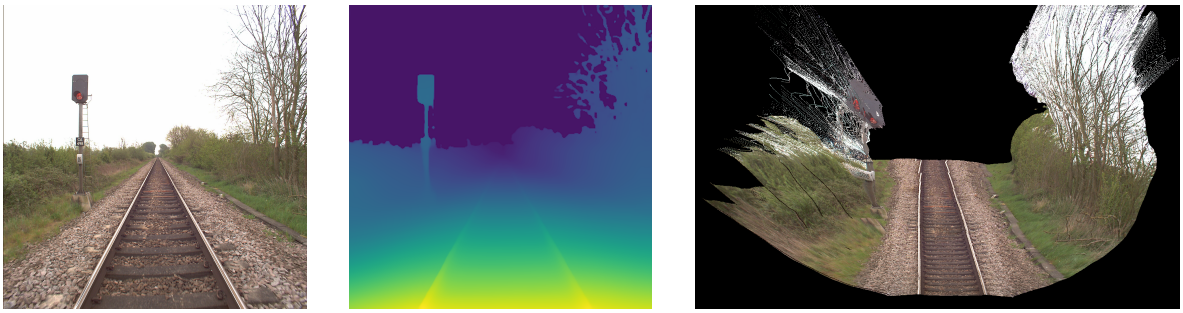


Figure 7.1: Pseudo-LiDAR point cloud (right) created from the camera image (left) and depth map (center) inferred by the MiDaS network.

7.2 Structure from Motion

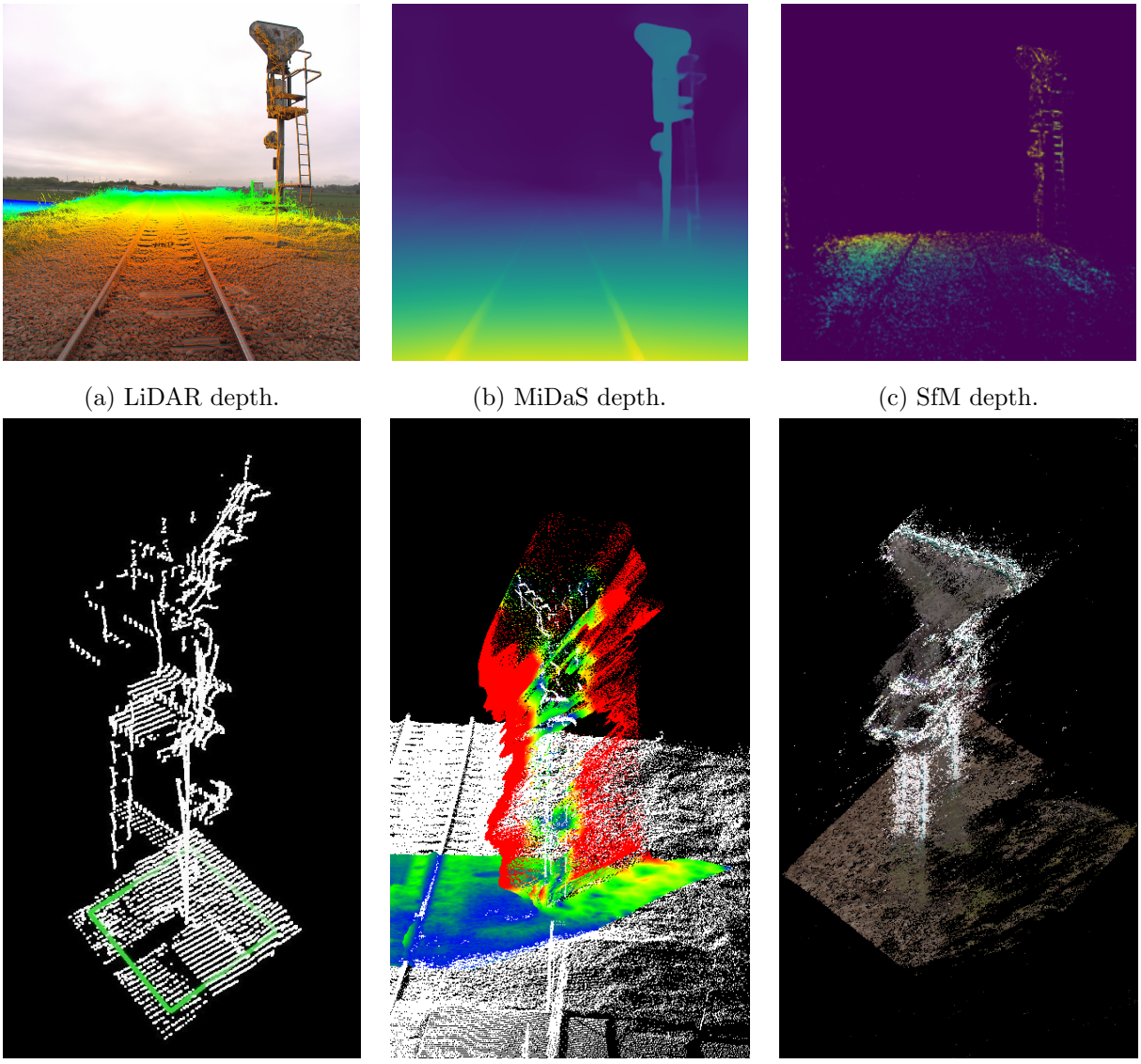
Compared to pseudo-LiDAR, the SfM point cloud in Figure 7.2 shows qualitative improvements, because SfM point cloud is more similar to the wayside object as seen in the LiDAR and camera image. A comparison of wayside objects per category can be found in the appendix in Figure C.1, Figure C.2 and Figure C.3. A few observations can be made, namely that geometry is more defined compared to LiDAR in the categories signal lights and markers. For example, in the last column of the SfM row, we can clearly see the complete circular shape and rectangle parts of the sign.

Objects with a more uniform surface or that are further away from the camera are less detailed, as for example seen in the second column in the SfM row of the cabinet category.

In contrast, the scale and GPS alignment is incorrect (see example in Figure C.5). The scaling and alignment of SfM did improve somewhat with GCP settings enabled, but not enough since the point cloud was still not roughly aligned with LiDAR (see example in Figure C.6). Scaling manually to the same size as the LiDAR point cloud caused new alignment issues, where it was not possible anymore to place objects in the correct position and would not be properly reproducible. Therefore, we roughly aligned the point clouds and we align them by using the ICP algorithm in CloudCompare with scaling enabled and using the default settings.

In addition, the point clouds also have quite some white noise from the sky background in the image and there is often no track present (example shown in Figure C.4). These are caused by limitations in the algorithm for reflective and uniform surfaces as discussed in subsection 3.3.3.

After the ICP transformation and computing the C2C distances for a part of the track with 30 meters in length, the quantitative results of SfM in Figure 7.3 show that that points closer to the ground generally have a better accuracy than higher points. In Figure 7.4, the histogram shows that many points are around 10cm away from their nearest neighbour or have a distance $>$ 50cm (C2C distances are capped at 50cm).



(a) LiDAR depth. (b) MiDaS depth. (c) SfM depth.

(d) LiDAR. (e) Pseudo-LiDAR. (f) SfM.

Figure 7.2: Top row shows the depth maps of LiDAR on top of the camera image, pseudo-LiDAR and SfM respectively and the bottom row the corresponding point cloud of the wayside object, a signal light. In (e), the colours (red-green) depict the distances to the LiDAR point cloud (white).

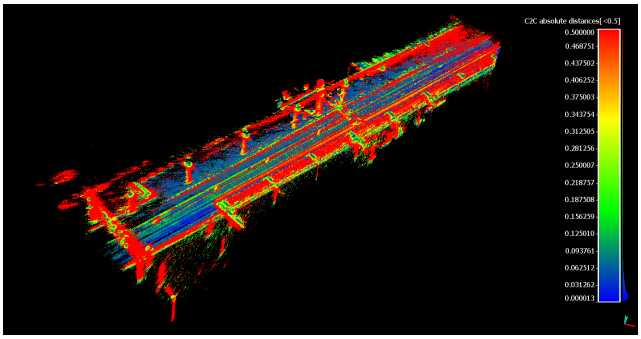


Figure 7.3: SfM versus LiDAR distances in a segment of a station, an area with relatively more objects and features than e.g. a grassland region. The SfM was translated and scaled using ICP.

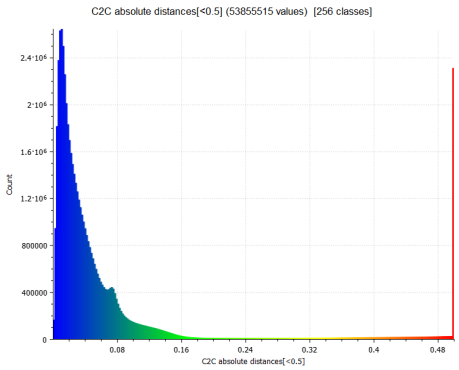


Figure 7.4: Cloud-to-cloud distances histogram. C2C distances larger than $>$ 50cm are automatically capped at 50cm.

7.3 3D object detection

The results from both the training graph in Figure 6.4 and evaluation metrics in Table 7.1 demonstrate that the SMOKE model was able to learn adequately on our data. The validation evaluation shows better performance than the testing split, since the testing split is unseen data only. This subset also contains many empty samples, so there are cases where the model predicted an object that is not present (see Figure 7.8).

A large area under the precision and recall curve indicates low false positive rate and low false negative rate for the matched bounding boxes. Figure 7.7 shows that the classification of the validation split has a higher precision and recall rate compared to test split. The precision classification of the test split is lower for the cabinet category compared to the other two. In a recall with error graph such as in Table 7.1, error lines increasing with the recall (the number of predictions made) indicate that the more relevant predictions are made, the higher the errors in the translation, orientation and scale become. If the errors stay relatively stable as the recall is increasing, it means that the model stays reliable when more relevant predictions are being generated.

While there are less signal lights than cabinets and markers, the performance is not worse, in the test split this category has the highest AP. These results are also reflected in the prediction samples in Figure 7.6. Failure cases are shown in Figure 7.8.

There exist prediction samples where wayside objects are inferred correctly from far away while the ground truth has no annotations because of class filtering or clipping (see Figure D.6). In addition, we show an example in Figure D.4 of objects that are not present in the LiDAR but detected by and visible in the SMOKE model and SfM point cloud.

NDS↑	**mAP↑**	**mATE↓**	**mASE↓**	**mAOE↓**
0.8199	0.8661	0.1657	0.5549	0.0508
Object class	**AP↑**	**ATE↓**	**ASE↓**	**AOE↓**
Signal Light	0.873	0.173	0.504	0.063
Cabinet	0.879	0.197	0.490	0.047
Marker	0.847	0.128	0.671	0.042

(a) Validation split.

NDS↑	**mAP↑**	**mATE↓**	**mASE↓**	**mAOE↓**
0.7159	0.7045	0.1614	0.5686	0.0655
Object class	**AP↑**	**ATE↓**	**ASE↓**	**AOE↓**
Signal Light	0.842	0.168	0.535	0.082
Cabinet	0.576	0.171	0.512	0.062
Marker	0.695	0.145	0.659	0.052

(b) Test split.

Table 7.1: NuScenes true positive evaluation metrics after training SMOKE for 72 epochs, where m indicates the mean score.

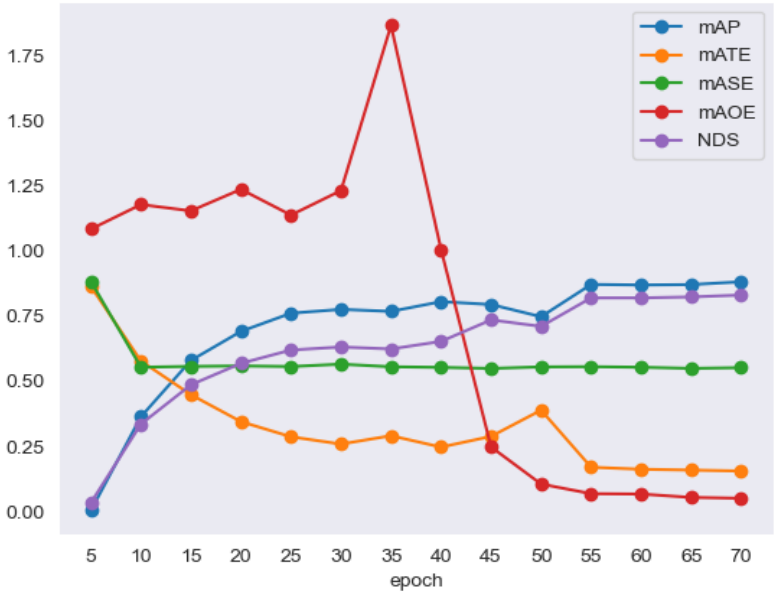
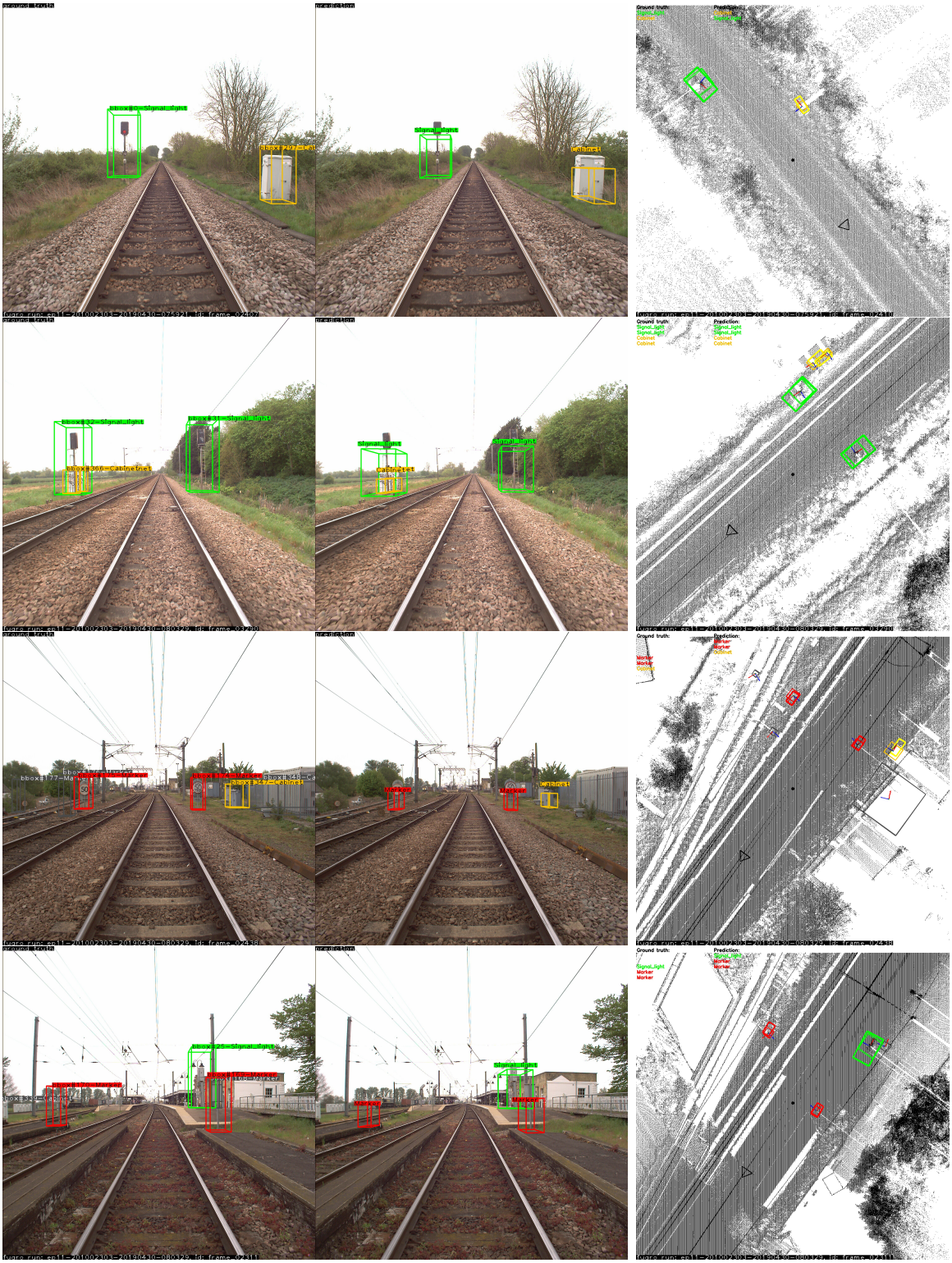


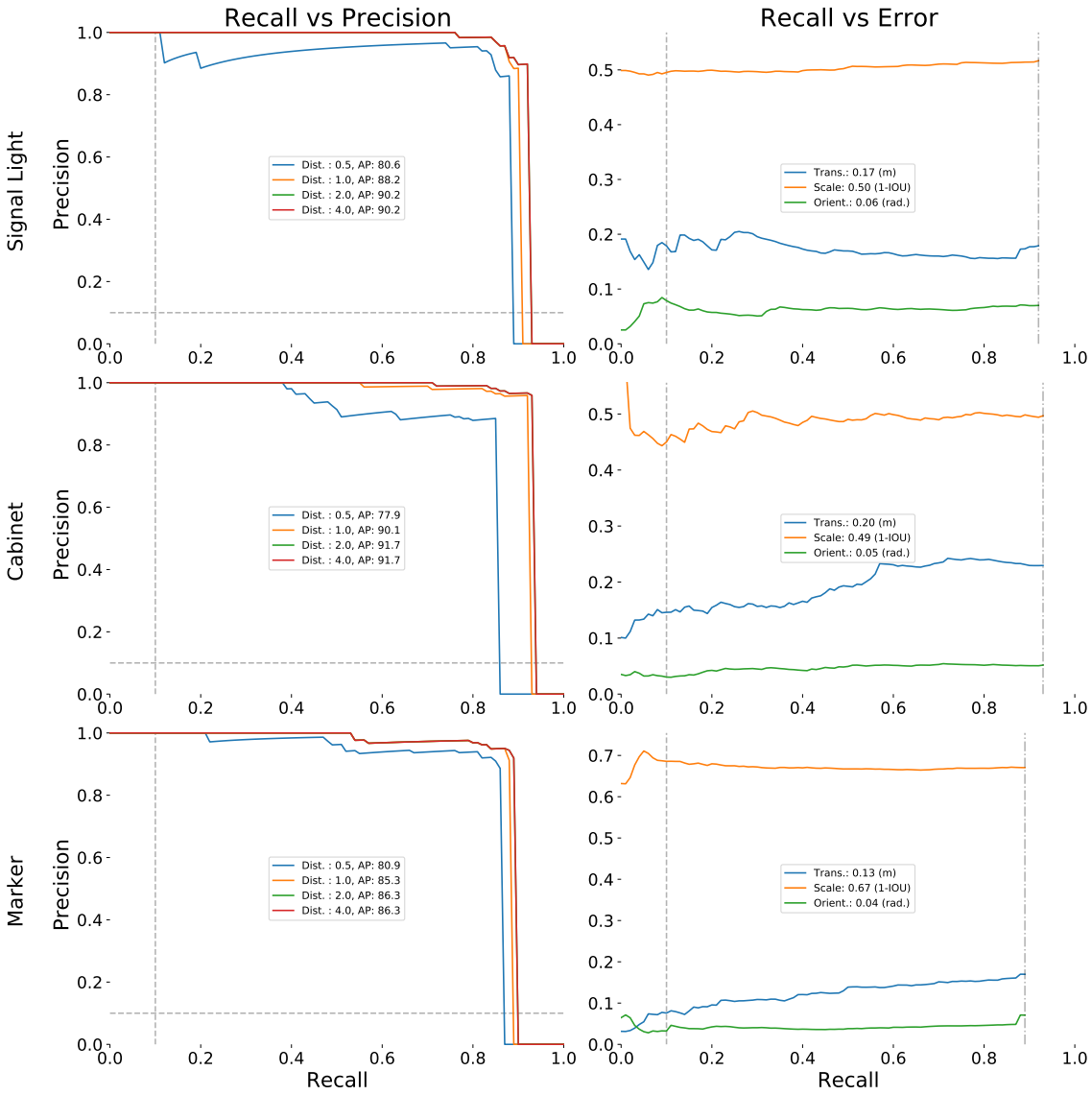
Figure 7.5: Validation scores and errors reported by the model during training. Higher mean Average Precision and NDS score is better, the other values should be lower.



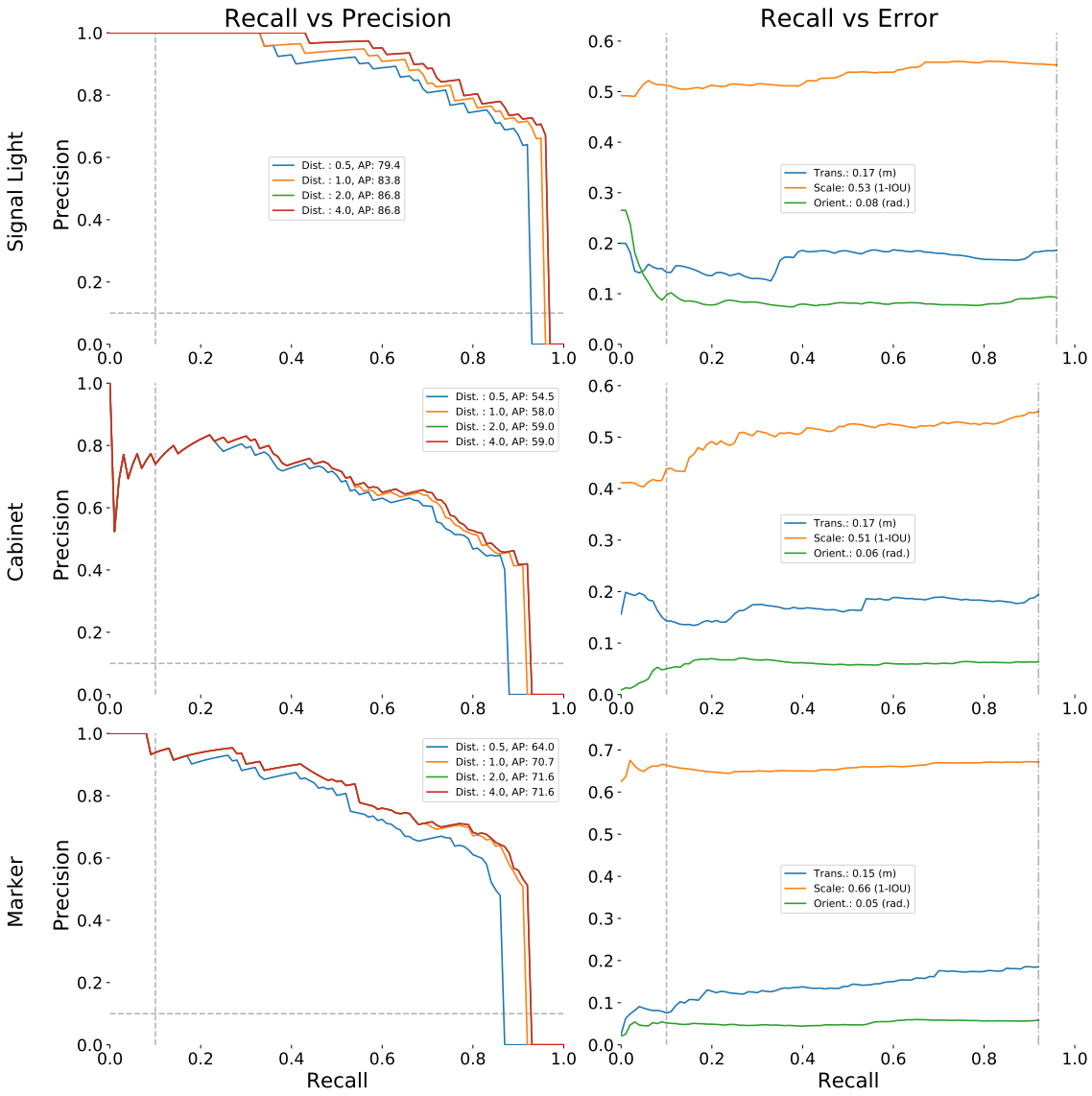
(a) On the left is the ground truth, right is prediction.

(b) Bounding boxes with thicker borders are predictions.

Figure 7.6: Prediction result examples on the test split as seen from camera and bird's-eye view (BEV).

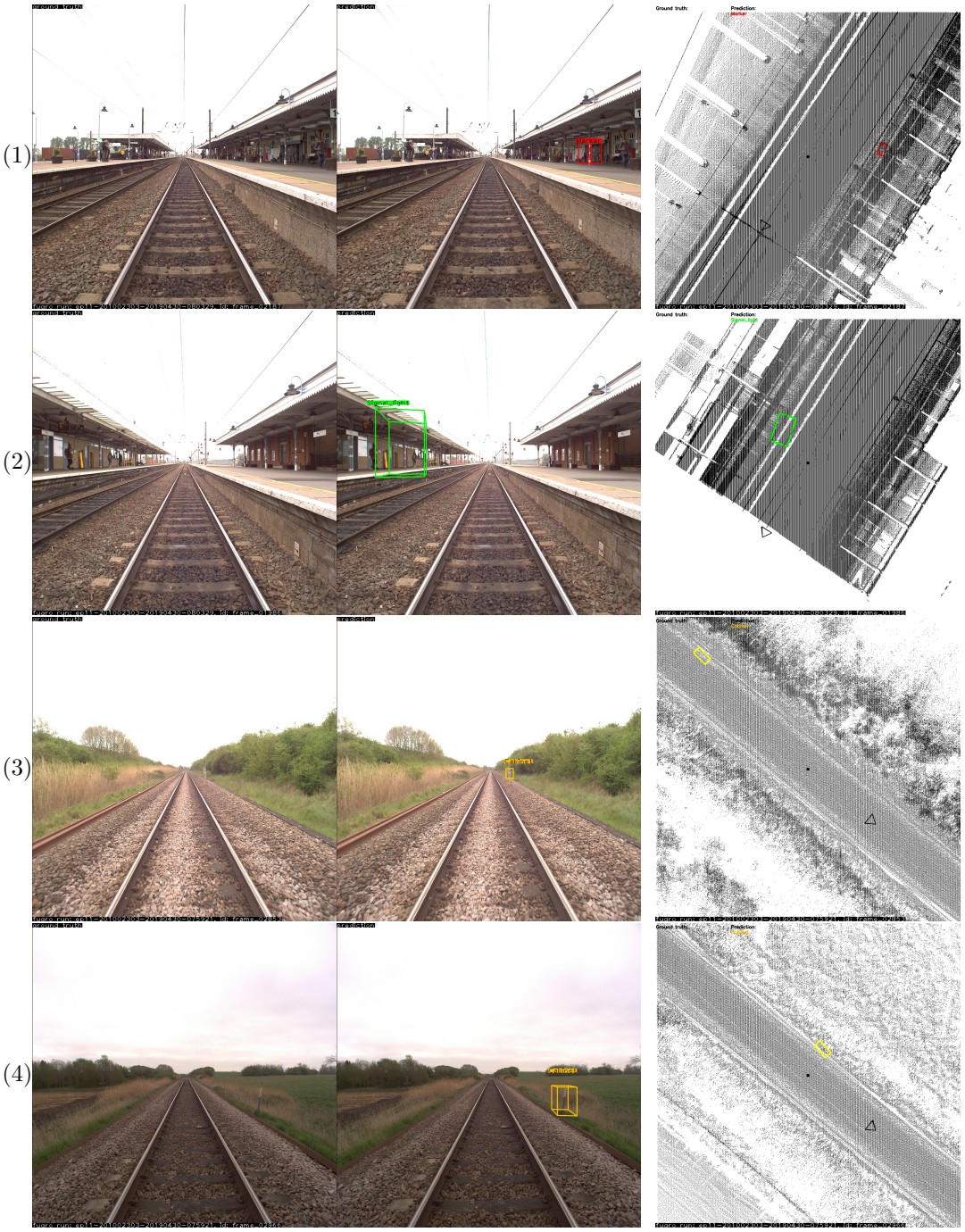


(a) Validation split.



(b) Test split.

Figure 7.7: Recall-precision and recall-error graphs of the SMOKE model trained for 72 epochs. A high area under the curve in the left graphs indicates a more accurate classifier. On the right, the graphs show that when the errors stay relatively stable even when the recall, so the number of predictions made, is increasing.



(e) Left is ground truth, right is prediction

(f) Bounding boxes with thicker borders are predictions.

Figure 7.8: Failure cases on the test split from top to bottom: (1) Predicted bounding box of marker is wrongly placed, scale and orientation of cabinet is incorrect. (2) Predicted bounding box is too close to the camera, there should not be a prediction in this frame. (3) Predicted bounding box has the correct class and position in 2D, but the incorrect 3D location, since the ground truth bounding box is already out of the frame. (4) Predicted bounding box has correct pose, but the object is incorrectly detected, since the ground truth does not contain any annotations.

Chapter 8

Discussion

In this chapter, we address our research question together with our results, limitations and future work.

Pseudo-LiDAR. Both the transfer learning and standard pseudo-LiDAR show poor qualitative and quantitative results compared to LiDAR. From the results of the training process, the distances of the track and the generated point clouds, we can give a few explanations. First, the MiDaS model was not able to learn much from our data. Even with a small dataset, we would have seen improvements during the training if the model was able to extract useful features. Due to these results, we decided to not implement a segmentation step and a style transfer as as discussed in subsection 4.2.1. The failed transfer learning could have several causes. First, the domain gap between synthetic dataset and the model might have been too significant. Although MiDaS was trained on synthetic data as well, it was from a 3D animated film and more character than object focused. Secondly, errors in our custom training pipeline could have also led to no improvements while fitting the model, and fixing these could yield more positive results.

Considering the results of previous research as discussed in Chapter 4, it is likely that even with a synthetic or a ground-truth dataset with enough representative features, depth estimation models are limited. Thus, depth maps inferred by monocular depth estimation is unlikely to improve and enhance the quality of pseudo-LiDAR point clouds accurately enough for production with current state-of-the-art networks.

Structure from motion. This geometrical approach produced point clouds that are qualitatively much more structured than pseudo-LiDAR in terms of the objects shape that coincide more with the camera and LiDAR.

In some cases the SfM point clouds show more completeness than LiDAR such as in Figure C.1 in the center column of the SfM row. The example of wayside objects in Figure D.4 not being present in LiDAR demonstrates this as well. SfM shows, however, more noise, because all the white points around the objects in Figure C.1 originate from the air around them in the camera images. However the quantitative results in Figure 7.4 reveal that SfM is not properly aligned with LiDAR and is not enough for accurate compared to LiDAR. The misalignment is possibly due to incorrect GPS and GCP settings that we were not able to correct within our time frame. Even though we show a limited area, it is representative of the behaviour of our SfM point clouds, we included more examples of these point clouds that are not aligned with laser scanner data in the code. In addition, as we mentioned in Chapter 6, comparing SfM to LiDAR is not a straightforward process because of the missing points in the LiDAR data. We decided to not implement a segmentation step as as discussed in Chapter 5.

3D object detection. The SMOKE model showed it was able to learn from the ground-truth annotations on our validation and testing set. As expected, the model performs better on the validation than on the testing set, because the model's hyperparameters have been tuned to fit the validation split specifically. The curves of the validation and testing graphs show similar results on

the recall axis, i.e. there is not a large recall gap, meaning that the model does not show significant but only slight signs of overfitting.

The cabinets object type is the second most occurring object in the dataset, after markers, but performs the worst on the testing split in terms of classification, with the lowest average precision. It could partly be explained by the physical properties of an electrical cabinet. The plain uniform colour of the boxes reoccurs often in other objects as well, which can lead to classification errors as seen in the bottom picture in Figure 7.8. This holds for the other objects as well in some cases, which could explain the incorrect predictions of bounding boxes where they should not exist. While one could argue these objects have the most rudimentary shape of all classes, it is also the class with the highest standard deviation in dimensions. The average dimensions of a class are used as baseline for prediction. Since the errors are relatively high for the scale metric compared to the translation and orientation error in all classes, it is clear that the model was not able to regress the final size of the bounding box correctly.

Furthermore, a relatively small dataset was used for training and the data is not geo-diverse, so the model might not be very general in terms of recognising wayside objects of other regions.

While the clipping of faraway objects from the dataset helps removing degenerate samples, it also skews the model training process. The model has been incorrectly penalised in some cases where the bounding boxes are further away than the clipping distance but the model was still able to accurately estimate the bounding boxes (see Figure D.6). The number of wayside objects per category also influences the training of the model. The model has seen fewer examples of signal lights than markers, so that might also explain the accuracy scores of each category. Furthermore, every bounding box in the dataset is oriented towards the camera at a perpendicular angle, so the model has not learned the appropriate features of the back of a signal light, for example. However, this flaw has not visibly affected the performance of the network.

Recalling the main research question:

> *How effective is depth estimation at reconstructing 3D geometry, location and orientation of wayside objects from monocular video data in a railway environment?*

From our results discussed above, we can summarise our answer to the research question: general depth estimation by predicting the (relative) depth of every pixel is not yet accurate enough to produce production reliable point clouds. Geometric depth estimation using SfM was also not accurate enough for production, but overall delivers point clouds with more object details than LiDAR. Finally, while estimating the depth of keypoints in an image to create a 3D bounding box around an object does not retrieve any information about an objects shape compared to point clouds, 3D object detection is able to achieve a decent 3D mapping of what object types are present alongside the track.

8.1 Limitations

One inherent limitation of every method is the available data. In our case, we are limited by the absence of a complete ground truth point cloud and 3D annotated bounding boxes.

Even if the ambition is to move away from laser scanner data, every method investigated in this thesis will still rely on ground-truth data for training to some extent. Apart from the unsatisfying results, the relative pseudo-LiDAR point clouds still need an absolute scaling to real-world coordinates.

SfM takes longer to run than the inference of the two other methods, which increases when adding more cameras and/or images. Nevertheless, reconstructions can be run in parallel, which can significantly reduce running times. However, the reconstructions contained little track geometry due to inherent limitations of the algorithm as explained in Chapter 3. In addition, the qualitative results show noise and holes for reflective and repetitive surfaces in the point clouds as seen in Table 7.2f and Figure C.4.

For 3D object detection, the fixed height of the 3D bounding boxes approximates the actual dimensions of every wayside object. This could be remedied by either manually adjusting the height of every object individually, subdivide the classes again or add attributes to the main class where every attribute contains its own height.

Moreover, one current drawback of the dataset is that it was split randomly. Considering the data is sequential, consecutive frames are spread across the different sets, meaning the model has seen very similar instances in the training, validation and testing data. Every frame is considered individually by the model, meaning the relationship between consecutive frames is absent. Bounding boxes are thus inferred independently and the same object might receive multiple bounding boxes. So manual checking and clipping bounding boxes is required after inference regardless. How to merge bounding boxes in 3D map: would require tracking [3] and or manual trimming of the bounding boxes. Finally, we ran our methods with data that was acquired during the day and in good weather conditions. Since all methods rely on camera input, they will most likely perform worse under heavy weather conditions or at night.

8.2 Future work and recommendations

Combining of multiple sources of data for training reduces over-reliance on one technology and increases generalisability of a learning-based model, from the literature research. We know that a fusion of multiple sources of training data provide reliable information for predicting depth in a monocular image sequence and is therefore commonly used in the field of autonomous driving. This partially explains why the first two methods, pseudo-LiDAR and SfM, that solely relied on camera images and camera calibration data, quantitatively is not effective and 3D objection detection, which was trained on accurate 3D annotated data, can be effective for recovering 3D information.

The large differences in the distance maps in Figure B.3b and Figure 7.4 demonstrate that LiDAR still proved to be the most accurate, albeit sparse. This sparseness could possibly be remedied by some hardware changes, especially if computer vision methods do not satisfy business requirements. First, the sensor configuration could include at least one stereo camera system to acquire geometrically computed depth maps directly from the environment. This opens up the possibility to train a monocular or stereo depth estimation model on dense ground truth depth maps. Additionally, a system with a (rapidly) rotating laser scanner while scanning or gearing the LiDAR sensor in the same viewing direction as the camera, could increase the completeness of smaller wayside objects.

Training on synthetic data with a different model could be done in future work. However, acquiring the synthetic data still required inefficient manual work. The game had to be run normally and the frame rate dropped significantly to extract the depth maps. Similar to CARLA for autonomous driving research, a simulator specifically for railway could be developed to automate and accelerate the data acquisition and training process, although creating such a software tool is very time-consuming.

If the SfM pipeline could be adjusted to be fully GPS aligned and scaled accordingly, it could be a promising method to create point clouds from merely camera data input. Likewise, advancements in the active research field of depth estimation networks will improve the quality of pseudo-LiDAR point clouds. In addition, colours are not lost with both methods using pseudo-LiDAR and SfM in contrast to LiDAR and bounding boxes. Unlike pseudo-LiDAR and SfM which only reconstruct a set of unordered and unclassified points in 3D space, the 3D object detection method also provides semantic information, because not only 3D information (in the form of a 3D bounding box) of an object is retrieved, but also the object type. Even in SfM point clouds, objects do not appear accurately, so this method can provide an alternative way to recover wayside objects. The example of wayside objects in Figure D.4 not being present in LiDAR, but visible in SfM and SMOKE, also indicates that these methods are promising.

For future work, the 3D object detection method can be extended in a couple of ways. As discussed in subsection 4.2.3, 2D detection might reduce localisation errors in the results. Thus, other 3D object detection that do use an intermediate 2D detection step can be implemented instead.

The SMOKE model trained under different conditions might be interesting for future work. For

example, add more training data, train for more epochs or add the left and right camera data. The dataset should be changed to include correct heights of bounding boxes in conjunction with adding different orientations relative to the camera to every bounding box. More object classes than three classes can be added to the dataset to increase its usability. The current method is not indicative, however, of how the model will perform with more classes.

Moreover, 3D object detection is not limited to rigid 3D bounding boxes. Methods such as 3D lane detection [21] can be used to recover track geometry.

Finally, if 3D bounding boxes are not complete enough as final format, as discussed in Chapter 4, shape-prior methods that first predict 3D bounding boxes and then the 3D shapes inside could be used alternatively.

Chapter 9

Conclusion

The retrieval of 3D object information from a single moving camera in a railway environment was investigated. Three different methods that involve estimating depth from sequences of images and comparing it with point clouds and annotated 3D data were explored: pseudo-LiDAR, Structure from motion (SfM) and 3D object detection.

Quantitatively, the results of both pseudo-LiDAR and SfM were not accurate enough for production as described in Chapter 6. Qualitatively, however, SfM shows promising results while the pseudo-LiDAR point clouds were not satisfactory. Furthermore, comparing either pseudo-LiDAR and SfM to LiDAR is not trivial because of the lack of ground truth data in the LiDAR point clouds. This makes it difficult to definitively measure by how much another point cloud has recovered compared to LiDAR.

Hardware design changes such as adding a stereo camera might improve the training data and the performance of monocular depth estimation. Given that 3D object detection methods focus inherently more on objects than general depth estimation in pseudo-LiDAR and SfM, it could be used as alternative to generate a 3D annotation map automatically instead of point clouds from a camera signal. We showed that the 3D object detection method is able to predict the class and 3D bounding boxes of three wayside object categories: signal lights, markers and electrical cabinets. The method is flexible enough such that the set of classes could be extended in future work.

If 3D bounding boxes are not satisfactory enough as final format for business operations, methods that focus on both 3D bounding boxes and 3D shapes of only the wayside objects could be implemented in the future. In case accuracy is critical, LiDAR is still an indispensable data source.

Bibliography

[1] balcilar (Muhammet). *Full Dense Depth Map Image for Known Positioned Camera from Lidar Point Cloud*. https://github.com/balcilar/DenseDepthMap. 2013.

[2] Matthias Adorjan. "OpenSfM: A Collaborative Structure-From-Motion System". PhD thesis. Wien, 2016.

[3] Adel Ahmadyan, Liangkai Zhang, Jianing Wei, Artsiom Ablavatski, and Matthias Grundmann. "Objectron: A Large Scale Dataset of Object-Centric Videos in the Wild with Pose Annotations". In: *CoRR* abs/2012.09988 (2020). arXiv: 2012.09988. URL: https://arxiv.org/abs/2012.09988.

[4] A. Atapour-Abarghouei and T.P. Breckon. "Real-Time Monocular Depth Estimation Using Synthetic Data with Domain Adaptation via Image Style Transfer". In: *Proc. Computer Vision and Pattern Recognition*. IEEE, June 2018, pp. 1–8.

[5] Baeldung. *What is a Learning Curve in Machine Learning?* URL: https://www.baeldung.com/cs/learning-curve-ml (visited on Mar. 30, 2022).

[6] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. "State of the Art in Surface Reconstruction from Point Clouds". In: *Eurographics 2014 - State of the Art Reports*. Ed. by Sylvain Lefebvre and Michela Spagnuolo. The Eurographics Association, 2014. DOI: 10.2312/egst.20141040.

[7] P.J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[8] Amlaan Bhoi. *Monocular Depth Estimation: A Survey*. 2019. arXiv: 1901.09402 [cs.CV].

[9] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. *Depth Prediction Without the Sensors: Leveraging Structure for Unsupervised Learning from Monocular Videos*. 2018. arXiv: 1811.06152 [cs.CV].

[10] Hui Chen, Yipeng Zuo, Yong Tong, and Li Zhu. "3D point cloud generation reconstruction from single image based on image retrieval". In: *Results in Optics* 5 (2021), p. 100124. ISSN: 2666-9501. DOI: https://doi.org/10.1016/j.rio.2021.100124. URL: https://www.sciencedirect.com/science/article/pii/S2666950121000729.

[11] Sicong Chen, Zhengbin Li, Lianxin Li, and Xiuyang Zhao. "Multi-Resolution Dense Network for Point Cloud Completion". In: *Proceedings of the 2020 4th International Conference on Electronic Information Technology and Computer Engineering*. EITCE 2020. Xiamen, China: Association for Computing Machinery, 2020, pp. 585–590. ISBN: 9781450387811. DOI: 10.1145/3443467.3443820. URL: https://doi.org/10.1145/3443467.3443820.

[12] Yaodong Cui, Ren Chen, Wenbo Chu, Long Chen, Daxin Tian, Ying Li, and Dongpu Cao. "Deep Learning for Image and Point Cloud Fusion in Autonomous Driving: A Review". In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–18. ISSN: 1558-0016. DOI: 10.1109/tits.2020.3023541. URL: http://dx.doi.org/10.1109/TITS.2020.3023541.

[13] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. "Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis". In: (2017). arXiv: 1612.00101 [cs.CV].

[14] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[15] Haoqiang Fan, Hao Su, and Leonidas Guibas. *A Point Set Generation Network for 3D Object Reconstruction from a Single Image*. 2016. arXiv: 1612.00603 [cs.CV].

[16] Wouter Florijn. "Single Camera 3D Reconstruction of Railway Environments". In: (2017).

[17] @nutonomy GitHub. *nuScenes detection task*. URL: https://github.com/nutonomy/nuscenes-devkit/tree/master/python-sdk/nuscenes/eval/detection (visited on Feb. 1, 2022).

[18] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. "Digging Into Self-Supervised Monocular Depth Estimation". In: (2019). arXiv: 1806.01260 [cs.CV].

[19] Jiayuan Gu, Wei-Chiu Ma, Sivabalan Manivasagam, Wenyuan Zeng, Zihao Wang, Yuwen Xiong, Hao Su, and Raquel Urtasun. "Weakly-supervised 3D Shape Completion in the Wild". In: (2020). arXiv: 2008.09110 [cs.CV].

[20] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. "Deep Learning for 3D Point Clouds: A Survey". In: (2020). arXiv: 1912.12033 [cs.CV].

[21] Yuliang Guo, Guang Chen, Peitao Zhao, Weide Zhang, Jinghao Miao, Jingao Wang, and Tae Eun Choe. "Gen-LaneNet: A Generalized and Scalable Approach for 3D Lane Detection". In: *CoRR* abs/2003.10656 (2020). arXiv: 2003.10656. URL: https://arxiv.org/abs/2003.10656.

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN". In: (2018). arXiv: 1703.06870 [cs.CV].

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Delving Deep into Rectifiers: Surpassing*
Human-Level Performance on ImageNet Classification. 2015. arXiv: 1502.01852 [cs.CV].

[24] Cansen Jiang, Dennis Christie, Danda Pani Paudel, and Cédric Demonceaux. "High quality reconstruction of dynamic objects using 2D-3D camera fusion". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 2209–2213. DOI: 10.1109/ICIP.2017.8296674.

[25] Lukas Koestler, Nan Yang, Rui Wang, and Daniel Cremers. "Learning Monocular 3D Vehicle Detection without 3D Bounding Box Labels". In: *CoRR* abs/2010.03506 (2020). arXiv: 2010.03506. URL: https://arxiv.org/abs/2010.03506.

[26] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. "Robust Consistent Video Depth Estimation". In: (2021). arXiv: 2012.05901 [cs.CV].

[27] Bryan Krauss, Gregory Schroeder, Marko Gustke, and Ahmed Hussein. *Deterministic Guided LiDAR Depth Map Completion*. 2021. arXiv: 2106.07256 [cs.CV].

[28] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. *Multi-Task Multi-Sensor Fusion for 3D Object Detection*. 2020. arXiv: 2012.12397 [cs.CV].

[29] Zechen Liu, Zizhang Wu, and Roland Tóth. "SMOKE: Single-Stage Monocular 3D Object Detection via Keypoint Estimation". In: *CoRR* abs/2002.10111 (2020). arXiv: 2002.10111. URL: https://arxiv.org/abs/2002.10111.

[30] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. "Point-Voxel CNN for Efficient 3D Deep Learning". In: (2019). arXiv: 1907.03739 [cs.CV].

[31] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. "Consistent Video Depth Estimation". In: (2020). arXiv: 2004.15021 [cs.CV].

[32] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. "Delving into Localization Errors for Monocular 3D Object Detection". In: *CoRR* abs/2103.16237 (2021). arXiv: 2103.16237. URL: https://arxiv.org/abs/2103.16237.

[33] Niall O' Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Adolfo Velasco-Hernández, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. "Deep Learning vs. Traditional Computer Vision". In: *CoRR* abs/1910.13796 (2019). arXiv: 1910.13796. URL: http://arxiv.org/abs/1910.13796.

[34] Priyanka Mandikal, K L Navaneet, Mayank Agarwal, and R. Venkatesh Babu. "3D-LMNet: Latent Embedding Matching for Accurate and Diverse 3D Point Cloud Reconstruction from a Single Image". In: (2019). arXiv: 1807.07796 [cs.CV].

[35] Mathworks. *Evaluating the Accuracy of Single Camera Calibration*. URL: https://nl.mathworks.com/help/vision/ug/evaluating-the-accuracy-of-single-camera-calibration.html (visited on Mar. 29, 2022).

[36] Mathworks. *Structure from Motion Overview*. URL: https://www.mathworks.com/help/vision/ug/structure-from-motion.html (visited on Mar. 29, 2022).

[37] Moritz Menze and Andreas Geiger. "Object Scene Flow for Autonomous Vehicles". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[38] N. Mitra, L. Guibas, and M. Pauly. "Partial and approximate symmetry detection for 3D geometry". In: *ACM Trans. Graph.* 25 (2006), pp. 560–568.

[39] OpenCV. *Perspective-n-Point (PnP) pose computation*. URL: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html (visited on Mar. 29, 2022).

[40] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *CoRR* abs/1612.00593 (2016). arXiv: 1612.00593. URL: http://arxiv.org/abs/1612.00593.

[41] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer". In: (2020). arXiv: 1907.01341 [cs.CV].

[42] Cody Reading, Ali Harakeh, Julia Chae, and Steven L. Waslander. "Categorical Depth Distribution Network for Monocular 3D Object Detection". In: *CoRR* abs/2103.01100 (2021). arXiv: 2103.01100. URL: https://arxiv.org/abs/2103.01100.

[43] Haoyu Ren, Aman Raj, Mostafa El-Khamy, and Jungwon Lee. "SUW-Learn: Joint Supervised, Unsupervised, Weakly Supervised Deep Learning for Monocular Depth Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2020.

[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: (2016). arXiv: 1506.01497 [cs.CV].

[45] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, and Elisa Ricci. *Are we Missing Confidence in Pseudo-LiDAR Methods for Monocular 3D Object Detection?* 2021. arXiv: 2012.05796 [cs.CV].

[46] Chahat Deep Singh. *Structure from Motion*. URL: https://cmsc426.github.io/sfm/ (visited on Mar. 29, 2022).

[47] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. *On the Importance of Stereo for Accurate Depth Estimation: An Efficient Semi-Supervised Deep Neural Network Approach*. 2020. arXiv: 1803.09719 [cs.CV].

[48] Mario Soilán, Ana Sánchez Rodríguez, Pablo del Rio, Carlos Pérez-Collazo, Pedro Arias, and B. Riveiro. "Review of Laser Scanning Technologies and Their Applications for Road and Railway Infrastructure Monitoring". In: *Infrastructures* 4 (Sept. 2019), p. 58. DOI: 10.3390/infrastructures4040058.

[49] Christopher Town. *Computer Science Tripos Part II*. URL: https://www.cl.cam.ac.uk/teaching/1011/CompVision/Town%20-%20ComputerVision%20-%20L1.pdf (visited on Mar. 29, 2022).

[50] Dahlia Urbach, Yizhak Ben-Shabat, and Michael Lindenbaum. "DPDist : Comparing Point Clouds Using Deep Point Cloud Distance". In: *CoRR* abs/2004.11784 (2020). arXiv: 2004.11784. URL: https://arxiv.org/abs/2004.11784.

[51] Patil Vaishakh, Wouter Van Gansbeke, Dengxin Dai, and Luc Van Gool. "Don't Forget The Past: Recurrent Depth Estimation from Monocular Video". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[52] Jingwen Wang, Martin Rünz, and Lourdes Agapito. "DSP-SLAM: Object Oriented SLAM with Deep Shape Priors". In: *CoRR* abs/2108.09481 (2021). arXiv: 2108.09481. URL: https://arxiv.org/abs/2108.09481.

[53] Qian Wang and Min-Koo Kim. "Applications of 3D point cloud data in the construction industry: A fifteen-year review from 2004 to 2018". In: *Advanced Engineering Informatics* 39 (2019), pp. 306–319. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2019.02.007. URL: https://www.sciencedirect.com/science/article/pii/S1474034618304683.

[54] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. "Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving". In: (2020). arXiv: 1812.07179 [cs.CV].

[55] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. "DeepSFM: Structure From Motion Via Deep Bundle Adjustment". In: *CoRR* abs/1912.09697 (2019). arXiv: 1912.09697. URL: http://arxiv.org/abs/1912.09697.

[56] Changchang Wu. "Towards Linear-Time Incremental Structure from Motion". In: June 2013, pp. 127–134. DOI: 10.1109/3DV.2013.25.

[57] Wenxuan Wu, Zhongang Qi, and Li Fuxin. "PointConv: Deep Convolutional Networks on 3D Point Clouds". In: (2020). arXiv: 1811.07246 [cs.CV].

[58] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration". In: *CoRR* abs/1605.03344 (2016). arXiv: 1605.03344. URL: http://arxiv.org/abs/1605.03344.

[59] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. "HVNet: Hybrid Voxel Network for LiDAR Based 3D Object Detection". In: (2020). arXiv: 2003.00186 [cs.CV].

[60] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. *Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving*. 2020. arXiv: 1906.06310 [cs.CV].

[61] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. "PCN: Point Completion Network". In: (2019). arXiv: 1808.00671 [cs.CV].

[62] Linqi Zhou, Yilun Du, and Jiajun Wu. "3D Shape Generation and Completion through Point-Voxel Diffusion". In: (2021). arXiv: 2104.03670 [cs.CV].

[63] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. "Unsupervised Learning of Depth and Ego-Motion from Video". In: *CoRR* abs/1704.07813 (2017). arXiv: 1704.07813. URL: http://arxiv.org/abs/1704.07813.

[64] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for
Point Cloud Based 3D Object Detection". In: (2017). arXiv: 1711.06396 [cs.CV].

Appendices

Appendix A

Camera setup

Figure A.1: Example of the three camera views from the railway data, where the overlap between the images is $< 50\%$.

Appendix B

Pseudo-LiDAR

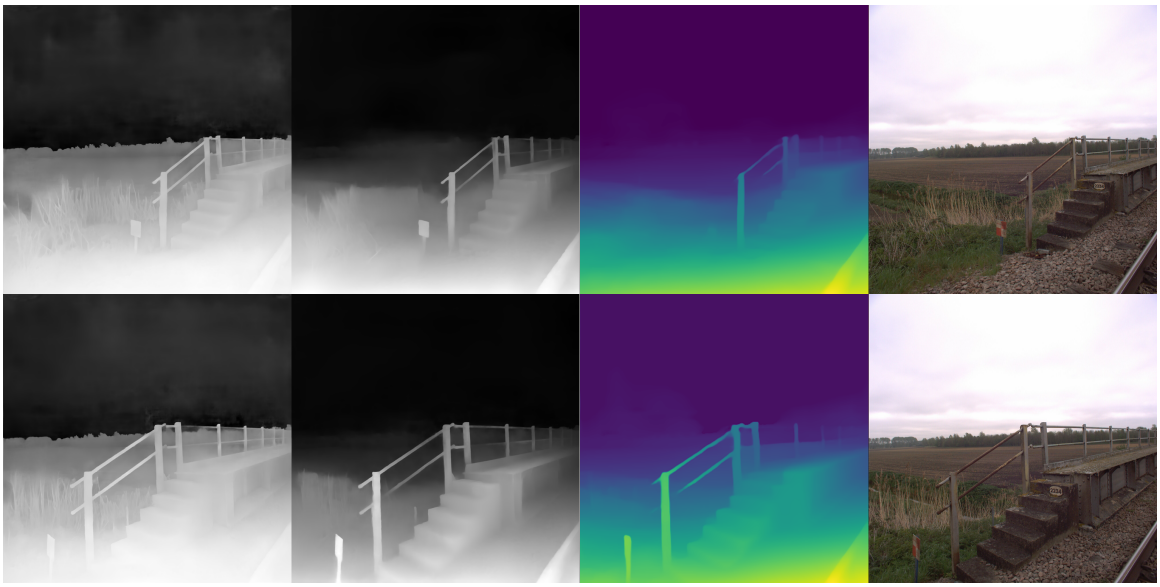


Figure B.1: Initial depth estimation tests on the railway data. Models used from left to right[1]: LeReS, Boost MiDaS and MiDaS v2.1.

[1]LeReS `https://github.com/aim-uofa/AdelaiDepth/tree/main/LeReS` and Boost MiDaS `https://github.com/compphoto/BoostingMonocularDepth`

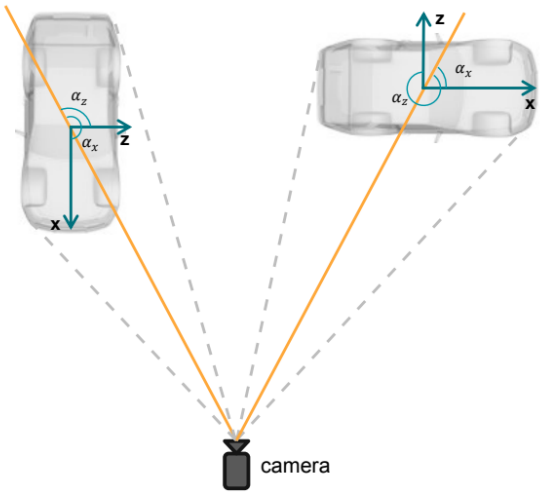
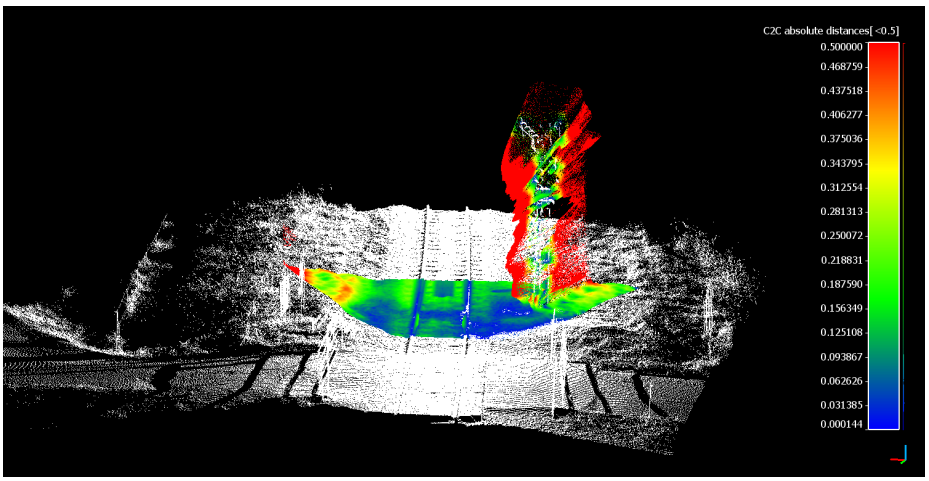
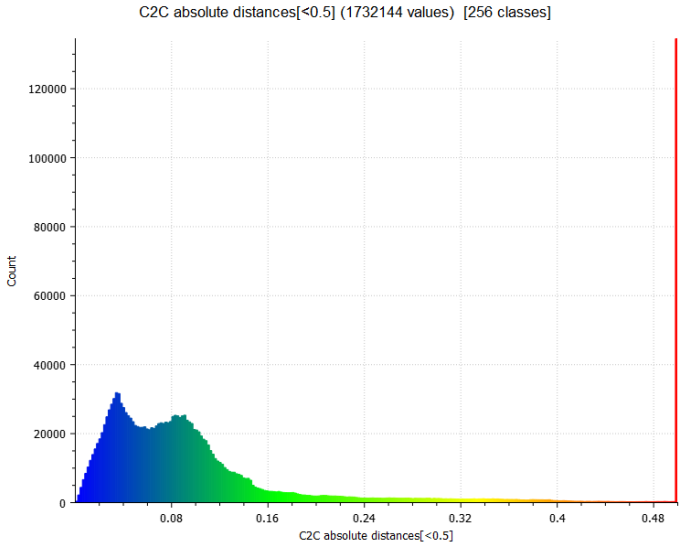


Figure B.2: KITTI dataset provides the angle of observation angle a_x relative to the camera. If a_x is zero, the object is always facing away from the camera. SMOKE computes the loss over the angle a_z. Image from [29].



(a) Pseudo-LiDAR versus LiDAR distances (white). Pseudo-LiDAR is already aligned with the GPS location of the camera, so only a fixed scaling to world coordinates was applied.



(b) Cloud-to-cloud distances histogram. C2C distances larger than $>$ 50cm are automatically capped at 50cm.

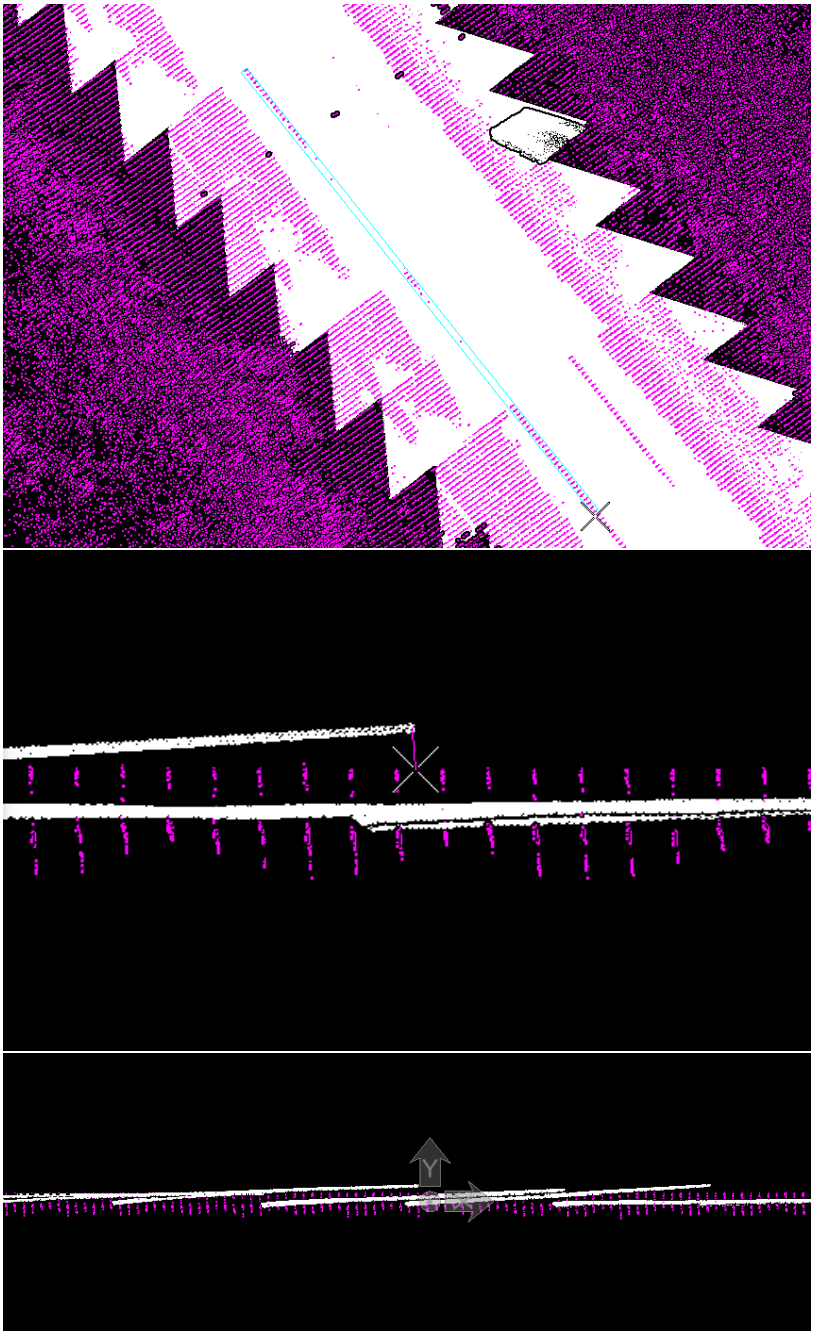


Figure B.4: Pseudo-LiDAR (white) versus LiDAR (purple) track side view zoomed in (top) and zoomed out (bottom). In the top image, the distance at the top of the LiDAR track and the pseudo-LiDAR measures 0.091m.

Appendix C

Structure from Motion

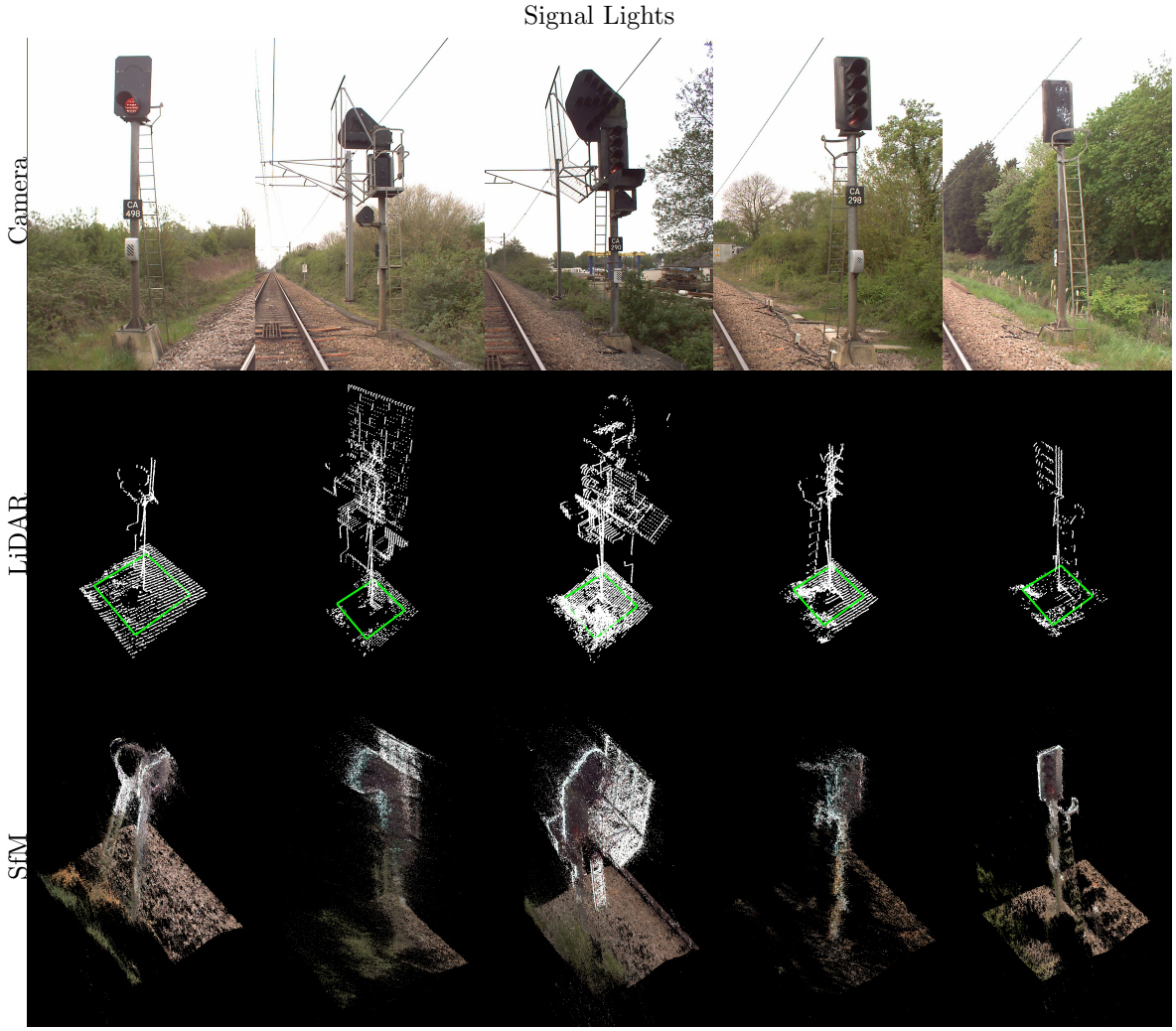


Figure C.1: Qualitative comparison of wayside objects in category *signal light* in the camera, LiDAR and SfM data. The coloured squares indicate the ground-truth LiDAR annotation data.

Figure C.2: Qualitative comparison of wayside objects in category *cabinet* in the camera, LiDAR and SfM data. The coloured squares indicate the ground-truth LiDAR annotation data.

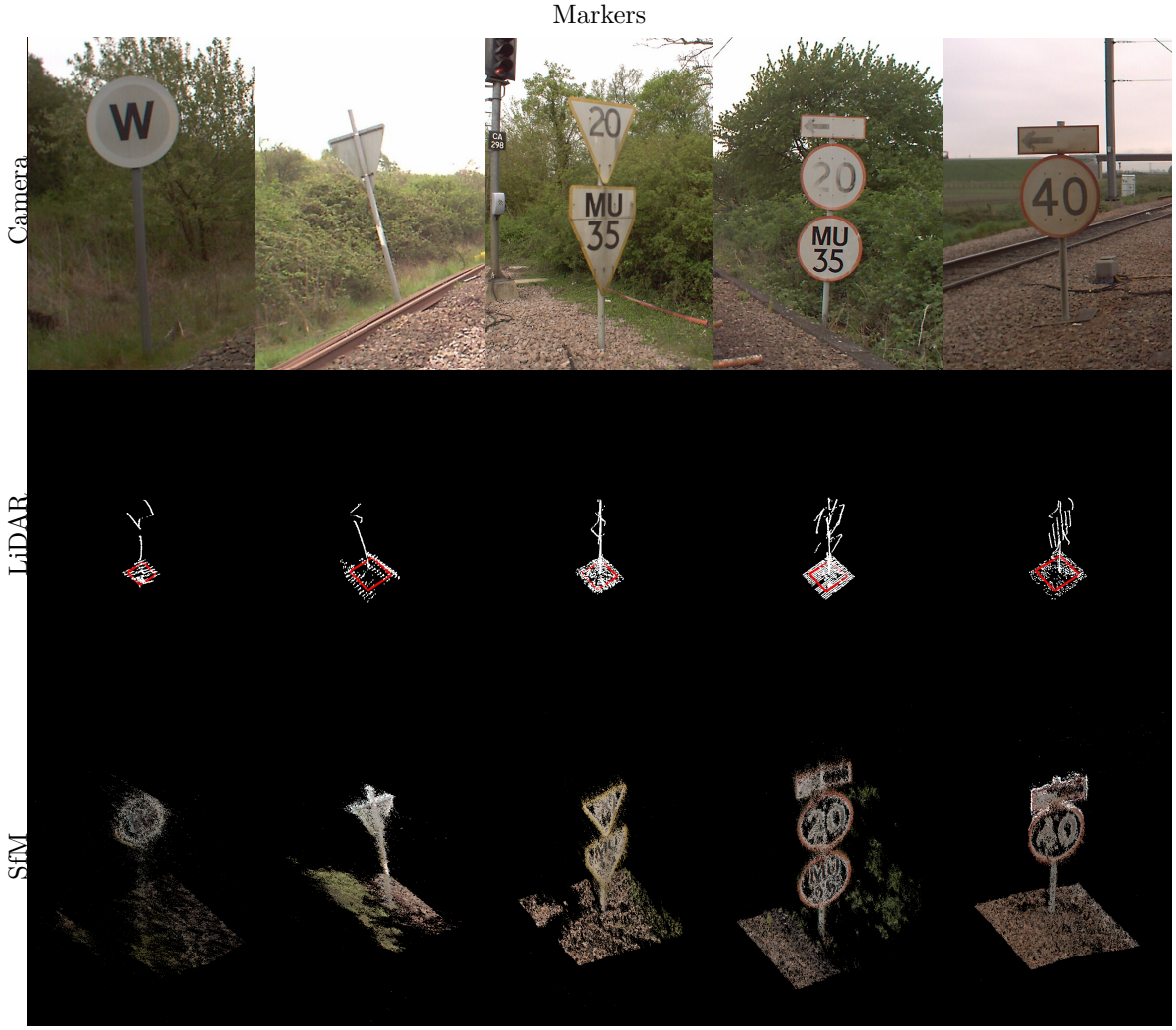


Figure C.3: Qualitative comparison of wayside objects in category *marker* in the camera, LiDAR and SfM data. The coloured squares indicate the ground-truth LiDAR annotation data.

Figure C.4: SfM top-view of track example.

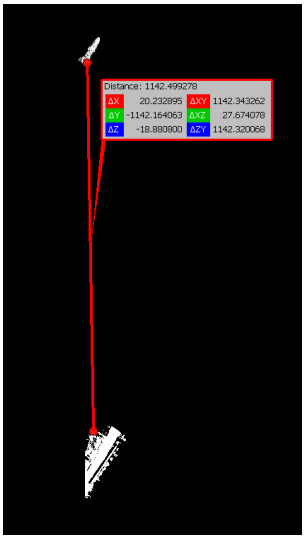


Figure C.5: Example from top view showing how a produced SfM point cloud (top) is not aligned with a LiDAR point cloud (bottom). This reconstruction was created with the GPS coordinates of the camera. Distance is in meters and a rough measurement.

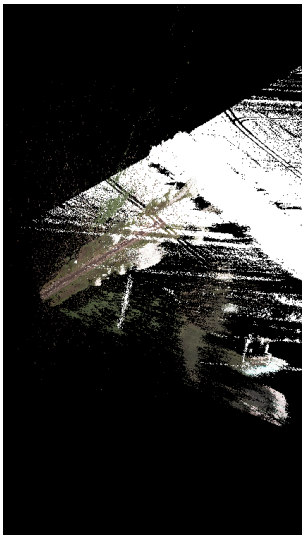


Figure C.6: Example from top view showing how a produced SfM point cloud (coloured) is rotated and not aligned with a LiDAR point cloud (white). This reconstruction was created with GCPs.

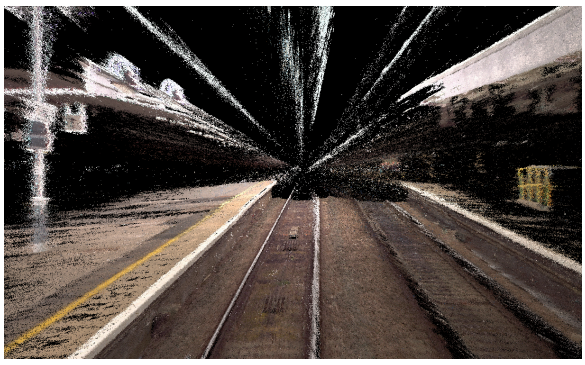


Figure C.7: Incorrectly reconstructed points in SfM point cloud due to features around the center of the image detected incorrectly.

Appendix D

3D object detection

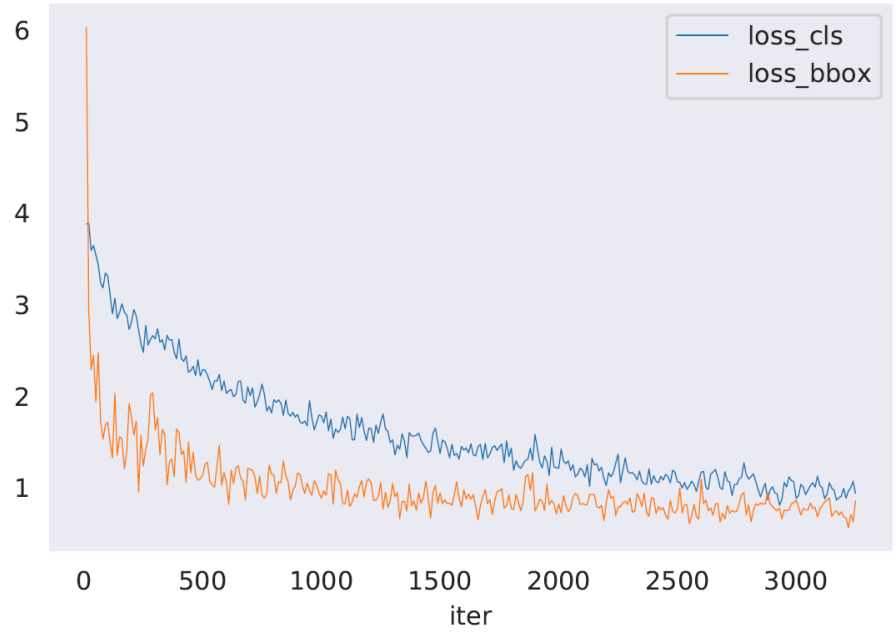


Figure D.1: SMOKE average training losses of object classification and bounding box estimation for every iteration.

NDS↑	**mAP↑**	**mATE↓**	**mASE↓**	**mAOE↓**
0.5838	0.7115	0.3413	0.5455	1.6795
Object class	**AP↑**	**ATE↓**	**ASE↓**	**AOE↓**
Signal Light	0.682	0.412	0.517	1.599
Cabinet	0.766	0.332	0.451	1.729
Marker	0.686	0.280	0.669	1.710

Table D.1: NuScenes evaluation metrics on the validation split after training SMOKE for 25 epochs, where 'm' indicates the mean score.

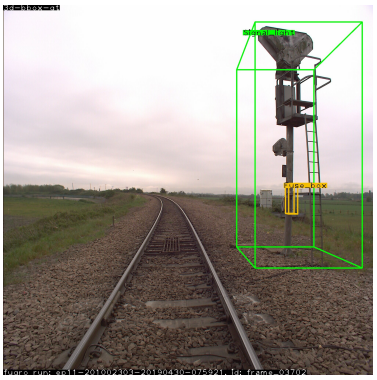


Figure D.2: Example where one bounding box from the ground-truth 3D annotations is blocking the other from the camera view. In this frame, the bounding box of the electrical cabinet should be removed.

Figure D.3: Example of 2D bounding boxes predictions by the SMOKE model. The left image shows ground truth and right image the prediction.

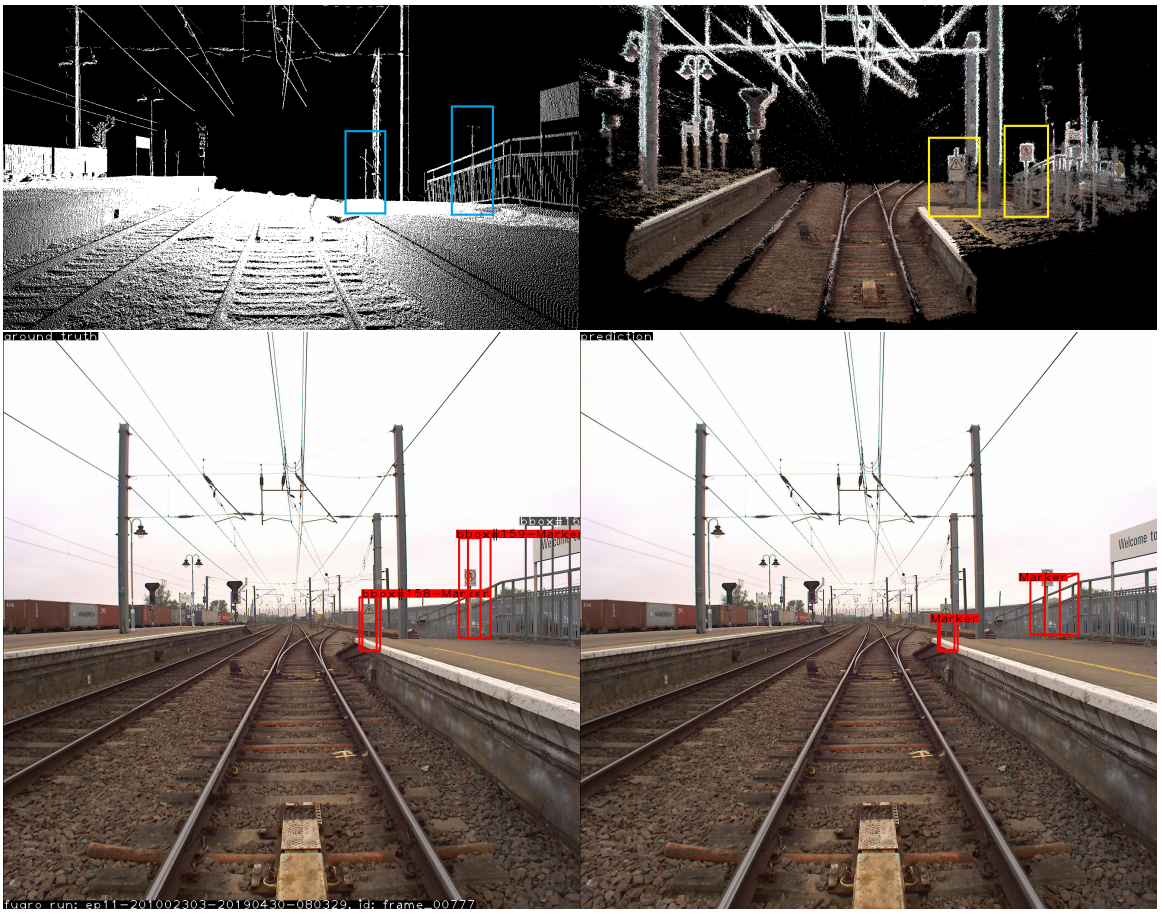


Figure D.4: Example of two marker objects that are not completely scanned in the LiDAR point cloud (top left) but are visible in the SfM point cloud (top right) and detected by SMOKE (bottom right). Ground truth of the 3D bounding boxes is shown in the bottom left.

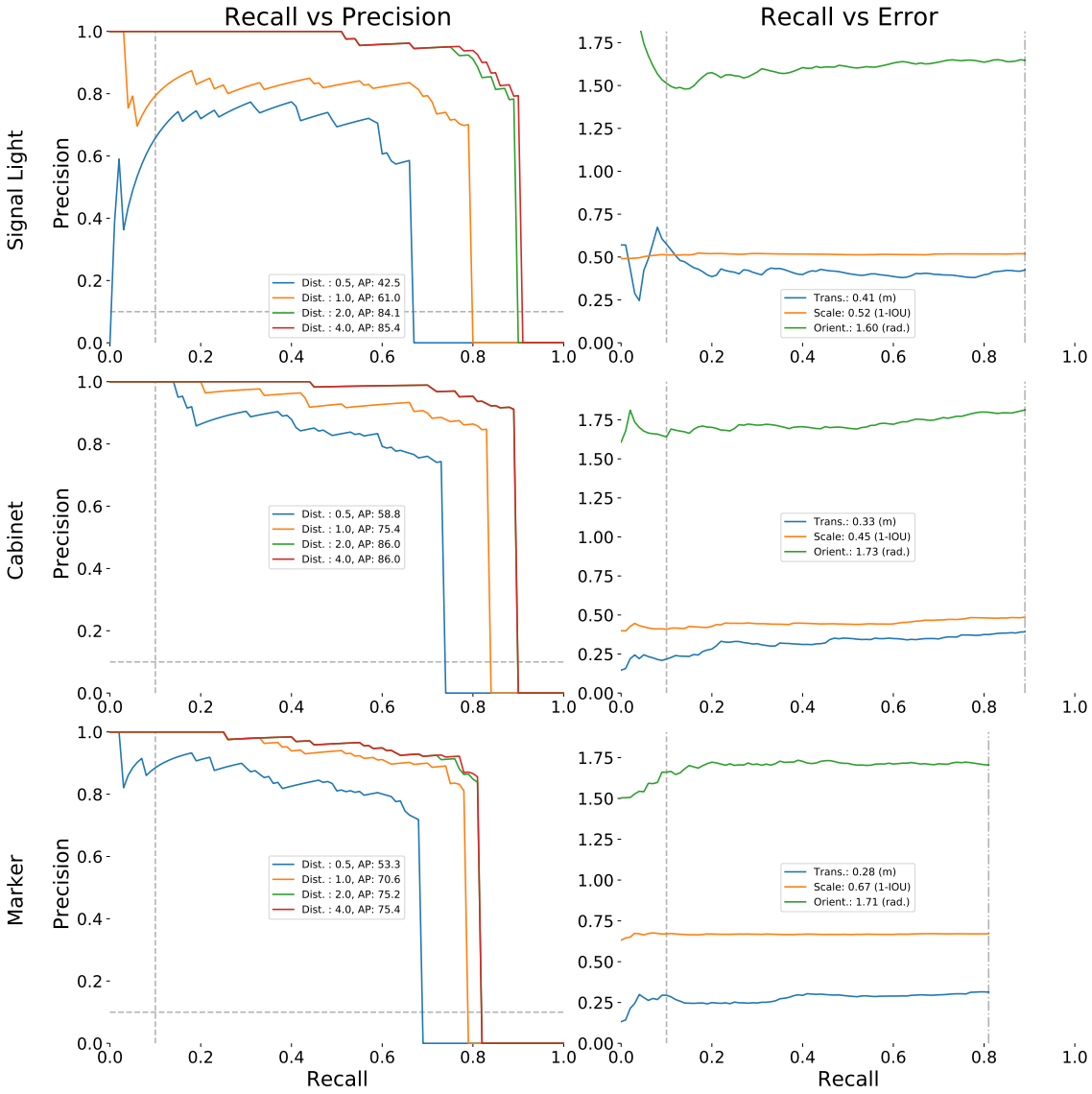
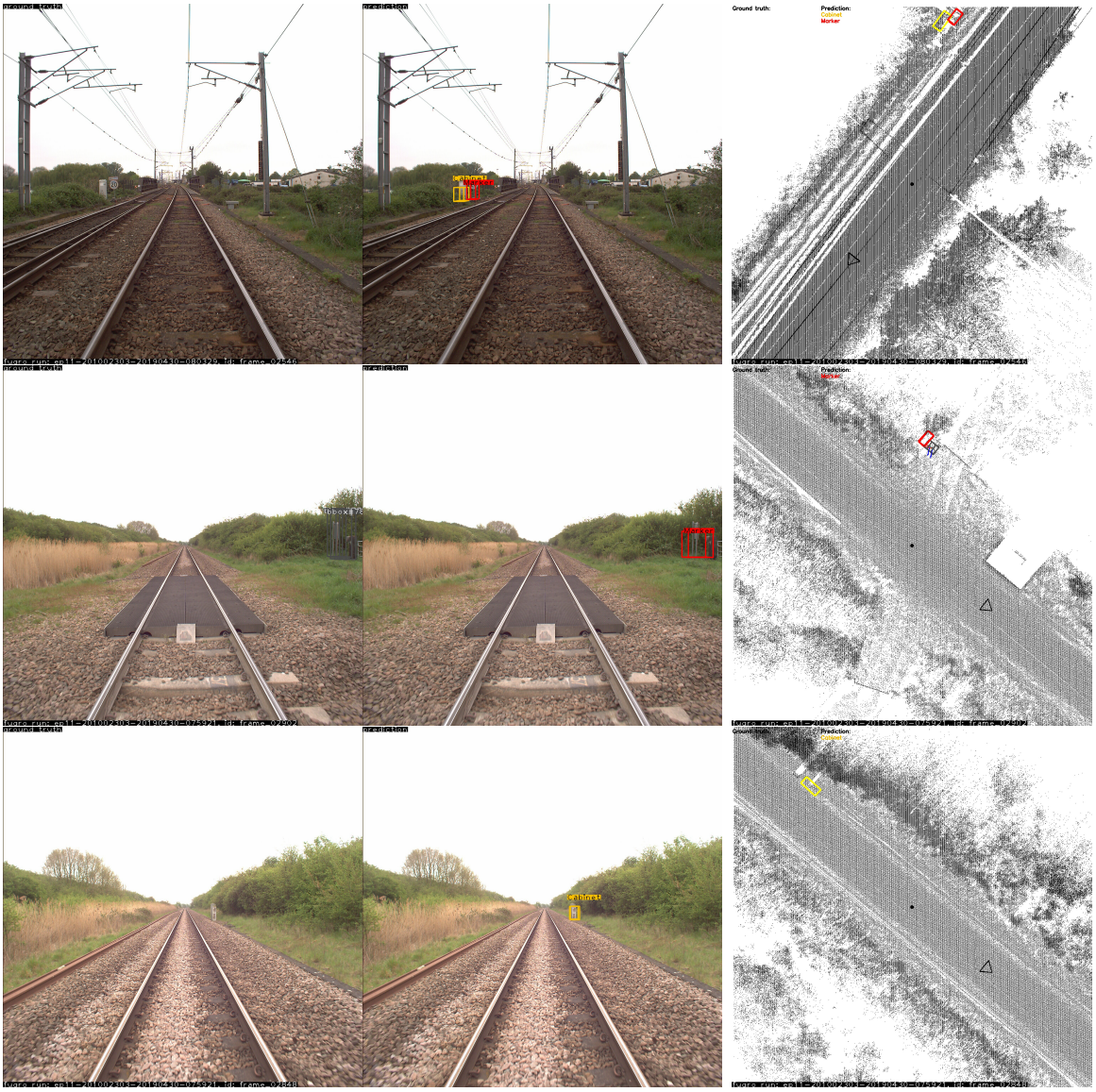


Figure D.5: Recall-precision and recall-error graphs on the validation split of the SMOKE model trained for 25 epochs.



(a) Left is ground truth, right is prediction

(b) Bounding boxes with thicker borders are predictions.

Figure D.6: Prediction result on the test split as seen from camera and bird's-eye view (BEV) where the ground truth did not have any annotations (due to clipping of faraway objects in data preparation), but the model was able to still predict the 3D bounding boxes correctly.