

Automated Detection of Track Gauge Deviations Using Video and Depth Cameras with Machine Learning

Connor Stonestreet*, Hwapyeong Song†, Husnu S. Narman‡, Pingping Zhu§, and Ammar Alzarrad¶

Email: *stonestree19@marshall.edu, †song24@marshall.edu, ‡narman@marshall.edu, §zhup@marshall.edu, ¶alzarrad@marshall.edu

Abstract—Ensuring the safety and reliability of railway infrastructure is crucial for transportation systems worldwide. This paper introduces a novel approach to detecting horizontal and vertical track height deviations in railways using video and depth cameras combined with machine learning. Track gauge deviation refers to the change in track gauge values from the expected to the current value. The primary objective is to reduce the time, human labor, and costs associated with inspecting large sections of railway for track gauge deviation by automating the process with machine learning. A dataset of relevant track images is selected and augmented using techniques such as grey scaling, blurring, brightness changes, and the addition of noise. This dataset is used to train several machine learning models. Various detection strategies were developed and considered, and a combination of converting pixels to real-world measurements and utilizing depth camera data was chosen. Preliminary results from our depth camera demonstrate promising levels of accuracy for estimating track gauge deviation. This machine learning approach offers a cost-effective and efficient solution for detecting track gauge deviation, thereby maintaining the safety of our railroad infrastructure.

Index Terms—Railway, Object detection, Gauge deviation, Machine learning

I. INTRODUCTION

Detecting defects and issues in railroad tracks is an essential part of the maintenance required to keep our railways safe and efficient. Without regular inspection and maintenance track gauge deviation can develop and pose significant safety issues. Track gauge deviation occurs when the distance between two rail lines grows or shrinks outside of an acceptable range. Track gauge deviation often occurs gradually over longer periods of time and when left untreated can have severe negative consequences.

To combat issues like track gauge deviation, railroad companies assign large crews of workers to inspect sections of rail track in their entirety. Inspecting every foot of a rail track takes a significant amount of time and resources, which could be optimized using a machine learning approach. A machine learning approach to identifying track gauge deviation would streamline the overall process and reduce the need for human crews and other resources, which would reduce railway maintenance costs by a significant amount.

Previous research has been conducted regarding different machine learning approaches to detecting track gauge de-

viation. For example, Lasisi and Attoh-Okine utilized the Weibull distribution to create machine learning models that predict when track gauge deviation may occur [1]. Liao et al. compared many of the most prominent machine learning techniques for railroad defect detection, including track gauge deviation [2]. Also, Sresakoolchai and Kaewunruen developed a model that uses supervised and unsupervised machine learning, which is based on track geometry [3]. While their focus was on missing bolts and bar cracks, their results and processes could potentially be applied to detecting track gauge deviation. Moreover, Falamarzi et al. conducted a comparative study between SVM and ANN classifications for predicting gauge deviation on both straight and curved lines, concluding that ANN demonstrated higher performance [4].

Popov et al. utilized an autoencoder paired with KMeans clustering to detect track geometry irregularities [5]. More recently, Pires et al. created and optimized nine different machine learning models to identify lateral and vertical track irregularities [6]. This study relied on a tool called "Optuna" which could prove useful in our study when developing our machine learning models.

Yu et al., in 2017, introduced CASENet, which is an edge detection algorithm that can categorize its detection [7]. This could potentially be adapted to help detect the edge of rail lines. In addition, in 2019, Rezatofghi et al. conducted a comparative analysis of different bounding box regression models within the frameworks of Faster R-CNN and Mask R-CNN [8]. These frameworks could be utilized to compare the actual position of the rail lines with the expected position to monitor their alignment. Tang et al. collected 81 scholarly articles that focus on railway inspection and maintenance using artificial neural networks, convolutional neural networks, and recurrent neural networks [9]. These articles provide a solid research base to explore for potential solutions.

While many of these works provide a solid foundation and a good starting point for our objective, they primarily focus on using various machine learning methods to identify defects or predict when track gauge deviation will occur. However, our *objective* is to analyze the effectiveness of machine learning models by incorporating object detection and sensor data to identify both vertical and horizontal track gauge deviations.

The key *contributions* of this paper are as follows. First, we develop a comprehensive training dataset using various

annotation strategies and preprocessing techniques. Second, multiple machine learning models are assessed to identify the most effective one for detecting railroad lines. Third, the selected model is trained with the created dataset and fine-tuned to excel in detecting railroad lines. Then, several methods are devised to utilize the model for detecting track gauge deviations from video footage. Finally, the effectiveness of the models is analyzed using datasets with and without cameras equipped with depth sensors. The *results* of our research provide a consistent and effective way to detect track gauge deviation in railroad tracks by using machine learning. Our results can assist organizations that monitor and repair railroads to increase their efficiency and decrease their costs by eliminating the need for extensive human-led track inspections.

The rest of the paper is organized as follows: Section II includes the methods of the paper, including the creation of the dataset, the selection of the machine learning model, the setup of the model training environment, and finally the creation and implementation of the track gauge deviation detection techniques. Section III discusses the results, which include the machine learning model training results and the results from the implemented track gauge deviation detection technique. Section IV contains the final remarks.

II. METHODS

Before creating and training machine learning models, it is essential to compile a dataset of suitable images. In this paper, the datasets used for training various models are created using Roboflow, an online tool for dataset creation and machine learning model training. Potential images are uploaded to the project on Roboflow and annotated using its annotation tools.

Images are selected based on their relevance to track gauge deviation. Ideal candidates have a clear view of both rail lines, preferably from a near-vertical angle, and are free from track obstructions or other obstacles. Images are compiled from several public datasets available on Roboflow [10]–[12]. A good example image can be found in Fig. 1.



Fig. 1. Example of a training image from the dataset.

A. Annotation Strategy

In order to prepare the image for training our machine learning models, annotations must be applied. Detecting track gauge deviations from a video feed requires the detection of each rail line in each image/video frame. Therefore, each rail line in each image is annotated using the class “railroad-lines”. Along with the rail lines, in approximately half of the total images, the ballast in and around the rail lines is labeled as “ballast”. The labeling of the ballast in some of the images allows the machine learning models to better differentiate between the railroad lines and the ballast, which leads to better overall predictions and performance.

The labeling of the ballast is only done on approximately half of the total images because of the time-consuming nature of manually annotating each image by hand. After the annotation process is complete, the total number of images is 132. After each image in the dataset is labeled, several preprocessing methods are applied to the dataset to increase the overall size. An example image with “railroad-lines” and “ballast” class annotations can be found in Fig. 2:



Fig. 2. Example training image with all annotations applied.

B. Preprocessing Techniques

After each image in the dataset is labeled, several preprocessing methods are applied to the dataset to increase the overall size. Using the built-in Roboflow preprocessing techniques, up to 3 outputs per training example can be created using selected augmentations. The following augmentations are applied to all 107 of the training images in the dataset, resulting in a new total of 321 training images; grayscale to 100% of images, brightness between -21% and +21%, blur up to 2.5px, and noise up to 1.01% of pixels. An example of a training image after preprocessing techniques have been applied can be found in Fig. 3.

Grayscale is applied to increase the training variance as the color information is not relevant to detecting the rail lines. The brightness values of each image vary between -21% and



Fig. 3. Example training image after preprocessing techniques have been applied.

+21% to give the model a more comprehensive view of real-world scenarios, where many different lighting conditions are expected to occur. Furthermore, blur up to 2.5px is applied to the images to make the model more resistant to potential camera focus and blur. Since this model will eventually be applied to instances where the video/images are taken at a high speed, it is essential that it can still make detections even if the view is blurry. Last, noise is added up to 1.01% of pixels to increase training variance and prevent overfitting.

Utilizing these preprocessing techniques significantly increases the total number of training images that are available in the dataset, and provides a more comprehensive collection of images that better represent real-world conditions.

C. Training Environment

In order to determine which model is the best for our applications, each model is trained and tested using identical conditions. Model training is done using Google Colab, with the selected GPU being the A100. Each model version is trained using the same dataset, the creation of which is discussed earlier in this paper. Model training is done using the built-in training functions that the YOLO library provides. All hyperparameters are kept default, with the exception of the image dimensions, which was set to a value of 640 pixels.

D. Machine Learning Model Selection

When selecting a machine learning model for detecting track gauge deviation, several factors are considered. The chosen model must excel in image segmentation and maintain minimal implementation complexity without sacrificing performance. After evaluating various options, several models from the YOLO library by Ultralytics were selected for comparison, including versions of YOLOv8, YOLOv9, and YOLOv11 [13]–[15].

The specific chosen model is from YOLOv9, and is the highest strength YOLOv9 segmentation model available, de-

noted by the “e” in the model’s name, “YOLOv9e-seg”. This model performs at a high level for both object detection and image segmentation and offers a pretrained model option.

The pretrained model option is pretrained using the Microsoft COCO dataset and can be fine-tuned using a custom-made dataset that is better fit for specific needs. The pretrained model is advantageous when the size of the training dataset is small. A comparison between the different versions of YOLO segmentation models can be found in the “Training Results” section of Section III.

E. Horizontal Track Gauge Deviation Detection Strategies

Several different methods of detecting track gauge deviation from a video source are examined to determine their effectiveness and consistency. First, detection is attempted using the railroad ties that span between the railroad lines as leveling guidelines to determine distance. In theory, measuring from one rail line to the other using the railroad ties as a guide should provide an accurate measurement, as the ties are ideally perpendicular to the rail lines. An example of close to perfect conditions can be found in Fig. 4. However, when observing



Fig. 4. Example of railroad ties in a close to perfect environment.

real-world conditions, these railroad ties are often heavily decayed, skewed, or missing entirely. The railroad ties are also often covered by the surrounding ballast on the track. An example image of obscured railroad ties can be found in Fig. 5. Additionally, this method would require the model to know where the railroad ties are to measure from those spots. This either requires our machine learning model to also detect railroad ties, or each tie needs to be spaced exactly the same distance apart. In a real-world scenario, each rail tie will never be equidistant from each other, and our machine learning model could never be consistent and accurate enough to identify each tie at the level required for sufficient results.

The second detection method we explored is utilizing a depth camera to measure how far each rail line is from the camera, and then comparing the values to see if one line is farther away than the other. This approach requires the



Fig. 5. Example of obscured railroad ties on rail tracks.

depth camera to be set up at exactly the center of where the two rail lines should be, as well as proper depth calibration, after which the height and angle of the camera must be kept consistent. This approach becomes difficult to replicate each time recording takes place.

The third detection method also relies on several aspects of the process being kept consistent. This approach measures the total number of pixels in between the centroids of each rail line detection, then applies a known scale factor to the pixel count, which results in an accurate distance calculation. One potential issue that arises with this approach is the centroids of each detection not being parallel to each other, unlike Fig. 6, which causes an incorrect measurement. Fig. 7 shows an example of this, which is caused by the camera angle including more of the left rail line than the right rail line. In order to

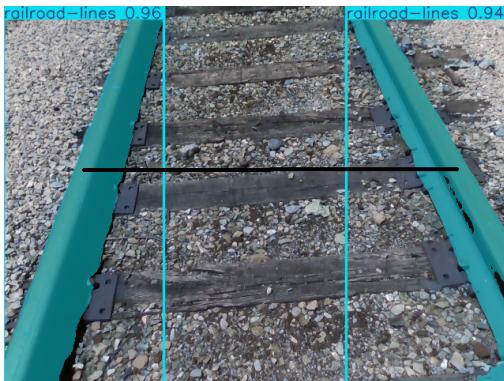


Fig. 6. Example of bounding boxes that give ideal centroid measurements.

combat this issue, instead of measuring the pixel count directly between each centroid, the pixel count from each centroid to the center of the image is collected, and then is combined to create the overall pixel distance. While this approach may cause the specific point on the rail being measured on each side to differ slightly in some cases, it prevents over measurements

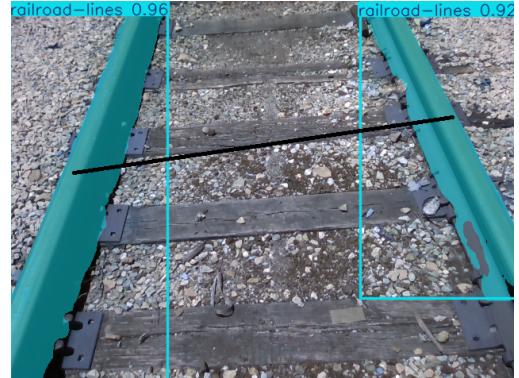


Fig. 7. Example of diagonal bounding box separation that results in an incorrect measurement.

such as in Fig. 7. A flowchart that details the horizontal track gauge deviation detection process can be found in Fig. 8.

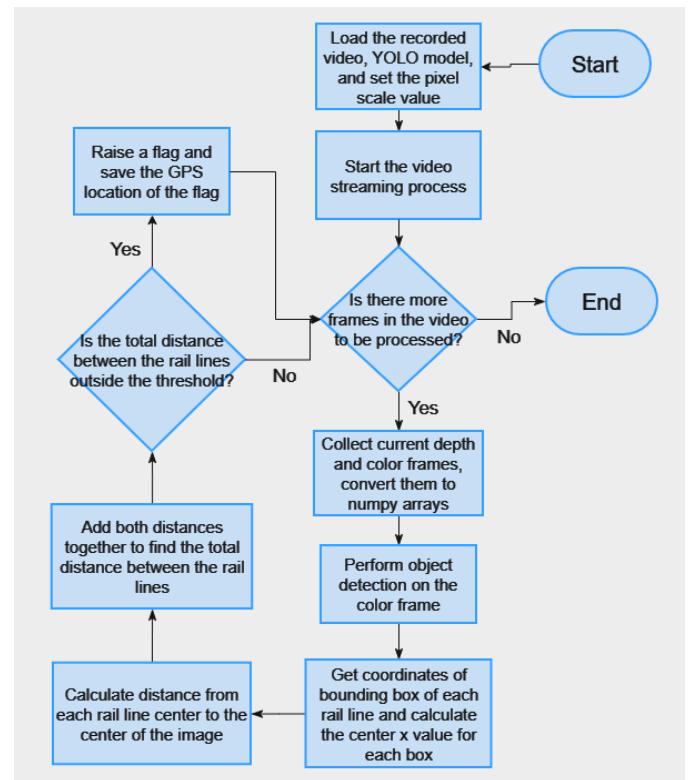


Fig. 8. Flowchart for the horizontal track gauge deviation detection process.

For this method to work, several things must be kept consistent. First, in order for the centroid of the rail line bounding box to be located on the surface of the track, the camera must be positioned directly vertical of the track. This makes sure that the rail line is not at an angle in the video, and therefore the center should always lie on the surface of the track. Second, the image/video dimensions and the pixel per meter scale factor must be determined for each instance. The dimensions of the image/video are important because they determine the pixel value where the vertical center of the

image is. The pixel per meter scale factor is used to convert the total number of pixels from each rail line to the center combined into a real-world measurement such as meters or feet.

The pixel per meter scale factor can be difficult to determine. A manual approach can be used, where a known distance between rail lines can be used to judge if the scale factor is appropriate. For example, the distance between the rail lines in Fig. 6 is known to be 55 3/4 inches. Knowing this, the pixels per meter value can be adjusted appropriately until the output measured distance is 55 3/4 inches. Alternatively, if the width of the rail line surface is known, the number of pixels the rail line surface makes up can be manually counted, which can then be converted into a scale factor. While the pixel per meter scale factor can be difficult and tedious to calculate, it allows for videos taken at different heights to be used, as long as the scale factor is properly set each time. While each track gauge detection method has its own shortcomings and limitations, mainly regarding how to keep the process consistent, the third method that measures the pixels has the most manageable factors. Results from this detection method can be found further in the paper, in the "Video Detection Results" section in Section III.

F. Vertical Track Height Deviation Detection Strategy

While horizontal track gauge deviation can be more obvious, vertical track height deviation is harder to detect but poses significant risks to the health of the railroad track. Vertical track height deviation occurs in two main ways: one rail line rising above the other, causing a sloped angle between the two rail lines, or both rail lines rising above the appropriate height.

In order to determine if one rail line has raised above the other and causes a sloped angle between the two, we employ a similar process to how we detect if both rail lines have raised above a threshold. Each corresponding depth and color frame are processed and converted to images that can be detected using the machine learning model. The "YOLOv9e-seg" machine learning model is used to detect rail lines in the color image, and the results are processed to find the depth data inside the detected bounding box.

The distance of the closest pixel inside the bounding box for each rail line is added to a list. Then, the two entries in the list, which correspond to the closest pixel in the detected bounding box for each rail line, are compared to each other. If the minimum depth pixel in the two rail lines being compared differ by a predetermined amount, it is judged that vertical track height deviation has occurred in at least one of the rail lines and a flag is raised.

Detecting if both rail lines have raised is significantly harder than only detecting one. One potential complication when trying to detect if both rail lines have raised is that our depth camera, which ideally is attached to a rail truck/car on the rail tracks, also raises the same amount. As the rail lines which the camera car is attached to raise, the entire camera contraption will raise the same amount, causing the rail lines to be a consistent distance from the camera, despite the significant

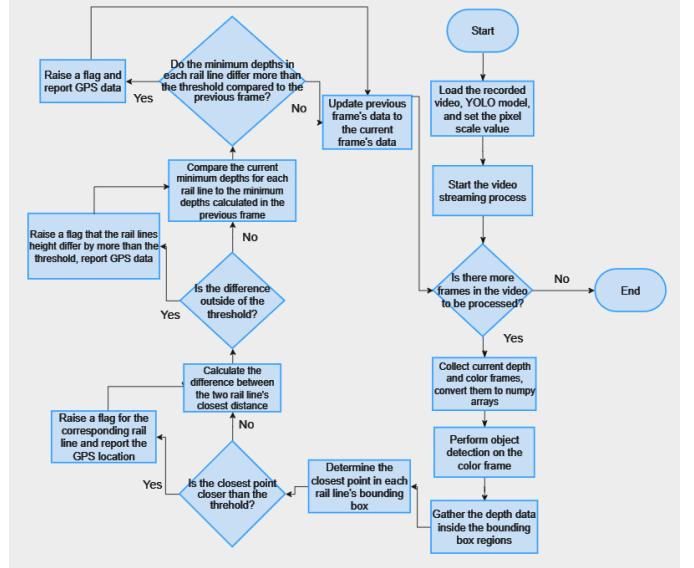


Fig. 9. Flowchart for the vertical track height deviation detection process.

change in elevation. This effect leads to missed detection of vertical track height deviation when both rail lines raise the same amount.

To combat this effect, the distances of each rail line are stored for the current frame. Then, the distances of the rail lines in the next frame are compared to the previous frame. A threshold is set for appropriate increases or decreases in track height, and if the current rail line's distances change more than the set threshold when compared to the previous frame, a flag is raised. A flowchart that details the process for vertical track height deviation can be found in Fig. 9.

III. RESULTS

A. Training Results

Each model version was trained for 500 epochs total, and each model version's metrics, including recall, precision, and F1 score, were recorded after the training process was complete. An important note, all of the metrics in Tables I are for the "railroad-tracks" class, and do not include any results from detecting the "ballast" class. The results of the training can be found below in Table I :

TABLE I
COMPARISON OF DIFFERENT YOLO SEGMENTATION MODELS.

Model	F1 Score	Recall	Precision	Time (ms)
YOLOv8x-seg	0.759	0.886	0.818	16.5
YOLOv8l-seg	0.754	0.856	0.674	4.5
YOLOv9c-seg	0.826	0.897	0.765	4.6
YOLOv9e-seg	0.869	0.875	0.864	10.4
YOLO11x-seg	0.799	0.819	0.78	7.3

The results from Table I show that the YOLOv9e-seg model version was able to achieve an F1 score of 0.869, which is the highest of the group. The other model versions demonstrate F1 scores significantly lower than the YOLOv9e-seg, with the closest being 0.826 from the YOLOv9c-seg model.

An additional important metric for our usage is recall. A high recall value is essential for our machine learning model as false negatives can be dangerous and costly, while a limited number of false positives can be accepted. The YOLOv9e-seg model version obtains a recall score of 0.875, compared to the highest recall value of 0.897 from the YOLOv9c-seg model. Despite the slightly higher recall value, when considering all of the training results, the YOLOv9e-seg model version performs the best and was selected as the model for our processes.

B. Video Detection Results

For the video detection results to be both consistent and accurate, a depth camera must be used to take the vertical video of both track lines. Our project utilizes the Intel® RealSense™ Depth Camera D435. Without a depth camera to provide distance measurements of each rail line, track gauge deviation detection from the video source would not be possible. An example of a color image and a depth image taken from the depth camera can be found in Fig. 10 and Fig. 11. The vertical track height deviation detection techniques rely



Fig. 10. Example of color video from the depth camera.

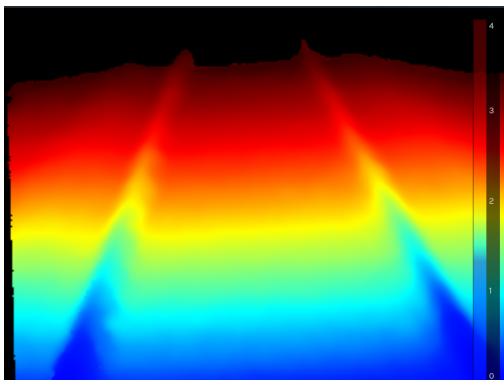


Fig. 11. Corresponding depth video from the depth camera.

on depth camera measurements to produce accurate results and properly detect when track height deviation has occurred. In addition to the depth camera requirement, several factors must be kept consistent between recordings to ensure consistency. First, the angle of the depth camera must be perpendicular

and vertical to the track surface. This will guarantee that the depth camera measurements are as accurate as possible and that the horizontal and vertical track height deviation detection techniques work as intended. Second, the height of the depth camera relative to the track should be kept the same. While the depth camera can account for different distances, it must be calibrated before each use, so it is safer to keep the camera height consistent.

Preliminary results using video taken using the depth camera in a handheld manner are promising and show the potential of video based track gauge deviation detection. An example frame after the machine learning model has applied its detections can be found in Fig. 12.



Fig. 12. Example of the detection being applied by the machine learning model using depth camera.

After the detection is applied, the calculations to determine horizontal and vertical track height deviation can be made. The distance between the two rail lines is estimated to be 1.44 meters, and neither rail line has raised above an acceptable amount. Additionally, the difference in height of each rail line is not outside of the threshold. Finally, the heights of each rail line have not changed significantly when compared to the previous video frame. Therefore, neither horizontal or vertical track height deviation is deemed to have occurred.

While this example is limited by the nature of the video recording, it still demonstrates the potential effectiveness of detection from video. Many of the consistencies that the video detection methods require are absent from this example, but a reasonable estimation can still be made. More precise measurements can be gathered through the use of a track car that holds the camera in a stable and consistent position across the length of the entire video.

C. Limitations

The process of detecting something as precise as track gauge deviation naturally comes with a large set of limitations and drawbacks. The majority of limitations stem from the many variables that come with the video recording process.

If the recorded videos are not shot at a vertical angle, or the camera is not positioned in the center of the rail lines, then incorrect measurements can be made due to incorrect depth camera measurements. In addition, any objects on the rail track

surface or surrounding the rail lines could possibly interfere with the machine learning model's ability to make predictions. These detection strategies rely on clear and unobstructed videos to perform at their highest capability.

Another significant limitation to these track gauge deviation detection methods is their ability to work with different types of track sections. The ideal scenario is straight track sections with no other rail lines around. Often times, especially at rail intersections, there are several sets of rail lines within close proximity of each other, which cause more than two rail lines to be detected, breaking the process.

Sections of curved track is another area where potential issues arise. When dealing with the horizontal track gauge deviation detection, the centroids of each detection bounding box are assumed to be located on the surface of the rail track. The more the section of track is curved, the less likely this is to be true. Therefore, these detection processes should only be applied in the context of straight sections of rail track, where only two rail lines are visible in the video footage. This ensures erratic and incorrect measurements are not made, and false flags and warnings are not raised.

Despite the limitations of the video detection method, the overall process significantly cuts down on the time and human labor needed to inspect rail track for track gauge deviation. Our ideal track scenario encompasses the biggest grouping of track, straight track, ahead of curved track and rail intersections. When keeping the limitations and necessary consistencies in mind, our detection techniques provide an efficient and reliable way to detect track gauge deviation.

IV. CONCLUSION

The integration of video and depth cameras with machine learning has demonstrated significant potential in detecting track gauge deviations in railway infrastructure. This innovative approach addresses the primary objective of reducing the time, human labor, and costs associated with traditional inspection methods. By automating the detection process, we can achieve more accurate and efficient identification of horizontal and vertical track height deviations, which are crucial for maintaining the safety and reliability of railways.

Our study has shown that, under controlled conditions, the use of video and depth camera recordings can effectively detect track gauge deviations. The combination of converting pixels to real-world measurements and utilizing depth camera data has yielded promising preliminary results, indicating high levels of accuracy in estimating track gauge deviations. This method offers a cost-effective and efficient solution, making it a valuable tool for railway maintenance. However, the approach is not without its limitations. Challenges such as inaccurate measurements on curved track segments, rail intersections, obstructions on the track, and inconsistent camera angles and distances must be addressed. These limitations highlight the need for further research to refine detection strategies and improve the robustness of the system. Future work should focus on developing specialized detection tech-

niques for complex track geometries and testing a broader range of machine learning models to enhance performance.

Despite these challenges, the benefits of this technology are substantial. The ability to conduct more frequent and thorough inspections will lead to a significant reduction in the time and human labor required for railway maintenance. This, in turn, will lower the overall costs and enable more regular inspections, thereby enhancing the safety and efficiency of railroads.

ACKNOWLEDGMENT

We extend our sincere gratitude to Engineer Research and Development Center for their generous support and funding.

REFERENCES

- [1] A. Lasisi and N. Attoh-Okine, "Machine learning ensembles and rail defects prediction: Multilayer stacking methodology," *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, vol. 5, no. 4, 2019.
- [2] Y. Liao, L. Han, H. Wang, and H. Zhang, "Prediction models for railway track geometry degradation using machine learning methods: A review," *Sensors (Basel)*, vol. 22, no. 19, 2022.
- [3] J. Sresakoolchai and S. Kaewunruen, "Railway defect detection based on track geometry using supervised and unsupervised machine learning," *Structural Health Monitoring*, vol. 21, no. 4, 2022.
- [4] A. Falamarzi, S. Moridpour, M. Nazem, and S. Cheraghi, "Prediction of tram track gauge deviation using artificial neural network and support vector regression," *Australian Journal of Civil Engineering*, vol. 17, no. 1, 2019.
- [5] K. Popov, R. De Bold, H. K. Chai, M. Forde, C. Ho, J. Hyslip, H. Kashani, R. Kelly, S. Hsu, and M. Rippin, "Data-driven track geometry fault localisation using unsupervised machine learning," *Construction and Building Materials*, vol. 377, 2023.
- [6] A. Pires, M. Viana, L. Scaramussa, G. Santos, P. Ramos, and A. Santos, "Measuring vertical track irregularities from instrumented heavy haul railway vehicle data using machine learning," *Eng. Appl. Artif. Intell.*, vol. 127, no. PA, 2024.
- [7] Z. Yu, C. Feng, M.-Y. Liu, and S. Ramalingam, "Casenet: Deep category-aware semantic edge detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [8] H. Rezatofighi, N. Tsai, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," *arXiv preprint arXiv:1902.09630*, 2019.
- [9] R. Tang, L. De Donato, N. Besinovic, F. Flammini, R. M. Goverde, Z. Lin, R. Liu, T. Tang, V. Vittorini, and Z. Wang, "A literature review of artificial intelligence applications in railway systems," *Transportation Research Part C: Emerging Technologies*, vol. 140, 2022.
- [10] V. CHENNAI, "Track crack damage detection dataset," Online, 2023, visited on 2024-09-03. [Online]. Available: <https://universe.roboflow.com/vit-chennai-japsi/track-crack-damage-detection>
- [11] MPMC, "Railway tracks dataset," Online, 2023, visited on 2024-09-03. [Online]. Available: <https://universe.roboflow.com/mpmc/railway-tracks-tpu61>
- [12] M. Learning and IOT, "Iotrads dataset," Online, 2023, visited on 2024-09-03. [Online]. Available: <https://universe.roboflow.com/machine-learning-and-iot/iotrads>
- [13] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [14] C.-Y. Wang and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," *arXiv preprint arXiv:2402.13616*, 2024.
- [15] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: <https://github.com/ultralytics/ultralytics>