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

2024 IEEE 15th Annual Information Technology, Electronics and Mobile Communication Conference (IEEE-IEMCON)
October 24-26, 2024

CONNOR STONESTREET, HWAPYEONG SONG, HUSNU NARMAN, PINGPING ZHU, AMMAR ALZARRAD

Outline:

- Introduction
- Methodology
- Results
- Limitations
- Conclusion

Problem:

- Track gauge deviation: edges of rail lines become too close together or too far apart
- Can cause trains to derail if left untreated
- Human-led track inspections are time-consuming and costly

Previous Solutions:

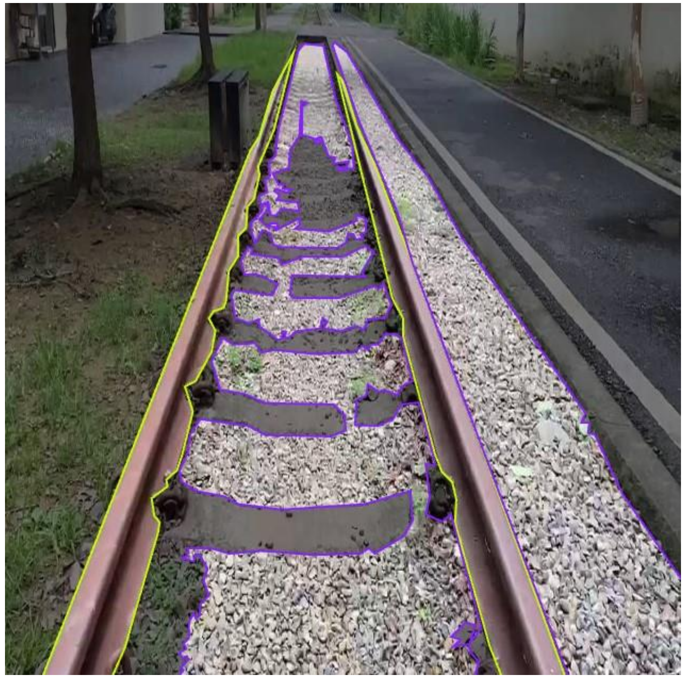
- Provide a solid foundation
- Focus on prediction of future
- Ex: Weibull, autoencoders, edge detection
- Our objective combines objection detection and **depth sensors**

Our Approach:

- Utilize machine learning to detect track gauge deviation directly from video source
- Train machine learning model to detect rail lines
- Use depth camera sensors to calculate distances
- Provides instant feedback of issues

Dataset Creation:

- Dataset assembled using Roboflow
- Images selected on real-world relevance
- Annotation done using Roboflow tools
- Preprocessing techniques applied to grow dataset size



Model Selection:

- Models trained using Google Colab notebook
- Models must excel at image segmentation
- Models from the YOLO (You Only Look Once) library were tested
- Simple Python implementation to extract results

Model Training Results:

- Each model trained for 500 epochs
- F1 score, recall, precision, and inference time per image recorded
- High recall essential
- Inference time largely ignored
- YOLOv9e-seg version chosen

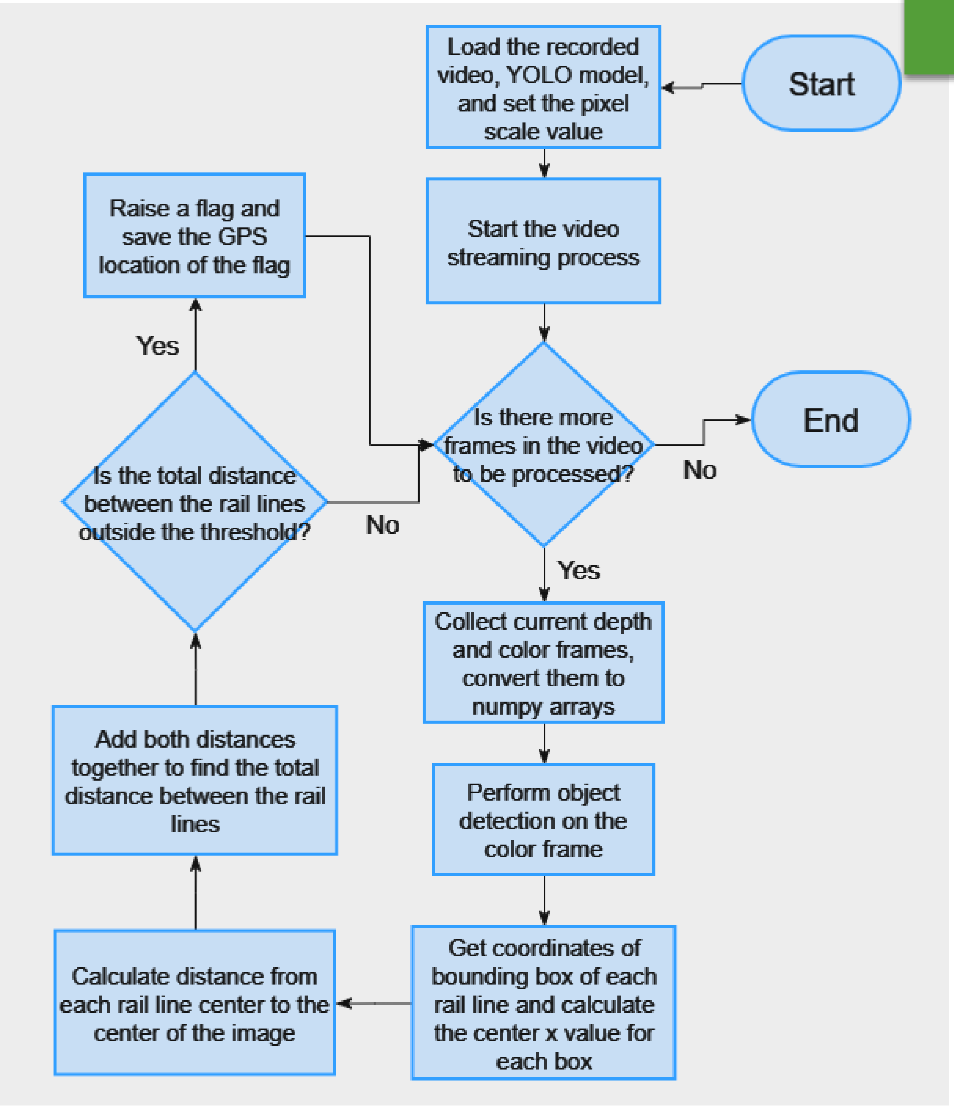
Model	F1 Score	Recall	Precision	Time (ms)
YOLOv8x-seg	0.759	0.886	0.818	16.5
YOLOv8l-seg	0.754	0.856	0.674	**4.5**
YOLOv9c-seg	0.826	**0.897**	0.765	4.6
YOLOv9e-seg	**0.869**	0.875	**0.864**	10.4
YOLO11x-seg	0.799	0.819	0.78	7.3

Track Gauge Deviation Detection Ideas:

1. Use the railroad ties guide for distance measurement
   - Often decayed, skewed, or missing
   - Requires model to detect rail ties
2. Use depth camera to find distance from camera
   - Camera must be set up perfect
3. Use pixels as a scale
   - Camera must be positioned vertical to the track
   - A scale factor must be determined each time
   - Easier to replicate across recordings

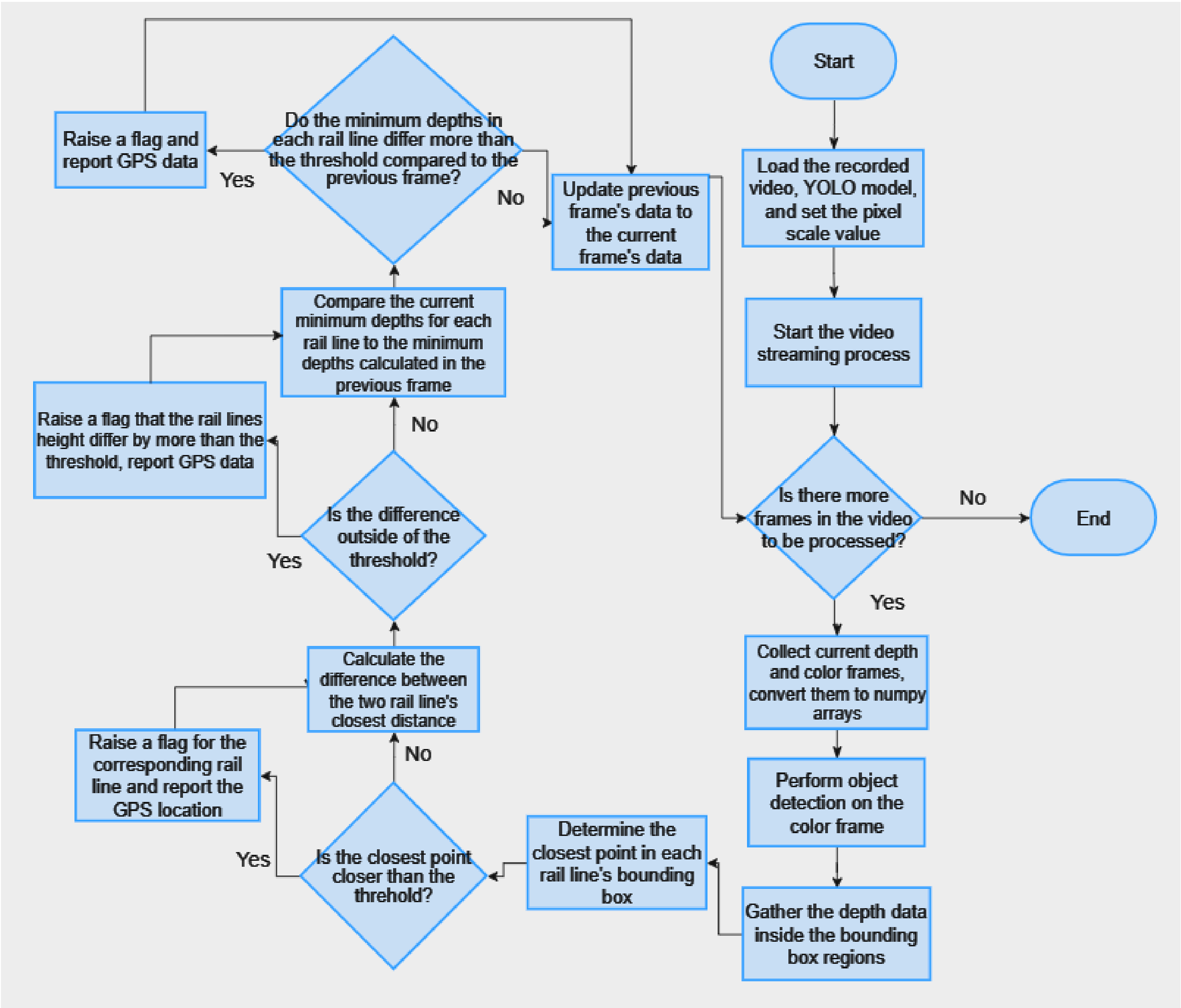
Implemented Solution:

- We chose to proceed with the third solution
- Representative flowchart can be seen on the right



Vertical Track Height Deviation:

- Track height can also change over time
- Strategy is to compare depth data across frames



Code Sample Output

```
0: 480x640 2 railroad-liness, 6051.4ms
Speed: 15.0ms preprocess, 6051.4ms inference, 39.2ms postprocess per image at shape (1, 3, 480, 640)
Rail line within bounding box (47,172)-(93,341) has no pixel closer than the threshold.
Rail line within bounding box (280,238)-(442,477) has no pixel closer than the threshold.
Rail 1 distance from center: 0.9368421052631579 meters
Rail 2 distance from center: 0.6305263157894737 meters
Total distance between the two rail lines: 1.5673684210526315 meters
Horizontal track gauge deviation detected, distance between rail lines: 1.5673684210526315 meters.
[1437, 1277]
Rail lines are within the acceptable depth difference.
```

Limitations

- Position of the depth camera must be vertical to the track surface
- Calculating the scale value can be difficult
- Track curvature increases error potential
- Rail junctions can cause errors

Conclusion

- Detecting issues such as track gauge deviation is essential
- Machine learning allows for more frequent and less expensive inspections
- Under controlled conditions the approach is effective
- Further work is needed to refine detections and improve the robustness
- Overall, our approach will enhance the safety and efficiency of railway systems

Questions?

MARSHALL™ | COMPUTER SCIENCE

MARSHALL UNIVERSITY
JOHN MARSHALL
1837