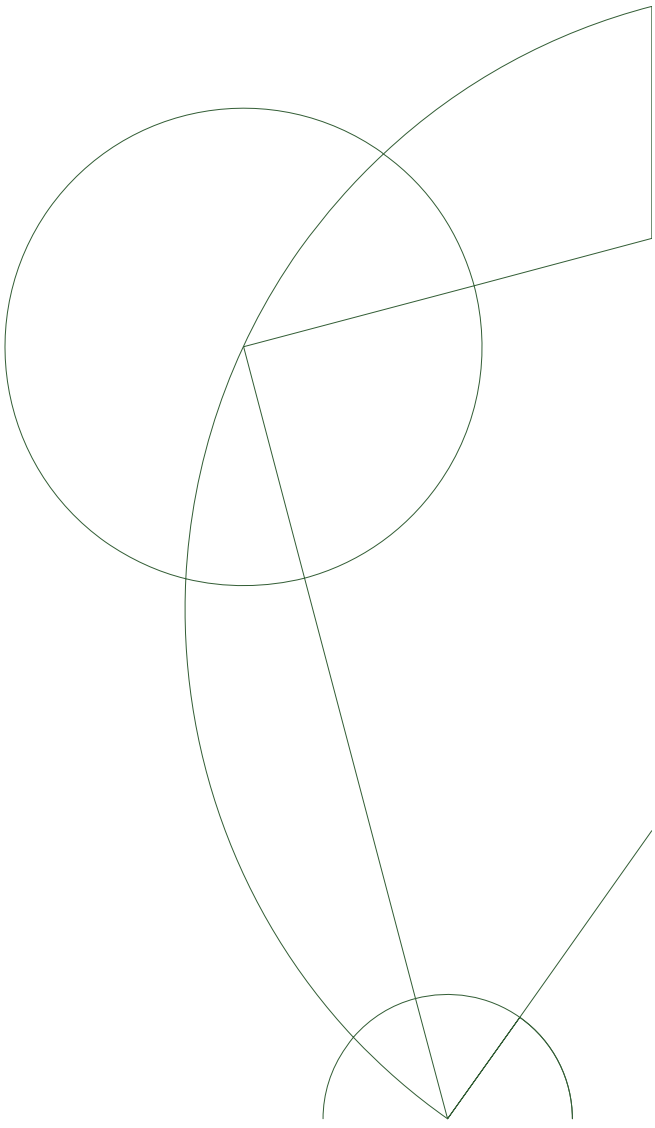
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Computer Vision in Railroad Asset Detection

Frederik Warburg
<dlk165@alumni.ku.dk>



Supervisors
Søren Ingvor Olsen <[illegible]>
Georgios Karagiannis

Abstract

Major railway lines need advanced management systems based on accurate maps of their infrastructure. Asset detection is an important tool towards automation of processes and improved decision support on such systems. Due to the lack of available data, limited research exists investigating railway asset detection despite the rise of Artificial Neural Networks and the extensive research on autonomous driving. In this project, we adapt the Faster R-CNN model to detect signs and signals in an annotated Railroad dataset from COWI. Through extensive experimental work, we find that modifying the model based on both prior knowledge about object sizes and the inclusion of the Focal Loss, improves the model performance. We further adapt the `Struct2depth` model to estimate a dense depth map for each frame in a subset of the Railroad dataset. As a proof-of-concept, we demonstrated that the inferred depth maps can be used to obtain 3D information about the detected railroad furniture.

Contents

Chapter 1

Introduction

1.1 Introduction

Major railway lines are assembled over centuries. They are build iteratively, which means that the placement of railroad furniture are exposed to deliberate modifications. Moreover, these railroad furniture can also be damaged by external conditions, such as the weather. Because of the sheer size of these railway lines, it has become very expensive to manually maintain an up-to-date map of the railroad infrastructure, including the position of each railroad furniture. Asset detection is an important tool towards automation of these processes. Moreover, asset detection can potentially improve the quality and refreshment rate of the infrastructural map, which may lead to improved safety and decision support for such systems. Limited research exists on asset detection for railway data despite the rise of Artificial Neural Networks and the extensive research on autonomous driving. This is mainly because there does not exist a publicly available annotated dataset for railroad sequences.

The North European consulting group, COWI, has collected and annotated numerous railroad video sequences from Queensland, Australia. This project will investigate these images and apply state-of-the-art convolutional neural networks to detect the railroad furniture and estimate their 3D positions.

1.1.1 Motivation

The 3D position of railroad furniture (including signs and signals) is very relevant for COWI since knowing the position can reduce the cost of maintenance and manual labor – as an up-to-date map can automatically and frequently be collected. Moreover, such a map can reduce the risk of accidents as the train driver can be supported by smart systems that are able to track and detect the railroad furniture or match the position of railroad furniture with the train's current position.

This project seeks to estimate the 3D position of railroad furniture. The project consists of two parts:

- Adapt and train a 2D object detection model - that takes an image as input and outputs a 2D bounding box together with class label for each object of interest in the image - for the entire Railroad dataset. Evaluate the performance of the trained model and investigate the causes of errors.
- Estimation of a dense depth map that allow COWI to measure the 3D positions of the detected objects. Adapt a model to infer depth predictions based on sequences of images and demonstrate how this depth information can be used to obtain 3D information about the detected objects. Evaluate the performance of the depth estimations.

1.1.2 Overview

The project is divided into two parts; where the first part addresses 2D object detection and the second part dense depth estimation.

The first part is structured as follows: Section 2.1 presents an overview of popular object detectors. Based on this initial research, it is decided to adapt the Faster R-CNN model for detection of railroad furniture. In Section 2.2, this model and its predecessors are explained in more detail, together with architectural alterations of the model. Section 2.3 provides an descriptive analysis of the Railroad dataset collected by COWI. It has been necessary to limit the amount of data to keep the scope of the project manageable. Based on the descriptive analysis, several design choices that decrease the amount of data and prepare the data for the training and testing are explained in Section 2.4. The Faster R-CNN is adapted to the Railroad data and several experiments are carried out and described in Section 2.5. In Section 2.6, the results from these experiments are reported. The main points are summarized in Section 2.7.

The second part of the report follows a similar structure. Section 3.1 and Section 3.2 describe related work and give an in-depth explanation of the state-of-the-art deep learning based depth estimators. Section 3.3 and Section 3.4 describe how the chosen depth estimator, `struct2depth`, is adapted to the Railroad data. This includes a description on how the intrinsic camera parameters have been reverse-engineered and how segmentation masks have been obtained. The experimental setups are described in Section 3.5. The results are analyzed in Section 3.6 and concluded upon in Section 3.7.

The overall applicability of the presented results for COWI is discussed in Section 4.1, together with suggested future work. Section 4.2 present the overall conclusion on the project.

1.1.3 Pre-requisites

It is assumed that the reader has a solid mathematical and statistical understanding. Moreover, the reader must have a basic knowledge about both Artificial Neural Networks and Computer Vision, e.g. Epipolar Geometry. However, it is not assumed that the reader has prior knowledge or experience with the specific task of object detection nor of depth estimation as was the case when I began this project.

Chapter 2

2D Object Detection

The goal of 2D object detection is to assign a bounding box and a label to each object of interest in an image. The bounding box is usually represented by (x, y, w, h) indicating a parameterization of the bounding box where (x, y) is the center of the box, w, h are the width and height of the box. The size of the bounding box should be minimized, however it must contain the entire object. This chapter will outline the landscape of object detectors. An object detector will be adapted to the Railroad datasets, which are used to to generate results obtained from an extensive experimental work. Finally, the accuracy of these results are analyzed.

2.1 Related work

The first object detectors were based on handcrafted features such as SIFT and HOG[2], [7], [45]. However, the recent research on object detection has shown that features learned by deep neural networks outperform the traditional handcrafted features, in particular for drastic appearance changes. These deep learning-based object detectors can be divided into two groups: one-stage detectors, which do not use regional proposals, and two-stage detectors, which do use regional proposals. This section seeks to provide an overview of some of the most popular object detectors within these categories.

Two-stage methods
The two-stage methods are characterized by first localizing several region proposals and then classifying which region belongs to each class. A characteristic with these regional proposal-based models is that they forward a varying number of region proposals through a Convolutional Neural Network (CNN). Among the most well known object detector belonging to this category is the Regional CNN (R-CNN)[12]. Girshick et al. presented this object detector, which first generates category independent region proposals using selective search, and then uses a pre-trained CNN to compute convolutional features for each region proposals. These features are used to classify each regional proposal using a one-to-many linear Support Vector Machine (SVM)[1]. Girshick later suggested Fast R-CNN[17] - a faster and more accurate object detector. This detector largely builds on the same ideas as R-CNN, however Girshick unified the three steps of R-CNN, which enables online training and the use of deeper pre-trained networks. Lately, another improvement to the R-CNN has been presented - namely the use of a Region Proposal Network (RPN)[46]. Yicheng et al. showed how the RPN could replace selective search for the Fast R-CNN detector. They named this improvement of Fast-RCNN for Faster R-CNN because of its increased speed. Faster R-CNN has become a popular and widely used object detector that currently - or variations of it as [27], [39] - is among the best performing object detectors on popular datasets for object detection[15], [43], [44].

One variation of Faster R-CNN is the Region-based Fully Convolutional Networks (R-FCN)[27] that enables full sharing of convolutional weights. This variation presents significant speed-up compared to Faster R-CNN while still achieves competitive results. Also, this variation enables full sharing of weights

by introducing position-sensitive score maps to better recognize the different parts of objects. The intuition behind these position-sensitive score maps is that by recognizing a part of an object (this could be an eye), we know the position of the object (the face). By calculating these score maps on the entire image allows the model to share information between the region proposals.

One-stage methods
The other group of object detectors is the one-stage methods, where a network is trained end-to-end to do both the localization and the classification. They are characterized by a fixed number of bounding boxes that are fed through an CNN. These networks must be trained end-to-end and outputs both a confidence estimate for each bounding box and a class distribution. These methods are in general faster than the region proposal methods, but often less accurate.

The first one-stage objector was You-Only-Look-Once (YOLO)[23]. The YOLO detector works by dividing the input image into a $S \times S$ grid, where S is the number of grid cells in each direction. For each cell, B bounding boxes are predicted - each with a confidence score that describes how well an bounding box includes an object - and a class distribution that describes the probability of that bounding box belonging to each of the given classes. This means that YOLO outputs $S \times S \times B$ bounding box predictions each of size $C \times 5$, where C is the number of classes and 5 is the number of parameters used in the parameterization of the bounding boxes. Since the release of YOLO, several architectural improvements have been proposed; first in YOLOv2[31] and recently in YOLOv3[41], which both improves the accuracy of the detector without compromising too much with the speed. The YOLO approach has become popular for object detection because it is very fast.

SSD presented in [20] is in many ways similar to YOLO. It uses a single deep neural network for the object localization and classification. Differently from YOLO, SSD uses a feature pyramid architecture that gradually reduces the spatial dimension and resolution of the image using convolutions. This feature pyramid helps discover objects at different scale spaces. For each feature pyramid, SSD detects objects using a set of default bounding boxes that have different ratios between height and width. The feature pyramid increases the capability of the model to detect different sized objects at the cost of computational speed.

Lin et al. found that one reason that the two-stage algorithms generally achieves higher accuracy than one-stage algorithms, is the extreme foreground-background class imbalance in one-stage algorithms, which is caused by the fixed number of regions forward through the network. This class imbalance makes the one-stage algorithms predisposed to classify a region as a well represented class, such as the background. They presented RetinaNet[34], an extension of SSD, that uses a Focal Loss that down-weights the loss assigned to the well-classified examples. This resulted in an object detector that achieved state-of-the-art accuracy and matched the speed of one-stage detectors.

In general, choosing between these two types of models is a trade-off between speed and accuracy, where detection methods that use regional proposals usually are much slower, but more accurate than the one stage methods. For this project, it has been chosen to adapt the two stage method, Faster R-CNN, to the task of detecting railroad furniture. A two-stage method is chosen because COWI is more interested in high accuracy than in real-time performance. Furthermore, it is accessed easier to incorporate the time and/or spacial information, which is evidently available in videos, into these networks as the task of detection is divided into two sub-tasks. Faster R-CNN is chosen since it achieves high performance, is richly documented by the community, and there exist several excellent Pytorch[37] implementations. Furthermore, from a learning perspective Faster R-CNN is a good starting point as it is base for many newer network architectures. Thus, a good understanding of this model is useful for working with other two-stage object detectors.

2.2 Theory

As mentioned Faster R-CNN largely builds on top of its predecessor R-CNN and Fast R-CNN. Thus, the purpose of this section is to give an in-depth explanation of the R-CNN trilogy in chronological order.

2.2.1 R-CNN

In [12], Girshick et al. present two key insights that laid the groundwork for modern object detectors; CNNs can be used for region proposals and pre-trained networks yield significant performance boost for object detection. These insights are incorporated into the R-CNN.

The R-CNN consists of two stages; localization and classification. Figure 2.1 gives an highlevel overview of the model.

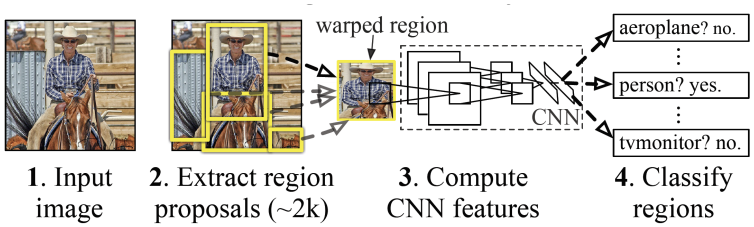


Figure 2.1: An overview of R-CNN [12]. Step 2 shows the region proposals found by selective search. Between step 2 and 3, the simple warping is used to fix the size of the Region of Interest (RoI). In step 3 the warped region is fed through a CNN. Step 4 uses a one-to-many SVM to calculate the probabilities of the object belonging to each class.

Localization The first step of R-CNN is to generate category independent region-proposals (Step 2 in Figure 2.1). The region proposals are generated using the selective search algorithm. Selective search works by recursively combining small pixel regions based on these regions' color, texture, size and shape similarities. The algorithm stops when a pre-defined number of regions is reached. At test time, the R-CNN generates around 2000 different sized region proposals for each input image. This method is rather inefficient and not trainable.

Classification In order to deal with the varying sizes of the region proposals, their sizes are fixed by a simple wrapping (Step 3 in Figure 2.1). The resized regions are fed into a CNN that outputs a 4096-dimensional feature vector (Step 4 in Figure 2.1). The chosen CNN is pre-trained for classification and during training fine-tuned to boost performance for the given detection task. The 4096-dimensional feature vector is fed into a set of category-specific Support Vector Machines (SVMs). This one-to-many SVM provides a confidence score for each region proposal for each class (Step 4 in Figure). Finally, non-maximum suppression is then used to remove the overlapping regions that have the lowest confidence scores.
In order to further boost the performance of R-CNN, a linear regression model is trained to predict a new region proposal given the features of the last pooling layer from the CNN and the original region proposal.

The bottleneck of R-CNN is that the 2000 region proposals discovered by selective search are individually fed through a CNN and then examined by a SVM. Thus, overlapping region proposals are examined more than once and do not share any information. This makes R-CNN very slow and infeasible for online end-to-end training due to very high memory requirements.

2.2.2 Fast R-CNN

Fast R-CNN is an improvement of R-CNN. The main idea is the same, however it is significantly faster and less memory demanding than R-CNN. It has a 9× faster training time, a 213× faster test-time and achieves a higher accuracy[17]. In this subsection, we will focus on the changes from R-CNN to Fast R-CNN.

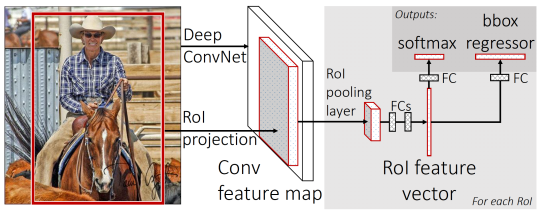


Figure 2.2: Fast R-CNN overview [17]. Both the image and the RoI are fed to a deep convolutional network. The RoI are projected onto the the feature map of this network, which are then resized using the RoI pooling layer. The network splits into two sibling layers that predicts respectively the class probability and the bounding box location for the given RoI projections.

Architectural changes Girshick identified that R-CNN was slow because the CNN had to calculate a feature vector for each region proposal - even though many of these were overlapping and thus shared information. To circumvent this bottleneck, the CNN in Fast R-CNN takes the entire image and the generated region proposals as input (See Figure 2.2). This means that the input image is processed only once. A pre-trained CNN (e.g. ResNet101 or VGG19) is used to generate a convolutional feature map; that is the features of the last convolutional layer in the pre-trained network. For each of the input region proposals, a Region of Interest (RoI) pooling layer is used to extract a fixed length feature vector from the feature map.

In Fast R-CNN, RoI pooling layers are used to fix the size of the region proposal. Thus, the RoI replaces the simple wrappings that was used in R-CNN. RoI pooling layers works by dividing a $h \times w$ sized region proposal into $h/H \times w/W$ sub windows, where H and W are chosen such that the output of the RoI layer is compatible with the input size of the subsequently fully connected layers. Max pooling is performed on each of the generated sub windows as shown in Figure 2.3.

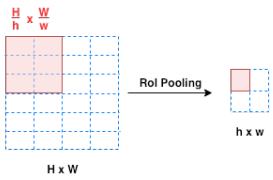


Figure 2.3: Shows an example of RoI pooling. In this example the size of the RoI is $H \times W = 4 \times 6$ and we wish to reduce the dimensions to $h \times w = 2 \times 2$. Thus, we divide the RoI into $\frac{H}{h} \times \frac{W}{w} = 2 \times 3$ sized sub windows (indicated by red square). We perform max pooling on this sub window to reduce its dimension to 1×1 cell.

Thus, the RoI pooling layer fixes the sizes of each region proposal such that they can easily be fed through a fully connected layer to generate a RoI feature vector. The network then branches out into a sibling layer; one produces a softmax probability for belonging to each class, the other produces a (x, y, w, h)

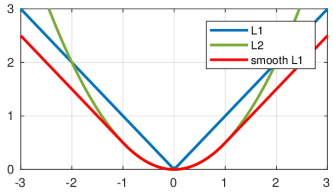


Figure 2.4: Shows the smooth_{L_1} loss used in Equation 2.3.

parametrization for the bounding box. Thus, the output size of the classification layer will be the number of classes and the output size of the regression layer will be the number of anchors times 4. This softmax probability layer replaces the set of linear SVMs. The softmax probability is convenient because the feature vector only needs to be fed through this layer once to produce a probability of belonging to each class compared to the one-vs-rest SVM approach in R-CNN, where the feature vector had to be fed through as many SVMs as there were classes.

By imposing these architectural changes, Girshick was able to unify the three individual parts of R-CNN into one network, which enable online end-to-end training of the model. The fact that the CNN operates on the input image and not on every region proposal significantly increases the speed of the localization as the region proposals can share weights. This increase in speed enables use of deeper, pre-trained networks such as VGG19 and ResNet101, which provides a performance boost of the model compared to R-CNN.

Loss Function Since Fast R-CNN uses a sibling layer that outputs respectively a bounding box prediction and an object probability, it is necessary to use a multi-loss function that consists of two parts; one for classification and one for localization.

$$L = L_{cls} + L_{loc} \tag{2.1}$$

The classification loss penalizes wrong classifications and the bounding box regression loss penalizes erroneous placements of the bounding boxes. The two loss functions are normalized such that their magnitudes become equal. The classification loss is the Cross Entropy, which is given by

$$L_{cls} = -\sum_i y_i^{cls} \log \hat{y}_i^{cls} \tag{2.2}$$

where y_i^{cls} is 1 for the true class and 0 otherwise and $\hat{y}_i^{cls}$ is the prediction probability for belonging to class i. Note that only when $y_i = 1$ there will be a loss, meaning that the CE only penalizes for the true class. The bounding box regression loss is given as

$$L_{loc} = \sum_{i \in \{x,y,w,h\}} \text{Smooth}_{L_1}(\hat{y}_i^{reg} - y_i^{reg}) \tag{2.3}$$

where y_i^{reg} and $\hat{y}_i^{reg}$ are respectively the ground truth and the predicted parameter for the bounding boxes where i runs over all 4 parameters in the parametrization of the bounding box. The smooth_{L_1} is a norm that is parallel to the L_1-norm, however in the range close to 0 it behaves smooth like the L_2-norm (Figure 2.4). This norm is chosen because it will care less about very small offsets of the bounding box predictions than the L_1, but it is more robust towards outliers than the L_2 norm.

2.2.3 Faster RCNN

The bottleneck in Fast RCNN is the selective search algorithm, which was used for generating region proposals. In [24], Ren et al. introduced Region Proposal Networks (RPNs) as a alternative to selective search. The RPNs were faster and trainable. Ren et al. showed that a RPN can be used as an attention mechanism for Fast R-CNN. This attention mechanism decreased the number of processed region proposals; improving both speed and accuracy. In the following RPN will be explained as well as how RPN works as an attention mechanism for Fast R-CNN.

Region Proposal Networks (RPN) A RPN takes a feature map of an image as input and creates as output a region proposal and an object score. It works by applying a sliding window on the inputted feature map. Each sliding window is fed through a small fully connected network that calculates a low dimensional feature vector. This feature vector is forwarded into a sibling layer; a box regression layer and a box classification layer - similar to the ones used in Fast-RCNN. As opposed to the classification layer in the Fast R-CNN detector, the classification in RPN provides an object score that describes the probability that the region proposal includes an object.

The RPN generates multiple region proposals per sliding window. Each of these region proposals are parametrized by four coordinates relative to an anchor. An anchor is defined by a center and a shape. In RPN, there is an anchor for each sliding window. Each anchor has a pre-defined size and aspect ratio that enables detection of overlapping objects. Thus, if three aspect ratios and three scales are pre-defined, we will have nine different anchors at each sliding window position. The RPN will provide a regression and an object score for each of these nine anchors.

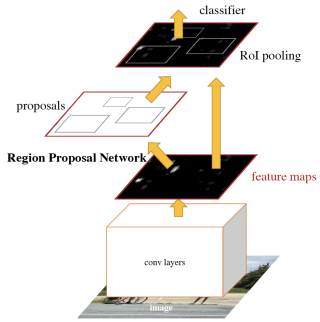


Figure 2.5: Faster R-CNN overview [24]. The image is fed through a deep convolutional network to calculate a feature map. The RPN finds regional proposals based on this feature map. These regions and the feature map are fed through the Fast R-CNN detector, which outputs bounding box predictions.

Architecture changes The Faster R-CNN consists of two stages; first, the input image is fed through a pre-trained CNN to generate a feature map (Figure 2.5). The RPN uses this feature map to generate several region proposals. Then, the feature map and these region proposals are fed into the Fast R-CNN detector. Thus, the main difference between Fast R-CNN and Faster R-CNN is that the selective search algorithm is replaced by the RPN. Note that because RPN and the Fast R-CNN detector operates on the same feature map, they can share weights (See Figure 2.5).

Loss Function Like Fast R-CNN, RPN is trained with the multi-loss function presented in Equation 2.1. As opposed to the classification loss in Fast R-CNN, the classification in RPN is binary and describes whether or not an object is present within a region proposal. In order for end-to-end training of Faster R-CNN the loss is set as the sum of the RPN loss and the Fast R-CNN loss.

$$L_{\text{Faster}} = L_{\text{RPN}} + L_{\text{Fast}} = (L_{\text{cls}}^{\text{RPN}} + L_{\text{reg}}^{\text{RPN}}) + (L_{\text{cls}}^{\text{Fast}} + L_{\text{reg}}^{\text{Fast}}) \tag{2.4}$$

where both L_{Faster} is the loss function for Faster R-CNN and L_{RPN} and L_{Fast} are both multi-loss functions described by Equation 2.1 for respectively the RPN and the Fast R-CNN detector. The magnitude of the individual loss functions are scaled to equal their importance.

As mentioned in Section 2.1, the Focal Loss is an alternative the the Cross Entropy. The Focal Loss greatly improves the performance of one stage models. Therefore, it is of interest how well it would integrate with a two stage method.

2.2.4 Focal Loss

The Focal Loss [34] was originally designed to handle the huge imbalance between foreground and background objects in one-stage methods. This section gives a mathematical formulation of the Focal Loss and describes how this loss function can be incorporated into Faster R-CNN.

The intuition of the Focal Loss is to down-weight easy examples and thus focus training on hard negatives. Thus, we wish to put more weight on classes that are poorly represented (in practice this is the case for all foreground objects). This is done by multiplying the commonly used Cross Entropy (CE) with a modulation factor $(1 - p_t)^\gamma$ where p_t is the probability of belonging to class t and γ is a tunable focusing parameter that decides how much focus should be put on hard examples. Thus, we can write the Focal Loss FL as

$$\mathrm{FL}\left(p_\mathrm{t}\right) = -\left(1 - p_\mathrm{t}\right)^\gamma \log\left(p_\mathrm{t}\right) \tag{2.5}$$

Figure 2.6 shows the Focal Loss for different values of γ. Note that the Focal Loss is equal to the Cross Entropy for $\gamma = 0$. The authors [34] have through experimental work found that $\gamma = 2$ works well.

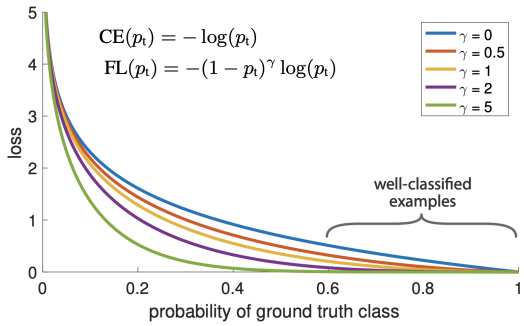


Figure 2.6: Shows the Focal Loss for different choices of γ [34].

From Figure 2.6, it is seen that by increasing γ less emphasis is put on well-classified examples. A numeric comparison between the Cross Entropy and the Focal Loss for $\gamma = 2$ is constructed to show the difference for well-classified and not well-classified examples.

For well-classified examples we have

$$\begin{aligned} \mathrm{CE}\left(p_\mathrm{t} = 0.9\right) &= 0.1 \\ \mathrm{FL}\left(p_\mathrm{t} = 0.9\right) &= 0.01 \end{aligned} \tag{2.6}$$

Whereas for not well-classified examples we have

$$\begin{aligned} \mathrm{CE}\left(p_\mathrm{t} = 0.1\right) &= 2.3 \\ \mathrm{FL}\left(p_\mathrm{t} = 0.1\right) &= 2.1 \end{aligned} \tag{2.7}$$

From Equation 2.6 and 2.7, it is seen that the Cross Entropy will penalize a poorly classified example $2.3/0.1 = 23$ times more than a well-classified example, whereas the Focal Loss will penalize a poorly classified example $2.1/0.01 = 210$ times more than a well-classified example. Thus, the Focal Loss puts more focus on hard examples.

Even though the class imbalance between foreground and background is not as extreme in two-stage methods as in one-stage methods, there is still a huge imbalance. In Faster R-CNN, the Focal Loss can directly replace the Cross Entropy in both the RPN and Fast R-CNN detector (Equation 2.4).
This section has explained the theory behind the experiments, which will later be performed on the Railroad data. The next sections will describe the dataset and explain how the experiments are executed.

2.3 Descriptive Analysis

This section provides an overview and description of the train journey dataset. Based on summary statistics, prior knowledge is obtained and data prepossessing choices are made.

2.3.1 Summary statistics

The dataset consists of $47,909$ panoramic images from a train journey in Queensland, Australia. The resolution of the images is 5400×1957 and the field of view (FOV) is 360 degrees. Figure 2.7 shows an image example from the train journey sequence.

Figure 2.7: An example of a panoramic image from the train journey sequence. 7 signs with red bounding boxes are located in the image. Are you able to find them?

The panoramic images are constructed by an unknown cylindrical or spherical projection of multiple images. From Figure 2.7 it is seen that this projection has severely distorted the area of the image to the left and right side of the tracks, while the area close to the train tracks' vanishing points (in both the front and rear direction) remains undistorted.

The railroad furniture along the train journey have manually been annotated with both bounding boxes and class labels (some bounding boxes are seen in Figure 2.7). There are $121,528$ annotated bounding boxes in the dataset. Each bounding box is assigned to one of 25 class labels. The class labels are explained in Table 2.1 and sorted after their number of occurrences.

Id	Description	Occurrences	Id	Description	Occurrences
141	Signals with four lamps, Front view	21987	210	Position lights with two white lamps and one red	2022
310	Standard speed sign	16093	250	Back view of any of the above Position lights	1977
131	Signals with three lamps, Front view	12442	330	Diverging right speed sign	1906
370	Back view of any of the above circular signs	9889	112	Signals with one lamp, Side view	1834
150	Back view of signals	9763	142	Signals with four lamps, Side view	1546
400	Signal number Sign	9526	220	Position lights with only two white lamps	1452
160	Other signals	9144	350	Speed train unit sign	1241
111	Signals with one lamp, Front view	4363	390	The back view of any of the above rectangular signs	1082
121	Signals with two lamps, Front view	3627	240	Side view of any of the above Position lights	1011
320	Diverging left speed sign	2932	132	Signals with three lamps, Side view	411
122	Signals with two lamps, Side view	2722	340	Diverging left-right speed sign	352
360	Other Speed signs	2103	230	Position lights with only two white lamps and a C letter	38
380	Back view of diamond signs	2065			

Table 2.1: Description and frequency of every class label (Id) in the dataset.

From Table 2.1 it is seen that some of the descriptions are very similar which might lead to erroneous manual annotation. It is also seen that some class labels are very rare e.g. class label 230 with only 38 occurrences in the entire dataset.

As an example, Figure 2.8 shows 6 crops of annotated speed signs. It is seen that the size, quality, and aspect ratios of these signs vary. Furthermore, it is seen that the bounding boxes are sometimes tightly fitted and other times more loosely fitted.

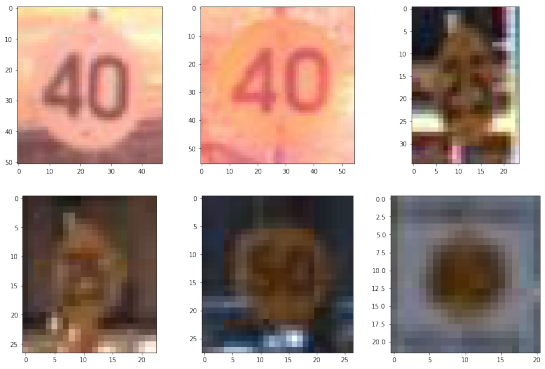
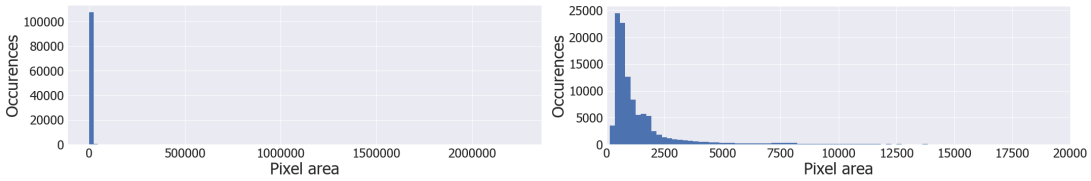


Figure 2.8: 6 crops of annotated speed signs are shown.

By investigating the size of the annotated bounding boxes, we find that most signs appear very small, however few, large bounding boxes exists. The distribution of the area of the bounding boxes is shown in Figure 2.9a. The biggest bounding box covers more than 20% of the full panoramic view. After manual inspection, it is found that this is a erroneous annotation, and therefore, it is decided only to consider bounding boxes with an area less than a 180^2 pixels. We look at the distribution of the area of annotated bounding boxes with an area less than 180^2 pixels ($\approx 0.3\%$ of the full panoramic view), which accounts for more than 99% of the annotations (see Figure 2.9b).



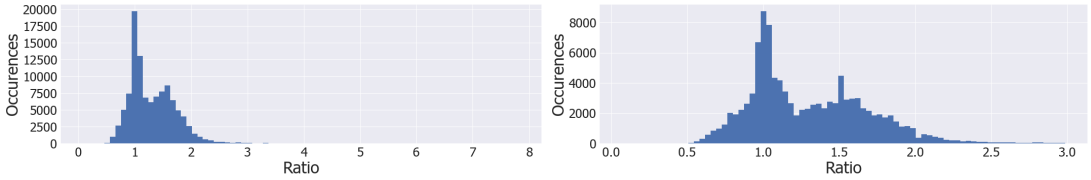
(a) All data

(b) 99% of data

Figure 2.9: Distribution of area of annotated bounding boxes.

From Figure 2.9 we see that most bounding boxes are small. We choose this threshold as it still accounts for 99% of the data. We find that the median area is $868 \approx 30 \times 30$ pixels. This is significantly smaller than median size of bounding boxes in most other object detection dataset [16], [43], [44]. This information will in Section 2.5 be used for modelling the anchor sizes for Faster R-CNN and evaluation purposes in Section 2.6.

We investigate the aspect ratios of the signs and signals in Figure 2.10. The figure shows the distribution of the aspect ratios obtained by dividing the height with the width of the bounding boxes.



(a) All data (b) 99% of data

Figure 2.10: Shows a distribution of the aspect ratios of the bounding boxes obtained by dividing the height with the width.

Again, we can describe more than 99% of the data by using a smaller interval. Figure 2.10b, it is seen that distribution of the aspect ratio have two peaks at respectively 1.0 - indicating quadratic bounding boxes - and 1.6. We will also use this prior knowledge to model the anchors in Section 2.5.

As the train's movement and the position of the railroad furniture are constrained by the train tracks, it is expected that the positions of the signs will be similar within most frames. To investigate this we create two contour plots. Figure 2.11 is a binary contour plot showing where the center positions of the bounding boxes have been observed. Figure 2.12 shows a heat map on top of an panoramic image. The heat map describes the density of observed bounding boxes across all panoramic images.

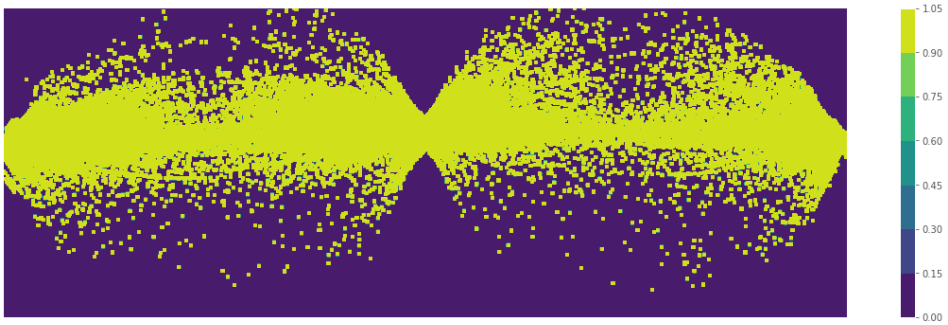


Figure 2.11: A binary map showing were the center coordinate of the annotated bounding boxes appeared across the entire dataset.

Figure 2.12: Heat map of density of annotated bounding boxes.

From Figure 2.11 it is seen that the positions of the centers of the bounding boxes follow the distortion field - closely gathered at the vanishing points and sparsely distributed in between the vanishing points. The heat map (Figure 2.12) shows that the density of observed railroad furniture is highest at the vanishing points of the tracks. This makes sense since railroad furniture at these locations are further away and thus their position in the image will change less between frames. This will make them appear in more similar positions over multiple frames.

In this section we have provided an overview of the data. We have further investigated the annotations. We found that some class labels are very rare, which will make learning difficult for these classes. We found that most bounding boxes are small and the median area is 30×30 pixels. We decided to remove all bounding boxes with an area more than 180^2 as these outliers are expected to be erroneous. We found the distribution of aspect ratios of the bounding boxes have a peak at 1.0 and 1.6. This prior knowledge about the bounding boxes can be used for fine turning the anchors in Faster R-CNN. Finally, we looked at the image positions of the bounding boxes. It was found that the centers of the bounding boxes are positioned along the field of distortion and that the density of bounding boxes is significantly higher at the vanishing points in both the front and rear direction.

2.4 Design choices

Training and testing on the entire dataset would take a very long time and is therefore outside the scope of this project. Consequently, it is chosen to decrease the size of the dataset such that more focus can be allocated to the implementation, evaluation and reportation. This section describes how the dataset is limited and prepared for training and testing the models.

2.4.1 Data limitation

The following procedure is implemented to reduce the size of the dataset.

Image classes The 25 classes have been reduced to 2 classes: Signs[1] and signals[2]. These two new classes have approximately the same size, thus the challenges with of a screwed class distribution is avoided.

Image size Instead of working with the full panoramic images, it is chosen only to use one 500×800 crop per image. This crop is taken from the center of the panoramic image where the distortion is small. Only bounding boxes, which are at least 1/4 inside this crop are considered. Bounding boxes that are located on the boundary of the cropped image are also cropped (See Figure 2.13).

2.4.2 Train, validation and test set division

In order to test a model and avoid overfitting, it is common practice to divide a dataset into a training, validation and test set. The model should be trained on the training set and hyper-parameter fine-tuning performed on the validation set. Finally, when a model has been decided, it should be trained on both the training and validation data and evaluated on an independent test set. As the test set is used to check the generalization of the model, it is important that the training and test set are independent. In the following part we will describe how the data is divided into a *trainval* (consisting of both training and validation set) and a test set. Appendix A.1 verifies that consecutive frames are correlated. Therefore, training and test set division must be handed with care.

[1] Sign ids from Table 2.1: 310, 370, 400, 320, 360, 380, 330, 350, 390, 340.
[2] Signal ids from Table 2.1: 141, 131, 150, 160, 111, 121, 122, 210, 250, 112, 142, 220, 240, 132, 340, 230.

Figure 2.13: Shows an image crop of 500×800 pixels taken from the center of the panoramic image. The red bounding boxes show the used bounding boxes. The green bounding box shows the original bounding box. From this figure it is seen that one bounding box has be cropped. The choice of crop means that we do not have to deal with the distortion observed in Figure 2.7.

`Trainval` and test division

As described in Appendix A.1, the frames in the video sequence are correlated in both time and space. E.g. if we have observed a stop sign in image i we will probably observe a scaled version of the same sign in image $i+1$ as seen in Figure 2.14. Especially, in our dataset this might cause a big problem as the train is sometimes stopped, thus two objects can appear extremely similar in two consecutive frames (Figure 2.14).

(a) Image $i-1$ (b) Image i (c) Image $i+1$

Figure 2.14: Shows 3 consecutive frames. From this short sequence, it is easy to observe that the frames and the position of the bounding boxes are correlated over time. Thus, it would not be appropriate to put image $i-1$ and $i+1$ in the *trainval* set and image i in the test set, because then *trainval* and test set would be correlated.

Based on the observed correlation between consecutive frames, it is chosen to divide the first part of the data into *trainval* set and the second part of the data into a test set. The *trainval* set and test set are separated by more than 2000 frames, and should therefore not be correlated in neither time nor space. The *trainval*-test division is illustrated in Figure 2.15 with a green and red bar under the x-axis.

In order to verify that the last frame in the training set and the first image in the test set are indeed uncorrelated, we show the two images in Figure 2.16.

From visual inspection of both the `trainval` and test set it is seen that they do not have a spatial overlap and thus should be uncorrelated in both time and space. Furthermore, it is found that the test set consists of a challenging environment with several tunnels, moving objects and drastic light

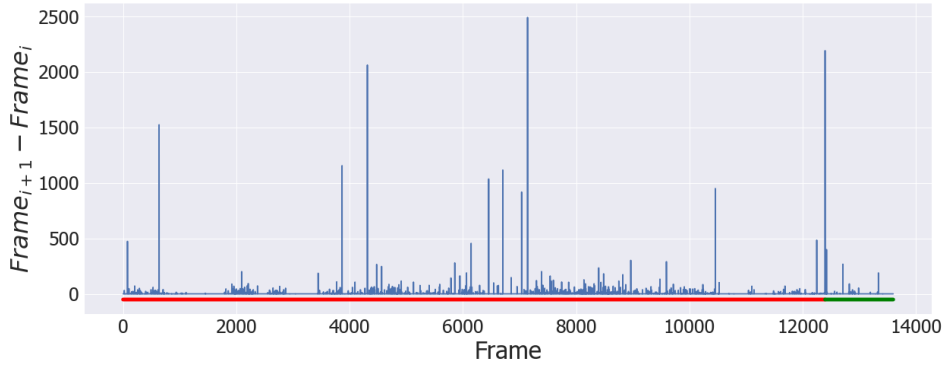


Figure 2.15: Shows the number of frames that are between the cropped image with annotations. The red and the green bars at the bottom indicate the portion of data used as respectively training and test set. It was found with manual inspection that the longer intervals without any railroad furniture were because the train has stopped or drove through a tunnel.

Figure 2.16: Shows the last frame in the training set and the first frame in the test set. As seen these images have little in common. To the left is image 12396 and to the right is image 12397. There are 2192 frames between the two frames, thus neither spacial nor time dependency.

changes that makes it representative for testing the generalization capabilities of the model. The test set consists of 1149 signs and 1693 signals of varying sizes. However, one must note that the test set only tests the generalization capabilities for Queensland-like urban environment. In order to evaluate generalization capabilities beyond this environment, a railroad sequence from different a different city, season or country should be recorded and annotated. Annotations of signs and signals appearing in the Nordland[13] dataset would provide this.

Validation and training division

For training and validation purposes, we only consider images that contain at least one annotation. These images have information about both the appearances of the objects and the background, whereas the images without annotations only have information about the background. This limits the size of the *trainval* set to $12,396$ images. By random sampling we divide the *trainval* set into a training and validation set of respectively 90% and 10% of the data.

2.5 Experiments

A Faster R-CNN implementation[3] was adapted to the railroad dataset. In this section we describe the five performed experiments.

Johnson et al.[4] made a thorough analysis on the performance of various CNNs. They found that ResNet performance was better than VGG in both computational time and accuracy for classification tasks. This is because ResNet is deeper, which is acheived through skip connection in the model architecture[19]. Furthermore, the Yang, who has implemented a Pytorch version of Faster-RCNN that we have adapted, also reports a significantly higher mAP by using ResNet101 than VGG16 on various datasets without too much increase in the computational time. Based on these findings, it is chosen only to consider the ResNet101 base network in all experiments.

To avoid overfitting, data augmentation has also been imposed in all experiments. The images are mirrored in their vertical axis such that we double the amount of annotated images. This data augmentation will hopefully learn the model invariance towards which side of the tracks the railroad furniture appear. If each of the original 25 classes were treated separately, it would not be recommended to mirror the images as useful information about, which side of the tracks the railroad furniture were located would be lost. However, as the categories are collapsed into just 2 (signs and signals) where there is no general information about the location of the objects, it is not expected that we will lose any information by mirroring the images.

Experiment 1 In this experiment, the default parameters from the implementation were used. The purpose of this model was to serve as a baseline for the latter experiments.

Experiment 2 Next, reduced the anchor scales to fit the prior knowledge obtained in Section 2.3. We also increased the learning decay, such that the learning rate was reduced every 10th epoch instead of every 4th. We did this because, the default implementation only trained for 10 epochs on a larger dataset, and as we train for 30 epochs on a smaller dataset, the learning rate should be updated less frequently.

Experiment 3 We replicated experiment 2, but reduced the number of anchor ratios. The anchor ratios were also based on prior knowledge obtained in Section 2.3.

Experiment 4 We tested experiment 2 with the Focal Loss with $\gamma = 2$.

Experiment 5 We tested experiment 3 with the Focal Loss with $\gamma = 2$.

All hyper parameters for the experiments are listed in Table 2.2. For better comparison between the performed experiments, we only vary one hyper parameters per experiment. In these experiments, the anchor scales and aspect ratios have been changed based on prior knowledge about the bounding boxes sizes (Figure 2.9b) and aspect ratios (Figure 2.10b). In this regard, it is important to note that the anchors are found on a feature map that is $1/16$ of the original image size, thus the scales should be multiplied by 16 to get the pixel sizes of the anchors.

Experiment 2 and 3 are replicated with the Focal Loss (FL) instead of the Cross Entropy (CE). A Focal Loss implementation[5] was adapted to the Faster R-CNN loss function. Both Experiment 4 and 5 use a Focal Loss with $\gamma = 2$.

Table 2.3 shows the training time for 30 epochs on a Nvidia Titan X. From the table, it is seen that small anchors increase the time to process an image, since it takes more windows to cover the feature

[3]Pytorch implementation of Faster-RCNN: `https://github.com/jwyang/faster-rcnn.pytorch`
[4]Benchmarks on popular CNNs: `https://github.com/jcjohnson/cnn-benchmarks`
[5]Focal Loss implementation: `https://github.com/clcarwin/focal_loss_pytorch`

Parameter	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
Base network	resnet101	resnet101	resnet101	resnet101	resnet101
Batch size	1	1	1	1	1
Epochs	30	30	30	30	30
Learning rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Learning decay	4	**10**	10	10	10
Anchor ratios	0.5, 1, 2	0.5, 1, 2	**1, 1.5**	**0.5, 1, 2**	**1, 1.5**
Anchor scales	8, 16, 32	**1.25, 2.5, 5**	1.25, 2.5, 5	1.25, 2.5, 5	1.25, 2.5, 5
Loss function	CE	CE	CE	**FL**	FL
Optimizer	SGD	SGD	SGD	SGD	SGD

Table 2.2: Hyper parameters and configurations. The bold numbers indicate which hyper parameters have been altered from the previous experiment. Note that a small batch size has been chosen across all experiments due to memory issues.

map (difference between Experiment 1 and 2). In Experiment 3, we only use 6 anchors, thus even though these are smaller than Experiment 1, the time to process the image is decreased.

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
Hours	55.55	75.57	52.98	76.05	55.48

Table 2.3: The training time for 30 epochs in hours on a Nvidia Titan X. Note that reducing the number of anchors and their sizes have a huge impact on the training time.

2.6 Results

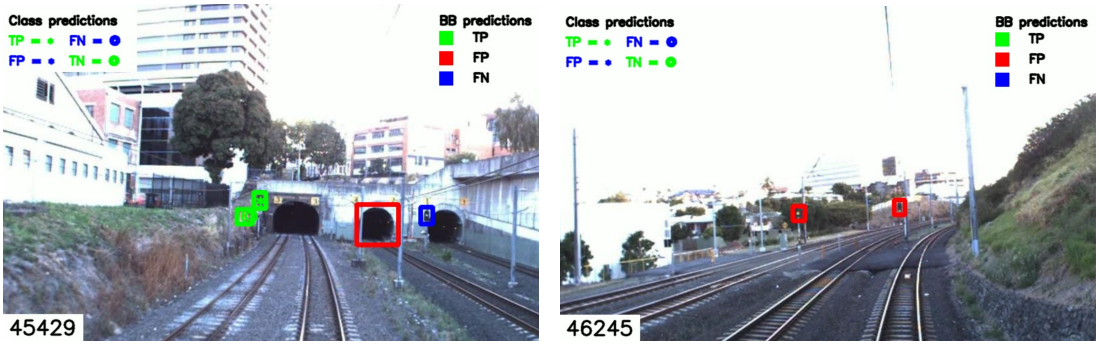
The experiments are evaluated on several metrics (Appendix A.2) for better understanding of the errors. Videos showing the results are available at `https://www.dropbox.com/sh/31haxfnn7c807f7/AAAui9ItaPF0Tn7t77bFmVcUa?dl=0`. During the test sequence the train passes 1149 signs and 1693 signals of varying sizes.

Table 2.4 shows the Average Precision (AP) for both classes and the mean AP (mAP) for each experiment of the five experiments described in Section 2.5. For an explanation of AP and mAP please consult Appendix A.2.

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
AP_{signal}	0.496	0.581	0.552	**0.666**	0.601
AP_{sign}	0.534	**0.610**	0.588	0.562	0.479
mAP	0.515	0.595	0.570	**0.614**	0.540

Table 2.4: The Average Precision (AP) for both classes and the mean Average Precision (mAP) for the experiments. See Section 2.5 for explanation of the different experiments.

From Table 2.4 we see that experiment 4 has the highest mAP. It is seen that both Model 2 and 3 perform better than Model 1, thus decreasing the anchor sizes, seem to improve the mAP. Surprisingly, the inclusion of the Focal Loss showed improvements for Model 4, but not for Model 5. As both Model 2 and 4 perform better than Model 3 and 5, it must be concluded that using 3 aspect ratios instead of 2 improves performance.



(a) It is seen that the model detects two objects correctly (green); respectively a sign (TP) and a signal (TN). The model makes a false positive prediction (red) as it believes that the tunnel entrance is a railroad furniture. Also, it makes a false negative prediction as it does not find the ground truth signal (blue).

(b) In this example we see that the model detects two railroad furniture, which are evaluated as false positives (red), whereas in fact, they should have been true positives, since the ground truth bounding box is simply missing. This is a common trend throughout the dataset.

Figure 2.17: Shows two examples from Experiment 4. Videos with the results from all five experiments can be found at[6]

We further investigate the error by studying the Precision Recall (PR)-curves in Figure 2.18. The procedure of calculating the PR curve is described in Appendix A.2. The PR curves are calculated for IoU greater than 0.5.

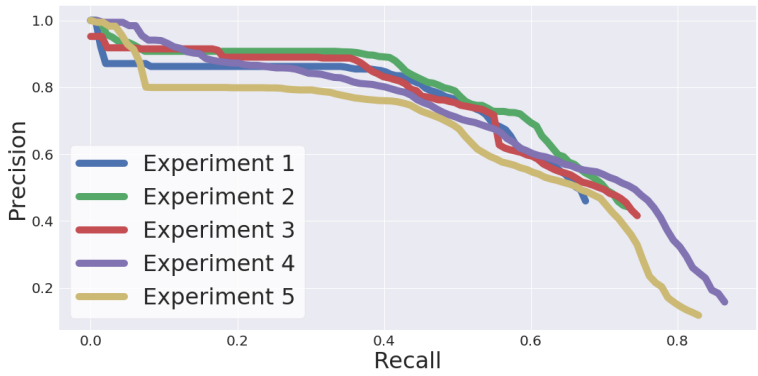
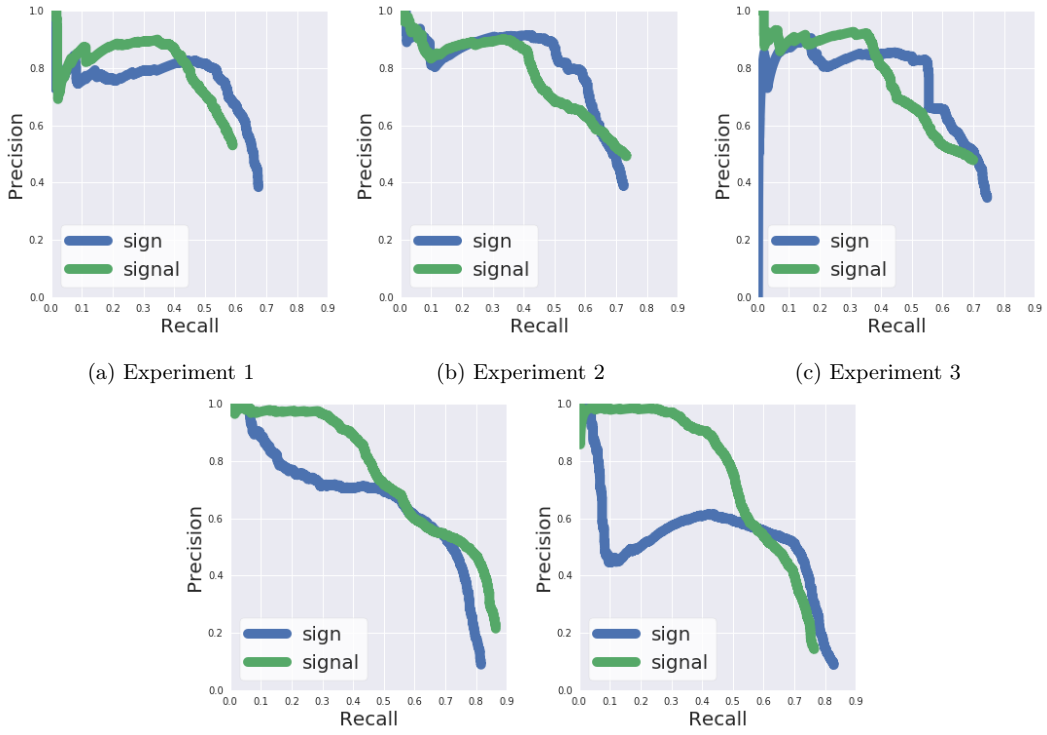


Figure 2.18: Shows the PR-curve for each experiment.

From Figure 2.18, we note that the maximum recall for experiment 4 and 5 are significantly higher than for experiment 1-3. The recall describes the model's ability to locate objects. Thus, from this metric it is seen that the Focal Loss helps localizing the railroad furniture. However, all the models have a recall of less than 0.85 meaning that we at most find 85% of the ground truth bounding boxes.
Furthermore, it is seen from the PR curves that experiment 4 differentiates from experiment 5 by having a higher precision for most recall values. This means that experiment 4 is better at classifying whether a bounding box prediction is a sign or a signal. This might be due to the increased number of anchors, which increases the differentiability between the bounding box predictions.

In Figure 2.19 each of the precision-recall curves are divided into a precision-recall for each class (signs and signals). Furthermore, it is chosen to plot these curves without the maximum interpolation from

Equation A.4 to better investigate for which prediction scores errors occur.



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

(d) Experiment 4 (e) Experiment 5

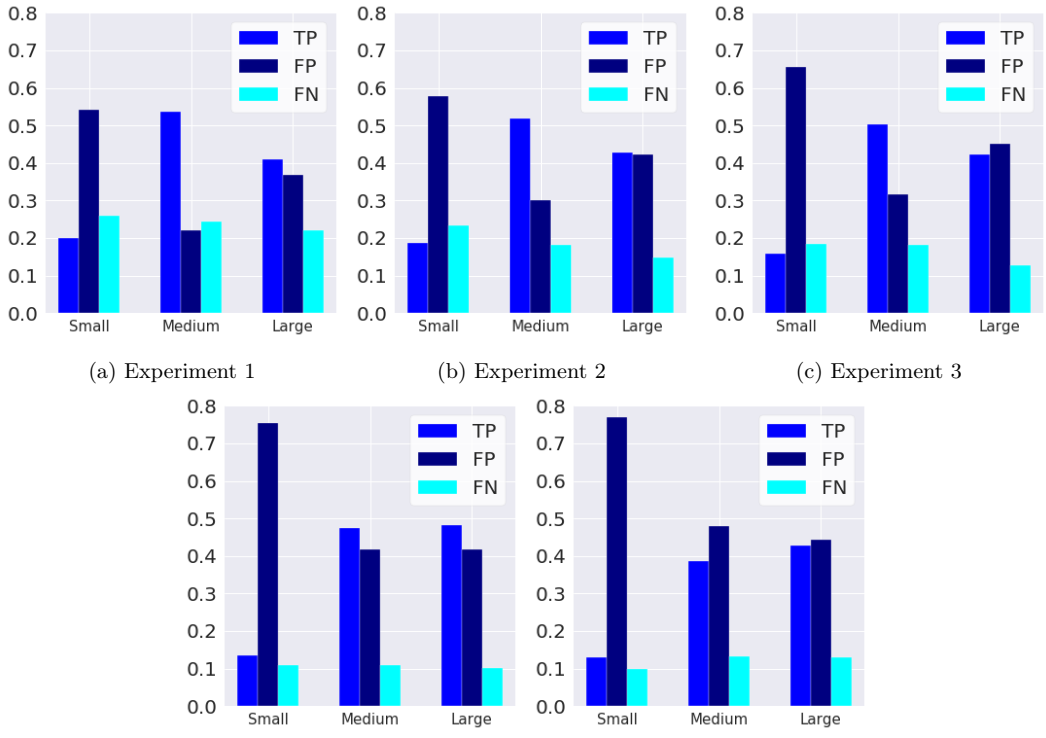
Figure 2.19: Precision and recall curves for signs and signals. See text for explanation.

From Figure 2.19, it is seen that in all five experiments, the model gets approximately the same recall for both classes. However, especially, in experiment 4 and 5, the model obtains significantly higher precision for signals than for signs. This trend, however, is not evident in the first three experiments. This might be because the Focal Loss does not penalize well classified objects much, and since inter-class dissimilarities for signs are larger than signals, the Focal Loss will have difficulties fitting to the more dissimilar signs.

Figure 2.20 together with Table 2.5 shows for which type of localization errors occur for different sized bounding boxes. We differentiate between TP, FP, FN for small, medium and large bounding box predictions, which have areas of respectively less than 20^2 pixels, between 20^2 and 40^2 pixels, and larger than 40^2 pixels. It must be noted that less 40^2 would all be considered small in standard terminology[7]. Figure 2.20 shows the distributions for each experiment.

Studying the within size-category relationship between TP, FP and FN between the different experiment, it is seen that the amount of FP is very high in all experiments; especially in experiment 4 and 5. This stems from the Focal Loss, which decreases the probability of predicting background (the most well-classified class), which leads to the prediction of more FP. Comparing the top row in Figure 2.20, it is seen that in both experiment 2 and 3 there are more FP than experiment 1. However, it is also observed that experiment 2 and 3 perform better as these have more TP and less FN than experiment 1. This might be more evident in Table 2.5. However, as seen from Figure 2.17 and by investigation of the result

[7]This thresholds are significantly lower than in popular dataset such as COCO that uses respectively less than 36^2, between 36^2 and 96^2, and above 96^2[15] to distinguish between object sizes



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

(d) Experiment 4 (e) Experiment 5

Figure 2.20: Shows the size-specific distribution of true positives, false positives and false negatives for small, medium and large bounding boxes, where small, medium and large refers to the whether the areas less than 20^2, between 20^2 and 40^2 or larger than 40^2. From the figure, we see that many FP are predicted. However, we are not too concerned about these as described in the text below.

videos[8], it is observed that many of the FP are caused by poor annotations and should in fact be TP - especially for the small railroad furniture this is a big problem. By visual inspection, it is observed that especially many of the small and distant railroad furniture are lacking annotations. Therefore, we are not concerned about many FP when evaluating the performance of the models as many of these might actually be TP.

	TP_s	TP_m	TP_l	FP_s	FP_m	FP_l	FN_s	FN_m	FN_l	TP_t	FP_t	FN_t
Experiment 1	150	1138	512	**407**	**469**	**461**	196	518	277	1800	**1337**	991
Experiment 2	151	1137	675	465	660	668	187	399	235	1963	1793	821
Experiment 3	152	1241	597	631	781	638	178	446	181	1990	2050	805
Experiment 4	168	**1308**	**719**	930	1150	620	**136**	**300**	**150**	**2195**	2700	**586**
Experiment 5	**185**	1229	628	1089	1526	651	141	419	192	2042	3266	752

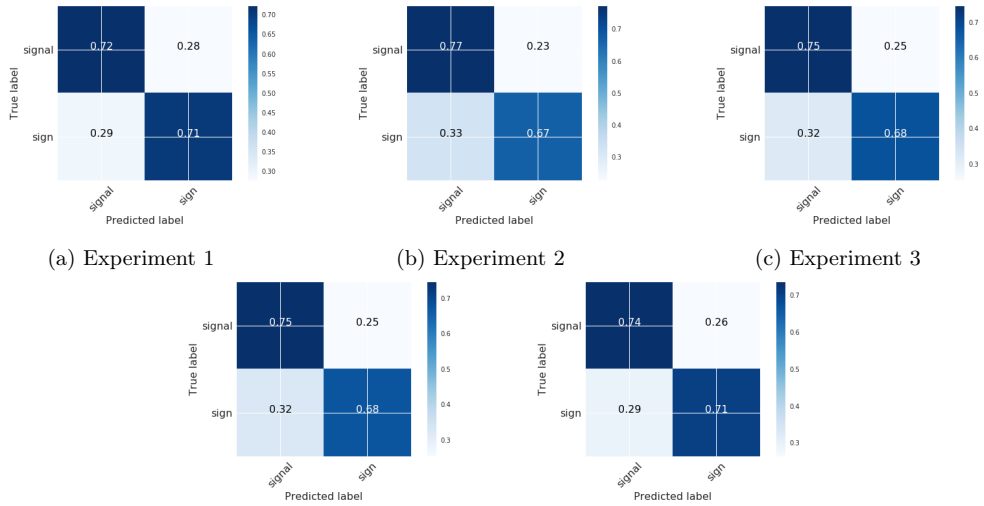
Table 2.5: Shows the TP, FP and FN for the different experiments and size-categories, where the subscripts s, m, l indicate respectively small, medium and large. t is the total across scales.

Based on Table 2.5 and Figure 2.20 it is seen that the models in general has most difficulties detecting small objects (less than 20^2 pixels) as we see a rather high percentage of FN. Furthermore, model 4 seems

[8] `https://www.dropbox.com/sh/31haxfnn7c807f7/AAAui9ItaPF0Tn7t77bFmVcUa?dl=0`

to be slightly better than model 5 at localizing objects. Surprisingly, model 5 performs best at localizing small objects, even though it does only have 6 anchors.

Lastly, we study the classification of signs and signals. From Figure 2.21 we show a confusion matrix from each experiment. From these confusion matrices, we see that in general it is easier for the models to get an high accuracy for signals than for signs. This suggests larger appearance variation for the signs in the test data. We observe that experiment 2 has the highest signal accuracy whereas experiment 1 and 5 has the highest sign accuracy.



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

(d) Experiment 4 (e) Experiment 5

Figure 2.21: Shows the size and class specific percentages of respectively TP, FP, TN, FN. See the text for more explanation.

The results from the five experiments have shown that models with the Focal Loss generally performed better than models with the Cross Entropy. Furthermore, we saw that exploiting the prior knowledge about the object sizes improved the mAP, however, it seemed to worsen the mAP to reduce the number of anchor boxes from 9 to 6. Model 4 achieved the highest mAP. This was mainly due to the improved capability localizing objects witnessed by its relatively high recall. Thus, it must be concluded advantageous to reduce the anchor sizes to fit the prior knowledge about the data.
In experiment 3 and 5, the number of anchor ratios were decreased to 2. This resulted in significantly faster training time, however slightly lower mAP after 30 epochs. The lower mAP was primarily based on lower precision for low confidence predictions. Investigation of the errors showed that model 4 and 5 predicted significantly more FP than the other models due to the Focal Loss assigning less probability to the background class.

2.7 Part Conclusion

In this chapter, we have adapted the Faster R-CNN model to detect signs and signals in the Railroad dataset. Through a descriptive analysis, the dataset's size and difficulty was reduced to fit the scope of the project, and prior knowledge about anchors were obtained. Several experiments have been carried out exploiting this prior knowledge as well as experimenting with the usage of the Focal Loss instead of the Cross Entropy.

Through evaluation of the performance of the trained models, it was found that the Focal Loss improved the mAP. Moreover, it was found that it improved the mAP to reduce the anchor size to fit the prior knowledge on the anchors, however reducing the number of anchors did decrease the training time. By investigation of the type and frequency of errors, it was established that all models, and especially the ones using the Focal Loss, found many false positives. However, many of these false positives were caused by erroneous annotations and should in fact have been true positives. Thus, there should not be put too much emphasis on this type of error, which makes the mAP an excellent performance estimator for this dataset.

There are several very interesting paths to continue the work on the 2D object detection. An obvious next step would be to extend the training and evaluation to the entire dataset. This raises several challenges, e.g. dealing with more classes (of which some are very poorly represented) and dealing with very large, distorted images. It is believed that the Focal Loss will be advantageous to handle the sparsely represented classes. The large panoramic images could be dealt with by crops, as it would not be feasible to feed the network the entire panoramic image due to memory issues. In this regard, interesting work remains in reverse engineering the distortion for the panoramic images.
A second interesting path would be to test and evaluate the performance of some of the one-stage methods mentioned in Section 2.1. Especially, the RetinaNet, which uses the Focal Loss, would be interesting as it is expected that this method will handle the sparsely represented classes well.
Finally, as we are dealing with a video, it would be interesting to exploit the sequential information described in Section 2.4. An simple approach would be to perform a majority vote across multiple frames to boost the performance of one of the single frame models. It is expected that this majority voting would reduce the number of false positives.

From the temporal dependencies between the frames in the video, it is furthermore possible to extract spatial information about the 3D location of the railroad furniture. The next chapter seeks to extend the 2D detection to a 3D location for the railroad furniture.

Chapter 3

Depth Estimation

The goal of the second part of this project is to enable an estimation of the 3D location of the detected railroad furniture. This is important for COWI as it will provide them with an up-to-date map of the positions of all the detected railroad furniture, which can improve safety and reduce maintenance cost. More formally, we intent to map the corner points, $q_i = [x_i, y_i]$, of the detected bounding boxes to their corresponding 3D locations, $Q_i = [X_i, Y_i, Z_i]$, in a global coordinate system. For monocular images, this is a difficult and ambiguous task since a 2D point, q_i, can be the projection of an infinitely number of 3D shapes. A unique solution to this ambiguous task can be found if the depth, Z_i, is known using the pinhole model[3]. Therefore, this chapter seeks to estimate a dense depth map for each frame in the Railroad dataset that will enable COWI to acquire the 3D position of the detected railroad furniture.

3.1 Related work

The toolchain for depth estimation from monocular images is widely studied in Structure from Motion (SfM). It is outside the scope of this project to account for the various methods within SfM to estimate the depth, however the interested reader can consults Hartley and Zisserman's Multiple View Geometry[3] or the following surveys [9], [10], [36].

Whilst the traditional toolchain for depth estimation is effective and efficient in many cases, it relies on accurate image correspondences. These image correspondences are traditionally obtained from hand-crafted features (e.g. SIFT[5], FAST[8], SURF[14], ORB[30]), which works well when there is a clear image correspondence, but fails when this is not the case e.g. occlusions of objects, lack of texture, non-static scenery or thin structures [36]. To address these issues, researchers have used learned-based features to handle several stages of the pipeline, e.g. feature matching[18]. These learned-based techniques are attractive as they can leverage external supervision during training, and hopefully become more robust towards the above issues when applied to test data.

Recently, it has been proposed to train networks end-to-end to learn the entire pipeline for depth estimation[28], [29]. These methods falls into the two categories; supervised and self-supervised depth estimation methods. An advantage with the self-supervised methods is that they do not require ground truth depth estimations as they can unsupervised learn depth through short sequences of consecutive frames. Furthermore, the self-supervised methods have shown superior performance[42]. The inferior performance of the supervised method is caused by the limited amount of depth annotated imagery and the inaccuracy of the ground truth estimations that are usually obtained by LiDAR scanners. Since, ground truth depth estimates are not available for the Railroad dataset, and because of the superior performance of the self-supervised methods this project solely focuses on these methods. The following section highlights recent achievements within self-supervised depth estimation for monocular image sequences.

3.2 Theory

Zhou et al. were the first to propose a self-supervised deep learning method for depth and ego-motion estimation[38]. Since their discovery, several researchers have suggested refinements that have further improved the accuracy, however, the fundamental approach that Zhou et al. proposed have not changed. Therefore, we will explain the `SfM-learner` suggested by Zhou et al. before we reveal two of the most recent improvements; respectively `vid2depth` and `struct2depth`.

3.2.1 SfM-learner

The `SfM-learner` consists of two networks; a depth CNN and a pose CNN that independently estimates respectively a dense depth map for an image and the relative camera pose between two frames. The two networks are able to learn respectively depth and ego-motion through a self-supervised learning approach, which is based on Depth-Image Based Rendering[4]. Depth-Image Based Rendering is the procedure of mapping a 2D point with an associated depth to a 3D point followed by a projection of this 3D point onto a 2D image plane from another viewpoint (Figure 3.1). The essential idea of this self-supervised method is to use the Depth-Image Based Rendering procedure as supervision for the model. This supervision enables the depth network and the ego-motion network to learn respectively depth and ego-motion as an intermediate task.

Depth-Image Based Rendering

Fehn [4] found that 2D points on the image plane of a camera can be reprojected into their corresponding 3D points given the depth of each of these points, and then projected onto the image plane of a "virtual" camera at another view position. Using Zhou et al.'s formulation, we have an image sequence $I_1 \dots I_N$ where an image I_t is the target view and the rest being source views I_s where $1 \leq s \leq N, s \neq t$. We will proceed by only considering one source image and one point correspondence to ease the notation, however, the method generalizes to dense images and to multiple source images. In practice the method uses 2 or 4 source images and 1 target image.

A 3D point M can be projected onto the image planes I_t and I_s using the pinhole model (Equation 3.1 and 3.2). We set the world coordinate system as the coordinate system of the target image I_t to simplify the equation.

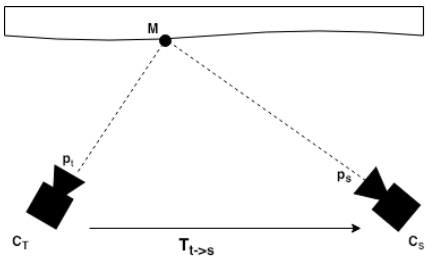


Figure 3.1: Shows the camera setup for one target camera c_t and one source camera c_s. M indicates a 3D point that is projected onto either the point p_t or p_s located respectively in image plane I_t or I_s. $T_{t\to s}$ is the transformation matrix that describes the rotation and translation from the target camera, c_t, to source camera, c_s. Note that if $T_{t\to s} = 0$, the point M would be observed without parallax, and thus its 3D location cannot be determined. Thus, we require motion between the frames.

$$\tilde{p}_t \cong K P_n \tilde{M} \tag{3.1}$$

$$\tilde{p}_s \cong K P_n T_{t\to s} \tilde{M} \tag{3.2}$$

where p_t and p_s are the projection of M into respectively the target and source image, K is the intrinsic camera parameters (for simplicity they are chosen to be the same), P_n is an 3×4 identity matrix. $T_{t \to s}$ is a 4×4 matrix that describe the rotation and translation of the source view relative to the target view. The tilde indicates homogeneous coordinates and $\cong$ indicate that these correspondences are identical "up to a scale factor". If $T_{t \to s} = 0$, the point M would be observed without parallax between the two frames. This would make it impractical to estimate its 3D position. Thus, we require motion between the consecutive frames. We expect this to be a huge problem in the Railroad dataset as we earlier observed that the train stops on several occasions. Furthermore, we require that the point M is visible in both frames.
In the remainder of this section, we aim to deduce the point correspondence between $\hat{p}_t$ and $\hat{p}_s$. In order to do so, we first multiply with P_n and rearrange Equation 3.1, such that we get

$$K^{-1}\tilde{p}_t \cong M \tag{3.3}$$

We then fix the scale by the depth z_t. In order words, we have a unique correspondence between a 3D point M and a 2D point $\hat{p}_t$ if we know both the camera intrinsics K and the depth z_t.

$$z_t K^{-1}\tilde{p}_t = M \tag{3.4}$$

where z_t is depth of the 3D point from the target view, which determines the scale. By substituting Equation 3.4 into Equation 3.2, we further get

$$z_s p_s = K T_{t \to s} z_t K^{-1} p_t \tag{3.5}$$

where z_s is the depth of the 3D point from the source view. As the depth CNN has estimated a dense depth map $\hat{D}_t$ given the target view, we can substitute $z_t = \hat{D}_t(p_t)$ and rewrite the equation

$$p_s \sim K T_{t \to s} \hat{D}_t(p_t) K^{-1} p_t \tag{3.6}$$

where the $\sim$ indicates that p_s is only known up to a scale factor. Thus, this equation describes a pixel-wise view synthesis of the target image seen from the source image. We use bilinear interpolation of p_s's four closest neighboring pixels in order to discretize p_s (Figure 3.2).

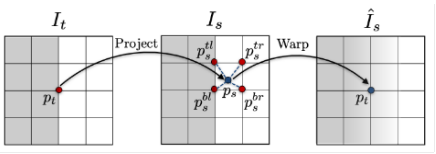


Figure 3.2: Shows the process of bilinear interpolation. A continuous estimate for p_s is obtained with Equation 3.6. This continuous estimate is discretized with bilinear interpolation, where the weighed average of the four neighboring points is used. The shaded area illustrate the discretization error that is introduced. Figure has been modified, but it originally stems from [38].

The discretization introduces a discretization error, which in some cases is not negligible. In Figure 3.2, the shaded area indicates that the bilinear interpolation can smooth out edges in an image as it samples from the four neighbouring pixels. We will return to this point in Section 3.2.2.

By applying Equation 3.6 for all pixels in the source image, we can synthesize the target image $\hat{I}_s$ from the source image s. We wish to synthesize the target image such that it looks similar to the true target image, thus we define our view synthesis loss as

$$\mathcal{L}_{vs} = \sum_s ||I_t - \hat{I}_s|| \tag{3.7}$$

This loss function is called the photometric reconstruction loss, and by minimizing this loss function, the relative pose between target and source view $T_{t \to s}$ and the depth map $\hat{D}_t$ will be learned as intermediate tasks.

Explainability mask

The exploited pixel-wise image correspondence from Equation 3.6 holds for static scenery without occluded objects, which is not realistic for real world applications. In order to improve the robustness towards these dynamic movements, an explainability network is included in the model pipeline. This network shares the decoder weights with the Pose CNN and outputs a pixel-wise softmax probability $\hat{E}_s$ for each target-source pair that describes the networks belief where direct pixel synthesis will be successfully modelled by Equation 3.6. This, explanatory mask is incorporated in the photometric reconstruction loss function (Equation 3.8).

$$\mathcal{L}_{vs} = \sum_s \hat{E}_s ||I_t - \hat{I}_s|| \tag{3.8}$$

As $\hat{E}_s = 0$ leads to a trivial solution, it is necessary to add a regularization term $\mathcal{L}_{reg}\left(\hat{E}_s^l\right) = -\sum_p \hat{E}_s(p)$ that encourages nonzero predictions by minimizing the cross-entropy loss with a constant label 1 at each pixel location p. A smoothness term $\mathcal{L}_{smooth} = ||\hat{D}_t||_1$ is also added to ensure smoothness in the depth map prediction. It is my belief that the 1-norm is chosen as it penalizes discontinuities less than the 2-norm, which leads to sharper depth estimations along edges. Thus, the final loss function consists of three terms. In practice, this loss function is calculated on four scales in order to capture the depth of both textureless regions and in order to get detailed estimates of texture-rich regions.

$$\mathcal{L}_{final} = \sum_l \mathcal{L}_{vs}^l + \lambda_s \mathcal{L}_{smocth}^l + \lambda_e \sum_s \mathcal{L}_{reg}\left(\hat{E}_s^l\right) \tag{3.9}$$

where l runs over the different image scales, and λ_s and λ_e are weighting for depth smoothness and explainability regularization respectively. In current open-source implementations, $\lambda_s = 0.5$ and $\lambda_c = 0$, which suggests that in practice the regularization term is not required.

Model architecture

As mentioned, the `SfM-learner` consists of two networks; depth CNN and pose CNN. Figure 3.3 gives an overview of the algorithm.

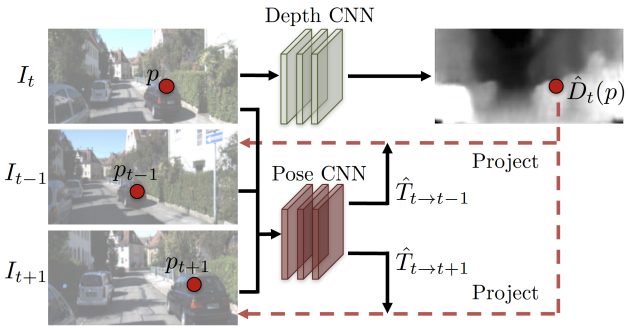


Figure 3.3: Shows an overview of `SfM-learner`[38]. In this illustration, the model operates on 3 consecutive images I_{t-1}, I_t and I_{t+1}. A pixel-wise depth estimation $\hat{D}_t$ is predicted from image I_t by the Depth CNN. The Pose CNN predicts the relative 4×4 extrinsic camera matrices $\hat{T}_{t\to t-1}$ and $\hat{T}_{t\to t+1}$ between image plane I_t and respectively image I_{t-1} and I_{t+1}. The intrinsic and estimated extrinsic camera matrices are used to project the densely estimated 3D points onto I_{t-1} and I_{t+1}. The model is updated using the photometric reconstruction loss (Equation 3.9) between the projections and the actual images I_{t-1} and I_{t+1}.

Depth CNN The depth CNN takes the target image I_t as input and outputs a dense depth prediction $\hat{D}_t$. It has an encoder-decoder architecture where the depth estimations are sampled from multiple scales from the decoder[1]. The many depth estimations are averaged to produce the final depth prediction. This multi-scale approach enables depth estimates from textureless regions. All the convolutional layers in the networks are followed by a ReLU activation function except for the last layer where a sigmoid activation function is used. This means that the predicted depth is constrained to be positive and within a reasonable and pre-defined range (usually set to 80 m).

Pose CNN The pose CNN, ψ_E, takes the target image, I_t, and a source image, I_s, as input and outputs a 6×2 vector (corresponding to the 3 Euler angles and the 3D translation) that describes the relative camera poses between consecutive images.

$$\hat{E}_{s \to t} = \psi_E(I_s, I_t) \tag{3.10}$$

The pose estimation network also has an encoder-decoder architecture. Global average pooling is applied at all scales to aggregate predictions across all spatial locations.

With the SfM-learner, Zhou et al. laid the foundation for depth estimation using self-supervised methods. They showed that depth and ego-motion networks could be supervised with the depth-image based rendering procedure. It is important to note that this procedure only works for a static scenery and if there is motion between consecutive frames.

3.2.2 Vid2Depth

Mahjourian et al. presented Vid2Depth in [40], where they suggest several refinements to Zhou et al.'s work, which yielded significant improvements in accuracy. Thus, this section will focus on these refinements.

Principle Masking

As mentioned earlier, an explanatory network learned to predict where view synthesis would be successful and unsuccessful in `SfM-learner`. It is a difficult task to train a network to find a general-purpose mask that can mask out dynamic objects, occluded objects, and edge regions, which are not visible from both views. Mahjourian et al. found that this general-purpose mask especially did not penalize the parts of the scene that were not covered in the new view, e.g. the image boundary in a forward going sequence. They found that this could result in degenerated depth estimates. Therefore, they proposed an analytically derived mask, which as its sole focus has these occluded boundary pixels. Given a pair of images I_t and I_s, one can obtain a pair of masks M_t and M_s that indicates which pixel coordinates are valid. The mask M_t can be obtained by projecting the 3D points estimate by I_s via Equation 3.6 onto image plane I_t. If the projected points are within the image boundaries of I_t, then the pixel coordinate is valid. The reverse process can be applied to obtain M_s. This analytically derived mask will ensure that the model does not perform model synthesis on the image boundaries that are occluded due to forward or backward motion, however, it will not penalize dynamic objects. Thus, in scenes with large dynamic motion, this can result in degenerated depth estimates, but this is not the case for the Railroad dataset.

Loss Function

Mahjourian et al. also suggest a new loss function. This loss function consists of four elements, which are described in the following paragraphs. Each loss is calculated over four different scales, similarly to Equation 3.9.

[1] The depth network is adapted from DispNet [22]

3D loss function Instead of solely penalizing on the projections of synthesized 2D points, Mahjourian et al. suggest to constraint directly on the 3D point clouds generated by Equation 3.1 and 3.2 for all pixels in respectively the target and source image. The 3D loss L_{3D} uses the Iterative Closest Point (ICP) method to find the transformation $T_{t\to s}$ that minimizes the distance between the two point clouds. The 3D points do not have an associated descriptor, thus point set registration methods are the most efficient way to solve for the point cloud alignment. ICP is the most used point set registration method, and it is my belief that ICP is chosen as it is intuitively easy to understand and rather efficient. The advantage of using the 3D loss is that it avoids the rather crude bilinear interpolation in Equation 3.6 and Figure 3.2.

Structured similarity Structured similarity SSIM[6] is a measure for similarity between two image patches, where 1 indicates complete similarity and 0 indicates complete dissimilarity. Thus, the loss L_{SSM} is defined as one minus the SSIM

$$L_{\text{SSM}} = \sum_{ij} \left[1 - \text{SSIM}\left(\hat{I}_s^{ij}, I_t^{ij}\right)\right] M_s^{ij} \tag{3.11}$$

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.12}$$

where x, y are two patches with mean, μ_x and μ_y, variance, σ_x^2 and σ_y^2, and covariance, σ_{xy}. c_1, c_2 are constant to stabilize low denominators. Equation 3.12 stems from 3 terms $\text{SSIM}(x,y) = l(x,y)c(x,y)s(x,y)$. The first term, $l(x,y) = \frac{(2\mu_x\mu_y+c_1)}{(\mu_x^2+\mu_y^2+c_1)}$, measures the luminance similarity between the two patches. Thus, if the average luminance of patch x is equal to the average luminance of patch y, we get $\mu_x = \mu_y \Leftrightarrow l(x,y) = \frac{2\mu_x^2+c_1}{2\mu_x^2+c_1} = 1$. The second term, $c(x,y) = \frac{(2\sigma_x\sigma_y+c_2)}{(\sigma_x^2+\sigma_y^2+c_2)}$, describes the contrast similarity. By the same argument as before, equal variance in the two patches will gives $c(x,y) = 1$. The third term, $s(x,y) = \rho_{x,y} = \frac{(\sigma_{xy}+c_3)}{(\sigma_x\sigma_y+c_3)}$, describes the structure similarity in the two patches by looking at their correlation. Using the above definition of the correlation, the denominator of the third term cancels out the numerator of the second term, leaving us with Equation 3.12. Hence, SSIM describes the similarity of two image patches based on their averages luminance, variances of luminance and structural luminance correlation. This ensures a more realistic view synthesis.

Gradient smoothness A refinement of the depth regularization term L_{smooth} is proposed. This refinement takes into account the gradients of the input image which allows sharp changes in the depth map for regions where there are also sharp changes in the input image.

$$L_{\text{sm}} = \sum_{i,j} \left\|\partial_x D^{ij}\right\| e^{-\left\|\partial_x I_t^{ij}\right\|} + \left\|\partial_y D^{ij}\right\| e^{-\left\|\partial_y I_t^{ij}\right\|} \tag{3.13}$$

The formula states that if there is a large change in the input image in the x direction then $\left\|\partial_x I_t^{ij}\right\| >> 0$, and thus $e^{-\left\|\partial_x I_t^{ij}\right\|} \approx 0$, thus a large change in the depth image $\left\|\partial_x D^{ij}\right\| >> 0$ will not be penalized much. Equivalently, for changes in the y-direction. Thus, if the sharp edges in the input and in the depth image are similar we will not penalize these, however if the sharp edges does not correspond between the two images, they will be penalized.

Total Loss By combining the three loss functions and the view synthesis loss function L_{vs} with the new explanatory mask, we get

$$L = \sum_l \alpha L_{\text{vs}}^l + \beta L_{\text{3D}}^l + \gamma L_{\text{sm}}^l + \omega L_{\text{SSM}}^l \tag{3.14}$$

where l runs over the different scales that the loss is calculated for, and $\alpha, \beta, \gamma, \omega$ are weightings for the different parts. The current implementation does not support ICP, thus the weights are set as follows $\alpha = 0.85$, $\beta = 0$, $\gamma = 0.05$, and $\omega = 0.15$. Thus, the reconstruction loss is by far the most important loss

term.

The refined mask suggested by Mahjourian et al. makes the depth estimations of vid2depth more robust for scenery where edges are not visible in the synthesized view, which often happens with e.g. forward motion. However, vid2depth do not penalize dynamic objects, and thus require the scene to be static, which is a significant drawback of this method. Most recently, Casser et al. circumvented this requirement with a further refinement of the model.

3.2.3 Struct2depth

Casser et al. [42] obtain state-of-the-art performance with an extension of Mahjourian et al.'s work where they add a motion model and impose an object size constraint. The motion model models all dynamic objects in the scene and the size constraint improves depth estimation of objects moving in front of the camera with similar speed as the camera. Once again, the theory will for simplicity be described for two images; one target image I_t and one source image I_s, however, the theory extends to multiple source images and in practice the model is fed 3 images.

Motion Model

The motion model, ψ_M, has the same encoder-decoder architecture as the ego-motion model, ψ_E (Equation 3.10), however it takes as input a target and source image (I_t, I_s) and their instance-aligned segmentation masks $(S_{i,t}, S_{i,s})$ where i runs over the number of segmented dynamic objects. The following procedure is implemented to obtain the depth estimates: First, a binary mask of the static background in each view is found by

$$O_0(S) = 1 - \cup_{i=1}^{N} S_i \tag{3.15}$$

where N is the number of segmented objects and S_i are the binary masks found with Mask R-CNN [33] in a single frame. 1 is a one matrix with the same size as the image.

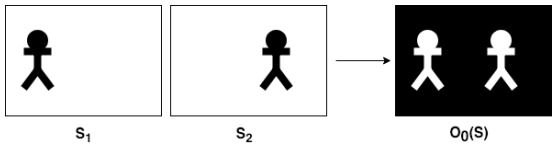


Figure 3.4: Shows an example of Equation 3.15 where one minus the union of the two masks S_1 and S_2 gives a mask for the static background $O_0(S)$. In the figure, black indicates ones and white indicates zeros.

Recall that our view synthesis only works between static scenes. Thus, we are only interested in the scene that is static throughout all images in the sequence as shown in Figure 3.5.

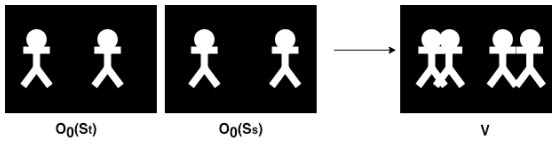


Figure 3.5: Shows an example of Equation 3.16 where the static background is obtained for two images. The union of these two masks gives the scenery that is static throughout the entire sequence. White indicate 0 (thus dynamic object) and black indicates 1 (thus static).

This can be calculated as

$$V = O_0(S_t) \odot O_0(S_s) \tag{3.16}$$

where V becomes a mask that filter all pixels that contains dynamic objects in the sequence. In this equation $\odot$ indicates element-wise multiplication. Using V, we can obtain the ego-motion only based on the static background

$$\hat{E}_{s\to t} = \psi_E(I_s \odot V, I_t \odot V) \tag{3.17}$$

Note that this is exactly the same as presented in Equation 3.6 except only the static points are used. This means that the masking of dynamic objects is applied directly on the images instead of on the predictions (Equation 3.8). Furthermore, this segmentation of dynamic objects enables us to improve predictions based on dynamic objects. As such, the motion ψ_M of the dynamic objects in the scene can then by calculated by

$$\hat{M}^{(i)}_{s\to t} = \psi_M(I_s \odot O_i(\hat{S}_{s\to t}), I_t \odot O_i(S_t)) \tag{3.18}$$

where i runs over all segmented dynamic objects[2]. Thus, as an intermediate calculation, we obtain a motion estimate for all dynamic objects in the scenery.

The final warping result is a combination of the ego-motion warping $\hat{I}$ and the warping for each individual moving object $\hat{I}^{(i)}$. This way, our image warping applies both the dynamic and the static scenery.

$$I^{(F)}_{s\to t} = \hat{I}_{s\to t} \odot V + \sum_{i=1}^{N} \hat{I}^{(i)}_{s\to t} \odot O_i(S_t) \tag{3.19}$$

where the first term is view reconstruction estimated from the static scenery and the second term reconstruction estimated from the dynamic objects. An appealing feature of the formulation is that we exploit both the information in the static and in the dynamic scenery for our synthesized view.

Object size prior

Another issue that Casser et al. discovered with the existing monocular depth estimation methods was that these models often modelled objects that were moving just in front of the camera and with the same speed as the camera at infinity depth. This is because these objects show no apparent motion - similarly to objects at infinity. As the model has no knowledge of the object scales, the movement of the object becomes ambiguous. E.g. a car moving with the same speed as the camera will appear similar in consecutive frames and minimizing the reconstruction photemetic loss will result in a prediction at infinity, as the car does not seem to change position. This is a limited problem for the Railroad dataset, however, a large problem for the KITTI dataset[11]. Therefore, Casser et al. propose to learn the objects' scales during training, which enables 3D modeling of the objects. In practice, this is done by imposing a weak prior on the height for each object class. As shown in Figure 3.6, the depth can be calculated for the segmentation mask with the focal length.

$$\hat{D}_{approx}(p, h) \approx f\frac{p}{h} \tag{3.20}$$

where f is the focal length, p is a learnable height prior in world units and h is the height of the respective segmentation in pixels.
We can then learn p by defining a loss term L_{sc} as in Equation 3.21. This loss term compares the depth approximation D_{approx} obtained for every segmented object via Equation 3.20 with the depth map for the object's respective position. The category specific height prior is updated such that Equation 3.21 is minimized.

$$L_{sc} = \sum_{i=1}^{N} \left\| \frac{D \odot O_i(S)}{\overline{D}} - \frac{D_{\text{ approx }}\left(p_{t(i)}; h\left(O_i(S)\right)\right)}{\overline{D}} \right\| \tag{3.21}$$

where $\overline{D}$ is the average depth of the middle frame, which is used as a scaling factor to avoid a degenerate solution. The sum runs over all objects in the scene, and $t(i)$ is a function that is used to identify a

[2]It is important to note that $\hat{M}^{(i)}_{s\to t}$ does not directly describe the object motion, but are in fact modeling how the camera would have moved in order to explain the object appearance. The actual movement of the objects can be obtained by tracking the object's voxel movements before and after the object movement transform in the respective region.

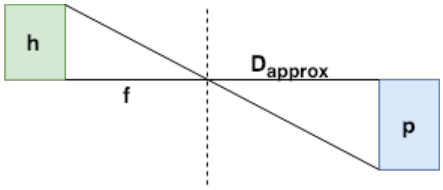


Figure 3.6: Visualizes Equation 3.20. Due to the triangular geometry, the ratio between an object's height in world unit and in pixels is the same as the ratio of the distance to the object over the focal length. Thus, $D_{approx}/f = p/h$, where f is the focal length, p the height of an object in world units, h the height in pixels and D_{approx} the distance between the camera and the object.

category id for each object i.

For this project, it has been chosen to adapt the `struct2depth` model to estimate depth. This model has been chosen because of its superior accuracy on the KITTI dataset[42], and the novel idea of using segmentation masks for treating the static background and dynamic objects separately. An open-source TensorFlow[21] implementation of this model is available at[3]. This model requires segmentation masks that can be obtain with the Mask R-CNN[33]. The Mask R-CNN model has been chosen as it is the open-source implementation that has the highest accuracy on the COCO detection challenge[16]. As the name suggest, the Mask R-CNN builds on top of the R-CNN trilogy, which was explained in Section 2.2. The next section will describe Mask R-CNN with emphasis on the changes from Faster R-CNN to Mask R-CNN.

3.2.4 Mask R-CNN

As mentioned, the Mask R-CNN[33] builds on top of the Faster R-CNN. It achieves state-of-the-art results on the COCO segmentation challenge[16]. A fundamental difference between Faster R-CNN and Mask R-CNN is that Mask R-CNN performs both bounding box detection and instance segmentation. The goal of instance segmentation is to assign an object class to each pixel value in an image. This section will focus on the changes from Faster R-CNN to Mask R-CNN.

Architectural changes The base RPN network is identically with the one used in Faster R-CNN, however, in the Fast R-CNN a segmentation prediction is implemented in parallel with the bounding box regression and classification as shown in Figure 3.7.

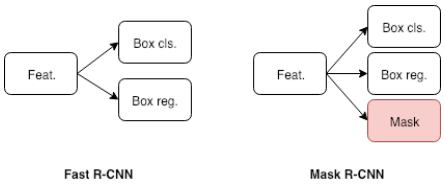


Figure 3.7: Shows the additional segmentation prediction that is calculated in parallel and thus independently of the bounding box and classification. Feat. indicates the features calculate from the RoI using RPN.

The mask prediction is performed by a fully convolutional network. The reason a convolutional network is used, as opposed to the non-convolutional networks used for the bounding box classification and regression, is that the convolutional network retains spacial orientation, which is crucial for location specific tasks such as pixel-wise segmentation. The convolutional network outputs a binary mask for each $m \times m$

[3]Implementation of `struct2depth`: `https://github.com/tensorflow/models/tree/master/research/struct2depth`

RoI for all K classes, thus the output is a $K \times m \times m$.

The parallel implementation of the segmentation decouples the segmentation prediction from the bounding box classification, which is crucial for the model's performance as it removes the competition among the class labels[33]. This means that the class prediction of a segmented region is based on the bounding box classification.

For training this model, the multi-loss function (Equation 2.1) is extended to consist of three normalized terms

$$L = L_{cls} + L_{loc} + L_{mask} \tag{3.22}$$

where L_{mask} is defined as the average binary cross-entropy loss, only including the k-th mask if the region is associated with the ground truth class k. Thus, only the estimate of the ground truth segmentation contributes to the loss. This ensures that there is no competition between the classes.

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \leq i,j \leq m} \left[y_{ij} \log \hat{y}_{ij}^k + (1 - y_{ij}) \log \left(1 - \hat{y}_{ij}^k\right) \right] \tag{3.23}$$

where K is the number of classes, m^2 is the area each of the K pixel-wise binary mask predictions. $y_{i,j}$ is the ground truth as position (i, j) and $\hat{y}_{i,j}^k$ is the predicted probability for class k. We divide by the object area m^2 such that bigger objects do not contribute more than smaller objects as this would be to favour close-by objects.

RoI Align RoI Alignment is a refinement of the RoI pooling layer used in Fast R-CNN, which have shown significant improvements in performance[33]. The problem with the RoI pooling is that data is often misaligned or lost, which becomes evident for pixel-wise segmentation. This happens because the RoI pooling discretizes the stride as illustrated in Figure 3.8. While this may not impact bounding box predictions, which are rather robust to small translations, it has a large negative effect on predicting pixel-accurate masks[33].

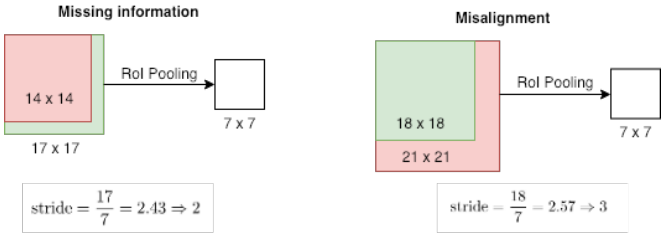


Figure 3.8: Shows the two problematic situations caused by the discretization of the stride for the RoI pooling. In the first situation, we have a 17×17 RoI that we want to reduce to a 7×7 RoI. Below the figure, the stride is calculated where $\Rightarrow$ indicate rounding to closest integer. Since the stride is rounded down, we will miss some information in the pooling process (indicated with green in the left figure). In the second scenario, our RoI is 18×18, thus the stride is rounded up to 3. This means that we select a too large area, which results in misalignment of the pixels (indicated with red in the right figure).

RoI Align, on the other hand, do not round the stride. Instead, each RoI is divided into bins; each with four sampling points as seen in Figure 3.9. Bilinear sampling (described in Figure 3.2) is then used to calculate a value for each of the sampling points. Maximum or average pooling are then performed on the four sampling points, such that each bin gets just one value.
Thus, the advantage with the RoI Align is that we do not lose information nor risk misalignment, because we do not perform any discretization of the RoI.

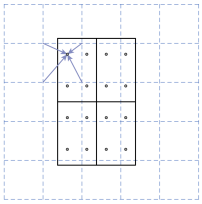


Figure 3.9: Shows an example of RoI Alignment[33]. The dashed lines represent a feature map, the solid lines the bins of a RoI (in this case there are only four bins), and the dots are the four sampling points per bin. The arrows indicate bilinear sampling for one sampling point. From the figure, it is seen that we do not lose any information even though the RoI is not aligned with the feature map.

In conclusion, Mask R-CNN enables object segmentation by an elegant extension of Faster R-CNN. In parallel, segmentation masks are made using the same feature map as the classification and bounding regression estimate, thus Mask R-CNN adds little computational overhead. The following Section will describe how segmentation masks are obtained for the Railroad dataset.

3.3 Motion Mask Estimations

The `Struct2Depth` model requires segmentation masks such that dynamic objects and the static background can be dealt with separately. These segmentation mask are calculated off-line and fed to the model during training.
A pretrained Mask-RCNN implementation[4] was adapted to obtain the motion masks for the Railroad dataset. This implementation was chosen as it is well-documented and had achieved state-of-the-art performance on the several datasets[32], including the KITTI dataset[11].

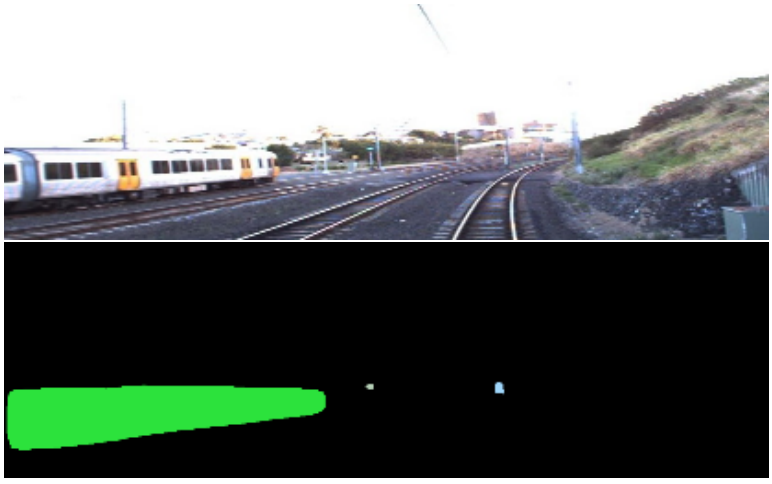


Figure 3.10: The top image shows the original image for the Railroad dataset, and the bottom image shows the corresponding segmentation mask. From this mask, it is seen that one train and two traffic signs are segmented.

The pre-trained Mask-RCNN had been trained on the COCO dataset [15] which have 91 categories[5]

[4]Pretrained Mask-RCNN model: `https://github.com/matterport/Mask_RCNN`
[5]Complete list of categories in the COCO dataset `https://tech.amikelive.com/node-718/what-object-categories-labels-are-in-coco-dataset/`

under 12 super-categories[6]. As most of these categories are obsolete for the Railroad dataset, it was decided only to include the segmentation masks from the three first super-categories (`person`, `vehicle`, `outdoor`). The pixel-wise largest and occurrence-wise most frequently dynamic object in the Railroad dataset are trains. The advantage of applying a Mask-RCNN model that had been trained on the COCO dataset is that this model already has learned to segment trains, and thus the model does not require fine-tuning.

Figure 3.10 shows an example of an obtained segmentation mask for the Railroad dataset. The major dynamic object in the sequence is the passing trains. In total, a train passes by 7 times where each train pass occupies between $50-200$ frames. From manual inspection, it was found than in all 7 instances, the segmentation mask correctly identified the train for more than 95% of the frames. Thus, the segmentation masks rather accurately identifies the dynamic objects in the railroad sequences. As the `train` category is the major dynamic object in the Railroad dataset, accurate segmentation on this category is by far the most important for the `struct2depth` model to learn accurate view synthesis.

A segmentation mask was calculated for all images in the Railroad dataset. These masks were used to segment respectively the static scenery and the dynamic objects in the `Struct2depth`-model during training and inference. In the next section, we will estimate the intrinsic parameters, which are also required for training the `Struct2depth`-model.

3.4 Estimation of the Focal Length

In order to reconstruct the depth information from the images, we need the intrinsic camera parameters. Unfortunately, these parameters are unknown for our uncalibrated railroad images sequence. However, with a number of assumptions, it is possible to estimate the most important intrinsic parameters, namely the focal length.

We will assume that the railroad tracks are parallel and runs on the planar x-z-surface (Figure 3.11). We further assume that the camera is mounted with an offset $(H, T, 0)$ relative to the Origin of the global coordinate system. We place the global coordinate system such that the z-axis is parallel to the rails and located halfway in between them. Thus, the x-axis becomes orthogonal to the rails. These assumptions are not controversial. The setup is shown in Figure 3.11.

A more controversial assumption, we are forced to make, is that the camera is mounted such that it is aligned with the tracks and thus it does not have any rotation. We furthermore ignore axis screw and radial distortion to reduce the number of parameters. This later assumption is not controversial. With these assumptions, we can write the projection of an homogeneous 3D point onto the 2D image plane as

$$\begin{bmatrix} ux \\ uy \\ u \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & H \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \pm X \\ 0 \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} c_x Z + f(H \pm X) \\ c_y Z + fT \\ Z \end{bmatrix} \tag{3.24}$$

The point pairs $Q_i = (\pm X, 0, Z)$ can be detected along the rails. For these points, we will assume that the z-value is given by $Z = Z_0 + nL$, where Z_0 is an unknown offset, L is a constant distance between the crossties and n is an unknown integer. We obtain both the inter-rail distance $2X$ and the inter-crossties distance L from a railroad lookup table. The Australian states of Queensland, where the railroad sequences are recorded, have narrow gauge railways[7], thus $2X = 1067$mm and for narrow-gauge railways

[6] The 12 super-categories in the COCO dataset: `person`, `vehicle`, `outdoor`, `animal`, `accessory`, `sports`, `kitchen`, `food`, `furniture`, `electronic`, `appliance`, `indoor`

[7] `https://en.wikipedia.org/wiki/Narrow-gauge_railway`

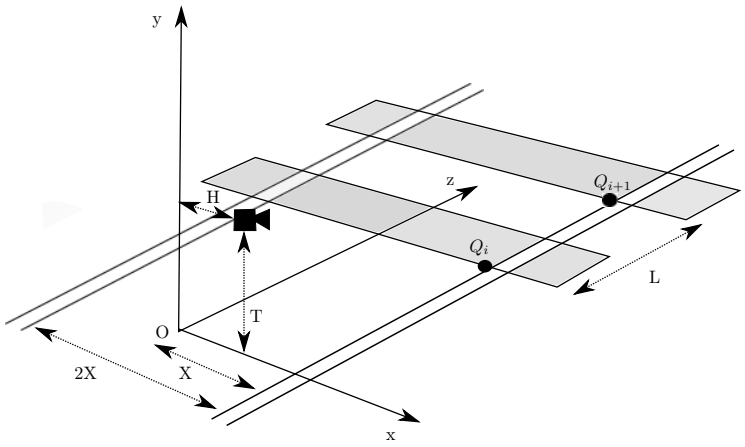


Figure 3.11: Shows a sketch of the assumed setup. The rails are assumed parallel and located on the x-z-plane. The global Origin, O, is placed halfway in between the rails and the camera's Origin is placed with an offset $(T, H, 0)$. The distance between the rails are $2X$ and is known as well as the distance between the wooden crossties L, which is also known.

the inter-crossties are most often $L = 667\text{mm}$[8], however this can vary depending on the material of the crossties and the forces to which the rails must be able to withstand.

It requires two vanishing point to estimate the principle point (c_x, c_y)[3], and as only one is visible in our setup (located at the green dot in Figure 3.12), we make an assumption that the horizontal coordinate c_x is the x-coordinate of the vanishing point and the vertical component c_y is the middle of the image as shown with the red axis on Figure 3.12. In practice, the choice of the principal points are often not crucial for obtaining a useful calibration. Figure 3.12 shows the described setup and the detected points.

In order to obtain the most accurate annotations, the points were located on a zoom as shown in Figure 3.13a. From this figure, it is seen that the annotations are not completely accurate. In future work, more accurate estimates can be obtained if the points are constrained to lie on a straight line. Even though, one point pair would be enough to obtain the focal length f and the cameras relative position (H, T), we annotate multiple points in order to improve numeric stability of the parameter estimate.

In order to estimate the parameters H and T, we find $(\tilde{x}, \tilde{y})$ by dividing with the scaling $u = Z$ and by changing the origin to the principal point (c_x, c_y), we get

$$(\tilde{x}, \tilde{y}) = \frac{f}{Z}(H \pm X, T) \tag{3.25}$$

where $\tilde{x} = (x - c_x)$ and $\tilde{y} = (y - c_y)$. By defining $r(\pm X) = \tilde{x}/\tilde{y} = (H \pm X)/T$, we get two equations with two unknowns, which are found as

$$\begin{aligned} T &= \frac{2X}{r(X) - r(-X)} \\ H &= X\frac{r(X) + r(-X)}{r(X) - r(-X)} \end{aligned} \tag{3.26}$$

[8] https://books.google.dk/books?id=Ttd1DwAAQBAJ&pg=PA478&lpg=PA478&dq=sleepers+667+mm&source=bl&ots=7ss1hxowdd&sig=8guGqbOaE7MmYT52hQVxDNbdQbM&hl=da&sa=X&ved=2ahUKEwiH1pjTx9HeAhUEL1AKHQ9ID1oQ6AEwB3oECAgQAQ#v=onepage&q=sleepers%20667%20mm&f=false

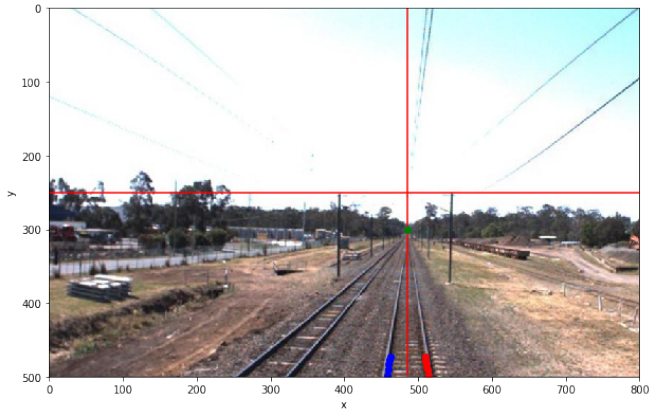
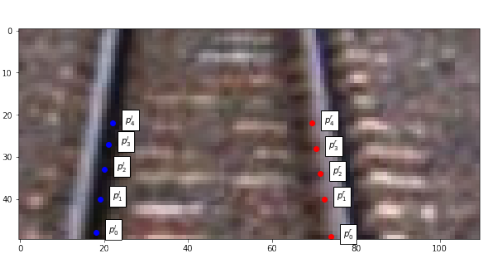
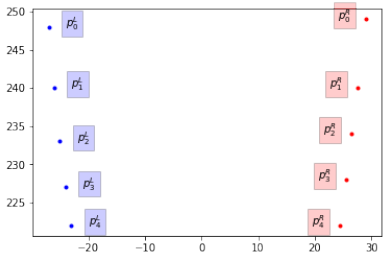


Figure 3.12: The green dot shows the selected vanishing point. The red axis shows the chosen principal points; C_x and C_y. C_x is chosen as the x coordinate of the vanishing point. C_y is chosen as half the image height. The blue and red points show the annotated points on respectively the right and left rail.



(a) The annotated points are shown as red or blue points.



(b) Points plotted to illustrate the perspective geometry. Note that this plot is rotated compared to the image.

Figure 3.13: Shows crops of Figure 3.12 used to annotating the points.

We now consider two neighboring points on the same rail, i.e. difference in z-coordinate by L. We let the y-coordinate of the two points be respectively y_i and y_{i+1}. Then

$$Z_0 + nL = f\frac{T}{y_i} \quad \text{and} \quad Z_0 + (n+1)L = f\frac{T}{y_{i+1}} \tag{3.27}$$

By subtraction, f can isolated

$$f = \frac{dZ}{T}\frac{y_i y_{i+1}}{y_{i+1} - y_i} \tag{3.28}$$

Thus, by Equation 3.28 and 3.26, we can estimate the three unknown camera parameters from Equation 3.24. As shown on Figure 3.12, we use five point pairs for this estimation. A simple average over the parameter estimates is used to increase stability. The obtained parameters are

$$\begin{aligned} f &= 1145.83 \pm 141.66 \text{ pixel} \\ H &= 15.83 \pm 1.35 \text{ mm} \\ T &= 4850.14 \pm 88.34 \text{ mm} \end{aligned} \tag{3.29}$$

Based on the geometry drawn in Figure 3.11 and the image shown in Figure 3.12, it makes sense that the H offset is small. It also makes sense that T is significantly larger than H, however the estimate T

states that the camera is located 4.85 m over the rails, which seems somewhat high. The focal length estimate does not seem unreasonably high, however, it is associated with a significant error (also seen in Figure 3.14).

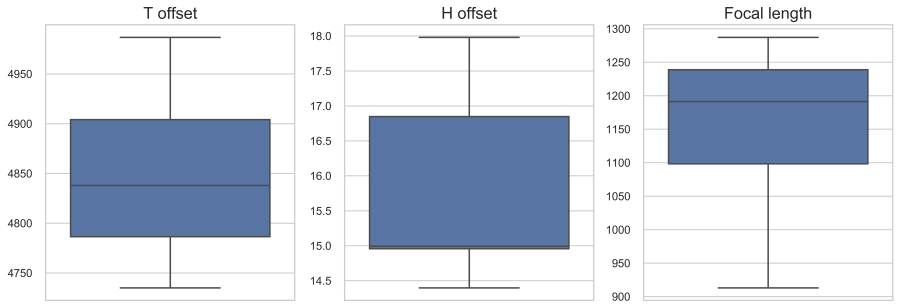


Figure 3.14: Shows the distribution of the estimated parameters. From the figure, it seen that the parameter estimates are rather uncertain. Improved sampling or parameter optimization would possible reduce the parameter uncertainty.

From Figure 3.14, it is seen that the parameter estimate are rather uncertain. The uncertainty of the parameter estimates can either be caused by erroneous assumptions, incorrect world-geometry or poor annotations. In the case of poor annotations, one could try to obtain better parameter estimates by annotating more points from a larger array of images (under the non controversial assumption that the camera parameters are constant throughout the image sequence), constrain the annotated points to lie on a straight line, and use a more robust optimization algorithm than the simple average, e.g. formulate the optimization of the parameters as a maximum likelihood problem.

3.5 Experiments

As described in Section 3.3 and Section 3.4, we have obtained both an estimate of the intrinsic camera parameters and a segmentation mask for each image in the Railroad dataset. These are used to train the `struct2depth` model in the two experiments.

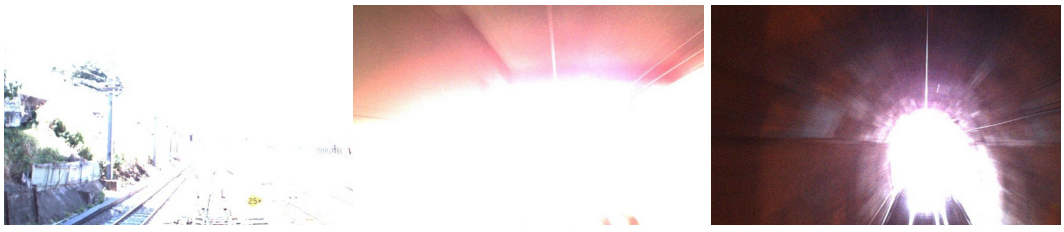


Figure 3.15: Three difficult scenarios for the Railroad dataset, that makes learning of both depth and ego-motion difficult, as they yield little information about the 3D scenery.

In the first experiment, we trained on the same training set as used in Section 2.5 consisting of more than 40000 images. As one might expect, the training did not converge, since this train sequence includes train stops (thus no motion between frames), tunnels (sequences where most pixels are black) and sequences where the camera is blinded by the sun. Figure 3.15 shows difficult examples from the Railroad dataset

that make learning of accurate view synthesis impractical.

In order to circumvent these difficulties, we used a small subset of the Railroad data in the second experiment. This subset was selected such that the train drove with approximately constant speed, only passed a few tunnels and the camera was not blinded for a longer time. The selected subset consisted of 363 consecutive frames. This is a very small dataset, and thus, we expect the model to overfit. In future work, it would be of interest to train on multiple subsets, which are selected with the same criteria as the one used on this experiment. The hyper parameters shown in Table 3.1 were used for training. Note that the weight for the 3D Loss is set to 0 as it had not yet been implemented and it was outside the scope of this project to implement it.

Hyper Parameter	Value
Sequence length	3
Reconstruction Loss	0.85
SSIM Loss	0.05
3D Loss	0
Smooth Loss	0.15
Batch size	4
Learning rate	0.0002

Table 3.1: The chosen hyper parameters for training. Similar to default parameters in[9]

The model was trained for 48 hours using the Adam optimizer and random image flipping. This data augmentation is introduced to compensate for overfitting. Figure 3.16 shows three loss function (3D loss was set to 0 as it was not available in the open-source implementation) and the total loss for the `struct2depth` model.

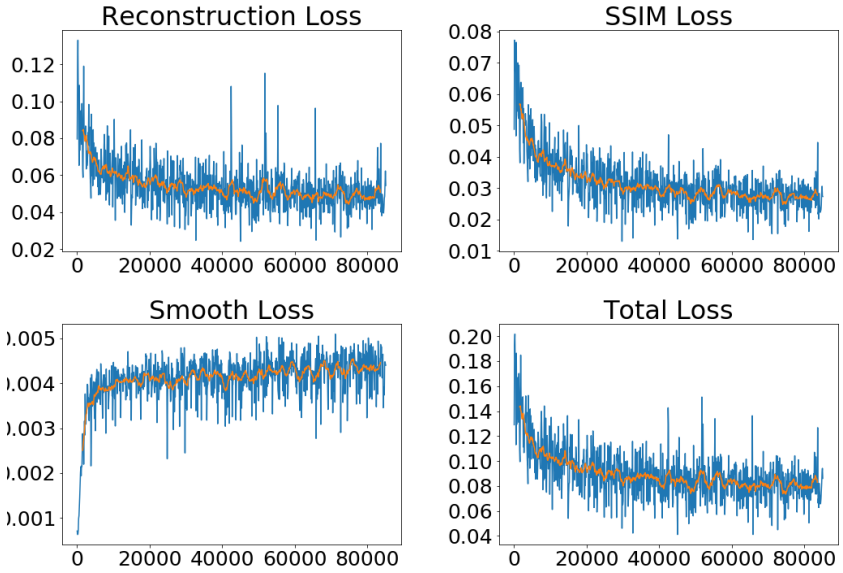


Figure 3.16: The reconstruction, SSIM and smooth loss weighted respectively 0.85, 0.15 and 0.05. The total loss is shown in bottom right corner.

From Figure 3.16, it is seen that both the reconstruction loss and SSIM loss decrease through training,

however the smooth loss increase. Recall that the smooth loss penalizes where the gradients in the input image and depth prediction does not coincides. The untrained depth map predictions is initialized as a constant and thus do not have any gradients, thus by Equation 3.13 the smooth loss will start out at zero. As more gradients are introduced, this loss term will begin to penalize.

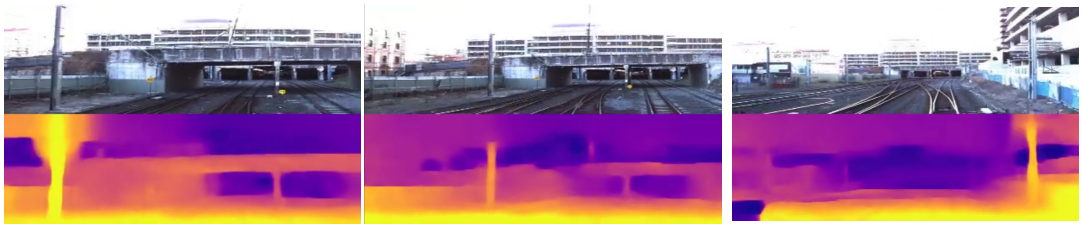
Finally, based on Figure 3.16, it seems that the model have converged on the subset, and thus we proceed to the evaluation of the model, which is presented in the next section.

3.6 Results

In this section, we seek to evaluate the performance of the trained model. This is not a trivial task, as there is not any ground truth depth nor ego-motion information available. Furthermore, the depth estimations are only predicted "up to scale". Thus, the evaluation is mainly based on visual inspection. A crude estimation of the global scale is used to demonstrate how a more sophisticated evaluation can be performed using the detected Railroad furniture's sizes. Video that shows the inference on the Railroad datasets is available at `https://www.dropbox.com/sh/31haxfnn7c807f7/AAAui9ItaPFOTn7t77bFmVcUa?dl=0`.

Figure 3.17 shows three images with their inferred depth estimations, where yellow is closer and blue is more distant. Note that the depth prediction is only known "up to scale", so we simply study the relative depths in the images. From the depth estimation in Figure 3.17a, the structure from the bridge is visible as well as the light pole in the foreground. However, the depth estimation of the top part of the light pole is rather blurred. This might be due to the fact that the horizontal gradients in the input image are smaller for this part of the light pole compared to the bottom part. As the network seeks to preserve respectively sharp and blurred edges from the original image, the small gradients in the original image might lead to blurred depth estimate.

The train is moving further away from the bridge in Figure 3.17b and 3.17c. This can be seen from the depth estimations that correctly identifies that the distance to the bridge increases through the sequence.



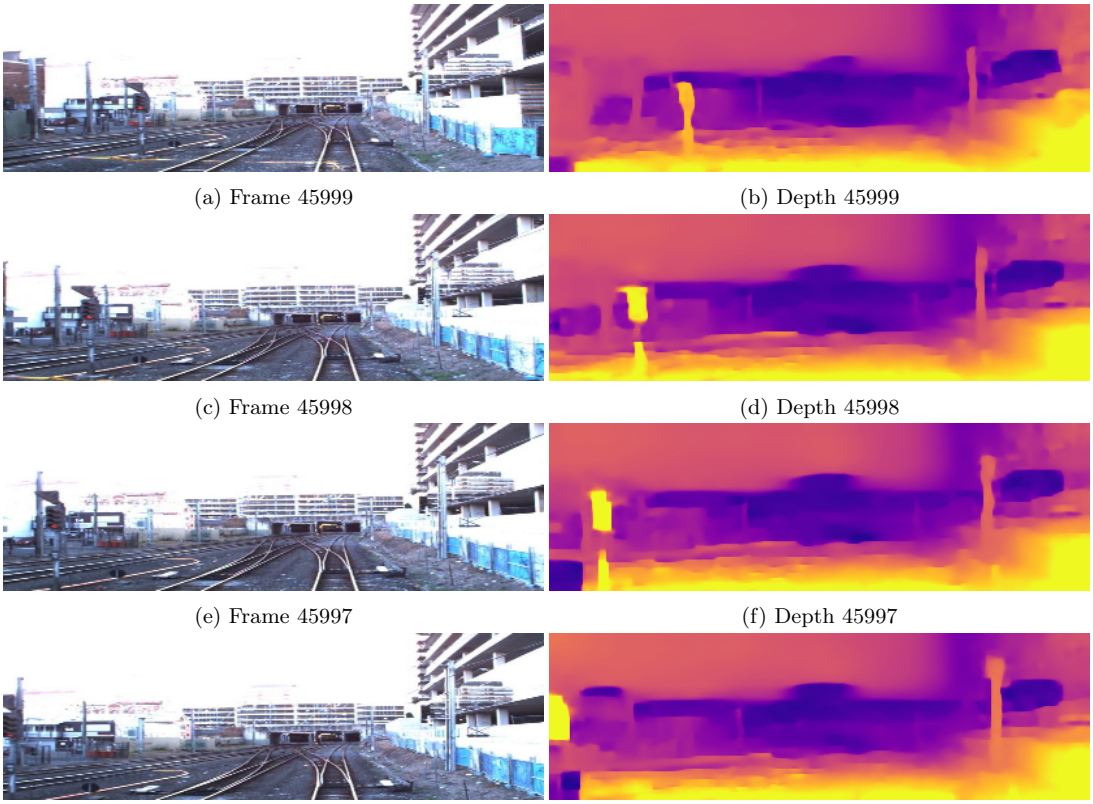
(a) Frame 45924 (b) Frame 45937 (c) Frame 45993

Figure 3.17: For each image, the top half shows the original image and the bottom half shows the depth prediction, where yellow is close and blue is distant. Note that the estimates are "up-to-scale".

In the lower left corner of Figure 3.17c is a dark square, indicating that the network predicts that this region is very far away. This is obviously wrong, as the ground in this region is close to the camera. This wrong prediction is properly caused by a very small change in the appearance of this region between the three consecutive frames. Small appearance variation is typical observed at low parallax, thus the network believes that this region is very distant.

From all three images, it is seen that the network has difficulties predicting the distance to the sky. The sky should be the most distant region, and thus be very dark blue, however it is estimated closer than the building in the background in both Figure 3.17b and Figure 3.17c. This is surprising since, the white sky does not exhibit any change in appearance, and thus would be expected to be observed at small parallax.

Next, we investigate the consistency between the depth estimations in Figure 3.18. As each depth estimation is inferred from a single frame, and since the predictions are only known "up to scale", it could lead to inconsistency between the estimation of the global scale from one image to another. However, from Figure 3.18, we see quite nice global consistency through the four images, e.g. tracking the sign in the left side of the image and the pole in the right side of the image. These two objects come closer through the sequence. The rest of the scenery also seem to be consistent through the sequence. In this sequence, it is again noticeable that the network have difficulties of estimating the depth to the sky. On the other hand, the network does surprisingly well at estimating the depth of the planar surface where the tracks lies. As expected, we see depth of this planar surface to constantly increase the further away the ground is to the camera.



(a) Frame 45999 (b) Depth 45999

(c) Frame 45998 (d) Depth 45998

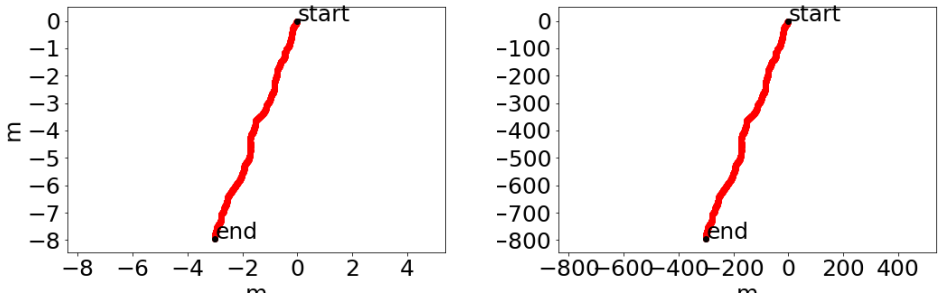
(e) Frame 45997 (f) Depth 45997

(g) Frame 45996 (h) Depth 45996

Figure 3.18: The left column shows four consecutive frames. The right column shows their respectively depth estimations. Yellow is close and blue is more distant.

The ego-motion model predicts the movement of the camera up to scale. In Figure 3.19a, the predicted path is drawn. The model correctly identifies that the train drove backwards. However, based on the video, it is expected that the path would be more curved. Unfortunately, GPS coordinates are not available. If GPS were available, the cumulative distance between the GPS and the predicted path would give an estimate for the error. Moreover, by inspecting the inferred video, it becomes clear that the global scale is too small, as the train definitely travels more than 9 meters during the sequence.

Even though the network does not know the global scale, these results might still be very useful for



(a) Not scaled

(b) Scaled with a factor 100.

Figure 3.19: The left figure shows the predicted ego-motion of the model. The right plot shows a scaled version of this prediction that coincides better with the global scale.

COWI. E.g. the global scale can easily be obtained via GPS coordinates or simple measurement on a map (this could be done off-line). The simplest method, would be to measure the distance between two distant landmarks e.g. two stations, and manually find the two frames that are associated with the selected landmarks. The scale factor would be the measured distance divided by the distance estimated by ego-motion models between these two frames.

As a proof-of-concept, we assume that the GPS coordinates of the start and finish point are known. Based on visual inspection of the video, it is assumed that the train travels approximately 900 m, and thus we scale the prediction with 100, such that we fit to the global scale (Figure 3.19b). This is a crude and controversial assumption, and it remains for future work to circumvent this assumption. However, it allows us to demonstrate how the global scale enables interpretation of the depth map as an absolute distance map in meters (Figure 3.20).

From Figure 3.20, we see that the model predicts the bridge to be approximately 40 meters away from the camera. Recall that we estimated the focal length to be 1145 pixel, which suggest quite a lot of zoom, thus the assumption about the global scale might be acceptable.

We can know use the pinhole model to estimate the size of the detected railroad furniture. The bounding boxes of the two railroad furniture elements detected by Faster R-CNN (Section 2.6) are shown in Figure 3.20. From the pinhole model (geometry shown in Figure 3.6 with slightly different symbols), the relation between pixel size and real world size can be found.

$$R = \frac{rd}{f} \tag{3.30}$$

where r is the side length of the bounding box measure in the image plane in pixel, R is the side length in world units, d is the depth to the object in world units, and f is the focal length in pixel. By this equation, we estimate the size of the two bounding boxes detected in Figure 3.20 to be respectively (31 cm, 29 cm) and (35 cm, 28 cm). These are approximately the size that we expect a speed sign to have. Thus, the selected global scale seem to be appropriate.

If the true size of the railroad furniture were known, and the global scale could be estimate via GPS coordinates or a map, the size of the railroad furniture could be calculate for the entire sequence, and their sizes compared to the ground truth sizes. This would be a more sophisticated evaluation method than visual inspection.

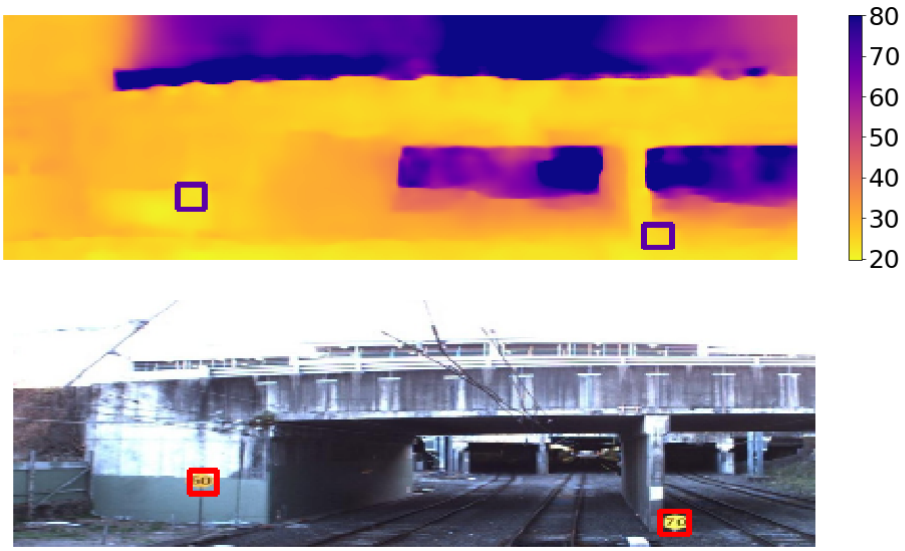


Figure 3.20: The predicted bounding boxes found with Faster R-CNN (Section 2.6) are shown in the top image. The bottom images shows the depth map that have been scale to fit the global scale. The color bar indicate the depth in meters.

3.7 Part Conclusion

In this chapter we adapted the `struct2depth` model to infer depth predictions for each frame in Railroad dataset. The model was trained on a very small subset that was selected manually, such that it did not include image pairs without motion, from tunnels nor images where the camera was blinded by the sun. It was found necessary for the convergence of the model to select this subset carefully. In future work, it would be interesting to extend the training data to multiple subset that are selected with these criteria for more training data.
Through visual inspection it was found that the depth estimations captured the close-by structures reasonably well. However, the model had difficulties with distant objects such as the sky. We found acceptable consistency between consecutive depth maps.

As a proof-of-concept, the global scale was crudely judged from the ego-motion estimates to be a factor 100 more than the predictions. In future works, this global scale factor can be estimated with GPS coordinates or by off-line measuring the distance between two recognizable landmarks, e.g. stations on a map. GPS coordinates are usually imprecise up to 10 m, however, as only the start and end points are necessary to obtain the global scale, and these probably are more than 1 km from each other, the measurement of the GPS will be negligible. This argument also applies for the landmarks selected on a map.

Using the global scale, a more sophisticated method for evaluation of the depth estimates was presented. This method enabled size estimation of the detected Railroad furniture using the pinhole model. Two speed signs were found to measure respectively (31 cm, 29 cm) and (35 cm, 28 cm), under the assumption of a global scale factor of 100.

The intrinsic parameters, which the `struct2depth` model used, were estimated using classical computer geometry. By using the parallel nature of the track, the vanishing point, the width of the tracks and distance between the crossties, it was possible to give a reasonable estimate of the intrinsic parameters.

The most controversial assumption was that the camera did not have any rotation with respect to the rails. It was found that the estimated parameters, and especially the focal length, were associated with rather high uncertainty, and a more thorough examination is required. In this regard, it is suggested to sample points from more images and use a more robust optimization algorithm.

Lastly, the Mask R-CNN model was used to infer masks on the Railroad dataset. This model had been trained on the COCO dataset, and was thus able to segment trains, which were the largest dynamic object in the Railroad dataset. From manual inspection, it was found than in all 5 instances, the segmentation mask correctly identified the train for more than 95% of the frames. A disadvantage with the `struct2depth` model is that these masks have to be calculated off-line, which adds a significant overhead in complexity and computational speed to the depth estimation pipeline compared to earlier methods. On the other hand, accurate segmentation allows the model to use both the static and dynamic scenery for the view synthesis, improving the precision of the depth estimates. In future work, the Mask R-CNN could completely replace the Faster R-CNN as it also can predict bounding boxes. This way, the 3D estimation pipeline, would be simplified and all output of the Mask R-CNN would be exploited.

Overall, the obtained results are interesting for COWI as a prototype for the possibilities of their data. As for now, the depth estimates are very sensitive towards tunnels and insufficient motion, and thus the model cannot be trained on the entire Railroad dataset. This being said, the model can most likely be trained on multiple well-behaved subsets to increase the robustness of the solution, and then inference can be performed on the sequences without motion. This is possible as the trained depth model can estimate the depth given a single frame and thus does not require motion. Thus, a depth estimate for all outdoor (and not blinded) images can be found.

Chapter 4

Discussion & Conclusion

4.1 Discussion & Future Work

The objective of this report has been to investigate the feasibility of automate the processes of generating an up-to-date map of the railroad infrastructure as this can reduce manual labor, reduce maintenance costs and potentially improve safety.

To approach this goal, we established a three-stage pipeline, where the railroad furniture was first selected in 2D, then a dense depth where estimated, and finally, 3D information was obtained based on the results from the two first stages. There exists networks, which are trained to perform 3D bounding box detection[25], [26], [35] end-to-end, thus basically unifying the suggested three-stage pipeline. The results from these methods are advantageous for estimation of objects that have a significant depth, such as cars, however, for thin structures, such as signs, the output would be very similar. On the other hand, the suggested pipeline is advantageous because both the first stage, 2D object detection, and the third-stage, depth-based projection, are mature research domains. Furthermore, the suggested pipeline does not require annotated 3D bounding, which is the case for the end-to-end 3D bounding box methods. Therefore, for future work, the most encouraging path to continue would probably be to refine the three stages of the pipeline.

For the first-stage, the Faster R-CNN was adapted for object detection of railroad furniture. Via modification of the loss function and the anchor sizes, this model was able to find 85% of the annotated bounding boxes. This might not seem sufficient for real world applications. However, as the railroad furniture appears across multiple frames, COWI should evaluate the performance according to how many of the unique railroad furniture that have been detected. It is expected that this score will be significantly higher, as all the railroad furniture will be close - and thus easier to detect - in at least in one frame, but probably more, while it is highly unlikely, that they will be missed in every frame. This however requires manual or semi-manual ID-assignment of the bounding boxes, which have been outside the scope of this project.

The adapted Faster R-CNN model discovered many false positives. These false positive falls into two categories as some are simply caused by erroneous annotations and in reality should have been true positives, and others are in fact false negatives. To handle the many false positives, it is recommended to post-process the bounding box predictions. E.g. exploit the spatial and temporal information about the bounding boxes to filter wrong predictions by performing a majority voting across a 3-7 frame sequence of images.

For future work, the models should be trained on the entire dataset; including the full panoramic images and all the 25 classes. This will yield more training data, which we expect will improve the performance of the model. The larger dataset will also yield challenges; dealing with very large distorted panoramic

images and screwed class distributions. The large images will probably not fit into memory, and thus the images should be cropped intelligently. Furthermore, it should be considered how to deal with the distorted regions. In regard to the screwed class distribution, it is expected that the Focal Loss will be advantageous for handling the sparsely represented classes. On this subject, it would be compelling to compare the performance with the one-stage method, RetinaNet[34], which also uses the Focal Loss.

In the second stage, the `struct2depth` was adapted to estimate a dense depth map for each frame. The model was chosen as it had shown superior performance on the KITTI dataset. Because of convergence difficulties, the model was trained on a very small subset of the Railroad dataset, which was selected such that it did not include consecutive triplets without motion or from within tunnels. The foreground of the estimated depth maps did for the most part resemble the expected depth, however, the depth estimation of the distant objects such as the sky, were clearly wrong. Based on this visual inspection of the estimated depth maps, more work is required to obtain depth estimates that are accurate enough for a real life application. It is assumed that in order for the model to generalize better, it has to be trained on more data. This can be done by manually selecting more subsets with motion and without tunnels. An alternative is to train on a similar dataset, such as the KITTI dataset. However, as the intrinsic camera parameters in the two dataset are not the same, it is uncertain how accurate the inferred depth estimate would be. Therefore, it will properly require fine-tuning on stable sequences from the Railroad sequence, in order for the model to learn the view synthesis with another set of intrinsic camera parameters.

The `struct2depth` method requires both segmentation masks and an estimate of the intrinsic camera parameters. The biggest and most often occurring dynamic objects in the Railroad dataset were trains. If not segmented appropriate, they could potential break the view synthesis. Through manually inspection, it was found that the Mask R-CNN trained on the COCO dataset, performed really well at segmenting the trains. This is very positive for COWI as this result means that it is not necessary to annotate segmentation masks for the entire Railroad dataset, which would have been a very labour intensive and expensive process.
The intrinsic camera parameters were estimated using several assumptions about the geometry of the scene. The most controversial being that the camera did not have any rotation with respect to the rails. The inference were estimated from just 5 point pairs and were associated with a rather large uncertainty. It is assumed that more accurate estimates can be found by using additional point pairs across multiple images. Furthermore, in order to improve the annotation accuracy, the annotated points could be constraint to lie on a line. On the other hand, in a real world application, it is not controversial to assume that the intrinsic camera parameters are known. More accurate intrinsic camera parameters would most likely improve the depth estimates, and it remains an important task for future work to either ask COWI for intrinsic parameters or improve the estimate of these.

The third stage, the positioning of the detected railroad furniture based on the 2D bounding box and the dense depth estimate, requires an estimate of the global scale. It is not controversial to assume that the global scale can be estimated with sufficient precision for real life applications relatively cost free, e.g. it could be found by measuring the distance between two landmarks, such as stations, on a map, and then match with the corresponding frames in the railroad sequence. The estimated ego-motion could then be scaled to coincide with the measure map, which would yield the scale factor to obtain the global scale.

In this project, a proof-of-concept for a refined method to evaluate the entire pipeline have been proposed. This methods use the bounding boxes predictions and the depth map estimate to calculate the 3D size of the signs and signals, which can be compared with the true size. For future work this novel evaluation method could be extended to multiple classes of railroad furniture, each associated with their own ground-truth size.

4.2 Conclusion

In this project, we have investigated the feasibility of automating the processes of generating an up-to-date map of the railroad infrastructure through 2D object detection and dense depth estimations.

The object detector, Faster R-CNN, was adapted and trained to detect signs and signals in the Railroad dataset. The model was modified for enhanced performance, especially with a focus on reducing the amount of false negatives, as it was discovered that many of the false positives were caused by erroneous annotations. Through several experiments, it was found that including the focal loss and exploiting prior knowledge about the size of the bounding boxes improved the model's performance. It is judged that COWI can use this technology without much improvements to find the signs and signals in the Railroad dataset.

We further adapted and trained the Struct2depth model to estimate a dense depth map for each frame in a subset of the Railroad dataset. In a proof-of-concept, we demonstrated that if the global scale is available, which is not a controversial assumptions in a real world setting, the depth map can be used to obtain 3D information about the detected railroad furniture. Through visual inspection of the depth map estimations, it was found the in general the foreground was well estimated, however, more distant object e.g. the sky were associated with high error. As the depth map for large parts of the scenery was inaccurate, it is considered that this technology requires improvements with regard to robustness in order for COWI to use the inferred depth maps for 3D positioning of the railroad furniture.

Bibliography

[1] C. Cortes and V. Vapnik, "Support-vector networks", Machine Learning, vol. 20, no. 3, pp. 273–297, Sep. 1995, ISSN: 1573-0565. DOI: 10.1007/BF00994018. [Online]. Available: https://doi.org/10.1007/BF00994018.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", 2001, pp. 511–518.

[3] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed. New York, NY, USA: Cambridge University Press, 2003, ISBN: 0521540518.

[4] C. Fehn, "Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv", Proc.SPIE, vol. 5291, pp. 5291 - 5291 - 12, 2004. DOI: 10.1117/12.524762. [Online]. Available: https://doi.org/10.1117/12.524762.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", Int. J. Comput. Vision, vol. 60, no. 2, pp. 91–110, Nov. 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. [Online]. Available: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[6] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity", IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, Apr. 2004, ISSN: 1057-7149. DOI: 10.1109/TIP.2003.819861.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in In CVPR, 2005, pp. 886–893.

[8] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth parametrization for monocular slam", IEEE Transactions on Robotics, vol. 24, no. 5, pp. 932–945, Oct. 2008, ISSN: 1552-3098. DOI: 10.1109/TRO.2008.2003276.

[9] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards internet-scale multi-view stereo", in The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, C 2010, pp. 1434–1441. DOI: 10.1109/CVPR.2010.5539802. [Online]. Available: https://doi.org/10.1109/CVPR.2010.5539802.

[10] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time", in Proceedings of the 2011 International Conference on Computer Vision, ser. ICCV '11, Washington, DC, USA: IEEE Computer Society, 2011, pp. 2320–2327, ISBN: 978-1-4577-1101-5. DOI: 10.1109/ICCV.2011.6126513. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126513.

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset", International Journal of Rol 2013.

[12] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", CoRR, vol. abs/1311.2524, 2013. arXiv: 1311.2524. [Online]. Available: http://arxiv.org/abs/1311.2524.

[13] NRK. (2013). Nordlandsbanen: Minute by minute, season by season, [Online]. Available: https://nrkbeta.no/2013/01/15/nordlandsbanen-minute-by-minute-season-by-season/.

[14] T. Wang and C. Chen, "Improved simultaneous localization and mapping by stereo camera and surf", in 2013 CACS International Automatic Control Conference (CACS), Dec. 2013, pp. 204–209. DOI: 10.1109/CACS.2013.6734133.

[15] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context", CoRR, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

[16] ——, "Microsoft COCO: common objects in context", CoRR, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

[17] R. B. Girshick, "Fast R-CNN", CoRR, vol. abs/1504.08083, 2015. arXiv: 1504.08083. [Online]. Available: http://arxiv.org/abs/1504.08083.

[18] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "Matchnet: Unifying feature and metric learning for patch-based matching.", in CVPR, IEEE Computer Society, 2015, pp. 3279–3286, ISBN: 978-1-4673-6964-0. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#HanLJSB15.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", CoRR, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector", CoRR, vol. abs/1512.02325, 2015. arXiv: 1512.02325. [Online]. Available: http://arxiv.org/abs/1512.02325.

[21] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[22] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation", CoRR, vol. abs/1512.02134, 2015. arXiv: 1512.02134. [Online]. Available: http://arxiv.org/abs/1512.02134.

[23] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", CoRR, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[24] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks", CoRR, vol. abs/1506.01497, 2015. arXiv: 1506.01497. [Online]. Available: http://arxiv.org/abs/1506.01497.

[25] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving", in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016.

[26] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving", CoRR, vol. abs/1611.07759, 2016. arXiv: 1611.07759. [Online]. Available: http://arxiv.org/abs/1611.07759.

[27] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks", CoRR, vol. abs/1605.06409, 2016. arXiv: 1605.06409. [Online]. Available: http://arxiv.org/abs/1605.06409.

[28] R. Garg, V. K. B. G, and I. D. Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue", CoRR, vol. abs/1603.04992, 2016. arXiv: 1603.04992. [Online]. Available: http://arxiv.org/abs/1603.04992.

[29] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency", CoRR, vol. abs/1609.03677, 2016. arXiv: 1609.03677. [Online]. Available: http://arxiv.org/abs/1609.03677.

[30] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras", CoRR, vol. abs/1610.06475, 2016. arXiv: 1610.06475. [Online]. Available: http://arxiv.org/abs/1610.06475.

[31] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger", CoRR, vol. abs/1612.08242, 2016. arXiv: 1612.08242. [Online]. Available: http://arxiv.org/abs/1612.08242.

[32] W. Abdulla, Mask r-cnn for object detection and instance segmentation on keras and tensorflow, https://github.com/matterport/Mask_RCNN, 2017.

[33] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN", CoRR, vol. abs/1703.06870, 2017. arXiv: 1703.06870. [Online]. Available: http://arxiv.org/abs/1703.06870.

[34] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection", CoRR, vol. abs/1708.02002, 2017. arXiv: 1708.02002. [Online]. Available: http://arxiv.org/abs/1708.02002.

[35] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, "3d bounding box estimation using deep learning and geometry", in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul. 2017, pp. 5632–5640. DOI: 10.1109/CVPR.2017.597.

[36] O. Özyesil, V. Voroninski, R. Basri, and A. Singer, "A survey on structure from motion", CoRR, vol. abs/1701.08493, 2017. arXiv: 1701.08493. [Online]. Available: http://arxiv.org/abs/1701.08493.

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch", 2017.

[38] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video", CoRR, vol. abs/1704.07813, 2017. arXiv: 1704.07813. [Online]. Available: http://arxiv.org/abs/1704.07813.

[39] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, Detectron, https://github.com/facebookresearch/detectron, 2018.

[40] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints", CoRR, vol. abs/1802.05522, 2018. arXiv: 1802.05522. [Online]. Available: http://arxiv.org/abs/1802.05522.

[41] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement", arXiv, 2018.

[42] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Unsupervised learning of depth and ego-motion: A structured approach", in AAAI-19, 2019.

[43] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, The PASCAL Visual Object Classes Cha http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[44] ——, The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[45] D. G. Lowe, "Distinctive image features from scale-invariant keypoints",

[46] C. Y. Yicheng An Jiafu Wu, "Cnns for face detection and recognition", [Online]. Available: http://cs231n.stanford.edu/reports/2017/pdfs/222.pdf.

Appendix A

Appendix

A.1 Time dependency

The Railroad dataset consists of low sample video sequence, however, despite of the low sample-rate, we expect the frames to be highly correlated over time. In Figure A.1 the number of objects within each cropped frame is presented for the entire time series.

Figure A.1: The number of objects appearing in each cropped frame. The blue line shows the total amount objects and the red and green lines shows respectively the number of signals and signs per frame.

From Figure A.1 we see that frames with objects cluster together. A stair case-alike step structure is visible describing how objects appear and disappear - one by one - from one frame to the next. This structure is more clearly identified in Figure A.2.

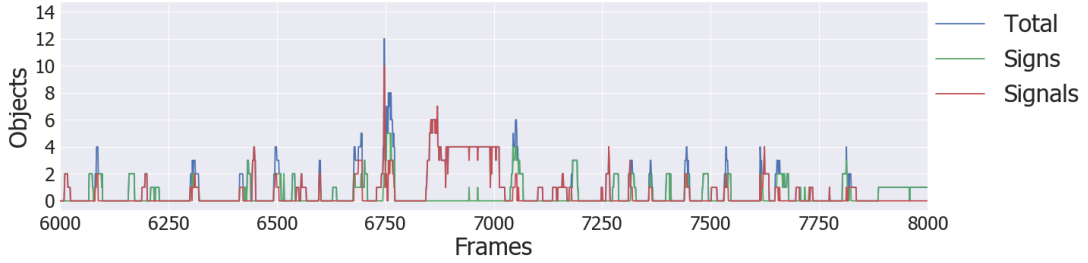


Figure A.2: Shows a zoom of Figure A.1. From this Figure the stair step structure of the number of objects becomes clear.

In order to confirm the time dependency of the number of objects per frame, the auto correlation function of the number of annotations per frame is shown in Figure A.3. The auto correlation function describes the correlation in time.

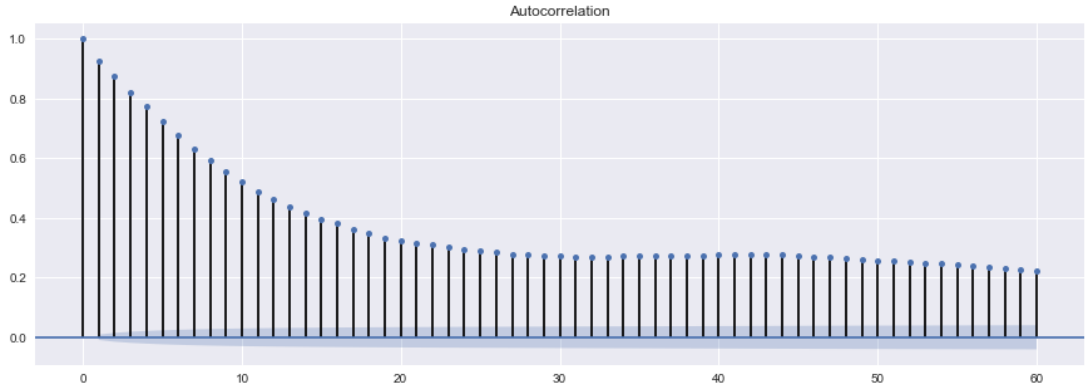


Figure A.3: The auto correlation function of the number of objects per cropped frame. The plot indicates a clear time dependency.

Figure A.3 tells us that the number of objects in a given frame is approximate 0.9 correlated with the previous frame. Thus, if we observe 2 objects in a frame, we would expect with 90% probability also to observe 2 objects in the next frame. Based on this observation, we should be careful when splitting data into training and test set as these must be independent.

A.2 Evaluation of 2D object detectors

Model evaluation in object detection is not a trivial task as it consists of two sub-problems - localization and classification - each with there own errors. Therefore, the notions on true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) can be confusion. The purpose of this section is to make this distinction more clear.

The intersection over Union (IoU) is used to determine how good the localization of an object is. It measures how much the predicted bounding box overlaps with the true bounding box. It is calculated as the ratio between the intersection and the union. The problem about this definition is that it does not take into account the size of the bounding boxes, for which we intuitively would allow much less error for small boxes than larger boxes. Usually in detection tasks, a threshold of 0.5 is used to distinguish between background and objects.

Localization :

- **TP:** The bounding box prediction and the ground truth bounding box have an IoU above 0.5.
- **FP:** The bounding box prediction has an IoU below 0.5 with all the ground truth bounding boxes.
- **FN:** A ground truth bounding box has an IoU below 0.5 with all bounding box predictions.

Classification (the two class problem of signals and signs (Section2.4) :

- **TP:** Signal prediction is actually signal.
- **TN:** Sign prediction is actually sign.
- **FP:** Signal prediction is actually sign.
- **FN:** Sign prediction is actually signal.

To comprise the evaluation of the two sub-task into one score, the Precision Recall (PR) curve is often used to evaluate object detectors. It is important to note that the precision and recall uses slightly different definitions of TP, FP and FN that encapsulates both information from the task of classification and detection.

Object detection :

- **TP:** The bounding box prediction and the ground truth bounding box have an IoU above 0.5 and the classification of the object is correct.
- **FP:** The bounding box prediction and the ground truth bounding box have an IoU above 0.5, but the classification of the object is incorrect.
- **FN:** A ground truth bounding box has an IoU below 0.5 with all bounding box predictions of the ground truth class.

With these definition settled, the recall and precision can be defined as

$$\text{Recall} = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{A.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{A.2}$$

where TP, FP and FN follows the object detection definition. The interpretation of the precision is how well the model classifies the found objects and the interpretation of the recall is how well the model localizes the objects. The PR-curve is calculated by sorting each predicted bounding box according to their confidence scores. We then calculate the recall and precision for each prediction. The results from these calculations are plotted against each other.

The Average Precision (AP) Intuitively, the AP can be understood as the area under the PR-curve. However, it is an approximation where we for each recall value take the maximum value of all previously observed precision, which gives a more smooth function. Furthermore, we divided the PR-curve into 11 equally sized bins and average over these. This can more formally be written as

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r) \tag{A.3}$$

$$p_{\text{interp}}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \tag{A.4}$$

where p is the function that maps a recall to a precision and $\tilde{r}$ is all recall values greater than the current recall value.

The mean Average Precision (mAP) The mAP is simply the mean of the AP for all classes.

$$\text{mAP} = \frac{1}{n} \sum_{i}^{n} \text{AP}_\text{i} \tag{A.5}$$

where n is the number of classes.