Vision Based Stationary Railway Track Monitoring System

Mirjam Klammsteiner*, Mario Döller*, Patrik Golec†, Michael Kohlegger*
Kufstein University of Applied Sciences
Kufstein, Austria
mirjam.klammsteiner, mario.doeller@fh-kufstein.ac.at
*Josef Ressel Center Vision2Move
†FFG Digital Innovation Hub West

Stefan Mayr
Bernard Technologies GmbH
Hall in Tirol, Austria

Ehtsham Rashid
University of Passau
Passau, Germany

***Abstract*—This paper presents a system for monitoring rail tracks in mountainous areas by using a stationary camera system to detect obstacles on the railbed. On-board obstacle detection systems on trains are already used for the purpose of obstacle detection, but due to the nature of mountainous areas these aren't suitable for our problem case. Our approach combines a Machine Learning approach using Yolov5 with a traditional Computer Vision approach to be able to detect obstacles that Yolo was not trained for such as avalanches, mudslides, etc. The system was tested on a single example setup which was set up with the help of the Austrian railway track company. Our experimental test runs already showed a promising low number of incorrect reports. The system, however, yet does not cope with specific conditions, such as extreme lighting or bad weather.**

I. Introduction

In 2020, 1331 major railway accidents were reported by the European Union [1]. There were 798 incidents involving unauthorized persons on track, along with that 412 level crossing accidents involving pedestrians. In total, 687 individuals lost their lives and 468 were severely injured in these accidents. Based on those numbers, obstacle detection on railway tracks is an important lifesaving issue.

There is promising research on both, vision- and other sensor-based hardware placed in front of the train to mitigate those accidents [2]–[4]. These sensors are similar to the sensors used in the automotive industry such as LiDAR, Ultra-sonic, Infrared and Thermal cameras, Stereo and RGB cameras, etc. However, most of these sensors are extremely costly, and strong computing hardware is required to interpret all essential data in real-time. Instead of expensive sensors, one can employ low-cost cameras with AI-based solutions. But, research on AI-based software implemented for trains is minimal. There is some literature available using traditional Computer Vision (CV) based methods for railway track and obstacle detection [5], [6].

In general, one can distinguish between two kinds of approaches: On the one side, research is focusing on obstacle detection based on vehicle-mounted sensors. There are various sensors (LIDAR, visual-based, etc.) that can be used. However, sensors mounted in front of a train have a major drawback. Depending on a train's length and speed it often needs to stop several kilometers before an obstacle to avoid hitting it.

Our contribution can be summarized as follows: (A) an overall concept for a railway track surveillance system capable of running on an embedded device in mountain areas is proposed (B) a combined approach consisting of ML as well as traditional CV-based method is evaluated (C) a data set on the real environment is introduced and challenges discussed.

The paper is organized as follows: Section 2 gives a short overview of already existing approaches to the problems of rail segmentation and obstacle detection. Section 3 introduces our contribution split into three sub-sections describing (A), (B), and (C) as mentioned previously. Section 4 details our evaluation approach and the dataset we used for it. The paper ends with a conclusion in Section 5.

II. Related Work

This section discusses previous research on obstacle and railway track detection. The topic of railway obstacle detection often consists of two parts. First, the railway tracks must be detected and then obstacle detection is performed. Sometimes these steps happen simultaneously. The primary purpose of object detection in railways is to detect objects (potential obstacles) on and near railway tracks to mitigate railway track accidents. The automotive industry has conducted a significant amount of research in obstacle detection and enormous advancement in AI and sensing technologies have been made. However, studies specifically on railway tracks have been less extensive. Some of the past work has used vision based as well as other sensors to detect obstacles or railway tracks [2].

The method used in [7] is a traditional CV-based method that identifies railway tracks using the Hough Transform technique, followed by the detection of obstacles using CV algorithms. Firstly, the Hough mechanism detects the lines representing the railway track. This task eliminates the irrelevant background objects by resizing the image and applying a rectangular mask to it. Secondly, the Canny edge-detection algorithm analyses the obstacles on railway tracks. The Canny algorithm first filters the image then the edge contours are extracted from the image. Finally, the systematic search of objects starts after filling the remaining contours. The research in [7] is then carried on in [8], where a new method is described for detecting dynamic objects derived from the idea of optical flow among frames. First, the approach finds potentially dangerous objects while ignoring unimportant background movement elements. Then, the trajectories of the potentially dangerous obstacles are tracked and used to determine if they might collide with the train in the future.

In [9], a CNN-based rail area detection method is presented. This method consists of two parts: extracting the railway area and further optimization. In the first part, a CNN was proposed for the pixel-level classification of the railway track area. The main improvement in their architecture was extracting the information from the rail line and their surroundings or even from the full image, thus achieving multi-scale feature extraction. In the second part, the author proposed an optimized polygon fitting method to refine the rail region further. The experimental results indicate that the model can perform well in different conditions, such as tunnels, shadows, reflections and rail switch scenes. This research covers the railway track detection part but lacks the obstacle detection problem in front of the train.

Jin Wang et al. [10] proposed a DL-based segmentation model named RailNet. This model consists of two networks, the segmentation network and the feature extraction network. The feature extraction network employs a pyramid design to generate a hybrid feature vector by spreading features from top to bottom. The segmentation network maps the railway track pixel by pixel using a convolutional network. To evaluate the RailNet model, the author created a Railroad Segmentation Dataset (RSDS). The dataset contains 3000 images, of which 2500 are for training, 200 are for validation, and 300 are for testing. The proposed method performed well on the dataset and accurately detected the railroad. This research covers the railway detection part but does not focus on obstacle detection.

Peng Yang et al. [11], in their paper, presented a DL-based method specific to object detection. In their research, a Faster R-CNN model [12] is used for detecting objects. The model is trained using 20,000 images of outdoor railroad track scenes. The dataset was divided into three classes: trains, people, and animals. Their research also handles the alert system (e.g. Warning and Normal). The proposed model detects people or animals who appear beside the train and, if it is, issues an alert. The method is capable of detecting multiple objects along with their confidence scores. However, the railway-track detection part is missing here.

In [13], a method named DisNet is proposed, which operates with an onboard RGB camera and detects the obstacles from the driver-eye view of the train. The method performs two tasks: the first task is the detection of possible obstacles using a DL-based model called YOLOv3 [14], and the second task is the distance estimation of obstacles from the camera using a multi-hidden layered neural network. YOLO was refined using a transfer learning technique and retrained on a custom dataset of 998 images in which 2238 objects are labeled with 4 different classes such as humans, cars, bicycles, and animals. An RGB SMART camera captured the images mounted on the Serbia Cargo train. The research results indicate that the model is functional for long and mid-range real-time obstacle detection. In contrast, we are developing a system by merging two fast methods that identify risk categories using the pixel-wise overlapping technique.

The main difference of our approach according to the presented related work can be summarized as follows. First, our approach also uses a CNN based network (YOLO [15]) for detecting well known objects (COCO [16]) and one for segmenting the target area (BiSeNet V2 [17]) similar to approaches such as [10]. In addition to that, we are focusing on monitoring rail tracks for obstacles such as mud, stones, trees, etc. where not enough data sets are available (for transfer learning). Therefore, computer vision approaches are evaluated and combined within an overall algorithm.

III. Architecture of monitoring system

A. Overall Architecture

As defined in the introduction, our approach aims on monitoring specific rail tracks in mountainous areas, where the probability of natural disasters is high. In Fig. 1, such a rail track is shown with the estimated camera systems that need to be installed. The different colors (green == 200m, yellow == 150m, orange == 100m) specify the line of sight that is observable according to the geologic environment.

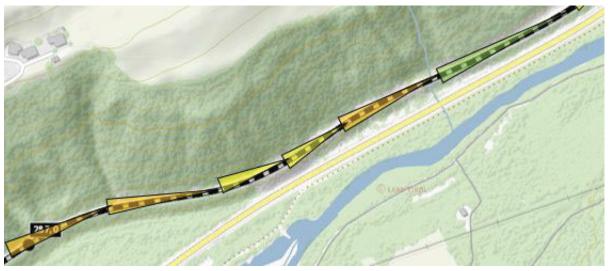


Fig. 1. Estimated camera positions and line of sight

Overall monitoring is established per camera, which reports binary status information (free/obstacle detected).

In the following subsections, the individual components of the overall architecture (see Fig. 2) are discussed. The overall architecture represents all algorithms that are executed on the individual cameras. This means, that every camera reports every specified time period (can be defined) about the status of the observed area.

B. Combined Approach: Track detection and Object classification

The approach is split into 5 sub-tasks, an overview of which can be seen in Fig. 2.

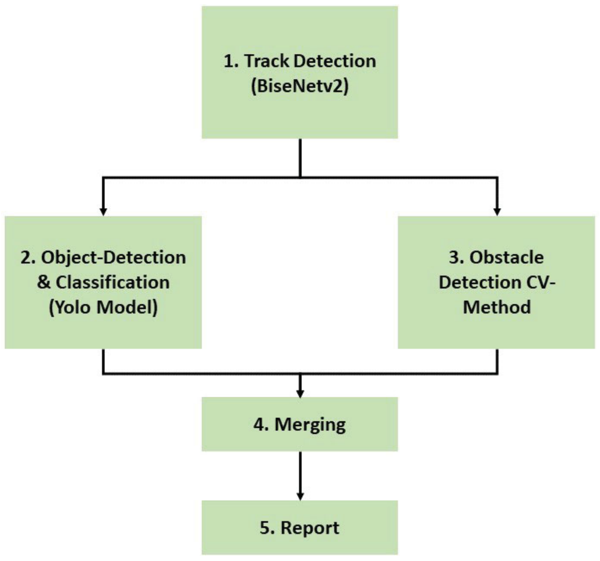

1. Track Detection (BiseNetv2)
2. Object-Detection & Classification (Yolo Model)
3. Obstacle Detection CV-Method
4. Merging
5. Report

Fig. 2. Components of the overall process

1) Track Detection: The first part of our approach focuses on railway segmentation and the identification of the track and its surrounding. This is realized by first using the BiseNetv2 model [17] to get the left and right rails as well as the railbed and subsequently adding some additional pixels to the left and right of the tracks to define a Region of Interest (ROI) which is then used as a mask for further steps, see Fig. 3

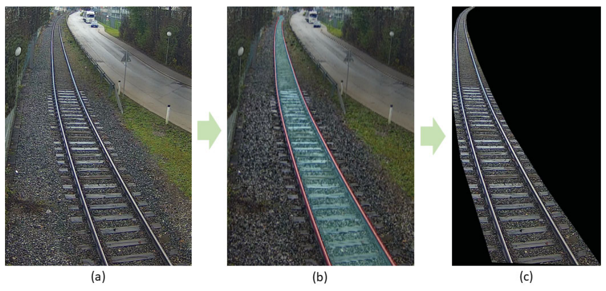


Fig. 3. (a) shows an image that is used to extract rails from (b) shows rail track detected by BiseNetv2 (c) shows final ROI after adding pixels to each side

2) Object Detection & Classification: The YOLOv5 model [18] is used to detect and classify known objects (known related to the COCO classes) in order to enable a binary classification in allowed and disallowed objects. For instance, a train that is detected on a track is an allowed object and should not end in sending an alarm. Whereas a person at a specific distance to the track or on the track is a disallowed object and needs to result in an alarm. Note, that our approach aims not in optimizing the track or object detection models as this is open for further research and can be exchanged easily in our overall architecture. In this proposal, the overall architecture of the system is the focal point.

3) Obstacle Detection: The main idea of the algorithm is to check for differences between the reference image and an image taken at the current time using an approach we dubbed Pixel4Pixel. In short, the difference is calculated by getting the absolute difference between the two frames and as an end result we receive one or more bounding boxes that show us the location of detected obstacles, as can be seen in Fig. 4.

There are several requirements that need to be fulfilled in order for the obstacle detection method to work. First of all, a masked ROI needs to be supplied, which can be taken from the track detection step that BiseNetv2 is used for. Additionally, a reference image of the tracks being empty is required which will be updated as the algorithm runs.

After receiving a current image of the track from the camera and having access to a saved reference image the algorithm can start. As a first step, we investigate both images for potential lens obstructions, examples of which can be seen in Fig. 6, which are often caused by bad weather. We use an algorithm that checks for blur in the image [19], if an image is too blurry, we assume that the lens is obstructed by something and stop the process. In the case of both images having been found clear of lens obstruction, we continue with the following steps. Because of lighting changes due to the movement of the sun and weather influence both images need to undergo several preprocessing steps at first such as greyscaling, adjusting contrast and brightness, and histogram matching. Afterward blur kernels with a differing radius are applied to sub-parts of the tracks to get rid of small changes we don't want to account for, so-called noise. The size of the kernel is changed depending on how close to the camera the part of the track is. Parts of the track closer to the camera are blurred with a wider kernel because small changes we don't want to account for are more visible closer to the camera. In contrast, a small kernel radius is used for the far end of the track. After the preprocessing steps, a pixel-for-pixel comparison between the image in question and the reference image takes place which results in an image that shows us the difference between the two images. Next, we try to enlarge possible obstacles as sometimes only rough outlines are detected. This happens by applying dilation kernels of different sizes on the image result from the previous step, here obstacles further from the camera are dilated with a smaller kernel. As a next step, the area of detected obstacles is calculated, if the result is bigger than a certain predefined size the obstacles's bounding box is calculated and passed onto the next step, which merges the results from the two different approaches we used.

Choosing a proper reference image is challenging because only images without obstacles in them are suited for that job as otherwise false positive alarms would be sent out. There are currently two ways we propose to get a new reference image, (a) taking the last non-obstacle image as a new reference image, (b) creating an average image from the last Y frames where newer frames are more important than older frames.

4) Merging: This step is meant to merge the results from the two methods that we used. From the Object Detection and Classification part (2.) we receive classification information on

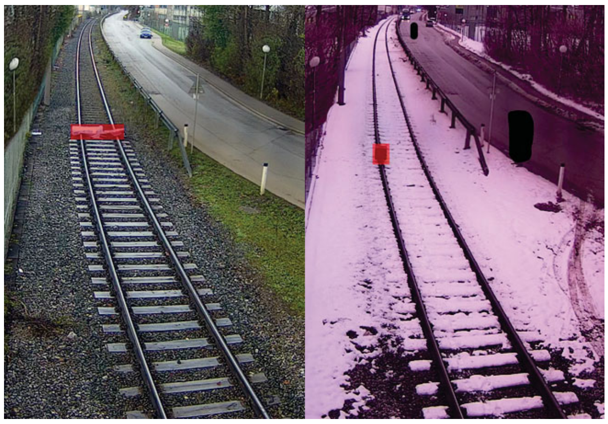


Fig. 4. Example images of obstacles that were detected highlighted by a red bounding box

the type of obstacle and bounding box coordinates whereas from the Obstacle Detection Method (3.) we only receive bounding box coordinates of a detected obstacle. We now combine these two results by checking whether the bounding boxes overlap, which can also be seen in the flowchart in Fig. 5. In the case of an overlap, we check whether the classified obstacle is detected as allowed (e.g. train) or disallowed (e.g. human) to be on the tracks. Disallowed obstacles result in a report, whereas allowed obstacles will be ignored. In the case of the Object Detection and Classification method not detecting any obstacles but the Obstacle Detection Method detecting something we still send a report, as we must assume that an obstacle that Yolo was not trained for (e.g. stones, avalanches, mudslides) is blocking the track.

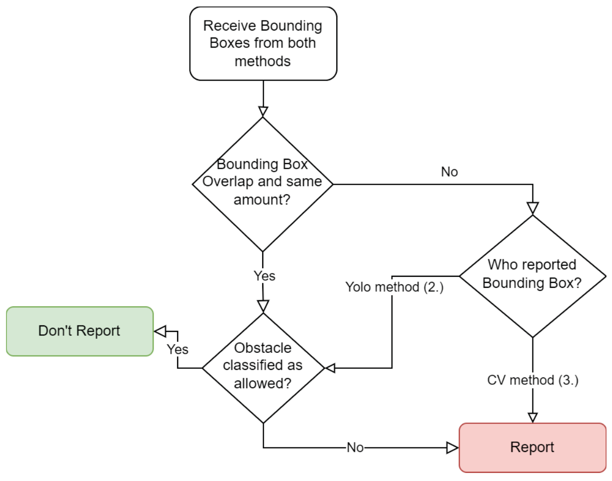

Receive Bounding Boxes from both methods
Bounding Box Overlap and same amount?
No
Yes
Who reported Bounding Box?
Yolo method (2.)
CV method (3.)
Don't Report
Yes
Obstacle classified as allowed?
No
Report

Fig. 5. Flow Chart showing merging decisions and results

5) Report: As a last step in the process, a report needs to be made. An image of the current obstacle with its bounding box as an overlay is sent to be checked by a human as a last step, who then needs to decide how to react. An example of what the results might look like can be seen in Fig. 4. We do not want to automate this step of the process as different steps need to be taken for different problems.

C. Challenges

Throughout the process of creating the system and evaluating approaches, we came upon several difficulties and challenges.

- Obstruction of the camera lens: Due to bad weather (e.g. snow, rain, ...) one difficulty we face is obstruction of the lens. While this might be solvable by physical constructions that avoid the obstruction to happen in the first place, for that purpose we use a blur detection approach as mentioned in 3) Obstacle Detection [19].
- Shadows: Because of the angle of the stationary camera it is difficult to distinguish between 2d and 3d objects, therefor shadows which ”move” onto the tracks due to sun movement could be falsely classified as obstacles. While there are some approaches to identifying and removing those shadows (e.g. [20]) we have not yet been able to evaluate the effect of such methods.
- Night Setting: Our approach has currently only been tested with images taken in a daylight setting, images taken at night might be difficult evaluate due to too little lighting.

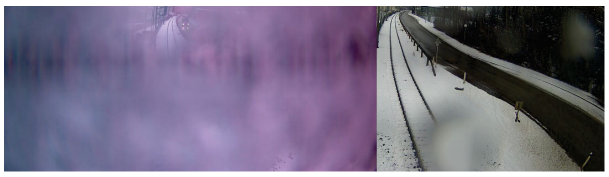


Fig. 6. Example images of the camera lens being obstructed

IV. Evaluation

For our evaluation, we used different kinds of images. For 2) Object Detection & Classification we used images from the RailSem19 data set [21] and some images downloaded from the internet (Youtube videos). We annotated 50 images manually and classified them into two classes (allowed/not allowed).

For the CV-approach images of our test environment, which has been set up with the Austrian railway transportation company, were used. We received 101 images showing the rail track in normal conditions and 186 images showing snowy conditions, see Fig. 7, from a camera that takes an image roughly every minute. We separated these 101 and 186 images into images containing obstacles in the ROI (69 and 136) and images not showing any obstacles in the ROI (32 and 50).

Afterward, we simulated a video stream, evaluating the images in chronological order but assuming that the first image is always free of obstacles. We then compared the results from that simulations to our labeling, resulting in the Confusion Matrices in Fig. 8, 0 meaning no obstacle and 1 meaning obstacle. The first image of each of the datasets is missing in the confusion matrix as we used it as a first reference image and therefore did not count it in the evaluation.

The confusion matrix shows false negatives for both of our approaches. We discovered that these errors occur when the contrast between obstacles and the background is too low, resulting in obstacles blending in with the background.

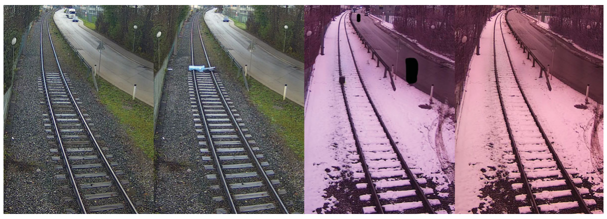
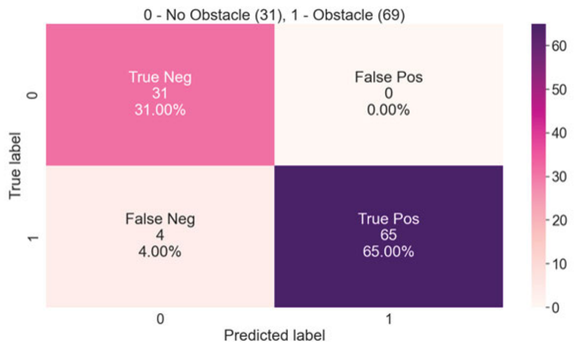


Fig. 7. Selection of images we received as test data



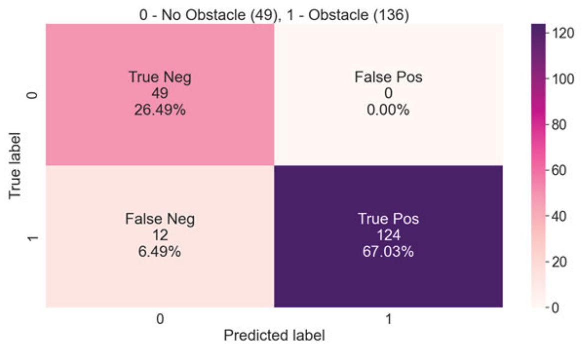


Fig. 8. Confusion Matrix of our simulation, top shows normal images, bottom shows snowy images

V. Conclusion

In this text, the authors present an approach for monitoring rail tracks in mountain areas using camera systems to detect obstacles. The approach is split into five sub-tasks, including (i) track detection, (ii) object detection and classification, (iii) obstacle detection, (iv) merging, and (v) alarming. The authors propose an overall architecture where all sub-tasks are executed on the individual cameras, with each camera reporting the status of the observed area. Obstacle detection is based on a Pixel4Pixel approach, which calculates the difference between the reference image and the current image to detect obstacles. The authors propose two methods for choosing the reference image, and they discuss the challenges associated with this step.

The presented work's contribution is not related to track and object detection itself, but rather to the overall architecture and the orchestration of sub-tasks. In addition to that, the presented research combines state-of-the-art ML techniques with traditional CV approaches. Currently, this approach lacks thorough testing in multiple real-world scenarios. So far, this approach has only been used in a stationary setup with dummy obstacles. Also, there is yet no data on how well this approach works under different lighting and vision conditions (e.g. in case of bad weather, very bright sunlight, or darkness). Overall, the authors present a comprehensive, yet applied approach for monitoring rail tracks in mountain areas that can be further optimized in future research.

Acknowledgements

This work has been partly supported by the following grants: FFG Digital Innovation Hub West as well as CDG Josef Ressel Center for multimedia analysis in the mobility domain

References

[1] European Union Agency for Railways, *Report on railway safety and interoperability in the EU, 2022*. Publications Office of the European Union, 2022.

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

[3] D. Ristić-Durrant, M. A. Haseeb, D. Emami, A. Gräer, V. Nikolić, I. Ćirić, M. Banić, B. Brindić, D. Nikolić, D. Radovanović, F. Eßfer, and C. Schindler, "SMART concept of an integrated multi-sensory on- board system for obstacle recognition," in *7th Transport Research Arena TRA 2018*. Zenodo, 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1446119

[4] A. Berg, K. Öfjäll, J. Ahlberg, e. R. R. Felsberg, Michael", and K. S. Pedersen, "Detecting rails and obstacles using a train-mounted thermal camera," in *Image Analysis*. Cham: Springer International Publishing, 2015, pp. 492–503.

[5] F. Kaleli and Y. S. Akgul, "Vision-based railroad track extraction using dynamic programming," in *Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2009, p. 1–6.

[6] H. Mukojima, D. Deguchi, Y. Kawanishi, I. Ide, H. Murase, M. Ukai, N. Nagamine, and R. Nakasone, "Moving camera background-subtraction for obstacle detection on railway tracks," 09 2016, pp. 3967–3971.

[7] L. A. Fonseca Rodriguez, J. A. Uribe, and J. F. Vargas Bonilla, "Obstacle detection over rails using hough transform," in *Proceedings of the XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA)*, 2012, p. 317–322.

[8] J. A. Uribe, L. Fonseca, and J. F. Vargas, "Video-based system for railroad collision warning," in *Proceedings of the IEEE International Carnahan Conference on Security Technology (ICCST)*, 2012, p. 280–285.

[9] Z. Wang, X. Wu, G. Yu, and M. Li, "Efficient rail area detection using convolutional neural network," *IEEE Access*, vol. 6, p. 77656–77664, 2018.

[10] Y. Wang, L. Wang, Y. H. Hu, and J. Qiu, "Railnet: A segmentation network for railroad detection," *IEEE Access*, vol. 7, p. 143772–143779, 2019.

[11] M. Yu, P. Yang, and S. Wei, "Railway obstacle detection algorithm using neural network," in *Proceedings of the AIP Conference Proceedings 1967.1*, 2018.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *CoRR abs/1506.01497*, 2015.

[13] D. Ristić-Durrant, M. A. Haseeb, M. Franke, M. Banić, M. Simonović, and D. Stamenković, *Artificial Intelligence for Obstacle Detection in Railways: Project SMART and Beyond*. Springer, 2020, iSBN: 978-3-030-58462-7.

[14] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[15] M. Horvat, L. Jelečević, and G. Gledec, “A comparative study of yolov5 models performance for image localization and classification,” in *Proceedings of the 33rd Central European Conference on Information and Intelligent Systems*, 2022.

[16] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *CoRR abs/1405.0312*, 2014.

[17] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, “Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation,” in *CoRR abs/2004.02147*, 2020.

[18] U. Nepal and H. Eslamiat, “Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs,” *Sensors*, vol. 22, no. 2, 2022.

[19] F. Crété-Roffet, T. Dolmiere, P. Ladret, and M. Nicolas, “The blur effect: Perception and estimation with a new no-reference perceptual blur metric,” in *SPIE Electronic Imaging Symposium Conf Human Vision and Electronic Imaging*, vol. 12, 2007, pp. EI–6492.

[20] A. Sanin, C. Sanderson, and B. C. Lovell, “Shadow detection: A survey and comparative evaluation of recent methods,” *Pattern Recognition*, vol. 45, no. 4, pp. 1684–1695, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320311004043

[21] O. Zendel, M. Murschitz, M. Zeilinger, D. Steininger, S. Abbasi, and C. Beleznai, “Railsem19: A dataset for semantic rail scene understanding,” in *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.