LineAP: A segment-based average precision metric for visual rail track detection

Annika Bätz and Martin Lauer

Karlsruhe Institute of Technology (KIT),
Institute of Measurement and Control Systems (MRT),
Engler-Bunte-Ring 21, Building 40.32,
76131 Karlsruhe, Germany

Abstract Reliable rail track detection is a prerequisite for safe automated train operation, yet evaluating it requires assessing two distinct aspects of prediction quality: (i) the geometric accuracy of predicted lines, and (ii) the correct instance-level grouping of polylines, e.g. accurate start and end points at railway switches. Existing metrics for polyline detection assess both subtasks simultaneously, losing insight into the error sources of a detection method: partially incomplete but geometrically accurate detections are penalized, while entire rail instances with inaccurate sections are rewarded. We propose LineAP, a segment-based average precision metric that assesses geometric accuracy independently from instance-level correctness by partitioning polyline predictions into fixed-length segments, each evaluated via joint distance and orientation thresholds. A novel track-width-based distance threshold further customizes LineAP to the safety requirements of railway applications, where geometric accuracy remains highly relevant even at large distances from the camera.

Keywords Rail track detection, polyline detection, evaluation metric, average precision, railway perception, lane detection

1 Introduction

Automated train operation requires reliable perception of the train's surroundings. A prerequisite for many downstream tasks – such as ego-path detection, obstacle recognition, scene understanding, and

map-based self-localization – is the precise localization of the railway tracks in camera images.

Railway tracks can be efficiently represented as polylines that mark the left and right rail of each track [1–3] (see Fig. 1). Due to this representation, rail track detection can be formulated as a 2D polyline detection task that is similar to the detection of road lane markings in the automotive domain. As a result, commonly used metrics for rail track detection originate from this domain, such as the metrics from the TuSimple [4] and CULane [5] benchmarks. However, all metrics commonly used for the evaluation of either rail track detection or lane marking detection, share two shortcomings.

First, the metrics treat entire predicted and ground truth instances as the unit of evaluation. As a result, errors such as incomplete or partially correct predictions, and one-to-many or many-to-one correspondences between predictions and ground truth, are not properly accounted for: a prediction is treated entirely as a true positive (TP) or a false positive (FP), regardless of the geometric precision of its individual parts.

Second, existing metrics for 2D polyline detection use fixed distance thresholds in pixel space, and disregard the fact that the same distance error can have different implications depending on the distance of the line object from the camera. As precise rail track detection in the far distance is particularly important for safe train operation, this can lead to misleading evaluation results. A natural alternative would be to evaluate polyline detections in bird's-eye view (BEV) using calibrated real-world coordinates; however, camera calibration is not available for most of the existing rail track detection datasets, such as the ones used in this work, RAIL-BENCH [1] and RailSem19 [2]. Moreover, a BEV projection additionally relies on a flat-world assumption, which not always holds.

To address these issues, we propose LineAP, a segment-based average precision metric that evaluates polyline detections at a finer geometric granularity than existing metrics. Although designed for rail track detection, it can be applied to any polyline prediction task such as road lane marking detection. With the addition of a variable distance threshold that approximates a fixed real-world distance based on the track width, LineAP can be further adapted to the specific requirements of rail track detection.

We demonstrate the benefits of LineAP in a case study on three



Figure 1: Example of rail track annotations in the RAIL-BENCH Rail dataset. Each colored line represents a separate rail instance. Rails of the same track are visualized in the same color. Best viewed in color.

network architectures finetuned on the novel RAIL-BENCH Rail dataset [1], where LineAP yields complementary rankings to the established ChamferAP metric. Together they provide a richer diagnostic than either metric alone.

2 Metrics for Polyline Detection Evaluation

We give a brief, non-exhaustive overview of existing metrics for polyline detection evaluation commonly used in the automotive and railway domains, focusing on those most relevant to our work.

In the TuSimple protocol, ground truth and predicted lines are sampled at uniformly distributed vertical image positions; a ground truth line counts as detected if the horizontal distance to a predicted line stays below a pixel threshold for 85% of these positions. TuSimple Accuracy is the fraction of correctly detected ground truth lines.

For CULane, ground truth and predicted lines are rendered as binary masks (drawn with a fixed line width) and matched via the Hungarian algorithm using pairwise IoU with a fixed threshold of 0.5; the result-

ing TP, FP, and false negative (FN) counts yield the CULane F1 score.

The Chamfer Distance (CD) – the symmetric mean nearest-neighbor distance between uniformly sampled points on two polylines – is a common choice for evaluating polyline predictions within an Average Precision (AP) framework in HD Map Construction [6], which we refer to as ChamferAP; unlike the previous metrics, it is designed for bird's-eye-view (real-world) coordinates. Matching is performed greedily, following the confidence ranking of the predictions and a distance threshold for the CD.

Note that all three metrics operate on the instance level, treating entire predicted and ground truth lines as the unit of evaluation – making it difficult to reward geometrically accurate but incomplete predictions, or to handle one-to-many or many-to-one correspondence errors. In addition, all three use a fixed pixel distance threshold that does not correspond to a consistent real-world distance (ChamferAP included, when applied to 2D image-plane predictions).

3 Track-width-based Distance Threshold

In order to define a variable distance threshold in image space that approximates a fixed real-world distance, we can exploit the fact that the track width is a fixed real-world constant that can be measured – at least approximately – in the image.

We define the track width t_w as the distance between the outside edges of the rail heads as this corresponds to the annotations in the RAIL-BENCH Rail dataset. This width combines the track gauge t_g and the width of the two rail heads t_r:

$$t_w = t_g + 2 \cdot t_r \tag{1}$$

The track gauge t_g is a fixed norm, while the width of the rail heads t_r varies depending on the rail type but can be assumed constant within a single image [7].

To measure the track width in an image, we use a simple heuristic that measures the horizontal distance Δw_k between corresponding rails at uniformly sampled image rows v_k (see Fig. 2). The measurements are then averaged for each image row ($\overline{\Delta w_k}$) and linear regression is applied to obtain a linear function $t_w(v)$ that approximates the track

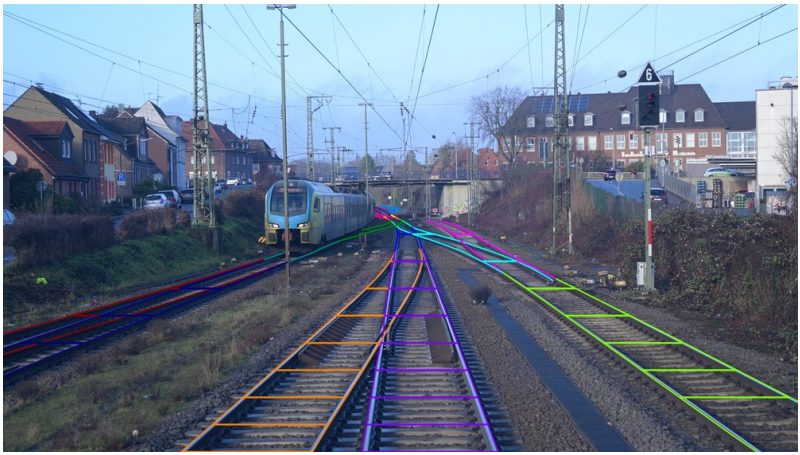


Figure 2: Track width measurements in an image from the RAIL-BENCH Rail dataset. At ten uniformly distributed image rows, the track width is measured as the horizontal distance between the outside edges of the rail heads. Note that the railway sleepers, which are perpendicular to the rails and therefore representative for the track width, are nearly parallel to the horizontal measurement lines. This supports the validity of the horizontal measurement heuristic.

width in image space as a function of the image row (see top right of Fig. 4):

$$t_w(v) = m \cdot v + b \tag{2}$$

Averaged over all images in the RAIL-BENCH Rail dataset, this per-image linear fit achieves a root-mean-square error of 7.6 px against a mean track width of 414.6 px, i.e. a relative error of about 1.8%, supporting its validity.

A variable distance threshold $\Delta d(v)$ can then be defined as a fraction α of the track width in image space:

$$\Delta d(v) = \max(\alpha \cdot t_w(v), \Delta d_{\min}) \tag{3}$$

A lower limit $\Delta d_{\min}$ is applied to avoid overly strict thresholds in the far distance. With the exception of the lower limit, this distance threshold is proportional to the track width and therefore approximates a fixed real-world distance in image space.



Figure 3: Distance threshold in image space: each ground truth polyline is extended by the corresponding distance threshold to both sides. Left: fixed distance threshold of 38 px. Right: variable distance threshold $\Delta d(v)$ of $\alpha = 5\%$ of the track width with a lower limit d_{min} of 5 px.

Figure 3 displays the difference between a fixed pixel threshold and our proposed track-width-based distance threshold $\Delta d(v)$. Note that the distance thresholds are similarly strict in the foreground, but in the background only the track-width-based distance threshold keeps the threshold boundaries close to the ground truth polylines and prevents overlap with neighboring tracks.

Our proposed variable distance threshold $\Delta d(v)$ can be used in any metric that requires a distance threshold in image space. In our experiments, we implement it both in our novel LineAP metric and in our implementation of ChamferAP.

4 LineAP Metric

LineAP builds on the established Average Precision (AP) framework [8]. Its computation involves three consecutive steps: *(A) instance construction*, *(B) instance matching*, and *(C) AP computation*. The latter step is standard and not specific to LineAP, so we focus on the first two steps that define the core of our metric. Figure 4 shows an illustrative toy example.

4.1 Instance Construction

Both ground truth and predicted polylines are partitioned into short segments of a fixed pixel length (the last segment of each line may be

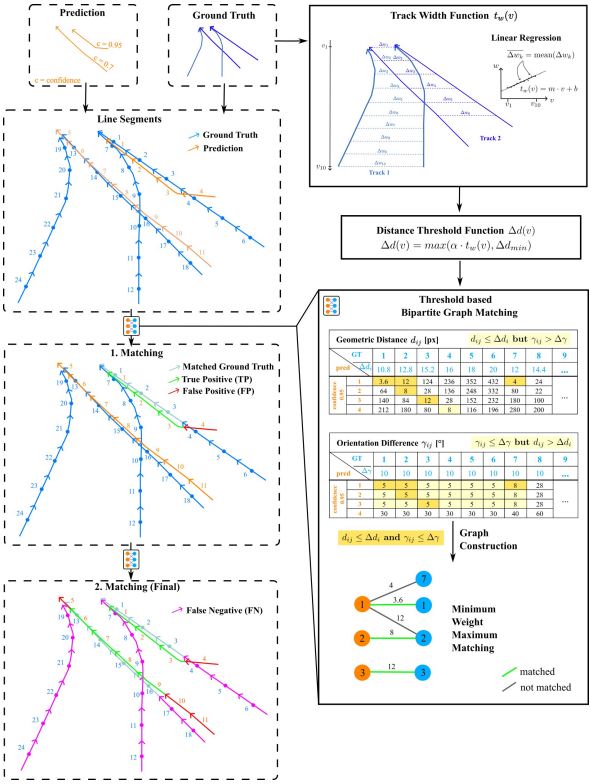


Figure 4: Instance construction and matching in LineAP. Top left: Ground truth (blue) and predicted (orange) polylines are partitioned into short oriented segments. Top right: Construction of the track-width-based distance threshold from the ground truth instances. Lower right: A bipartite graph is constructed based on the variable distance $\Delta d(v)$ and orientation $\Delta\gamma$ thresholds. Minimum-weight maximum matching yields the optimal one-to-one assignment for that confidence level, which is repeated for the next confidence level until all predictions are processed. Best viewed in color.

shorter). Each predicted segment inherits the confidence score of its parent polyline. The resulting ground truth and predicted segments are treated as independent instances that undergo the AP matching process.

4.2 Instance Matching

Matching follows the standard AP protocol: each ground-truth and predicted segment can be matched at most once, and predictions are processed in descending confidence order. For each group of predictions with the same confidence score, we first collect all *valid pairs* of unmatched ground-truth and predicted segments. A pair is valid when two criteria are simultaneously satisfied: (1) the Euclidean distance from the center point of the ground-truth segment to the nearest point on the predicted segment is below the track-width-based distance threshold $\Delta d(v)$, where v is the v-coordinate of the ground truth center point; and (2) the angular difference between the two segment orientations is below another threshold $\Delta\gamma$. This second threshold is fixed across all instances.

Valid pairs are added to a bipartite graph, with nodes representing segments and edges identifying all valid pairs. Minimum-weight maximum matching yields the optimal one-to-one assignment. The process repeats for the next lower confidence group until all predictions are processed. The resulting precision-recall curve is integrated to obtain the AP score (*C: AP computation*).

5 Experiments

5.1 Model Training

We use three different network architectures – PINet [9], YOLinO [10], and CLRNet [11] – to illustrate a range of model capabilities and possible erroneous predictions. All three models are first pretrained on the RailSem19 dataset [2], and then finetuned and evaluated on the novel RAIL-BENCH Rail dataset. We use the predefined train/val/test split of the RAIL-BENCH Rail challenge.

PINet is pretrained and finetuned with a learning rate of 1×10^{-3} for 55 and 35 epochs, respectively. YOLinO is pretrained with a learning rate of 1×10^{-4} for 30 epochs, and then finetuned for 15 epochs with a smaller learning rate of 1×10^{-5}. The raw output of YOLinO, a set of short line segments, is post-processed into the final rail line predictions via greedy graph-based merging of nearby segments. For CLRNet we use a ResNet-18 backbone [12], loss weights of $w_{cls} = 2$, $w_{xytl} = 0.2$,

$w_{LIoU} = 2$, and $w_{seg} = 0.5$, and a non-maximum-suppression threshold of 5. The model is pretrained for 50 epochs with a learning rate of 2×10^{-3}, finetuned on RAIL-BENCH Rail for 40 epochs with a learning rate of 2×10^{-4}.

5.2 Evaluation Settings

In our experiments, we evaluate all models with both LineAP and ChamferAP on the RAIL-BENCH test set. For ChamferAP we keep the general evaluation pipeline, but replace the constant distance threshold with our new track-width-based threshold. For that, we assign to each ground truth line an individual distance threshold using the mean v-coordinate of the ground truth line to compute the track-width-based distance threshold $\Delta d(v)$. The minimal distance threshold is set to 10 px. We use 50 sample points along each predicted and ground truth line to compute the CD.

For LineAP we choose a segment length of 2% of the image width, which ranges between 76 and 109 px, to account for different image resolutions. The orientation threshold is set to 10° and the minimal distance threshold to 5 px. Following standard evaluation protocols, we limit the number of predictions to the 100 with the highest confidence scores for both LineAP and ChamferAP.

6 Results

Table 1 reports the quantitative results. CLRNet achieves the overall highest mAP_{avg}, followed by PINet and YOLinO. Most notably, YOLinO achieves the highest LineAP but the lowest ChamferAP score. As explained in Section 5.1, YOLinO's rail instances are assembled from short line segments via a separate post-processing step, which more often yields incomplete or interrupted lines than PINet or CLRNet – lowering its ChamferAP score. LineAP, however, credits partially correct predictions whenever the predicted segments are sufficiently close to the ground truth and within the orientation threshold, illustrating exactly the diagnostic insight lost when evaluating polyline detections only at the instance level.

Figure 5 shows a qualitative comparison across all three models.

For PINet, only LineAP penalizes the incorrect background predictions. For YOLinO, several incomplete predictions correspond to a single ground truth line and are rewarded only by LineAP. For CLRNet, conversely, two predicted lines correspond to four ground truth lines. ChamferAP still matches both lines to the blue ground truth track in the easy setting, but in the hard setting the CD of the right rail exceeds the threshold and the line is marked incorrect – whereas LineAP fully credits the geometrically correct sections of both rails while penalizing the large gap on the left rail.

Table 1: LineAP and ChamferAP results on the RAIL-BENCH test set. Distance thresholds @α are given in percentage with respect to the track width. mAP is the mean AP across all thresholds for each individual metric and mAP_{avg} is the average between mAP of LineAP and ChamferAP. **Best results** are shown in bold, second-best results are underlined.

	LineAP						ChamferAP						mAP_{avg}
Model	@3	@5	@7	@10	@20	mAP	@5	@10	@20	@30	@50	mAP	
PINet	35.5	45.3	51.0	55.9	63.3	50.2	14.3	27.4	36.8	**40.3**	**43.6**	32.5	41.3
YOLinO	**51.9**	**56.1**	**59.2**	**62.5**	**66.5**	**59.2**	8.8	16.6	22.6	26.7	32.8	21.5	40.4
CLRNet	49.4	53.1	55.7	58.5	62.5	55.9	**22.9**	**32.5**	**37.7**	39.7	41.6	**34.9**	**45.4**

7 Conclusions

Our experiments show that LineAP evaluates polyline predictions at a finer geometric granularity than existing metrics. Combined with an instance-based metric like ChamferAP, we can achieve richer diagnostics of model performance than each metric alone. While LineAP highlights geometric accuracy, an instance-based metric is essential for understanding a model's instance construction capabilities.

Our novel track-width-based distance threshold further allows to customize distance-based metrics to the specific requirements of rail track detection, where precise localization of distant lines is particularly important for safe train operation.

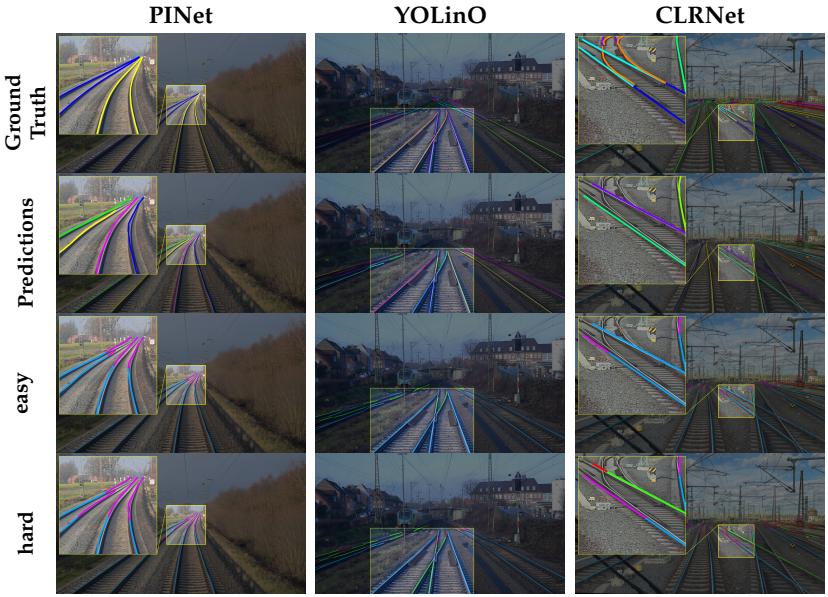


Figure 5: Ground truth (1. row) and predicted rails (2.-4. row) from PINet, YOLinO, and CLRNet, evaluated with LineAP and ChamferAP on RAIL-BENCH Rail in two difficulty levels: easy (LineAP@10 vs. ChamferAP@30) and hard (LineAP@5 vs. ChamferAP@10). Blue: both metrics agree that the prediction is correct. Red: both metrics agree that the prediction is incorrect. Green: identified as correct by LineAP only. Pink: identified as correct by ChamferAP only. Particular interesting cases are highlighted with a yellow box. Best viewed in colors.

Acknowledgements

This work was supported by the Federal Ministry of Transport of Germany within the mFUND program under the grant 01F1178A.

References

1. A. Bätz, P. Klasek, S.-Y. Ham, P. Neumaier, M. Köppel, and M. Lauer, "Railway artificial intelligence learning benchmark (RAIL-BENCH): A benchmark suite for perception in the railway domain," 2026. [Online]. Available: https://arxiv.org/abs/2604.22507

2. 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 Conf. on Comput. Vis. and Pattern Recognit. Workshops*, 2019, pp. 1221–1229.
3. T. Hiebert, F. Hofstetter, C. Thomas, S. Mushtaq, E. Kaan, and B. Parameswaran, "Labels4Rails: A railway image annotation tool and associated reference dataset," *Data*, vol. 10, no. 12, 2025.
4. S. Yoo, H. Lee, H. Myeong, S. Yun, H. Park, J. Cho, and D. H. Kim, "End-to-end lane marker detection via row-wise classification," 2020, arXiv:2005.08630.
5. X. Pan, X. Zhan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," 2026, arXiv:1712.06080.
6. Q. Li, Y. Wang, Y. Wang, and H. Zhao, "HDMapNet: An online HD map construction and evaluation framework," 2022, arXiv:2107.06307.
7. J. Holzfeind, J. Liu, and F. Pospischil, Eds., *Handbuch Eisenbahninfrastruktur*. Springer Berlin Heidelberg, 2025. [Online]. Available: https://link.springer.com/book/10.1007/978-3-662-68960-8
8. M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *Int. J. of Comput. Vis.*, vol. 111, no. 1, pp. 98–136, 2015.
9. Y. Ko, Y. Lee, S. Azam, F. Munir, M. Jeon, and W. Pedrycz, "Key points estimation and point instance segmentation approach for lane detection," *IEEE Trans. on Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8949–8958, 2022.
10. A. Meyer, P. Skudlik, J.-H. Pauls, and C. Stiller, "YOLinO: Generic single shot polyline detection in real time," in *Proc. of the IEEE/CVF Int. Conf. on Comput. Vis. Workshops*, 2021, pp. 2916–2925.
11. T. Zheng, Y. Huang, Y. Liu, W. Tang, Z. Yang, D. Cai, and X. He, "CLRNet: Cross Layer Refinement Network for Lane Detection," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA: IEEE, 2022.
12. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:206594692