Railway Tracks Inspection

---

1. Problem Statement

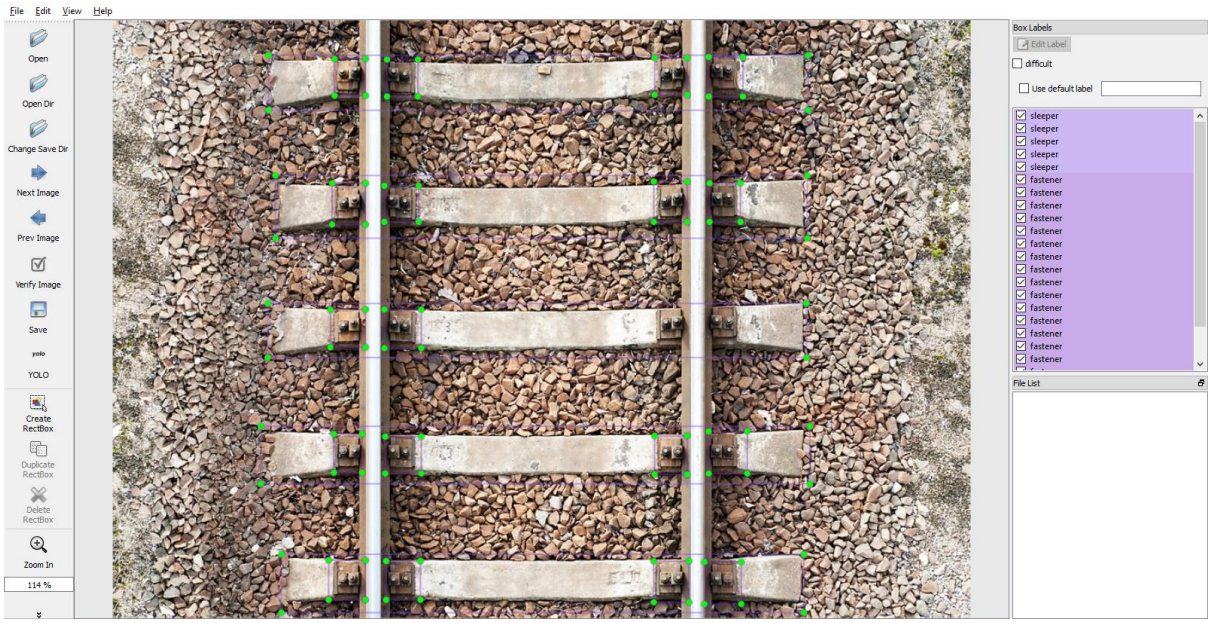
Operations of Railways have always placed “safety” as the priority. Managing “public safety” requires integral risk monitoring and management. Rail inspection is the practice of examining rail tracks for flaws that could lead to catastrophic failures.

2. Proposed Approach

We use a four-step process comprising of data collection, data labeling, image processing, image segmentation, detection of sleepers and identifying object on tracks.

The data to be collected are the raw images or image frames of a live video footage of railway tracks captured by drones. The image below is a sample of the image required.

Data collected will be labelled using an open-source tool called LabelImg. The screenshot below is how the track would look like after labelling.



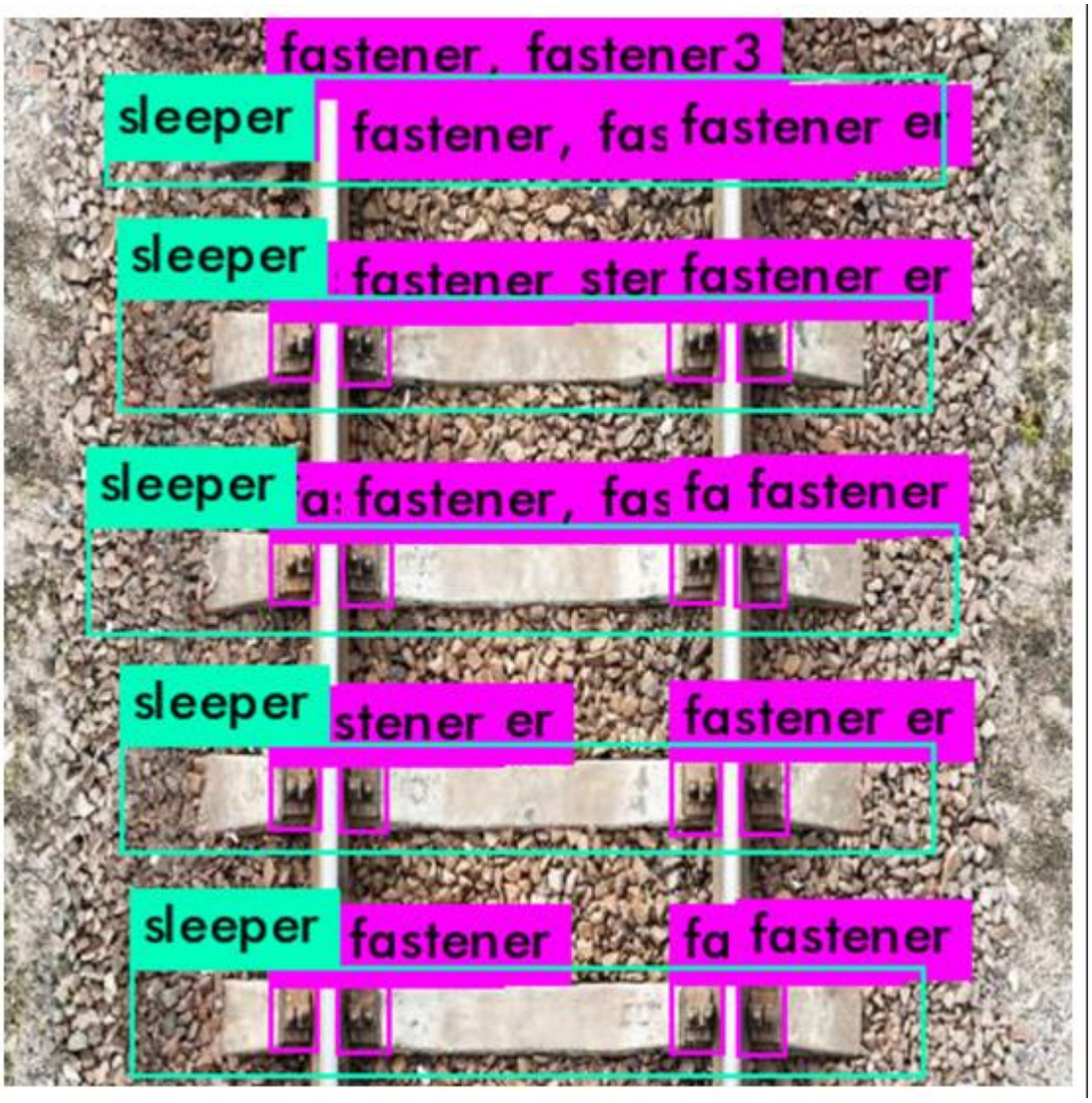
3. Proposed Models

3.1. Detecting fasteners on railway tracks

For detecting the fasteners on the railway tracks, the images are manually labeled using labeling. The model to be used here is YOLOv4 tiny object detection model. YOLO, short for You Only Look Once, are popular for fast object detection.

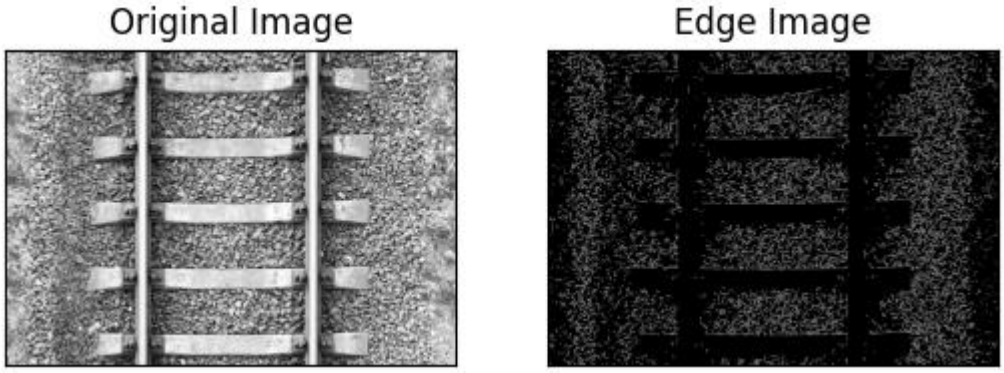
Tiny Yolo object detection has a slight accuracy tradeoff, but it makes it up with its speed. It is much faster than other standard yolo object detection models.

To detect a missing fastener, for every frame, number of detected fasteners should be equal to 4 times of the number of sleepers, otherwise it will give an alert.



3.2. Railway Sleeper Crack detection

For detection of Cracks on Railway Sleepers, canny edge detection can be used. Any visible crack can be seen through canny edge detection.



3.3 Railway Track Obstacle detection

Depending on the objects we need to detect, model can be trained accordingly using Yolo v3 algorithm. Objects near a particular threshold value of the rail track will get detected others will be ignored.

4. Project Structure

```
|──custom_images
|     |──1.jpg, 2.jpg, …. (training images, annotations for Bounding Box Coordinates)
|     |──classes.NAMES
|     |──creating-files-data-and-name.py
|     |──creating-train-and-test-txt-files.py
|──tracks_img
|     |──info.jpg
|     |──processed.jpg
|──yolo_model_cfg
|     |──yolov3_custom.cfg
|──yolo_model_weights
|     |──yolov3_custom_final.weights
|──custom.py
|──main.py
```

custom_images contain the images for training the model along with labeled bounding box annotations, classes.NAMES file for pretrained YOLO model for finding ROI.
creating-files-data-and-name.py and creating-train-and-test-txt-files.py are used for splitting the images and annotations files into train and test sets.

tracks_img contains the cropped ROI and transformed image to be fitted to model.

yolo_model_cfg and yolo_model_weights contain the custom YOLO model's configuration and weights file after training – to be used for testing on images.

custom.py is used for detection and main YOLO's algorithm with appropriate confidence thresholds.

main.py contains the python script.

5. Project Requirements

- Dataset for the above use cases is needed.
- Drones that can capture high resolution images of the tracks.
- Ubuntu 16.04 or later (64-bit) Operating System
- macOS 10.12.6 (Sierra) or later (64-bit) (no GPU support)
- Windows 7 or later (64-bit) Operating System
- GPU support requires a CUDA®-enabled card (Ubuntu and Windows)

6. Challenges/Issues Faced

- Cracks not clearly visible might not get detected.
- Shadow of the track can hide fasteners.
- Improper lighting conditions for images.

- Images at different angles that do not capture the information clearly.

7. Current Status

Currently working on collection of data.

References

1. https://github.com/tzutalin/labelImg
2. https://justin-liang.com/tutorials/canny/
3. https://pjreddie.com/darknet/yolo/