Towards safe obstacle detection for autonomous train operation: Combining track and switch detection neural networks for robust railway ego track detection

Philipp Jass, Carsten Thomas, Tina Hiebert, Gustav Plettig

Towards safe obstacle detection for autonomous train operation: Combining track and switch detection neural networks for robust railway ego track detection

Philipp Jass*, Carsten Thomas, Tina Hiebert and Gustav Plettig

Department of Energy and Information — Computer Engineering
HTW Berlin, University of Applied Sciences
Berlin, Germany
*philipp.jass@htw-berlin.de

***Abstract*—Similar to autonomous driving on the road, automated and autonomous train operation also offers many advantages. These include relieving the burden on train drivers, as well as a possible increase in line capacity or the redevelopment of previously unprofitable route sections. One of the most important tasks of an autonomous train control system is to monitor the surroundings and, above all, the route to be traveled. This must be continuously monitored for possible obstacles in the train's path, just as a human train driver does. In order to perform this task, sensors are required that record data about the train's surroundings. Such sensors in autonomous systems are usually cameras, radar or lidar sensors. To detect obstacles on the track, the critical zone must first be identified. For trains, this area is called the clearance gauge and describes the space that the train occupies when traveling on a track. In complex scenes with switches, the section of track that the train travels through – the ego track – must be determined depending on the status of the switches. This paper presents an image-based approach for embedded on-board ego track determination, combining track and switch information in order to achieve a more robust ego-track prediction.**

***Index Terms*—On-Board Railway Track Detection, Machine Learning, Computer Vision, Robustness**

I. Introduction

Rail transportation is considered as a safe and energy-efficient mode of transport. But the technology needs to evolve to cope with increasing demands for traffic density and flexibility. Requests to channel more trains in closer sequence on the same network infrastructure and the interweaving of long distance and regional passenger traffic with cargo transportation creates new challenges for control and protection systems, and for train drivers and other involved personnel. On-board systems for driver assistance and systems enabling automated train operation are elements helping to respond to these challenges.

Funded by the German Federal Ministry of Education and Research (BMBF) (Grant Number 01IS22029C), within the scope of Project 'Certifiable machine learning based controls for safety-critical applications (CertML)' as part of the program 'KI4KMU - Research, Development and Use of Artificial Intelligence Methods in SMEs'.

Such systems shall either support the driver in their key operative duties, or even replace the driver and fully assume these tasks. One of the key capabilities required for such systems is the safe and reliable detection of obstacles on the train path. The obstacle detection performance required must be at a level similar or better compared to that of human train drivers. Consequently, the systems must be able to cover a range of several hundred meters distance (direct line of sight), operating conditions that include different light conditions such as shadows, rain and fog, backlight, and dawn. Safety integrity levels (SIL) for the covered functions can be derived through the analysis of use cases and associated hazards.

Railway track topology includes switches that connect individual railway tracks (either merging or splitting), and crossover points where individual tracks cross each other. In general, obstacles may be detected on any of the tracks in front of a vehicle. Taking all these into account would potentially lead to many "false positive" detection events, leading to unnecessary alarm or braking reactions and rendering the system unusable. Instead, the system should be able to understand which path the vehicle will actually take considering switch positions and crossings (the "ego track"), and to take only those obstacles into account that are situated on the ego track.

In principle, ego track detection could be based on a combination of train localization information, track topology data, time table and route planning information. Yet, each of these information elements may be either not accurate enough or not up-to-date. E.g., track-exact localization by GNSS (Global Navigation Satellite System) is extremely difficult to realize and infrastructure-based localization means are not available everywhere. Therefore, ego track detection must be either completely based on or at least substantiated by on-board means.

In order to perform both, ego path detection and obstacle detection, sensors are required that observe the railway infrastructure in front of the train. On-board perception systems usually use combinations of daylight and/or infrared cameras,

and sometimes additional detection means such as LiDAR or radar sensors. Based on the acquired sensor information, the ego path is identified taking into account the sensed track topology (potentially including all visible railway tracks), switches and crossings, switch positions, and railway signals. Such ego track then may be considered as region-of-interest that may be subsequently scanned for obstacles or other anomalies.

In this paper we present an approach for ego track detection based on combining deep neural networks (DNNs) for track detection and switch detection, thereby implementing a two-step approach involving plausibility checks that enhance its robustness considering complex track topologies. In the following sections, we first present related work and explain the need and the challenge of robustly identifying the ego track as a precondition for safe obstacle detection. In the next section, we explain our approach that is based on combining two specialized neural networks to detect tracks and switches, and to combine the detection results to properly identify the ego track. Further we present experimental results for the different approach variants implemented and provide our conclusions.

II. Related Work

Compared to the automotive sector, less research has been carried out in the field of advanced driver support systems and autonomous driving in the railway sector. Yet, during the last years, the subject has started to attract a lot of attention. Railway operators invest into these topics, state funding is being made available, and several large research consortia have been established like "Railenium" in France and "Digitale Schiene" in Germany. Momentum has been established that led to a considerable number of approaches and initial solutions in this area.

Several studies that have examined the importance and necessity of autonomous trains for our society, as well as the related risks and challenges, e.g. Trentesaux et al. [1]. Hyde et al. [2] and others worked out that obstacle detection is necessary for trains with a higher grade of automation (GoA). Since in those systems the safety-critical monitoring function of the driver is replaced by a software control system, those systems need to fulfill safety regulations and requirements. Safety regulations for automated driving of metros do exist (EN 62267), whilst the similar regulative framework for main-line railway – which is a much more diverse and challenging environment compared to metros – is still under development. Initial analyses for safety requirements are already available. One such analysis performed for the German railway authority focusing on ATO at GoA level 3 determined required safety integrity levels of SIL0 to SIL2 depending on the task taken over by the automated system, with SIL3 required in exceptional cases [3].

For railway obstacle detection, both classical computer vision (CV) methods and approaches based on machine learning (ML) have been presented in the literature. Ristić-Durrant et al [4] have carried out a comparative literature analysis on the different approaches. This work and other more recent work [5] indicate that ML-based methods are better suited for obstacle detection tasks due to their robustness regarding complex scenes and diverse operating conditions. On the other hand, traditional approaches have advantages with respect to the certifiability for safety-critical applications such as automated train operation.

In [6]–[10], neural networks are proposed for track detection as a first task in an obstacle detection pipeline. Semantic segmentation is used to identify rails or complete tracks, by assigning each pixel in the image to a semantic class.

Many published obstacle detection approaches do not explicitly aim to identify the ego track, but rather include all visible tracks in front of the train as region of interest for obstacle recognition [11], or compute all possible paths starting from the track immediately in front of the vehicle (i.e., excluding all those tracks without a direct connection to the current train position) [12]. As explained earlier, this may lead to many "false positive" identifications of obstacles that are truly obstacles, but situated on tracks that will actually not be used by the train.

To identify the ego track in the set of all visible tracks, switches and switch positions have to be identified. Jahan et al. [13], e.g., present an object recognition network that can recognize switches in railroad scenes, including their status.

Identifying the ego track with a single neural network, extracting the characteristic features for rails and tracks as well as the features for switches and switch position information is extremely challenging. Our own experiments in this direction indicated that the approach works for simple scenarios, but has difficulties to properly identify the ego track in more complex rail topologies. Therefore, we propose a different approach in this paper, combining two specialized neural networks for the identification of tracks and the detection of switches and switch positions (see section III-A). However, a paper recently published by Laurent [14] shows very promising results also for a single-network ego track detection.

A major challenge for machine learning approaches in the railroad sector is the availability of annotated training and test data, as underlined by [4] and many other publications in this field. Very few datasets in this field are publicly available. One of the most widely used public datasets is the Railsem19 dataset with 8500 annotated scenes [15]. Other, rather novel ones are RailSet from the "Railenium" context with 6600 images [16], and the Open Sensor Data for Rail dataset, published by the "Digitale Schiene" initiative and containing approx. 1500 multi-sensor frames [17].

III. Combined Ego Track Detection Approach

A. Approach

As explained in section II, safe and reliable obstacle detection requires a robust approach for the detection of the ego track. Detecting the ego track with a single network approach is very challenging and potentially will not achieve the required detection performance for complex scenes. Therefore, in this paper we propose to split this task into several subtasks. The idea behind this is that each of the subtasks

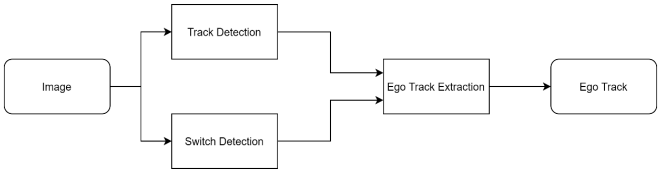


Fig. 1: Combined Architecture for robust on-board ego track detection. Blocks with rounded corners represent input and output data. Rectangular blocks represent architectural elements.

can be optimally performed with a corresponding specialized neural network. For detecting the tracks, the widely used and practice-proven approach of treating rail detection as a semantic segmentation task is retained. This approach has the advantage that the image areas in the image that belong to the track are provided as direct result of such a recognition. These image areas are also the areas that need to be scanned for potential obstacles. In typical images containing several railway tracks, the ego track cannot easily be identified. Only the operational context of the moving train determines which of the contained track segments represents the ego track. Therefore, instead of detecting the ego track, it is convenient to let the segmentation network detect all tracks contained in the image. Thus, rail detection is a sub-task of ego track detection.

In order to be able to decide which of the identified tracks is the ego track and which tracks are neighboring tracks, it is necessary to recognize and understand the switches themselves and their position. By the position of a track we mean the position of the switch blade, which determines the direction in which the train will travel at a switch. The second sub-task of ego track recognition is therefore switch recognition. A switch can be identified as an object in the image by its characteristic structure. In this paper, we propose to use a neural network for object recognition for this sub-task.

Finally, the third sub-task is the extraction of the ego track by combining the previously generated track and switch recognition results. In order to better meet existing safety requirements and objectives, it is advisable to program this task based on rules and not to solve it with a neural network. This allows for better testability and thus verifiability of the system's safety properties. The resulting architecture of the proposed combined system for robust on-board ego track detection is shown in figure 1.

B. Custom Ego Track Detection Dataset

For our work, we used a specifically created dataset that comprises track data, differentiating ego track and other tracks, and switch data including switch position data. The images of the dataset are frames from the "minute by minute" documentary from the Norwegian Broadcast Corporation, a video from the driver's perspective of the railway between Trondheim and Bodo [18]. An example of a annotated images is displayed in igure 2, containing track labels and switch labels.

Track labels are segmentation masks which use different labels

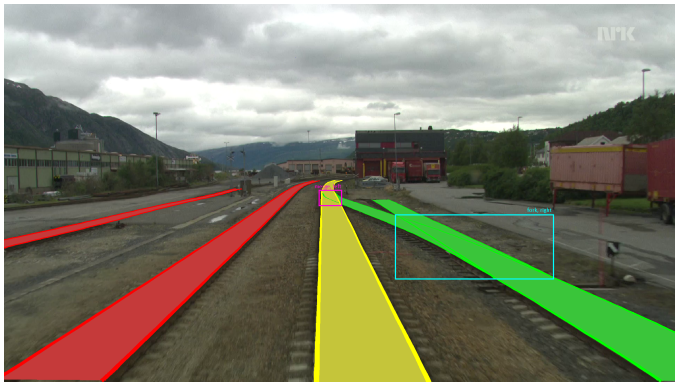


Fig. 2: Labelled image. The yellow track is the ego track, red and green tracks are left and right neighbor tracks. The blue box is a fork label, the pink box is a merge label.

for ego track, left neighbor track(s) and right neighbor track(s). Furthermore, rails and trackbed have different labels. These labels allow for a variety of training scenarios: a neural network can be trained on the ego track only or include neighbor tracks as well while taking into account only the rails or the whole track.

Switch labels are bounding boxes and also provide information on the type of switch, merge or fork, and the relative direction: left, right or unknown. The unknown direction is used for obscured or far off switches, where the direction cannot be determined.

There are 6802 images with labelled tracks and 2334 images with labelled switches. The datasets with labelled tracks and switches have an intersection of 424 images where both, tracks and switches, are labelled.

To increase the number of images for our experiments, we additionally used the RailSem19 dataset [15] by adapting the available labels to our labelling scheme. The resulting combined dataset contains 15302 images.

C. Neural Networks

a) Rail Detection: We used a MobileNet-SegNet architecture – which had shown good performance in our earlier work – as DNN for rail detection It consists of a backbone network for feature extraction (the encoder) and a decoder network. The encoder part uses a MobileNet architecture for extracting features from the input image. The architecture was described in a paper by Howard et al. [19] and was developed for computationally efficient image processing in mobile applications. A SegNet architecture, as described by Badrinarayanan et al. [20], was used as decoder. Based on the extracted features, this decoder generates a segmentation mask, which marks the detected track area. For the implementation of the network we used the code from Gupta [21].

The described network was trained using stochastic gradient descent with batch size 8 and learning rate 0.01 for 100 epochs. For the training process we used the combined dataset of 15302 images described in section III-B which we split into training (70%), validation (15%), and testing (15%) sets (the RailSem19 images were used only in the training set).

We trained the network twice. The first training was done for direct (single-shot) ego track detection. The network was trained with the images of the training dataset and masks marking only the ego track in the image. The resulting DNN is referred to as MS-ego in the following. For the second training, the same images were used, but with masks that marked all tracks in the image. This trained DNN is referred to as MS-all in the remainder of the paper. In this way, we obtained one network that tries to detect the ego track directly on the image and one that tries to detect all the tracks in the image.
Those trained neural networks predict the rails, i.e. the track area, as a segmentation mask. Such a segmentation mask is a gray-scale image representing the confidence of the DNN for every pixel, that it belongs to the track area, which means black pixels in this image represent background pixels (0% track) and white pixels (pixel value 255) represent confident track pixels (100% track). All pixel values in between represent a respective track confidence. To obtain a binary mask, which is required for the ego track extraction algorithms, the segmentation masks need to be preprocessed. In our approach this included thresholding with threshold 0.5, morphological closing with kernel size (10,10) and filtering out small contours with size less than 200 pixels, as those areas are too small to represent correct track areas.

b) Switch Detection: To detect the switches, we used a DNN based on the YOLO architecture, which is a well known architecture for object detection tasks. We used YOLOv8 [22] for object detection. YOLOv8 gives a choice of several different model sizes (n,s,m,l,x). For our task we used the largest (i.e., x) of the available models to get more accurate results.
For training the YOLOv8 switch detection DNN, we used our previously described switch dataset (see section III-B, split into training (70%), validation (15%) and testing (15%) set. We trained the network to recognize the type of the switch (fork, merge) and also the direction it is set for (left, right, unknown), resulting in a total of 6 different classes.
To improve the training results, we tuned the training parameters using the built-in tuning algorithm from the ultralytics package. This algorithm automatically mutates the parameters and tests them to analyse the fitness of the model. This tuning ran for 300 iterations of 10 epochs each.
We used the *patience* parameter for the final training. This parameter allows to set the number of epochs after which the training will be stopped if there is no significant change in the training process. For training of the YOLO network, we used batch size 20 and trained for 600 epochs.
The trained YOLO switch detection DNN predicts switches in images as a bounding box marking the switch location and a respective class of the switch describing its attributes.

D. Ego Track Extraction

The task of ego track extraction is to extract the ego track, i.e., the track that the train will follow, from the prediction results of the rail and switch detection. For our work, we use the following characteristics of the ego track:

Fig. 3: Example image with track segments (colored lines with marked start and endpoint) and ego track (white overlay).

The ego track is always a single track. Consequently, its mask contains only one contour. For an on-board ego track detection system with forward-looking sensors, we can assume that in case multiple track contours are present in the image, the ego track contour is always the closest one to the image center at the bottom of the image. This contour covers the entire area of the track to be traveled. In our approach, the ego track includes both the rails and the track bed. At switches, the switch position determines the further course of the ego track. If a switch position is unknown or if the switch is set in such a way that the train cannot cross it, the safely detected ego track ends at this switch. Otherwise, the ego track continues beyond the switch for the rail section indicated by the switch position.
The ego track consists of an arbitrary number of track segments, but at least one. In the following, a track segment refers to a continuous track section without branches. Such a segment ends at a switch, i.e. a branching point, and one or two new segments begin depending on the type of switch. Figure 3 illustrates these terms in an example picture.

Two different approaches for ego track detection where implemented during our work. Both algorithms are explained in detail in the following and compared to each other there after.

a) Early Fusion Approach: The early fusion approach aims to extract the ego track directly from the binary segmentation mask generated by preprocessing of the neural network output. This approach is based on convexity defects of the segmentation masks, so only the contours of the segmentation mask are analyzed. The following steps are executed:

Step 1 - Determine Contour Containing the Ego Track: Neighbor tracks without a connection to the ego track may lead to multiple contours in the image. As described above, the contour containing the ego track will be the one closest to center at the bottom of the image. Therefore each contour close to the bottom of the image is analyzed to determine the left and right bottom points. The mean of the left and right bottom points is assumed to be the center bottom point

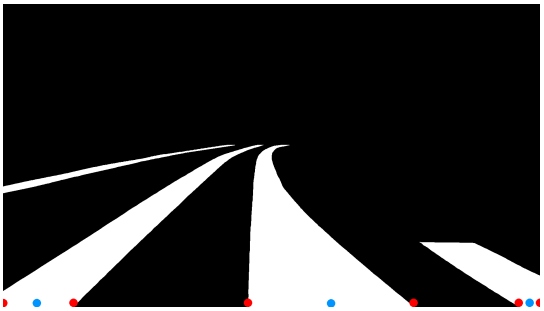


Fig. 4: Early fusion: Multiple track segmentation mask with detected bottom points (center bottom point – blue, left / right bottom points – red).

4. The contour with the center bottom point closest to the image contains the ego track.

Step 2 - Find Switches on Ego Track Contour: To find the switches that are on the ego track contour, the intersection of the ego track contour and the bounding box of each switch is calculated. Only if the area of the intersection exceeds a certain value, the switch is assumed to be on the ego track contour. If there are no switches on the ego track contour, the algorithm stops as area of the ego track contour contains only the ego track. Otherwise the ego track contour is processed further to separate forking and merging neighbor tracks.

Step 3 - Find Frog Point of the Switch: Usually the contours of tracks with switches have a significantly deep convexity defect with the frog point at the farthest point. Curved tracks also show a convexity defect on the inner side of the curve, but such defect is relatively shallow compared to its extent. Therefore only convexity defects with a large depth and a small extent are considered as frog points. In an ideal case, were the tracks fork uniformly to the left and right, the frog point would be at the farthest point of the defect. As this is not always the case, points with a high curvature within the convexity defect are added to the possible candidates for the frog point. From these candidates, the point is selected as the frog point of the switch, which is closest to

- the top line of the bounding box of the switch, if the switch is a fork.
- the bottom line of the bounding box of the switch, if the switch is a merge.

Step 4 - Split Contour at the Switch: To split the contour at the switch, the other end of the switch – more precisely a point opposite to the frog point – must be determined. The other end of a switch is assumed to be at the bottom line of the fork bounding box or the top line of the merge bounding box, respectively. Along this line the opposite point is on the right or left side of the track, depending on the setting of the switch. The opposite point of the switch is determined by following the contour until the level of the bottom line of the fork bounding box or top line of the merge bounding box is reached while ensuring that the line between the frog point an the opposite point is completely within the track contour.

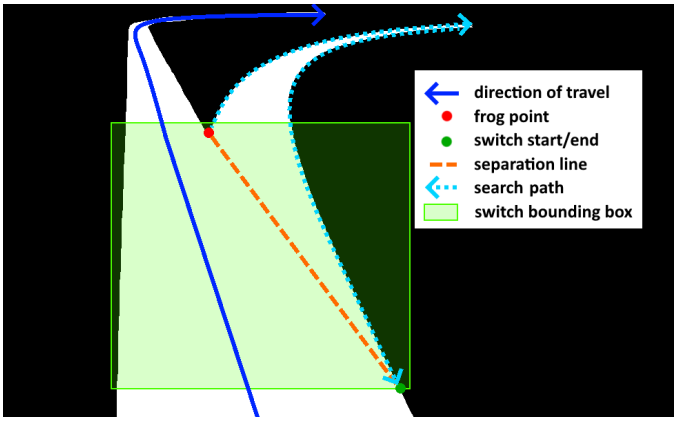


Fig. 5: Early fusion: Contour splitting at a left directed fork.

The frog point and opposite point are marked with red and green dots respectively in Figure 5. The contour is split along a separation line between the frog point and the opposite point.

These steps are repeated until all neighbor tracks are separated from the ego track and the algorithm terminates in step 2 as no more switches can be found on the ego track contour.

b) Late Fusion Approach: In contrast to the early fusion approach, the ego track is not determined directly from the segmentation mask output by the neural network in late fusion ego track extraction. Instead, the results of the rail detection are first preprocessed to describe the rail areas as compactly as possible.
A track is a very simple structure. Its course is determined exclusively by the two rails. These always have a constant distance in the real world. This makes it possible to describe the track along a single centerline. This centerline runs parallel to the two rails in real world coordinates and has the same distance to both. In an image, the centerline is not parallel to the rails anymore due to the perspective of the camera, but still always is in the middle between both rails. The idea of the late fusion approach is therefore to determine the ego track based on the centerlines of the detected track areas. The procedure for this approach is described below.

Step 1 - Find Centerlines: The first step in this process is to calculate the centerlines of the segmentation mask. For each row of the binary images, resulting from the segmentation mask preprocessing, contiguous sections of white pixels are determined, and the mean value between the first and last pixel coordinate of the area is calculated for each of the sections.
This calculation is performed separately for each contour in the binary image in order to obtain the centerline for each track segment individually. If the contour contains a switch, a Y-shape can be recognized. In this case, the individual track segments can be separated from each other by splitting the contour at the point where the number of center points

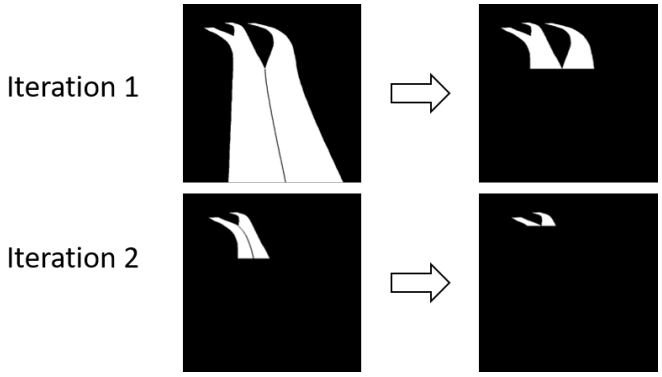


Fig. 6: Late fusion: Splitting of segmentation mask into individual track segments.

changes in between adjacent image rows, i.e. the switch frog (see Figure 6). If a contour contains multiple switches, this procedure is applied recursively until all track segments are separated from each other.

Step 2 -Filter Centerlines: Once the centerlines have been calculated, they need to be filtered. Due to fuzzy edges at to top and the bottom in the segmentation masks, very short centerlines can appear at the beginning and ending of tracks, as well as outside of the track area. Those erroneous centerline fragments are filtered out. Furthermore, centerlines are merged if the upper end of a segment is located very close to the lower end of another segment. This is done in order to describe each track segment with only one centerline, if it was split into multiple individual centerlines due to irregularities in the contour edges.

Step 3 - Identify Switches in the Rail Detection Results: Using the centerlines found in this way, it is possible to identify switches by its characteristic centerline structure independently of the DNN switch detection results. A switch is a point, where three centerlines are starting or ending close to each other. Two centerlines always end at exactly the same y-level, and a third begins in the adjacent pixel row (see Figure 7).
The exact arrangement of the centerline ends even allows a distinction between merging and forking switches. However, the exact setting of the switch cannot be determined using the centerlines. This is one of the reasons why the object detection network is used for switch detection.

Step 4 - Check Plausibility of switches: Since there are now two switch detection results after step 3, it is obvious to check their plausibility against each other. Both the switch detection network and the rail detection network results provide the position of the switch in the image as well as the switch type. Therefore, both can be compared with each other. For this plausibility check, a certain tolerance zone has to be defined for the switch position based on the detected bounding box from the object detection, because depending on the position of the switch in the image, the centerline ends are not always

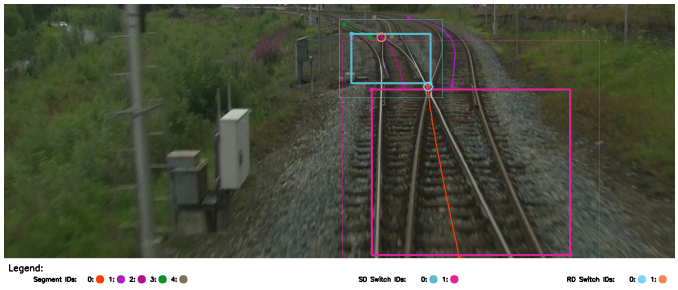


Fig. 7: Late fusion: Identification of switches in rail detection results. Colored lines mark detected centerlines. Solid circles mark segments start and end points. Circles around start/end points mark a switch detected from rail detection result. Bold rectangles mark switch detection bounding box results. Narrow rectangles mark tolerance area around switch bounding box. The legend below the image shows Element IDs for the respective colors.

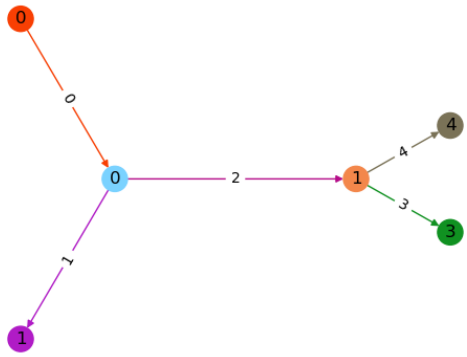


Fig. 8: Late fusion: Track network graph generated from rail detection and switch detection results for the example image in fig. 7 .

within the bounding box, but in the immediate vicinity. If both the position and the type of the switch match, it can be confidently assumed that the switch is correct. The setting of the switch can then be taken from the switch detection result.

Step 5 - Create Track Network Graph: Once the rail segments and switches have been located in the image, the tracks contained in the image can now be represented in a network. This is done using a directed graph. The rail segments are represented as edges and the switches as nodes of the graph. In addition, start and end nodes are created for segments that do not start or end at a switch. This graph (see example in Figure 8) represents the information about the connection of the track segments.

Step 6 - Find Ego Track Segments: Once a graph of the track network has been created from the detection results, it can be used to determine the segments belonging to the ego track. Using the assumptions on the ego track position introduced initially in section III-D, the start node that matches these assumptions best in the graph can be

determined. Starting from this node, the next node of the directed graph is searched recursively until an end of the track is reached. Such an end can either be an end node or a switch where the track ends. With this procedure, a list of segments and nodes belonging to the ego track can be created

Step 7 - Determine Spline Points: As can be seen in the figures for calculating the centerlines (see figures 6 and 7), the centerline found in the switch area does not describe the correct course of the two track segments entering or leaving the switch. For this reason, the centerlines of the corresponding segments cannot simply be output to create the ego track centerline. The incorrect centerline is always located in the section of the switch area that has only one centerline. This area extends from the switch blade to the switch frog. Interpolation is required to reconstruct the correct track centerline there. The entire switch area is defined by the bounding box that is provided by the switch detection results. However, in order to keep the interpolated area as small as possible and thus the interpolation as precise as possible, not the entire area in the bounding box is interpolated, but only the area of the single segment from the edge of the bounding box to the center of the switch. For all other areas, the centerlines of the segments are assumed to be correct.
For the interpolation, interpolation points are selected at regular intervals on the y-level from these correct track segments. For the example image, these interpolation points are shown in figure 9. This figure also shows the gap between the interpolation points in the area of the switch.

Step 8 - Interpolate Ego Track Centerline: Finally, spline interpolation is used to generate the ego track centerline. With this method, a quadratic function is adapted to the interpolation points and can then also be evaluated for the switch areas in order to calculate the correct centerline of a track. Figure 9 shows the calculated centerline of the ego track for the example image.

Step 9 - Generate track mask: Based on the centerline, the mask describing the ego track can be reconstructed using the track width. In the real world, this track width is constant. However, since the camera has a perspective distortion, the track width decreases with increasing distance from the camera to the back of the image. However, an examination of the track width for different images has shown that it decreases linearly over the course of the image. The corresponding linear function is used to calculate a corresponding track width for each row in the image. Figure 10 show the reconstructed track area for the example image.

IV. Results

This section presents the experimental results of our investigation. We compared a direct (single-shot) ego track detection approach with our proposed combined approach consisting of rail and switch detection. For both approaches,

Fig. 9: Late fusion: Interpolated centerline using spline interpolation shown as red line. For reference spline points used for the interpolation are shown as well as white dots.

Fig. 10: Late fusion: Generated ego track area. Yellow area marks track area, red area marks safety area around track (according to [23] annex 1 to §9). Blue line marks left ego track edge, red line marks right ego track edge and green line marks ego track centerline.

a neural network was trained with the presented MobileNet-SegNet architecture (see section III-C) for rail detection. For direct ego track detection, this network was trained with masks containing only the ego track for rail detection. This approach is referred to as *singleshot* in the following, utilizing the MS-ego DNN from section III-C. For the combined approach, the same network architecture was used, but trained on masks containing all tracks of the scene (MS-all from section III-C). For the combined approach, both proposed ego track extraction algorithms – early and late fusion described in section III-D – are tested separately. Table I provides an overview of the individual components of the approaches described, as well as the test labels which we use in the following.

For the evaluation of the these approaches, the same test set of images is used for all variants. This test set consists of 2296 images of the dataset described in section III-B. Since

TABLE I: Approaches to be compared and their characteristics and labels

Approach	**Rail Detection DNN**	**Switch Detection DNN**	**Ego Track Extraction**
singleshot	MS-ego	-	-
combined-EF	MS-all	YOLOv8	Early Fusion (EF)
combined-LF	MS-all	YOLOv8	Late Fusion (LF)

TABLE II: Fusion algorithm IoU results on ground truth data.

Image set	Early Fusion	Late Fusion
all scenes	**0.972**	0.943
switch scenes	0.896	**0.921**

the proposed combined approach for ego track detection aims to provide an improvement over the state of the art especially for complex scenes including switches, such scenes are of particular interest for the evaluation. In the used dataset, there are significantly more images without switches than with switches. As a consequence, also the test set contains rather few images with switches (only 143). This fact later was identified as problematic (see the result discussion below).
In order to evaluate and compare the different approaches investigated in this paper, we performed primarily a quantitative analysis using the Intersection over Union (IoU) metric, complemented by a qualitative analysis for selected images. The IoU metric calculates the ratio of correctly detected areas of a segmentation mask to incorrectly detected areas. The prediction is always compared to the ground truth mask. With GT as the segmentation ground truth mask of the track area and $Pred$ as the predicted mask of the track area, the IoU is calculated using the following equation:

$$IoU = \frac{Area\,of\,Intersection}{Area\,of\,Union} = \frac{|GT \cap Pred|}{|GT \cup Pred|} \quad (1)$$

First, the two fusion algorithms were tested on the ground truth data. This data can be considered as ideal recognition results and should therefore serve to demonstrate how the algorithms work in the ideal case. Both algorithms perform well on the ground truth data. The results can be seen in table II.

Looking at all images of the test dataset, i.e. the first row of the table II, we can see that the early fusion approach has a significantly higher average IoU, indicating a better ego track detection performance. However, the test dataset contains considerably more images without switches than with switches. On such scenes, the basic functional principle of this approach performs better than the late fusion approach. In scenes without switches, the early fusion approach only needs to select the correct contour belonging to the ego track. This usually works very reliably, as the test results show. However, the late fusion approach also shows good overall performance in ego-track detection across all test data. Reconstruction of the ego track area from the centerline seems to be generally feasible. The lower IoU value of the late fusion approach is mainly due to the use of an approximated track width to generate the mask from the centerline. As a result, the predictions are not as accurate as those of the early fusion approach.
However, the second row of the table shows that the late fusion approach performs better when only scenes with switches are considered. Here, the IoU value decreases slightly, but not as much as with the early fusion approach. The detection of the convexity defect for the mask splitting seems to be difficult in some scenes, but the algorithm generally allows a correct

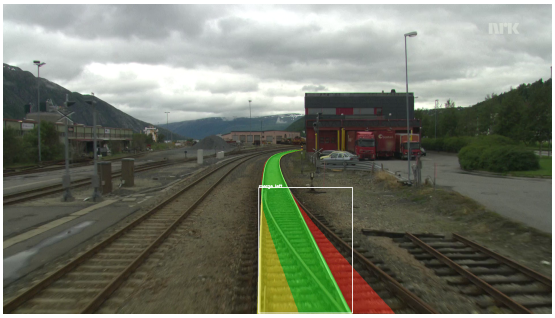


Fig. 11: Late fusion: Incorrect ego track calculation due to (a) missing spline points (b) partially visible merging switch. Green areas=True Positive, Yellow areas=False Negative, Red areas=False Positive. White rectangles represent results of the switch detection network.

(a) (b)

Fig. 12: Positive Examples of ego track detection for (a) Early Fusion algorithm (b) Late Fusion algorithm. color meanings as in Figure 11.

detection of the ego track, as indicated by an IoU of almost 0.9. Also for the late fusion approach, the switch scene IoU is lower than that for all scenes. This can be explained by scenes such as the one shown in figure 11. If a merging switch is only partially shown at the bottom of the image because the train has already entered the switch area, the rail detection results will produce a contour that is wider than a single track. In the late fusion approach, the centerline is found and the ego-track mask is generated based on the approximated track width. In this case, however, the centerline no longer runs along the correct course of one of the two track segments leading into the switch, but between them. Since these problem areas are always located at the lower edge of the image, the resulting incorrect areas are quite large and therefore have a major impact on the IoU. As indicated in section V, we intend to improve the late fusion algorithm for handling these specific scenes.

Still, the average IoUs of both algorithms are quite high for the ground truth data, i.e., assuming optimal input from the network predictions. With these positive initial validation results, we consider both algorithms as being able to detect the ego track from a fusion of the detection results of the rail and switch detection and to generate a corresponding mask. Figure 12 shows positive examples for both algorithms for reference.

In the following, we tested the algorithms using the actual recognition results of the two neural networks.
Unexpectedly, the results of the generated ego track

(a) (b)

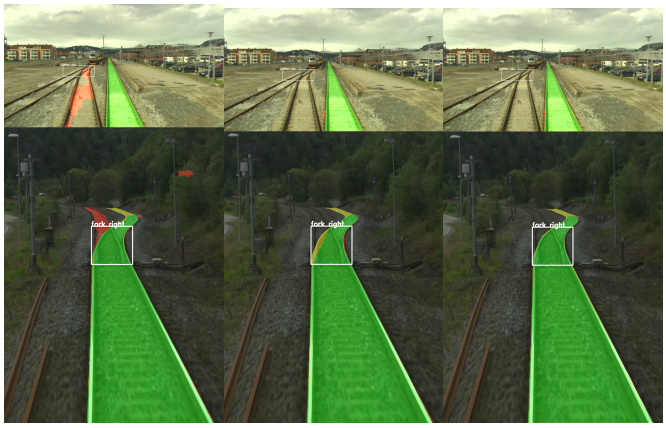
Fig. 13: Negative Examples of MobileNet-SegNet all-tracks detection results. color meanings as in Figure 11.

predictions indicate that the singleshot approach, which employs a single network to detect the ego track, yields the most favorable outcomes (see table III). The singleshot approach exhibits a remarkably high IoU of approximately 0.94 across the entire test set and across all subsets examined. Additionally, the network's predictions are remarkably consistent, as a comparable average IoU value is achieved for scenes with and without switches. Investigating the reasons for this unexpected result, we found three problematic areas regarding our experiments.

Firstly, the MS-all network results shown in table III demonstrate that – in comparison to direct ego track detection – the task of detecting all tracks in the image presents a considerable challenge for the MobileNet-SegNet DNN. The image areas to be detected are larger in this case, and as the neighboring tracks are more likely to be at the edges of the image, there is also a greater distortion caused by the camera. Especially high-complexity scenes are challenging for the MS-all network. For these scenes, it occasionally produces incomplete or inaccurate predictions, including holes in the predicted masks and incorrectly recognized contours outside the actual track area (see figure 13). As both ego track extraction algorithms are predominantly geometry-based, their IoU value is considerably impacted by a low IoU of the MS-all network due to those irregularities.
Secondly, the switch detection DNN performs less well than expected. In the entire test set, there are 143 images with switches that were labeled in the ground truth data. Of these, only 126 images (i.e., 88%) were properly identified by the switch detection network. The performance gap may be attributed to the rather small size of the training data set. As the combined approaches strongly depend on the switch detection correctly identifying switches and their direction, the performance gap likely has a strong impact on the results.
Thirdly, it is important to note that the excellent recognition results achieved by the singleshot approach may be to some extent attributable to the fact that the test set predominantly comprises relatively simple scenes. The number of switches in the scenes is small, with the majority of the ego track switches leading to a straight continuation of the ego track. It is therefore possible that the MS-ego network may have learned this fact as a result of overfitting. This is exemplified by a qualitative analysis of switch scenes in which the ego

TABLE III: Mean IoU results of the fusion algorithms compared to the ego track detection network. GT switch scenes are all scenes with labelled switches. Pred. switch scenes are all scenes with predicted switches from switch detection network. The results of the MS-all DNN are given for reference in the last column, since the combined approaches are based on these results.

Image set	**singleshot**	**combined-EF**	**combined-LF**	*MS-all*
all scenes	**0.945**	0.939	0.916	0.937
GT switch scenes	**0.941**	0.859	0.833	0.907
Pred. switch scenes	**0.943**	0.867	0.846	0.912



(a) (b) (c)

Fig. 14: Examples of ego track detection errors of the singleshot approach. a) shows singleshot output, b) shows combined-EF output and c) shows combined-LF output respecitvely for the same input images. color meanings as in Figure 11.

track takes a turn at a switch (see figure 14). In the second row of this figure, it can be observed that the singleshot approach utilizing the MS-ego network selects the incorrect track segment that emerges from the switch. In the first row, the singleshot approach fails to detect the ego track shortly after a merging switch. In contrast, the combined approaches manage to detect the ego track correctly in both scenes. It can therefore be assumed that the detection performance of the singleshot approach would decrease on more complex scenes than those primarily included in the test data set.

Despite these identified problem areas, both combined approaches demonstrate a performance on the actual DNN recognition results that is not significantly below that of the segmentation ground truth data used for initial algorithm validation. We found the combined ego track extraction algorithms being capable of detecting the ego track also on these non-optimal segmentation masks. On the actual DNN recognition results, the combined-EF approach is slightly superior to the combined-LF approach as indicated by the

slightly higher IoU. From table III it is also evident, that the detection results of the switch detection play a significant role in both approaches.

V. Conclusion

In this paper, we have proposed two algorithms that are able to identify the ego track by combining the neural network detection results for track detection and switch detection.

Our current evaluation of detection performance of these two approaches in comparison to a direct approach utilizing a single neural network indicates that the singleshot approach achieves good results in the analysis of low-complexity scenes. We assume that for higher complexity scenes, the combined approaches presented in this paper are more robust. Whilst qualitative analysis of typical complex scenes supports this assumption, we were not yet able to quantitatively substantiate this assumption due to limitations of the dataset (being still too small and containing primarily low-complexity scenes).

As our analysis has shown, the performance of the two fusion algorithms depends very much on the quality of the recognition results of both DNN, for track segmentation and for switch detection. Therefore, the training of the neural networks must be improved in the future. Our main task in this area will be to develop a larger image database for both training and evaluation. In particular, more images with labeled tracks (ego track and all tracks) and more complex scenes need to be obtained. Also the fusion algorithms must be further amended to counteract their susceptibility to error. With these future improvements, we expect the combined algorithms to perform at a similar or better level than the singleshot approach.

Since – in light of the safety requirements for autonomous train operation – our ultimate aim is a provably safe approach towards ego track detection, we consider the use of an explicitly defined algorithm for detection result combination as an advantage, as such explicit algorithm can be developed according to traditional software safety regulations such as EN 50657. In addition, the combined approaches – specifically the combined-LF approach – offer more possibilities to check the plausibility of the predictions during operation. Some of these possibilities have been presented in this paper, others we will investigate in more detail in the future. Thus, in our view, the presented combined approaches to ego track detection are more suitable for use in safety-critical applications supporting autonomous train operation than singleshot approaches, provided that comparable ego track detection performance can be achieved.

References

[1] D. Trentesaux, R. Dahyot, A. Ouedraogo, D. Arenas, S. Lefebvre, W. Schon, B. Lussier, and H. Cheritel, "The autonomous train," in *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. IEEE, 62018, pp. 514–520.

[2] P. Hyde, C. Ulianov, J. Liu, M. Banic, M. Simonovic, and D. Ristic-Durrant, "Use cases for obstacle detection and track intrusion detection systems in the context of new generation of railway traffic management systems," *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, vol. 236, no. 2, pp. 149–158, 2022.

[3] J. Braband, B. Evers, M. Kinas, L. Lindner, D. Mihailescu-Stoica, F. Rexin, F. Adebahr, B. Milius, and H. Schäbe, "Risikoakzeptanzkriterien für das automatisierte fahren auf der schiene."

[4] D. Ristić-Durrant, M. Franke, and K. Michels, "A review of vision-based on-board obstacle detection and distance estimation in railways," *Sensors (Basel, Switzerland)*, vol. 21, no. 10, 2021.

[5] M. Ziegler, V. Mhasawade, M. Köppel, P. Neumaier, and V. Eiselein, "A comprehensive framework for evaluating vision-based on-board rail track detection," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–8.

[6] H. Li, Q. Zhang, D. Zhao, and Y. Chen, "Railnet: An information aggregation network for rail track segmentation," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 72020, pp. 1–7.

[7] Z. Tao, S. Ren, Y. Shi, X. Wang, and W. Wang, "Accurate and lightweight railnet for real-time rail line detection," *Electronics*, vol. 10, no. 16, p. 2038, 2021.

[8] 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.

[9] S. Belyaev, I. Popov, V. Shubnikov, P. Popov, E. Boltenkova, and D. Savchuk, "Railroad semantic segmentation on high-resolution images," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 9202020, pp. 1–6.

[10] 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.

[11] S. Khrueakhrai and J. Srinonchat, "Railway track detection based on segnet deep learning," in *TENCON 2023 - 2023 IEEE Region 10 Conference (TENCON)*, 2023, pp. 409–413.

[12] M. Ghorbanalivakili, J. Kang, G. Sohn, D. Beach, and V. Marin, "Tpenet: Track point extraction and association network for rail path proposal generation," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 2023, pp. 1–7.

[13] K. Jahan, J. Niemeijer, N. Kornfeld, and M. Roth, "Deep neural networks for railway switch detection and classification using onboard camera images," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1252021, pp. 01–07.

[14] T. Laurent, "Train ego-path detection on railway tracks using end-to-end deep learning," 2024.

[15] 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, 62019, pp. 1221–1229.

[16] 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.

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

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

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications."

[20] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation."

[21] D. Gupta, "Image segmentation keras: Implementation of segnet, fcn, unet, pspnet and other models in keras," *arXiv preprint arXiv:2307.13215*, 2023.

[22] ultralytics, "Yolo v8." [Online]. Available: https://github.com/ultralytics/ultralytics

[23] Bundesamt für Jusitz, "Eisenbahn-bau- und betriebsordnung: Ebo," 1967.