MASTER THESIS

Thesis submitted in fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Mechatronics/Robotics

Rail Track Prediction
Arbeitstitel

By: Sebastian Goebel, BSc.

Student Number: University of Applied Science: 51912403
Luzern University of Applied Sciences and Arts: 23-565-161

Supervisor: Prof. Dr. Björn Jensen

Vienna, November 8, 2024

Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz /Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Vienna, November 8, 2024 Signature

Kurzfassung

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Schlagworte: Schlagwort1, Schlagwort2, Schlagwort3, Schlagwort4

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Keywords: Keyword1, Keyword2, Keyword3, Keyword4

Acknowledgements

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Erste Überschrift der Tiefe 1 (chapter)

Etwas Text... Hier kommen noch einige Abkürzunge vor zum Beispiel Alphabet (ABC),world wide web (WWW) und Rolling on floor laughing (ROFL).

1.1 Erste Überschrift Tiefe 2 (section)

blindtext

1.1.1 Erste Überschrift Tiefe 3 (subsection)

blindtext

Erste Überschrift Tiefe 4 (subsubsection)

blindtext

2 Zweite Überschrift der Tiefe 1 (chapter)

blindtext

2.1 Zweite Überschrift Tiefe 2 (section)

blindtext

2.1.1 Zweite Überschrift Tiefe 3 (subsection)

blindtext

2.1.2 Dritte Überschrift Tiefe 3 (subsection)

blindtext

Zweite Überschrift Tiefe 4 (subsubsection)

blindtext
Querverweise werden in LaTeX automatisch erzeugt und verwaltet, damit sie leicht aktualisiert werden können. Hier wird zum Beispiel auf Abbildung 1 verwiesen.

Figure 1: Beispiel für die Beschriftung eines Buchrückens.

Figure 2: 2. Beispiel für die Beschriftung eines Buchrückens.

Und hier ist ein Verweis auf Tabelle 1. Das gezeigte Tabellenformat ist nur ein Beispiel. Tabellen können individuell gestaltet werden.

Table 1: Semesterplan der Lehrveranstaltung „Angewandte Mathematik“.

Datum	Thema	Raum
20.08.2008	Graphentheorie	HS 3.13
01.10.2008	Biomathematik	HS 1.05

Table 2: 2. Semesterplan der Lehrveranstaltung „Angewandte Mathematik“.

Datum	Thema	Raum
20.08.2008	Graphentheorie	HS 3.13
01.10.2008	Biomathematik	HS 1.05

Hier wird auf die Formel 1 verwiesen.

$$x = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q} \tag{1}$$

$$x = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q} \tag{2}$$

Literaturverweise sollten automatisch verwaltet werden, vor allem, wenn es viele Quellenverweise gibt. Beispiele sind [1] [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. Das verwendete Zitierformat (bzw. das Format des Literaturverzeichnisses) ist entspechend der Vorgaben der Studiengänge zu wählen.

3 Dritte Überschrift der Tiefe 1 (chapter)

Hier wird etwas Quellcode dargestellt:

```
#include <iostream>

void SayHello(void)
{
    // Kommentar
    cout << "Hello World!" << endl;
}

int main(int argc, char **argv)
{
    SayHello();
    return 0;
}
```

Listing 1: Hello-World

3.1 Algorithms

Use a defined environment for algorithms.

Algorithm 1 is an example from the gallery (https://www.overleaf.com/latex/examples/euclids-algorithm-an-example-of-how-to-write-algorithms-in-latex/mbysznrmktqf) .

Algorithm 1 Euclid's algorithm

procedure EUCLID(a, b) ▷ The g.c.d. of a and b
  $r \leftarrow a \bmod b$
  while $r \neq 0$ **do** ▷ We have the answer if r is 0
    $a \leftarrow b$
    $b \leftarrow r$
    $r \leftarrow a \bmod b$
  return b ▷ The gcd is b

4 Introduction

After a drop in rail passenger transport in 2020, it quickly recovered and increased by almost 71%, resulting in 328,3 million passengers in 2023 [16]. The Wiener Linien and the ÖBB show similar behaviors with an increase of 218 million and 115,4 million passengers from 2020 to 2023 respectively [17] [18].

The increase in train traffic is bound to a rise in accidents. In Austria, 34 rail traffic accidents with 38 victims were recorded in 2021. Of them 24 were seriously injured and 14 were killed [19]. Incidents include a train derailment in the area of a signal box, where there usually are many switches [20]. In 2022 in Münchendorf a train drove too fast over a switch, resulting in a derailment accident, with one passenger dead and 25 injured [21]. Also in other countries like the Czech Republic, an accident occurred, where a head-on collision of two trains resulted in four deaths. It is suspected that an incorrect switch caused that incident [22]. The cause of all accidents mentioned can be traced back to either an incorrect set switch or a derailment at a switch. This leads to the idea that correctly identifying the switch states may have prevented these accidents.

(a) Accident in Münchendorf, Austria. The train drove too fast over a switch and derailed, resulting in one death and 25 injured. [21]

(b) Accident in the Czech Republic. Head-on collision of two trains. The suspected cause is an incorrecty set switch, resulting in four deaths. [22]

Figure 3: Train accidets in **(a)** Austria and **(b)** the Czech Republic, where misinformation of switches resulted in deaths.

Simultaneously, autonomous vehicles are becoming increasingly seen as revolutionary technology for the future [23]. Also indicated by the "Autonomous Vehicle Readiness Index" [24] in most countries including Austria [25]. Especially autonomous road vehicles are rapidly evolving and could improve safety [26] [1], with machine-learning algorithms being a key trend [26]. The rail domain on the other hand, has gotten little attention [26]. This is because, unlike other

modes of transportation like airplanes or cars, trains follow a predetermined path defined by the tracks. Nevertheless, trams or other low-speed trains operate in dynamic environments where autonomous systems are still needed to respond to unpredictable events [1]. Therefore autonomy for these modes of transport will gain importance in the future [27].

One particular task of an autonomous train system involves filtering out the area in front of the vehicle, which can be defined as the danger zone. This area represents where the train is headed and the space, which is occupied next.

The problem is, that many state-of-the-art techniques in the rail domain consider all rails or rail tracks, and often make no distinction between the train's track and other tracks [1]. This can be observed in figure 5, in the right two images of figure 6 and in figure 7. While the segmentation of rails can be used for obstacle detection [27], the output of these methods can easily be misinterpreted [1]. Train infrastructure is complex, with multiple tracks often crossing, merging, or splitting at switches. Therefore, it is insufficient to recognize all tracks in a single image. A more targeted system is needed, which only detects the train's track and predicts the upcoming path at switches [1]. Such a system identifies the most relevant danger zone and when it is real-time capable it could be used as a preprocessing step for other algorithms or safety features [1] [27]. Leading to better-informed decisions [1], possibly preventing accidents like the ones mentioned before and potentially save lifes. Figure 4 shows a schematic visualization of a more targeted system.

This is why in the line of this work a distinction between *rail detection* and *rail track prediction* is made. On the one hand, *rail detection* in this context is filtering out rails or rail tracks with no further information about the own or adjacent rails. On the other hand, *rail track prediction* answers the question of where the train will be in the next few moments, particularly when switches are present that split the track. In figure 4 a *rail detection* would ouput both the green and the red marked tracks as one class and a *rail track prediction* outputs just the green track.

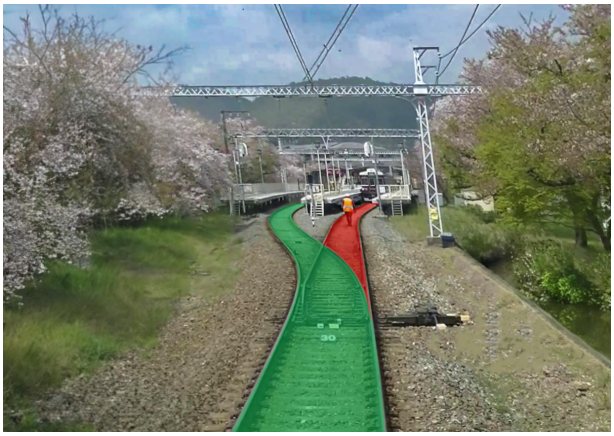


Figure 4: Example of a diverging rail track at a switch. The trains path is marked in green. The path the train cannot take is marked in red. This path is be obstructed and would be unsafe [1]. *rail detection*-output: green and red track; *rail track prediction*-output: green track

The focus of *rail track prediction* extends beyond solely camera-based detection of the train's path. It particularly emphasizes the accurate identification of switches where the train's path diverges, as illustrated in Figure 4.

Consequently, the goal of this work is the development of a rail track prediction system, which correctly predicts the most relevant danger zone and the direction in which the train is moving This is done with an emphasis on scenarios involving track splits through switches present in the Field Of View (FOV). Since the use case primarily takes place in dynamic environments, another goal, besides achieving an acceptable accuracy, is to ensure it operates in real-time and is therefore lightweight [1].

As the goal is to perceive the environment in front of the train, it is most advantageous to use front-view cameras [1] [27]. Images are captured in the driving direction from the driver's cabin because it is the best view of the tracks [1].

The main contributions of this work can be roughly divided into three parts.

1. Firstly, an initial approach was taken that aimed to combine object detection and semantic segmentation. However, due to unsatisfactory results, this methodology was deemed ineffective, leading to the pursuit of a fundamentally different solution.

2. Secondly, TEP-Net [1] was chosen as the new baseline for further experiments. Improvements are implemented with a specific focus on correct path prediction in the presence of switches.

3. Thirdly, as further improvements were anticipated through the use of the temporal component with RNNs, the system was adjusted to accept video information rather than single frames as input. Thus, the data loading logic was adapted and models were modified. Additionally, a temporal dataset was created. To minimize the workload for annotation, an auto labeler was developed utilizing the improved single-frame-based model.

5 State of the Art describes different approaches, to how the train track prediction problem can be solved. In this section, thorough research is done, which gives an overview of various approaches, models, and fitting datasets. Additionally, a couple of papers that could serve as a baseline are discussed in more detail. In **6 Methodology** the used datasets as well as the hardware and software frameworks used for training and evaluation are described. Additionally, this work includes the development of a temporal dataset with partly auto-labeled annotations. The labeling process and algorithms for auto-labeling are also described in this section. After that carried-out experiments are described in **7 Experiments** and their results are described in **8 Results** following a discussion in **9 Discussion**. In the final section **10 Conclusion and Outlook**, the work is summarized and further possible ideas for improvements are presented.

5 State of the Art

Etwas Text... Hier kommen noch einige Abkürzunge vor zum Beispiel ABC,WWW und ROFL.

5.1 Different solutions approaches for rail prediction

möglichkeiten - rail segmentation (detection und keine direction prediction) hat normalerweise probleme wenn mehrere rails in der scene sind –> weichen schwer trennbar weil der pixelhaufen dazwischen irgendwas ist - rail line detection (detection und keine direction prediction) anzahl der rails in einer scene ist das problem - TEP-Net

5.1.1 Ensemble Learning

Real-time Object detection

- Yolov9
- Yolov7
- ...

Semantic Segmentation

- DeepLab V3+
- AdapNet++
- ...

5.1.2 Panoptic segmenation (vielleicht)

5.1.3 Rail Segmentation

5.1.4 Line detection algorithms

Rail Detection: An Efficient Row-based Network and A New Benchmark –> hat auch Lane Detection & Railroad Detection

5.1.5 Backbone architectures

- EfficientNet
- ResNet
- MobileNet
- DenseNet

5.1.6 mögliches Baseline Paper

5.1.7 verwendetes Baseline Paper

5.1.8 Temporal Models

- LSTM
- GRU
- ...

5.2 Datasets

Datasets are essential for the development of autonomous driving systems, particularly for training and testing algorithms or neural networks. Typically, raw sensor data is collected from real-world driving scenarios, providing a realistic environment that reflects potential situations the system may encounter in future applications. This allows for more accurate modeling and evaluation of the system's performance under real conditions. A common approach to solving problems in autonomous driving systems is using vision-based machine learning algorithms [26, S. 1221]. These applications typically rely on camera-based data to address various challenges [26, S. 1221].

Since, autonomous vehicles are increasingly considered a groundbreaking technology of the future [23] a lot of work is put in the development of such systems. However often the main focus of this quickly evolving field is on road vehicles, like cars or trucks. Therefore, most publicly available datasets focus on this application and primarily reflect scenarios in road traffic [26, S. 1221].

Classification Datasets

There are a couple of public datasets for different Computer Vision (CV) applications. The data of the following datasets are especially gathered for classification tasks.

Table 3: Datasets for Classification

Dataset	Labels	Number of relevant images
CIFAR 100	trains	600
PASCAL VOC2012	trains	544
Microsoft COCO	trains	3745
1000 ImageNet	electric_locomotive, steam_locomotive, bullet_train	6722
Open Images Dataset V4	train	9284

CIFAR 100 dataset [28] consists of small images with pixel size 32x32. This dataset includes 600 images with the label *trains*. In *PASCAL VOC2012* [29] there are 544 images labeled *trains*. *Microsoft COCO* [30] has 3745 images with the class *trains*. Additionally, there are classes like *traffic lights* and *stop sign*, however theses are not useful to the task of this work because they are from the street domain and not the rail domain. *1000 ImageNet* [31] includes labels like *electric_locomotive* (4330 images), *steam_locomotive* (1187 images) and *bullet_train* (1205 images). *Open Images Dataset V4* [32] consists of more than 9.2 million images, annotated with bounding boxes. Included are 10506 *train*-labels in 9284 images. However, there are no

other significant labels relevant to the rail domain. Additionally, the dataset contains labels for toy trains, which could pose potential challenges.

Semantic Segmentation

Semantic segmentation labels are often refereed to as dense or pixel-wise annotated data. These datasets are characterized by the fact that each pixel in their images is assigned to a class.

Table 4: Datasets for Semantic Segmentation

Dataset	Labels	Number of relevant images
Cityscapes	rail track, train	284
Mapillary Vistas	construction-flat-rail-track, object-vehicle-on-rails	710
COCO-Stuff	platform, railroads, train	8615
KITTI	rail tracks, train	65

The *Cityscapes* dataset [33] is commonly used for benchmarks when it comes to road scenes. It has 35 different labels of which two are *rail track* (131 labels in 117 images) and *train* (194 labels in 167 images). The *rail track* does not differ between the rails and track bed. *Mapillary Vistas* [34] also has more labels, but again only two are rail related ones. *construction-flat-rail-track* is annotated in 710 images and *object-vehicle-on-rails* occurs 272 times. *COCO-Stuff dataset* [35] includes the same 182 classes like the *Microsoft COCO* [30] but as dense labels. The rail relevant ones are *railroad* (2839) and *train* (4761). There is a third rail related label *platform*, however this is a very general label because this can by any plane. *KITTI* [36] has the same dense labels like *Cityscapes* [33]. Likewise, the rail relevant ones are 47 *rail track* and 18 *train* labels.

These are commonly used datasets in CV tasks. However there are there are three main issues when it comes to solving the track prediction use case presented in this work. Firstly, there is not enough data because the amount of included rail relevant labels is relatively low in each dataset. Secondly, the labels present are not suitable for training a track prediction algorithm. In this case only the rails, rail tracks or track beds are needed. Thirdly, the images of the presented datasets are taken out of passengers and pedestrian views. Additionally there are some road views [37].

The datasets mentioned before are very general with a vast amount of different labels. However there are some datasets specially captured for the rail domain. As with the other datasets, it is important to consider what specific tasks the datasets are intended to be used for. There are some datasets captured in the birds eye view to detect damages like cracks in rails [38] [39] or even some to detect garbage in grooved rails [40]. Then there are datasets in ego-

perspective of the train driver like *FRSign* [41] or *GERALD* [42], which are created for detecting different traffic lights on French and German railways.

This work deals with Rail Track Prediction, so the system should predict the direction of the track in front of the train. For this particular use case it is most advantages when the dataset is recorded out of the driver cabin in ego-perspective, because it offers a clear sight of the rails in front of the train. Therefore the data has to reflect scenarios, which are comparable. Since the view of the captured images represents a key factor, the before mentioned dataset become unsuitable. An additional reason why these datasets cannot be used for this work is the fact that they are created for different use cases. Other datasets which deal with the perception in a rail domain environment and are captured in the right perspective are discussed in the following paragraphs.

RailSem19

RailSem19 [26] is the first publicly available dataset fitted for environments in the rail domain. It consists of 8500 annotated images which are gathered from YouTube videos. All of these images are captured in the ego perspective of the train driver, which makes it suitable for the use case of this work. Additionally there are both bounding box labels and dense labels included for object detection and semantic segmentation. The bounding box labels are: *guard-rail; rail; traffic-signal-front; traffic-signal-back; traffic-sign-front; crossing; train; platform; buffer-stop; switch-indicator; switch-static; switch-left; switch-right; switch-unknown*. Important for predicting the direction of the train are the switch labels, because it gives valuable information. The *switch-unknown* label is used when there is a switch visible but it is unclear in which direction the train would proceed. The presents of this label is mainly due to the high noise levels of the images of YouTube videos. The dense labels of RailSem19 are: *road; sidewalk; construction; tram-track; fence; pole; traffic-light; traffic-sign; vegetation; terrain; sky; human; rail-track; car; truck; track-bed; on-rails; rail-raised; rail-embedded; void*. In the case of dense labels the labels *tram-track; rail-track; track-bed; on-rails; rail-raised; rail-embedded* are of importance for predicting the direction of trains and trams. Another advantage is that very diverse environments have been used for this dataset. The creators of *RailSem19* took images from 38 different countries in all four seasons and weather conditions. Additionally, the focus was not only on rails but also on trams, providing a very diverse reflection of rail scenarios and not limiting the use on a specific use case.

Figure 5: RailSem19 dataset examples. First row raw images. Second row dense Ground Truth (GT) [26].

RailVID

Another dataset that focuses on the detection of rails is the *RailVID* dataset [43]. The goal of this project is to detect rail tracks and obstacles on the rails, which can lead to possible hazardous situations. With a functioning system, fully automatic train operation is aimed for. The *RailVID* dataset is a collection of 1071 images with the following labels: *background, railway, car, people*. Since, the area of application is on the "Suzhou Rail Transit Line 1" in Jiangsu Province, China all data is captured there. *RailVID* is a collection of infrared data captured with the AT615X infrared thermal instrument from InfiRay. The decision to use infrared data and not RGB images is because it is more robust against challenging imaging conditions, like darkness at night, fog, rain and direct light disturbance. Since, this dataset only consists of infrared data and in this work Red Green Blue (RGB) data should be used, *RailVID* cannot be used fro training. An additional issue is, that the dataset is recorded only on a specific Chinese line. This is advantageous for this particular use case, but it could become an issue if the system were to be deployed elsewhere.

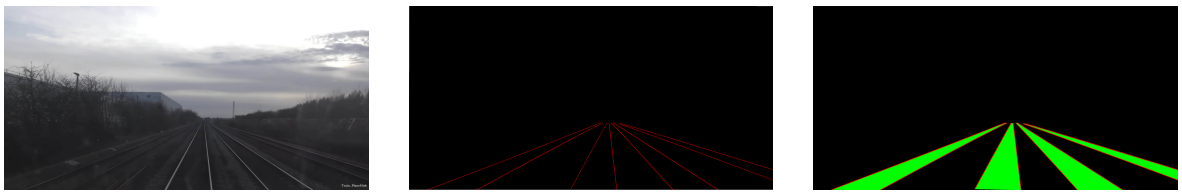
Figure 6: Example images and GT of RailVID dataset [43]

RailSet

[44] and [45] presented the *RailSet* dataset, which is divided into two sub sets: RailSet-Seg for segmentation and RailSet-Ano for anomaly detection [44]. Both of them are captured in the ego perspective of the train driver [44] [45]. The idea is to firstly detect the railways using semantic segmentation and secondly using positional information of the prediction for the creation of the anomaly dataset [44]. RailSet-Ano is a collection of 1100 images of railway defects like rail discontinuity and holes in the rail bed. Some anomalies are taken from a other images and pasted on images of RailSet-Seg and RailSem19 dataset others are generated with a network [44]. Since, RailSet-Ano deals with a different use case than the one in this work, this particular data cannot be used.

On the other hand, RailSet-Seg fits the problem. It consists of 6600 images of normal situations. The images are collected from 23 YouTube videos with a collective duration of 15 hours. It includes two labels: *rail* and *rail-track*. Besides the use of RailSet-Seg for the creation of RailSet-Ano, an additional motivation is to include more complex scenes of the rail domain than RailSem19. That is why the focus of RailSet lies on scenarios with poor weather conditions or lighting conditions. Furthermore, images are included in which the rails are not visible at all, like in tunnels without lighting or in snowy scenes. Additionally, it was ensured that the videos were recorded by different cameras and from different mounting positions [44] [45].

An advantage of this dataset is that it can be joined with RailSem19, when combining four specific labels. *trackbed* and *rail-track* from RailSem19 have to be transformed into RailSet's *rail-track* label and *rail-raised* and *rail-embedded* become the *rail* label. This method leads to more data for the training and validation which could be an advantage. However, it also shows the disadvantage of this dataset for the specific use case of this work. RailSet exclusively addresses railway and not tram scenes [45]. Since, the goal is to target a broad applicability, leaning towards tram scenes this dataset is not used for this work.



(a) (b) (c)

Figure 7: RailSet-Seg example with annotations [44] [45]: **(a)** raw-image, **(b)** rail class, **(c)** rail and rail-track class

RSDS

The creators of the Railroad Segmentation Dataset (RSDS) [27], tried to solve the railroad detection problem with a segmentation approach. Because there was no publicly available dataset for this task at the time, they had to construct their own. RSDS is captured from the

ego perspective of the train driver and is a collection of 3000 images. They used 2500 for training, 200 for validation and 300 for testing. The dataset only includes one *railroad* label. It is described that the labels only incorporate pixels between the two rails and intentionally ignore railway sleepers outside this area. Images are of size 1920 x 1080. Although it is not mentioned in the paper, it seems as if all images of RSDS are captured from a specific rail line in China. Additionally, from the images in the paper alone one can tell that this particular rail line has very distinctive structural characteristics. The colors are bright and it seems like the area between the rails is mostly concrete, which is unusual for most railways. 8 shows that structures besides the rails are specific too. One additional detail of this dataset, which can present a disadvantage for this work, is that switches are not addressed. Since this dataset only covers very specific railroads with unique characteristics and additionally does not address switches or other situations where the track splits, RSDS is not further considered for this work.

(a) (b)

Figure 8: RSDS example with annotation [27]: **(a)** raw-image, **(b)** ground truth

Rail-DB

A very similar dataset is Rail-DB [46]. This dataset is a collection of 7432 images, which are taken from 15 videos. The labels in this dataset are consist of poly lines, which represent all existing rails in an image. Additionally the the poly lines all have different classes. This way there is not only one rail class, but as many classes as there are rails in one image. The labeling policy specifies that the central rails are marked with the annotations 1 and 2. Additional rails are labeled with rising numbers. Since this dataset includes poly lines and not binary masks, this dataset can be used for training line detection algorithms. Compared to RSDS, this dataset includes not only lines and curves, but also rail switches. Additionally, the images are taken in various conditions and scenes. 9 shows example images of Rail-DB's different scenes. However, it seems that the images again have very specific characteristics like in RSDS. 9 also shows, that all images are captured on a Chinese line. Even though [46] presents a very interesting approach for solving the rail detection problem, because of the before mentioned specific characteristics it is not used for this work. Even the author of [46] states in the GitHub repository [47], that this project fails to generalize on example videos, where the scene looks

different from the one the dataset is captured on.

Figure 9: Rail-DB [46] images with annotations in different conditions

OSDaR23

Another dataset it the OSDaR23 [48]. This dataset is a collection of 21 sequences, which are split into 45 subsequences. Several sequences are short ones with 10 frames each, some are longer with 40 to 100 frames. In total there are 1534 labeled scenes in this dataset. Since, OSDaR23 is captured with 9 cameras (RGB and Infrared (IR)) in different angles, one lidar and one radar sensor, the total number of frames are 1534 * 11 = 16874. Additionally, position and acceleration sensors are used. Because also lidar and radar is used, this dataset offers 3D data as well. OSDaR23 consists of 20 different labels. These labels include the rail context, like *track*, *switch* or *train* and the environment, like *person*, *animal*, *bicycle*, *smoke*, *flame* or *crowd*. The annotations for the environment can be used for safety applications. For more details please refer to *TABLE V* in [48]. 10 shows the 11 different frames of each sensor and 11 shows an example of an annotated scene. As illustrated in 11 the *track* label and the *switch* label only offers positional information about their presents, but do not include any information on the direction of the train. Furthermore, there is no information that distinguishes the train rails from the adjacent rails. Therefore OSDaR23 can only be used for the detection of rails, but not the track prediction. Moreover, the capturing of this dataset only took place on rail roads in Hamburg, Germany. No tram scenes are included. Due to the necessity of re-labeling for this work and the fact that only data from Hamburg is offered, OSDaR23 is not used.

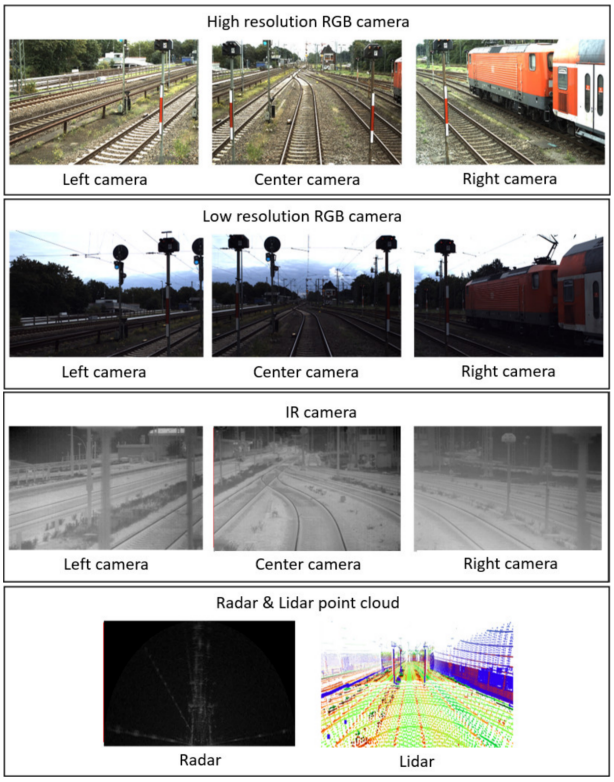
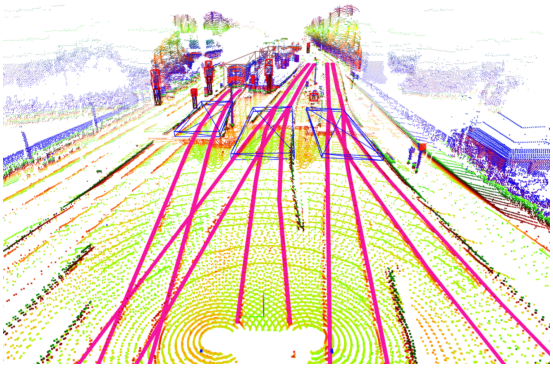
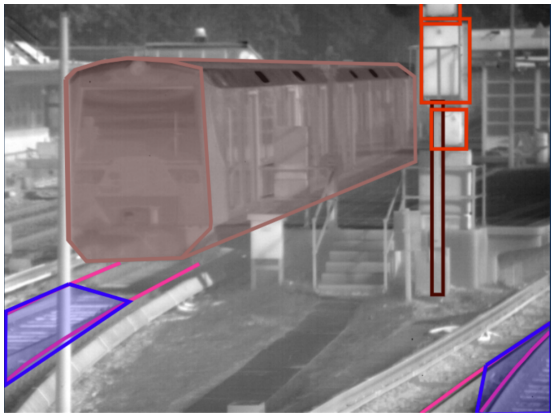


Figure 10: OSDaR23 data from all different sensors [48]

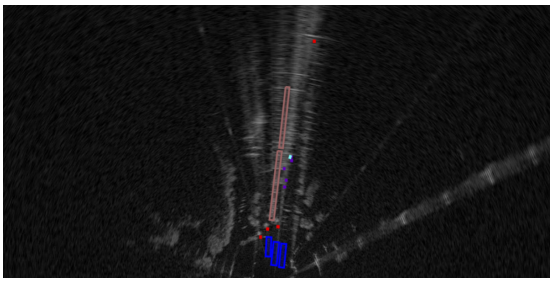
(a) RGB center camera



(b) merged lidar point cloud



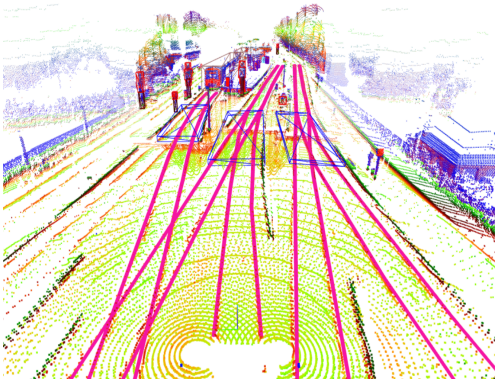
(c) IR center camera



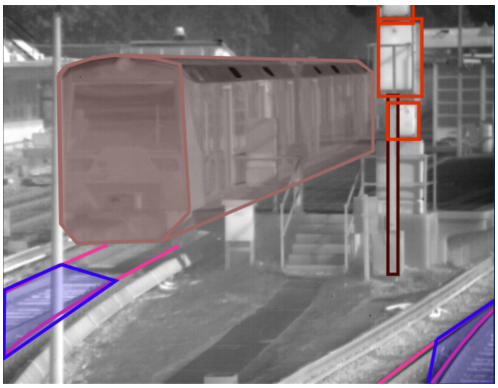
(d) Radar (zoomed)

Figure 11: OSDaR23 annotated scene [48]

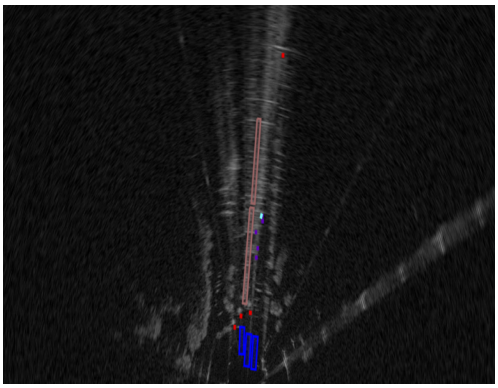
(a) Bild 1



(b) Bild 2



(c) Bild 3



(d) Bild 4

Figure 12: OSDaR23 labels skaliert

TEP-Net dataset

[1] presents the Train Ego Path (TEP)-Net dataset. The problem this paper aims to solve is rail track prediction. This differs from all previously mentioned datasets. No dataset but [1] provides information, which distinguishes possible rails in the image from the one the train actually follows. Since RailSem19 is the most popular dataset in the rail domain, it is used as initial point. A total of 7917 images were taken from RailSem19. The remaining 583 were excluded because they are taken from unusual perspectives or it is unclear which track the train is on or would continue on. Examples of such images are shown in 13. These images are excluded simply by not annotating them. For annotation a new labeling format is created. Two classes *rail_right* and *rail_left* give information about where the tracks of the train run. All other rails which might be in the image are ignored. These two classes consist of poly lines, which are annotated by the corresponding x and y pixel coordinates of the image. Only the tracks on which the train is located are labeled. Even if a switch appears in the image and the train would travel over it, only the tracks in the correct direction are further labeled. This way, switches are indirectly included in this dataset, providing information on the direction a train would continue, even though there is no explicit label for switches. The poly lines start from lowest pixel row

of the images and extend up to a specific horizon line. Above this horizon further labeling is not possible. This may occur for various reasons. The first reason can be an obstruction of the view by the environment or other trains, as shown in 14. Since the images are sourced from YouTube videos, it is also possible that the resolution becomes too low in the distance for separately identifying both rails. Another reason is when it is not possible to determine the direction in which the track continues based on a switch present in an image. This can happen due to low resolution or unfortunate camera angles. These cases may be labeled as *switch-unknown* in the original RailSem19 dataset for example. Here, the polylines stop before the switch. The polylines from this annotation can then be converted into several different labels using preprocessing algorithms. On one hand, they can be directly used as polylines. On the other hand, they can be transformed into a mask for segmentation tasks by filling in the area between the lines. A third application would be to convert this mask into a grid for classification tasks.

Figure 13: Examples images from RailSem19, which are not included in the TEP-Net dataset due to unclear circumstances about the trains direction. [1]

Figure 14: TEP-Net dataset example images with annotation [1]

notes wo alles ist

- Railroad Segmentation Dataset (RSDS) –> in RailNet: A Segmentation Network for Railroad Detection fertig
- RailSem19 in RailSem19 fertig

- RailVID in RailVID fertig
- RailSet in RailSet & Application of Rail Segmentation in the Monitoring of Autonomous Train's Frontal Environment fertig
- Rail-DB (compared to RSDS) –> Rail Detection: An Efficient Row-based Network and A New Benchmark fertig
- OSDaR23 in OSDaR23: Open Sensor Data for Rail 2023 fertig
- TEP-net dataset fertig
-
- RailSet -> Segmentation & Anomaly detection
- Application of Rail Segmentation in the Monitoring of Autonomous Train's Frontal Environment -> RailSegmentation (only rails and trackbed)

5.2.1 Erste Überschrift Tiefe 3 (subsection)

blindtext

Erste Überschrift Tiefe 4 (subsubsection)

blindtext

5.3 Baseline Paper

blindtext

5.3.1 Description of Baseline Paper

blindtext

5.3.2 Results and Limits of Baseline Paper

blindtext

6 Methodology

blindtext

6.1 Datasets

blindtext

6.1.1 RailSem19 and used subsets

blindtext

6.1.2 vielleicht: Rail-DB for multi rail setection

blindtext

6.1.3 used annotations

blindtext

6.2 Switch evaluation dataset

blindtext

6.3 labeling task for temporal models

CVAT

Autolabler

blindtext

6.4 Setup used for Training CNNs

For training Convolutional Neural Network (CNN)s a powerful hardware setup is necessary. In this work, one main setup is used to train all models. This work is based on a project proposed and supported by the Institute for Computer Technology (ICT) at the Technical University Vienna (TU), which also provides the necessary resources. The department has a server with two NVIDIA Tesla V100S-PCIE-32GB Graphics Processing Unit (GPU)s [49]. Since the GPUs utilized are equipped with 32 GB of memory and this is sufficient for the needs of this study, all trainings are done on a single GPU. No multi-GPU setup is needed. For this project, CUDA version 12.6 is used, which provides an environment for developing GPU-accelerated applications [50]. The server employs two Intel(R) Xeon(R) Gold 5118 CPUs @ 2.30GHz [51]. These Central Processing Unit (CPU)s have 24 threads and 12 cores each.

All trainings are logged with the "Weights & Biases" [52], a developer platform for training and fine-tuning machine learning models. [52] is utilized for logging configurations and results in this work. The *train loss* and *validation loss* of each training is tracked and visualized in a graph. The *test Intersection Over Union (IoU)* and the *best validation loss* are displayed at the end of each training. Additionally, the GPU's power usage and allocated memory are tracked and graphs are plotted. Moreover, "Weights & Biases" [52] assigns a unique name to each training session, a critical feature when starting hundreds of training sessions. All training logs are available at https://wandb.ai/sebiorganization/train-ego-path-detection.

6.5 Measuing Inference on NVIDIA Jetson Device

Due to the nature of this work's use case and potential applications for the system encompassing safety features or preprocessing steps for autonomous trains, the system has to ensure real-time capability when deployed on embedded devices. In more detail, one goal for the track prediction is to be deployed on an Nvidia Jetson device, because of their high computing power and their low power consumption. Additionally, through NVIDIA's software ecosystem, rapid deployment and latency measurements are possible. Jetson devices are suitable for applications in autonomous systems and computer vision tasks [53].

6.5.1 Hardware Setup for Measuring Inference

For this study, the NVIDIA Jetson AGX Xavier is chosen because of several reasons. This platform achieves up to 32 TOPS by utilizing a GPU with 512 cores and 64 Tensor cores, which is advantageous for parallel data processing and neural network inference [54]. Even though there are more powerful devices like some of the NVIDIA Jetson Orin series, the AGX Xavier is sufficient for this application and cheaper [55]. Additionally, the Xavier has a lower power consumption than the Orin. Finally but yet important is the ease of integration. Since, track prediction is a use case of a company, that most commonly uses the Jetson AGX Xavier

platform. Conducting tests on this specific device is appropriate. The technical specifications of the NVIDIA Jetson AGX Xavier are shown in table 5.

AI Performance	32 TOPS
GPU	512-core NVIDIA Volta GPU with 64 Tensor Cores
CPU	8-core NVIDIA Carmel ARM v8.2 64-bit CPU | 8 MB L2 + 4 MB L3
Memory	32 GB 256-Bit LPDDR4x | 136.5 GB/s
Storage	32 GB eMMC 5.1
DL Accelerator	(2x) NVDLA
Power	10 W - 30 W

Table 5: Jetson AGX Xavier technical specifications [54]

6.5.2 Optimizing models with TensorRT

To fully leverage the used hardware platform Jetson AGX Xavier, NVIDIA introduced TensorRT [56]. This ecosystem is developed to allow faster inference times when deploying deep learning models. Also, the baseline paper [1] demonstrates through latency measurements that TensorRT consistently outperforms PyTorch in the context of speed. TensorRT is approximately six times faster than PyTorch, which aligns with the claims made by NVIDIA [1] [56]. Consequently, all latency measurements in this study are conducted using TensorRT. This framework optimizes inference with methods like quantization, layer and tensor fusion, and kernel tuning [56]. This can be done for various types of NVIDIA GPUs.

Quantization can optimize an already trained model. This technique shows a small reduction in accuracy but minimizes latency significantly. While PyTorch uses PF32 for the inference of its standard models, TensorRT allows to use GPUs and Tensor Processing Unit (TPU)s to their maximum capacity by permitting FP8, INT8, and INT4.

Layer and Tensor fusion are used from TensorRT for further optimizing inference. Often specific layer sequences include two consecutive layers that can be mathematically combined into a single layer. Resulting in a reduction of unessential computations.

Kernal tuning is a process that seeks the optimal configuration of available kernels in the Jetson device. The selected kernels depend on the specific machine-learning application and the used device. In more detail, the model is executed on the device several times using different CUDA kernels in each run and the best combination is utilized. Since, Kernel tuning is an iterative process it usually takes a couple of minutes even with compact models.

TensorRT engine from PyTorch model

Since TensorRT does not support PyTorch models, a workaround has to be made. First, the PyTorch models are converted into an Open Neural Network Exchange (ONNX) format [57]. After that, the `.onnx` files are optimized by TensorRT with the techniques mentioned before.

ONNX is an approach for easier access to hardware optimization and to make interoperability possible [57]. Often machine learning applications are locked in the framework they are developed in, which can present some hurdles. The ONNX format aims for a standardized representation of machine learning models and is therefore commonly used in the community. Once converted to ONNX, a model utilizes standard data types and a set of built-in operators. The ONNX format version, known as the "opset", defines which operators are used [57]. Therefore the TensorRT version must be compatible with the ONNX opset number.

The TensorRT version installed on the Jetson AGX Xavier must support the used ONNX opset. Therefore in this work, the opset 11 is used for all model exports.

```
# Export the model to ONNX
torch.onnx.export(
    model,                          # Model to be exported
    input_tensor,                   # Input to the model
    "onnx_file_path/model.onnx",    # Output file path
    opset_version=11,               # ONNX version to export the model to
    export_params=True,             # Store the trained parameter
                                    # weights inside the model file
)
```

Listing 2: Exporting a PyTorch model to ONNX format

In this work, all PyTorch models are converted to `.onnx` files with the built-in torch exporter. Listing 2 shows that the conversion is done with a single `torch.onnx.export()` python line [58]. After an ONNX model format is created it can be optimized by TensorRT. This can be executed with the `trtexec` console application.

```
trtexec –onnx=model.onnx –saveEngine=model.engine
```

This command provides an example, in which the model.onnx is optimized with TensorRT techniques mentioned above. Furthermore, the `model.engine` is saved and can be executed from now on without the need to create it again. After a couple of minutes, TensorRT outputs a performance summary with many different latencies, like the min, max, mean, or median. Of

these values, the median inference time is considered as the final latency rather than the mean, because it is more robust against outliers.

7 Experiments

Etwas Text... Hier kommen noch einige Abkürzunge vor zum Beispiel ABC,WWW und ROFL.

7.1 Ensemble Learning Approach

blindtext

7.2 improved TEP-Net

blindtext

7.2.1 Autocrop

blindtext

7.2.2 Backbones

blindtext

7.2.3 Pooling Layers

blindtext

7.2.4 Prediction Heads

blindtext

7.2.5 Sliding Window Approach

blindtext

7.2.6 Temporal Models

blindtext

7.2.7 Erste Überschrift Tiefe 3 (subsection)

blindtext

7.2.8 Erste Überschrift Tiefe 3 (subsection)

blindtext

Erste Überschrift Tiefe 4 (subsubsection)

blindtext

8 Results

Etwas Text... Hier kommen noch einige Abkürzunge vor zum Beispiel ABC,WWW und ROFL.

8.1 Ensemble Learning Approach

blindtext

8.2 improved TEP-Net

blindtext

8.2.1 Autocrop

blindtext

8.2.2 Backbones

blindtext

8.2.3 Pooling Layers

blindtext

8.2.4 Prediction Heads

blindtext

8.2.5 Sliding Window Approach

blindtext

8.2.6 Temporal Models

blindtext

8.3 Erste Überschrift Tiefe 2 (section)

blindtext

8.3.1 Erste Überschrift Tiefe 3 (subsection)

blindtext

Erste Überschrift Tiefe 4 (subsubsection)

blindtext

9 Discussion

Etwas Text... Hier kommen noch einige Abkürzunge vor zum Beispiel ABC,WWW und ROFL.

9.1 Erste Überschrift Tiefe 2 (section)

blindtext

9.1.1 Erste Überschrift Tiefe 3 (subsection)

blindtext

Erste Überschrift Tiefe 4 (subsubsection)

blindtext

10 Conclusion and Outlook

Etwas Text... Hier kommen noch einige Abkürzunge vor zum Beispiel ABC,WWW und ROFL.

10.1 Erste Überschrift Tiefe 2 (section)

blindtext

10.1.1 Erste Überschrift Tiefe 3 (subsection)

blindtext

Erste Überschrift Tiefe 4 (subsubsection)

blindtext

Bibliography

[1] T. Laurent, *Train ego-path detection on railway tracks using end-to-end deep learning*, 2024. arXiv: 2403.13094 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2403.13094.

[2] H. Kopka, *LaTeX, Band 1: Einführung*, 3rd ed. München: Pearson Studium, 2005.

[3] ——, *LaTeX, Band 1: Einführung*, 3rd ed. München: Pearson Studium, 2005. [Online]. Available: http://www.pearson-studium.de (visited on 07/06/2011).

[4] M. Goossens, F. Mittelbach, and A. Samarin, *Der LaTeX Begleiter*. Bonn: Addison-Wesley Deutschland, 2002.

[5] S. Teschl, K. M. Göschka, and G. Essl, *Leitfaden zur Verfassung einer Bachelorarbeit oder Master Thesis*, FH Technikum Wien, 2014. [Online]. Available: www.technikum-wien.at (visited on 08/04/2014).

[6] M. Humenberger, D. Hartermann, and W. Kubinger, "Evaluation of stereo matching systems for real world applications using structured light for ground truth estimation," in *Proceedings of the Tenth IAPR Conference on Machine Vision Applications (MVA2007)*, Tokyo, Japan: MVA Conference Committee, May 16, 2007, pp. 433–436.

[7] M. Humenberger, C. Zinner, M. Weber, W. Kubinger, and M. Vincze, "A fast stereo matching algorithm suitable for embedded real-time systems," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1180–1202, 2010.

[8] C. Zinner, W. Kubinger, and R. Isaacs, "Pfelib: A performance primitives library for embedded vision," *EURASIP Journal on Embedded Systems*, vol. 2007, pp. 1–14, 2007. [Online]. Available: http://downloads.hindawi.com/journals/es/2007/049051.pdf (visited on 07/06/2011).

[9] H. Hemetsberger, *Ait stereo sensor im einsatz während der darpa urban challenge 2007*, AIT Austrian Institute of Technology, 2007.

[10] Siemens Automation Technology, *Simatic*, 2011. [Online]. Available: http://www.automation.siemens.com/mcms/topics/de/simatic/Seiten/Default.aspx (visited on 07/06/2011).

[11] ——, *Simatic*, [Online] Verfügbar unter: <http://www.automation.siemens.com/mcms/topics/de/simatic/Seiten/Default.aspx> [Zugang am 17.10.2014], 2014.

[12] International Standards Office, *Iso 690 – information and documentation: Bibliographical references: Electronic documents*, Genf: International Standards Office, 1998.

[13] Atmel Corporation, *Atmel atmega16 – 8-bit microcontroller with 16k bytes in-system programmable flash*, San Jose, United States: Atmel Corporation, 2011. [Online]. Available: http://www.atmel.com/dyn/resources/prod%5C_documents/doc2466.pdf (visited on 07/06/2011).

[14] M. Humenberger, *Real-time stereo matching for embedded systems in robotic applications*, Wien: Technische Universität Wien, Fakultät für Elektrotechnik und Informationstechnik, 2011.

[15] J. Pohn, *Condition monitoring systeme für die zustandorientierte instandhaltung von windkraftanlagen*, Wien: FH Technikum Wien, Masterstudiengang Innovations- und Technologiemanagement, 2010.

[16] Statista-Austria. (2024). Anzahl der beförderten personen im schienenpersonenverkehr in österreich von 2009 bis 2023 (in millionen) [graph], [Online]. Available: https://de.statista.com/statistik/daten/studie/296524/umfrage/schienenpersonenverkehr-der-eisenbahn-in-oesterreich/ (visited on 11/05/2024).

[17] Wiener-Stadtwerke. (2024). Anzahl der fahrgäste der wiener linien von 2006 bis 2023 (in millionen) [graph], [Online]. Available: https://de.statista.com/statistik/daten/studie/714378/umfrage/fahrgaeste-der-wiener-linien/ (visited on 11/05/2024).

[18] Österreichische-Bundesbahnen. (2024). Anzahl der fahrgäste der österreichischen bundesbahnen (öbb) von 2012 bis 2023 (in millionen) [graph], [Online]. Available: https://de.statista.com/statistik/daten/studie/299369/umfrage/fahrgaeste-der-oesterreichischen-bundesbahnen/ (visited on 11/05/2024).

[19] Weninger, *Verkehrstatistik 2022*. 2022. [Online]. Available: https://www.statistik.at/fileadmin/user_upload/Verkehr-2022-barr.pdf (visited on 11/05/2024).

[20] red, *Zug entgleist: 40 passagiere gerettet*, 2024. [Online]. Available: https://noe.orf.at/stories/3260056/ (visited on 11/05/2024).

[21] Stefan Schwarzwald-Sailer, *Tödlicher zugsunfall bringt öbb in erklärungsnot*, 2024. [Online]. Available: https://noe.orf.at/stories/3259860/ (visited on 11/05/2024).

[22] red, *Mehrere tote bei zugsunglück in tschechien*, 2024. [Online]. Available: https://orf.at/stories/3359814/ (visited on 11/05/2024).

[23] Fraunhofer Institute for Cognitive Systems IKS, *Autonomous driving - fraunhofer iks*, 2024. [Online]. Available: https://www.iks.fraunhofer.de/en/topics/autonomous-driving.html (visited on 11/05/2024).

[24] KPMG. (2020). 2020 autonomous vehicles readiness index, [Online]. Available: https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2020/07/2020-autonomous-vehicles-readiness-index.pdf (visited on 11/05/2024).

[25] B. Impey. (2024). Entwicklungsstand im bereich autonomer mobilität nach ländern weltweit nach autonomous vehicle readiness index[1] (stand: 2020 und vergleich zum vorjahr) [graph], [Online]. Available: https://de.statista.com/statistik/daten/studie/1113863/umfrage/entwicklungsstand-im-bereich-autonomer-mobilitaet-nach-laendern-weltweit/ (visited on 11/05/2024).

[26] O. Zendel, M. Murschitz, M. Zeilinger, D. Steininger, S. Abbasi, and C. Beleznai, "Railsem19: A dataset for semantic rail scene understanding," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, 2019, pp. 1221–1229, ISBN: 978-1-7281-2506-0. DOI: 10.1109/CVPRW.2019.00161.

[27] Y. Wang, L. Wang, Y. H. Hu, and J. Qiu, "Railnet: A segmentation network for railroad detection," *IEEE Access*, vol. 7, pp. 143 772–143 779, 2019. DOI: 10.1109/ACCESS.2019.2945633.

[28] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," Toronto, ON, Canada, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744.

[29] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, pp. 98–136, 2015.

[30] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:14113767.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Apr. 2015, ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y. [Online]. Available: http://dx.doi.org/10.1007/s11263-015-0816-y.

[32] A. Kuznetsova, H. Rom, N. G. Alldrin, J. R. R. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4," *International Journal of Computer Vision*, vol. 128, pp. 1956–1981, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:53296866.

[33] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, " The Cityscapes Dataset for Semantic Urban Scene Understanding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2016, pp. 3213–3223. DOI: 10.1109/CVPR.2016.350. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.350.

[34] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, “The mapillary vistas dataset for semantic understanding of street scenes,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5000–5009. DOI: 10.1109/ICCV.2017.534.

[35] H. Caesar, J. Uijlings, and V. Ferrari, “Coco-stuff: Thing and stuff classes in context,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018. DOI: 10.1109/cvpr.2018.00132. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00132.

[36] H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, “Augmented reality meets computer vision: Efficient data generation for urban driving scenes,” *International Journal of Computer Vision*, vol. 126, pp. 961–972, 2018. DOI: https://doi.org/10.1007/s11263-018-1070-x.

[37] M. A. Hadded, A. Mahtani, S. Ambellouis, J. Boonaert, and H. Wannous, “Application of rail segmentation in the monitoring of autonomous train’s frontal environment,” in *Pattern Recognition and Artificial Intelligence*, ser. Lecture Notes in Computer Science, M. El Yacoubi, E. Granger, P. C. Yuen, U. Pal, and N. Vincent, Eds., vol. 13363, Cham: Springer International Publishing, 2022, pp. 185–197, ISBN: 978-3-031-09036-3. DOI: 10.1007/978-3-031-09037-0{\textunderscore}16.

[38] Z. Zhang, S. Yu, S. Yang, Y. Zhou, and B. Zhao, *Rail-5k: A real-world dataset for rail surface defects detection*, 2021. arXiv: 2106.14366 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2106.14366.

[39] S. Ma, K. Song, M. Niu, *et al.*, “Cross-scale fusion and domain adversarial network for generalizable rail surface defect segmentation on unseen datasets,” *Journal of Intelligent Manufacturing*, vol. 35, pp. 367–386, 2024. DOI: 10.1007/s10845-022-02051-7.

[40] X. Huang, K. Ye, Z. Fang, Y. Xie, X. Ma, J. Ji, and Q. Wu, “Research on grooved rail garbage identification algorithm based on improved yolov3,” *Journal of Physics: Conference Series*, vol. 1827, no. 1, p. 012 185, 2021. DOI: 10.1088/1742-6596/1827/1/012185. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1827/1/012185.

[41] J. Harb, N. R’eb’ena, R. Chosidow, G. Roblin, R. Potarusov, and H. Hajri, “Frsign: A large-scale traffic light dataset for autonomous trains,” *ArXiv*, vol. abs/2002.05665, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:211096970.

[42] P. Leibner, F. Hampel, and C. Schindler, “Gerald: A novel dataset for the detection of german mainline railway signals,” *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, vol. 237, no. 10, pp. 1332–1342, 2023.

[43] H. Yuan, Z. Mei, Y. Chen, W. Niu, and C. Wu, “Railvid: A dataset for rail environment semantic,” *ICONS*, vol. 2022, 17th, 2022.

[44] A. Zouaoui, A. Mahtani, M. A. Hadded, S. Ambellouis, J. Boonaert, and H. Wannous, "Railset: A unique dataset for railway anomaly detection," in *2022 IEEE 5th International Conference on Image Processing Applications and Systems (IPAS)*, vol. Five, 2022, pp. 1–6. DOI: 10.1109/IPAS55744.2022.10052883.

[45] M. A. Hadded, A. Mahtani, S. Ambellouis, J. Boonaert, and H. Wannous, "Application of rail segmentation in the monitoring of autonomous train's frontal environment," in *International Conference on Pattern Recognition and Artificial Intelligence*, Springer, 2022, pp. 185–197.

[46] X. Li and X. Peng, "Rail detection: An efficient row-based network and a new benchmark," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 6455–6463.

[47] S. Lee, *Rail-detection*, GitHub repository, 2022. [Online]. Available: https://github.com/Sampson-Lee/Rail-Detection.

[48] R. Tilly, P. Neumaier, K. Schwalbe, P. Klasek, R. Tagiew, P. Denzler, T. Klockau, M. Boekhoff, and M. Köppel, *Osdar23: Open sensor data for rail 2023*, de, 2023. DOI: 10.57806/9MV146R0. [Online]. Available: https://data.fid-move.de/dataset/3d7e7406-639f-49f6-bbca-caac511b4032.

[49] NVIDIA Corporation, *Nvidia v100 gpu architecture data sheet*, 2024. [Online]. Available: https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf (visited on 11/07/2024).

[50] ——, *Cuda toolkit*, 2024. [Online]. Available: https://developer.nvidia.com/cuda-toolkit (visited on 11/07/2024).

[51] Intel Corporation, *Intel® xeon® gold 5118 processor data sheet*, 2024. [Online]. Available: https://ark.intel.com/content/www/de/de/ark/products/120473/intel-xeon-gold-5118-processor-16-5m-cache-2-30-ghz.html (visited on 11/07/2024).

[52] Weights & Biases, *Weights & biases*, 2024. [Online]. Available: https://wandb.ai/site (visited on 11/07/2024).

[53] NVIDIA Corporation, *Nvidia jetson for next-generation robotics*, 2024. [Online]. Available: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/ (visited on 11/08/2024).

[54] ——, *Nvidia jetson xavier technical specifications*, 2024. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/ (visited on 11/07/2024).

[55] ——, *Buy the latest jetson products*, 2024. [Online]. Available: https://developer.nvidia.com/buy-jetson?product=all&location=DE (visited on 11/08/2024).

[56] ——, *Nvidia tensorrt*, 2024. [Online]. Available: https://developer.nvidia.com/tensorrt (visited on 11/08/2024).

[57] ONNX, *Open neural network exchange*, 2024. [Online]. Available: https://onnx.ai/ (visited on 11/08/2024).

[58] PyTorch, *Torch.onnx*, 2024. [Online]. Available: https://pytorch.org/docs/stable/onnx.html (visited on 11/08/2024).

List of Figures

List of Tables

Quellcodeverzeichnis

Abkürzungsverzeichnis

ABC Alphabet

WWW world wide web

ROFL Rolling on floor laughing

CV Computer Vision

RGB Red Green Blue

RSDS Railroad Segmentation Dataset

IR Infrared

TEP Train Ego Path

GT Ground Truth

FOV Field Of View

CNN Convolutional Neural Network

ICT Institute for Computer Technology

TU Technical University Vienna

GPU Graphics Processing Unit

IoU Intersection Over Union

CPU Central Processing Unit

TPU Tensor Processing Unit

ONNX Open Neural Network Exchange

A Anhang A

B Anhang B