Certified Control for Train Sign Classification*

Jan Roßbach
Heinrich-Heine-Universität Düsseldorf
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik
jan.rossbach@uni-duesseldorf.de

Michael Leuschel
Heinrich-Heine-Universität Düsseldorf
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik
leuschel@uni-duesseldorf.de

There is considerable industrial interest in integrating AI techniques into railway systems, notably for fully autonomous train systems. The KI-LOK research project is involved in developing new methods for certifying such AI-based systems. Here we explore the utility of a certified control architecture for a runtime monitor that prevents false positive detection of traffic signs in an AI-based perception system. The monitor uses classical computer vision algorithms to check if the signs – detected by an AI object detection model – fit predefined specifications. We provide such specifications for some critical signs and integrate a Python prototype of the monitor with a popular object detection model to measure relevant performance metrics on generated data. Our initial results are promising, achieving considerable precision gains with only minor recall reduction; however, further investigation into generalization possibilities will be necessary.

1 Introduction and Motivation

Artificial intelligence has been increasingly used in various sectors, including transportation [16]. One particular area where artificial intelligence (AI) has gained attention is the development of autonomous driving systems for railways [12]. The results already achieved in other transport sectors, mainly automotive, have encouraged the development of AI in the railway industry [19].

While this technology holds high economic interest, reliable certification methods are necessary to ensure safe and regulated access to these innovations [12]. Traditional verification approaches such as formal methods have faced difficulties in this area due to the opaque nature of AI, particularly in computer vision where class definitions for classification tasks based on raw pixel values have been considered challenging.

The KI-LOK [1] research project addresses these challenges by developing certification methodologies for autonomous AI-based railway systems. As part of this, a case study [5] on train movements during shunting movements is being analyzed. A formal B [1] model has been developed [5] to analyze the environment and ensure the safety of the deterministic steering system through model checking with the ProB [11] model checker. The safety of the system was found to be conditional on correct results from the AI-based perception system. In this work, we attempt to move towards verification of part of this perception system using a runtime monitor with a certified control [8] architecture. This architecture reduces the part of the system requiring formal verification compared to traditional monitor architectures putting a more formal analysis back into reach. In particular, we focus on a subset of the train sign classification component. It is responsible for detecting and classifying signs in the shunting yard to ensure safe train movements. False recognition of a 'track-free' (Sh1) signal has been determined to have

*This research is part of the KI-LOK project funded by the "Bundesministerium für Wirtschaft und Energie"; grant # 19/21007E.

[1] https://ki-lok.itpower.de

M. Farrell, M. Luckcuck, M. Schwammberger, and
M. Gleirscher (Eds): Fifth International Workshop on
Formal Methods for Autonomous Systems (FMAS 2023)
EPTCS 395, 2023, pp. 69–76, doi:10.4204/EPTCS.395.5

safety implications. We aim to significantly reduce or eliminate such false positives for some of the most critical classes by defining a sign-specific ontology and checking it at runtime. For this we introduce such a specification and show the potential performance gains by evaluating a prototype implementation in Python on a custom dataset.

2 Background and Related Work

The case study[5] being considered has been developed by Thales (now Ground Transportation Systems) and focuses on a train during shunting movements. The system includes an AI-based perception system and a deterministic steering system. The role of the perception system is to detect and classify obstacles (persons, animals, vehicles, ...) and railway infrastructure elements. The steering system then makes appropriate decisions about moving the locomotive based on that information.

There was a set of requirements provided with the case study, including the correct detection of several shunting train signs. In order to increase confidence in the perception system we aim to check the recognized signs with a runtime monitor. This will give strong confidence that detected signs are correct. In order to safeguard against unrecognized signs we will need to lean on other measures taken by the project, like a thorough environment ontology and systematic test case generation [4].

2.1 Certified Control

Certified Control[8] is an architectural framework for the real-time validation of autonomous systems. It distinguishes itself from conventional monitoring components by omitting its reliance on independent perception and instead counting on the controller to provide a *certificate* containing all essential information. This certificate serves as input for the runtime monitor, which assesses the accuracy of system behavior against specified criteria. By adopting this approach, the architecture establishes a trusted foundation that can potentially be subjected to a rigorous formal verification process.

The controller, which is not included in the *trusted base*, can utilize sophisticated algorithms such as neural networks without needing explicit formal verification. By separating the tasks of generating visual insights and ensuring safety, established verification methods can continue to be used with minimal adjustments. To accomplish this, a formal acceptance specification for the certificate is necessary to ensure compliance with safety requirements like *the detected lane lines are parallel* or *there are no objects on the track for 100m*. This reduces the amount of code needing verification and allows the AI components to go unverified.

While the effectiveness of this architecture in lane line detection for regular vehicles is promising [8], its applicability to other autonomous perception tasks such as sign classification and object detection remains uncertain. Therefore, we aim to investigate the applicability and effectiveness of such a certified control architecture in the context of the case studies train control perception system.

2.2 Related Work

Other attempts at verifying an autonomous train perception systems notably include [12]. The authors propose a multi-sensor pipeline relying on the statistical independence of the different perception mechanisms to control hazards and ensure suitable model performance. The goal is to show possible ways of certifying according to the ANSI/UL 4600 [6] standard, which provides a framework for integrating AI into fully autonomous systems. The standard gives practical guidelines and advice for a possible safety case, notably including the entire autonomy pipeline and AI algorithms. We also hope to provide methods

to aid with a verification according to this standard, while a full certification is currently out of reach. Other approaches to formal runtime monitor verification of AI systems have been done in the field of reinforcement learning using safety shields [10]. But these approaches focus on training agents to choose optimal policies depending on given environmental factors, which is similar to the traditional steering system in our model. There have also been proposals for formalizing image specification, including spatial model checking [2] and attempts to formalize vision ontology [13, 14].

3 Specification and Ontology

The selected sign classes for verification are Sh0, Sh1, and Wn7 as depicted in Figure 1. While these look similar, the semantic content is different. Sh0 means stop and the others signal safe passage. This makes properly distinguishing them a safety-critical issue. To ensure that the train comes to a stop when encountering a Sh0 sign on the current track, it is crucial to accurately detect and locate it. To achieve this, we employ an AI object detection system in the controller. Subsequently, the monitor verifies if the bounding box image aligns with the expected ontology. This provides additional confidence in the accuracy of the result.

(a) Sh0 (b) Sh1 (c) Wn7

Figure 1: Train Control Shunting Signs

It is often challenging to provide a precise formal definition of an image class based solely on its features. Instead, we focus directly on detectable image characteristics. In this context, we can observe that the images include two semi-circles with only orientation as the distinguishing feature. This characteristic feature allows us to define the sign using the contours and orientation angles of the feature.

For a given image tensor I with height h and width w, consider the set of contours (sets of points) denoted as $C(I)$, which are identified by a contour detection algorithm. Let S_0 be the set of images belonging to the Sh0 class. Also define $A : C(I) \to \mathbb{R}^+$ as the area function, which calculates the area of a given contour. Similarly, let $\sigma : C(I) \to \mathbb{Z}^+$ be an orientation function that determines the angle between the contour and the horizontal axis. We can then express membership of an image to one of the classes by considering an image a member of the set S_0 if it contains a pair $(c_1, c_2) \in C(I) \times C(I)$, which fullfills all the following conditions, given some pre-determined error tolerances $\delta_i, i \in \{1,2,3,4,5\}$[2] and an expected angle a that depends on the class in question.

1. $A(c_1)(1-\delta_1) \leq A(c_2) \leq (1+\delta_1)A(c_1)$
2. $(1-\delta_2)\sigma(c_1) \leq \sigma(c_2) \leq (1+\delta_2)\sigma(c_1)$
3. $\delta_3 h \leq A(c_i) \leq \delta_4 h, i \in 1,2$

[2]In the prototype implementation the tolerance values used were $\delta_{1,2,5} = 0.2$, $\delta_3 = 0.1$ and $\delta_4 = 0.3$

4. $\delta_3 w \leq A(c_i) \leq \delta_4 w, i \in 1,2$
5. $c_1 \cap c_2 = \varnothing$
6. $|\sigma(c_i) - a| \leq 90\delta_5, a = 0$

For the remaining two classes, the expected angle a in the final condition varies to 45 for Sh1 and 90 for Wn7. Otherwise, the definitions are identical. The conditions one to six define an Sh0 sign as an image with two contours that have similar angles and orientations. The orientation should be within a certain error threshold. Also, the definition expects, that the areas do not overlap. While ideally, we expect an orientation of zero, variations can occur due to different photo angles. Thus, the inclusion of an error term accounts for this discrepancy in measurement accuracy.

This definition is not flawless and permits the possibility of false positives. This implies that there may be instances where images that do not depict the intended sign could potentially be accepted (see Figure 2a). However, incorporating this check reduces the likelihood of such occurrences compared to those without it. The stringency of the monitoring process needs to be weighed against the decrease in true positives to strike a suitable balance. Adjustments can be made by selecting appropriate δ values within certain limits. Now we can define a requirement for a correct implementation.

REC: The implementation accurately verifies whether an image meets the ontology requirements of a specific class.

4 Implementation and Experiments

While the following implementation is not yet verified in terms of *REC*, we aim to do so in future work. Here we provide a prototype, which is developed enough to indicate the potential usefulness of such an implementation. Given an image and an expected class, it either validates or rejects the image. We then integrated it with a YOLOv8 object detection model and measured the influence on common performance metrics (see. Table 2b). In the following sections, we present details on the implementation and the performed experiments.

4.1 Implementation

The controller component is a simple wrapper for the YOLOv8[3] implementation of an object detection model known as YOLO [15]. The outcomes obtained from this model are packaged into a certificate and transmitted to the monitor. To have the model detect the signs in question, we created and labeled a custom sign-detection dataset [9], on which we trained three model variants. These were the nano, small and medium versions of the model with 3.2M, 11.2M and 25.9M parameters respectively. The training was done for 200 epochs with a batch size of 16. They achieved mAP50 values of 0.827, 0.90 and 0.93 on the test set.

From the model results the controller generates a *certificate* – in the sense of certified control (see Section 2) – consisting of the following components:

1. The original image.
2. The assigned class result.
3. The bounding box, represented as a tuple in the format (x, y, w, h), with values normalized to fit the dimensions of the image.

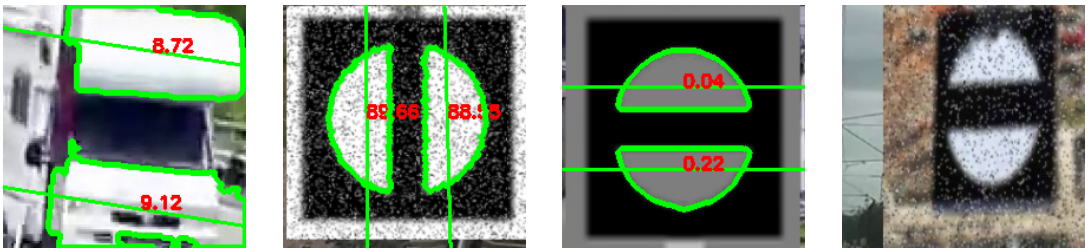
[3] `https://github.com/ultralytics/ultralytics`

This Python object is then given to the monitor. In a production implementation, it would be preferable to serialize and send this data to a statically typed version of the monitor for optimal security.

The monitor implementation utilizes Python's OpenCV [7] library to apply simple and well-tested computer vision algorithms to the given images. To begin, the bounding box image is resized to 206x206 and converted to grayscale to facilitate contour detection. Subsequently, a filtering process is applied to the contours to ensure their area falls within the specified size boundaries (refer to Section 3). We then need to calculate the area and orientation of the detected contours to determine if some of them fit the requirements for the ontology. The area of each contour is extracted using an available function within OpenCV. In addition, we utilize OpenCV once again by fitting a line through each contour as a means of determining its orientation. With that, we can calculate the orientation using the following equation.

$$\sigma(c) = \frac{180 \arccos(\vec{e_1} \cdot \vec{v})}{\pi |\vec{v}|}$$

Next, we evaluate the remaining contours in pairs to determine if they satisfy the similarity conditions for area and orientation (refer to Section 3 for details). If a pair is found that meets these conditions, we then verify if its orientation aligns with the expected orientation for the corresponding class. If it does, the monitoring system considers this as a valid certificate. However, if any of these criteria are not met, the certificate will be rejected.



(a) FP Sh0 accepted (b) FP Sh0 rejected (c) TP Sh0 accepted (d) TP Sh0 rejected

Figure 2: Visual Examples of Successful and Failing Monitor Checks

4.2 Experiments

In contrast to the automotive field, which benefits from large-scale image datasets like KITTI [3] for efficient object detection model evolution using road scene images, the railway industry faces limitations in terms of relevant datasets. Recently, interesting multi-sensor benchmark datasets [20] have started to emerge, but do not fit our particular use case. This lack of labeled, high-quality data poses a challenge when it comes to training and validating AI-based systems for this particular case. When evaluating the performance of the prototype, we have to confront this lack of data in the field. Since the relevant publicly available datasets do not cover the classes in question, we resort to custom labeling for training and a data generation approach for the evaluation of the system. For the generation, we chose a small number of base images of the signs in question, which are put through different random perturbation combinations and then pasted in random amounts – one to four – onto images from train footplate rides, gathered from the web. By this method we generated 28283 unique images containing 43638 signs. There are up to four signal per picture, which is typical of a shunting yard. For this work we ignore the selection of relevant signals and only focus on detection. The following perturbations were applied:

1. Horizontal Flip

Model	Detected	TP	FP
n	30111	25514	4597
s	30335	26790	3545
m	28672	22728	5944

(a) Results without Monitor

Model	Detected	TP	FP
n	21716	21714	2
s	22834	22831	3
m	20460	20460	0

(b) Results with Monitor

Table 1: Raw numbers for Models on Generated Data

Model	Precision	Recall	F_1 score
n	0.85	0.58	0.69
s	0.88	0.61	0.72
m	0.79	0.52	0.63

(a) Results without Monitor

Model	Precision	Recall	F_1 score
n	1.00	0.50	0.67
s	1.00	0.52	0.68
m	1.00	0.47	0.64

(b) Results with Monitor

Table 2: Model Metrics on Generated Data (values rounded to two decimal places)

2. Gaussian Noise (Salt and Pepper with Levels of 0.05 and 0.075)
3. Scaling (Up and back down to square images of 50,100,213,416 and 832 px)
4. Blur (normalized box filter with kernel sizes 3, 5, 7)
5. Brightness change (levels 0.5,1.5)

In Figure 2 we see examples of these images with monitor visualizations applied. It shows cut YOLO bounding boxes with the contours, lines and corresponding orientations detected by the monitor. Figure 2a shows one of the few remaining false positives. The image fits all the defined criteria of the S_0 ontology for these δ values but is not actually of that class. Given stricter tolerances (e.g. $90\delta_5 < 8$) this mistake would not occur. Overall the results seen in Table 2b show a slight reduction in model performance in terms of recall and an evenly weighted F-score compared to the prior results in Table 2a. The concrete detection numbers can be found in Table 1. The drop in recall and F-score is expected due to the reduction in true positives. However, almost all false positives have been recognized and can thus be prevented. The tolerances can be adjusted to further reduce false positives, at the cost of more recall and F-score, or to allow more leeway to the perception system. In terms of runtime performance, the monitor checks a certificate in approximately 0.7 ms on an Intel i5-12600K processor. In comparison, the inference of the YOLOv8 model will range from 2 ms – for the nano model variant – to 8 ms for the m version. This means that the performance overhead is likely not a major concern in a production environment.

5 Conclusion and Future Work

In conclusion, this study demonstrates the potential utility of certified control runtime monitoring for object detection of formally definable and safety critical classes. The resulting trade-off in our tests is promising enough to warrant further investigation into different application possibilities. However, further research is necessary to fully validate its implementation in a type-safe language following the *REC* guidelines. The obtained results should be verified in appropriate field test for any real world application. Additionally, it should be noted that a significant portion of the perception system remains unverified. Moving forward, our future work will involve evaluating the applicability of a similar architecture for other components of the perception system such as obstacle detection. This evaluation will include examining different sensor types such as LIDAR and radar on benchmark datasets.

References

[1] J.R. Abrial & A. Hoare (2005): *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, doi:10.1017/CBO9780511624162.

[2] Vincenzo Ciancia, Diego Latella, Michele Loreti & Mieke Massink (2016): *Model Checking Spatial Logics for Closure Spaces*. *Log. Methods Comput. Sci.* 12(4), doi:10.2168/LMCS-12(4:2)2016.

[3] Andreas Geiger, Philip Lenz & Raquel Urtasun (2012): *Are we ready for autonomous driving? The KITTI vision benchmark suite*. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, doi:10.1109/CVPR.2012.6248074.

[4] Jürgen Grossmann, Nicolas Grube, Sami Kharma, Dorian Knoblauch, Roman Krajewski, Mariia Kucheiko & Hans-Werner Wiesbrock (2023): *Test and Training Data Generation for Object Recognition in the Railway Domain*. In Paolo Masci, Cinzia Bernardeschi, Pierluigi Graziani, Mario Koddenbrock & Maurizio Palmieri, editors: *Software Engineering and Formal Methods. SEFM 2022 Collocated Workshops*, Springer International Publishing, Cham, pp. 5–16, doi:10.1007/978-3-031-26236-4_1.

[5] Jan Gruteser, David Geleßus, Michael Leuschel, Jan Roßbach & Fabian Vu (2023): *A Formal Model of Train Control with AI-based Obstacle Detection*. In Birgit Milius, Simon Collart-Dutilleul & Thierry Lecomte, editors: *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*, Springer Nature Switzerland, pp. 128–145, doi:10.1007/978-3-031-43366-5_8.

[6] Underwriters Laboratories Inc (2020): *4600 Standard for Evaluation of Autonomous Products*. Technical Report, Underwriters Laboratories Inc.

[7] Itseez (2015): *Open Source Computer Vision Library*. https://github.com/itseez/opencv.

[8] Daniel Jackson, Valerie Richmond, Mike Wang, Jeff Chow, Uriel Guajardo, Soonho Kong, Sergio Campos, Geoffrey Litt & Nikos Aréchiga (2021): *Certified Control: An Architecture for Verifiable Safety of Autonomous Vehicles*. *CoRR* abs/2104.06178, doi:10.48550/arXiv.2104.06178. arXiv:2104.06178.

[9] KILOK (2023): *Sign Detection Dataset*. `https://universe.roboflow.com/kilok/sign-detection-4oqe4`. Visited on 2023-08-09.

[10] Bettina Könighofer, Florian Lorber, Nils Jansen & Roderick Bloem (2020): *Shield Synthesis for Reinforcement Learning*. In Tiziana Margaria & Bernhard Steffen, editors: *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, Springer International Publishing, Cham, pp. 290–306, doi:10.1007/978-3-030-61362-4_16.

[11] Michael Leuschel & Michael Butler (2003): *ProB: A Model Checker for B*. In: *Proceedings FME*, LNCS 2805, pp. 855–874, doi:10.1007/978-3-540-45236-2_46.

[12] Jan Peleska, Anne E. Haxthausen & Thierry Lecomte (2022): *Standardisation Considerations for Autonomous Train Control*. In Tiziana Margaria & Bernhard Steffen, editors: *Leveraging Applications of Formal Methods, Verification and Validation. Practice*, Springer Nature Switzerland, pp. 286–307, doi:10.1007/978-3-031-19762-8_22.

[13] Daniele Porello, Marco Cristani & Roberta Ferrario (2013): *Integrating ontologies and computer vision for classification of objects in images*. In: *Proceedings of the Workshop on Neural-Cognitive Integration in German Conference on Artificial Intelligence*, pp. 1–15.

[14] " K. K. Thyagharajan R. I. Minu" (2014): *Semantic Rule Based Image Visual Feature Ontology Creation*. *International Journal of Automation and Computing* 11(20140504), doi:10.1007/s11633-014-0832-3.

[15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick & Ali Farhadi (2016): *You Only Look Once: Unified, Real-Time Object Detection*. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, Los Alamitos, CA, USA, pp. 779–788, doi:10.1109/CVPR.2016.91.

[16] Danijela Ristić-Durrant, Marten Franke & Kai Michels (2021): *A Review of Vision-Based On-Board Obstacle Detection and Distance Estimation in Railways*. *Sensors (Basel, Switzerland)*, doi:10.3390/s21103452.

[17] Claudio Filipi Gonçalves dos Santos & João Paulo Papa (2022): *Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks*. CoRR abs/2201.03299, doi:10.1145/3510413. arXiv:2201.03299.

[18] Satoshi Suzuki & KeiichiA be (1985): *Topological structural analysis of digitized binary images by border following*. Computer Vision, Graphics, and Image Processing 30(1), pp. 32–46, doi:10.1016/0734-189X(85)90016-7.

[19] Ruifan Tang, Lorenzo De Donato, Nikola Besinovic, Francesco Flammini, Rob M.P. Goverde, Zhiyuan Lin, Ronghui Liu, Tianli Tang, Valeria Vittorini & Ziyulong Wang (2022): *A literature review of Artificial Intelligence applications in railway systems*. Transportation Research Part C: Emerging Technologies 140, p. 103679, doi:10.1016/j.trc.2022.103679.

[20] Roman Tilly, Philipp Neumaier, Karsten Schwalbe, Pavel Klasek, Rustam Tagiew, Patrick Denzler, Tobias Klockau, Martin Boekhoff & Martin Köppel (2023): *Open Sensor Data for Rail 2023*, doi:10.57806/9MV146R0.