Automatic Detection of Railway Track Obstacles using a Monocular Camera

Guilherme Kaname Reis Kano
guilherme.kano@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

June 2020

Abstract

Railway tracks are one of the most important factors for the social-economical development of a nation. Due to its unquestionable impact, railway monitorization should be defined as a high priority task. In this work two solutions are developed and tested, based on computer vision and *deep learning* for the automatic obstacle detection in railway tracks, such as fallen trees or large dimension rocks resulting from rockslides. The computer vision-based algorithm employs image processing techniques such as edge detection and image segmentation using superpixels. The *deep learning* approach uses a pretrained network (transfer learning) which is modified and retrained over the target dataset. In this dissertation two types of obstacles are considered: direct and indirect obstacles. The first refers to elements directly obstructing the railway rails. The second refers to large obstacles located in the vicinity of the rails, susceptible of colliding with a passing train. Both algorithms perform upon RGB images of railway tracks, captured by a single monocular camera. Since no data set depicting railway track obstacles is known to exist, it was required to artificially create one to develop and test both methods developed. In a final phase, both algorithms were tested on the scale model composed of a *webcam* and railway tracks models. It was verified that the best results were obtained with the *deep learning* algorithm.

Keywords: Railway Tracks, Obstacles, Automatic Detection, Computer Vision, Convolutional Neural Networks

1. Introduction

From public transportation, to international trade of goods, rail transportation industry has an undeniable impact on a nation's social and economic development. To illustrate this, its impact on United States' gross domestic product (GDP) was a total amount of $74.2 billion to the U.S GDP in 2017 (approximately 0.4 percent of the total U.S. economy)[1].

While being a proven essential asset for a nation's development, rail transportation accidents have serious and social impacts. The derailment and collision of a freight train can incur high expenses and losses, while accidents involving human passenger vehicles may lead in injuries and casualties[2]. Railway companies have been continuously aiming to the improvement of safety and maintenance measures. These efforts are reflected in official reports, such as the one issued by UIC, a worldwide association composed by several European railway institutions, which reported that 81.50% of the accidents registered in 2017 were due to third parties and weather and environmental causes [3]

Preventive measures to minimize rock slide based collision often lay on the installation of protective barriers and signal fences. However, this approach can be expensive if the area under monitorization is extensive, as if the case for the Norwegian rail lines, located in mountainous areas that extended over 4000km. Alternative solutions such as automatic inspection using autonomous vehicles could play a vital role in railway monitoring.

1.1. Railway Maintenance and Inspection

One of the fields in railway track maintenance consists on the rail's surface inspection. Due to the multiple cyclical loads that rails are subjected to, their surface is prone to the effects of fatigue. Regarding this topic, many researchers have been developing deep learning base solutions [1, 2].

The detection of obstacles in railway tracks is a different field, that is still focused on railway track

[1] Oxford Economics. The economic impact of railway suppliers in the U.S.- Technical report,2018

[2] E. Beswick. Around 40 injured in Austria train accident, say police. Euronews, 2018. Accessed: 2019-08-27

[3] Significant accidents 2017 public report. Retrieved April 10, 2019.

monitorization. Obstacle detection solutions often rely the use of LiDAR systems, which are used for monitoring level crossing, one of the location where those incidents are most prone to occur.

To avoid the use of high cost sensors, often limited by an effective range, solutions often drift to imaging sensors and imaging processing techniques. maire2010obstacle

Many researchers [3, 4, 5] have focused on the use of computer vision techniques, such as edge detectors, to extract the rail edges, as the starting point for the detection of obstacles. In [5], authors, use a window slider technique that encompass a portion of the detected rail edges. At each window, the orientation and average intensity gradient of the rail edge is computed. If both the angle and intensity between two edges of adjacent windows do not lay under a similarity threshold, the rail is said to be obstructed.

Other approaches, such as the one explored by Ukai et al. in [6], combine the information retrieved from different sensors. Authors developed an on-board monitorization device that employs both image recognition and high resolution radar techniques to detect obstacles in the railway tracks.

Contrary to the available research of surface and railway track components maintenance using deep learning algorithms, it appears that no significant search is being conducted in obstacle detection task using the same methods. While the rail surface inspection allows for the collection of a vast amount of data, subsequently provided to the research community, the same do not extends to the obstacle detection field, as there is a significantly smaller number of registered observations, as mention by Nakhaee et al. in[7].

1.2. Objectives and Contributions

This thesis aims to develop, evaluate and compare computer vision and deep learning approaches for the automatic detection of obstacles in railway tracks, while being limited to the analysis of RGB railway tracks images captured by a single monocular camera.

The proposed detection algorithm operates upon images of railway tracks taken by an inspection vehicle mounted camera whose orientation is kept approximately constant over the inspection course. This constraint dictates the normal operation conditions on which the detection algorithm operates upon: the railway tracks' relative position and orientation within the captured image, with the image vertical bisector being approximately aligned with the middle of the railway rails.

Two types of obstacles are considered: direct obstacles and indirect obstacles. Direct obstacles refers to objects obstructing the raiway rails. Indirect obstacles are objects that, despite not being obstructing the rails, are located in the vicinity of the rails, susceptible of colliding with a passing train.

The main contributions achieved by this work are the creation and collection of a novel dataset of railway track images with and with no obstacles, that can be used by the academic community. Secondly, the algorithm developed and addressed in section 5 proposes an alternative to railway rails segmentation based on the detection of straight lines with the Hough transform. This topic is further addressed by Kano et al. in [8].

Finally, this work addresses main advantages and drawbacks when adopting vision and deep learning in the topic of automatic detection in railway tracks.

2. Theoretical Background: Computer Vision

2.1. Linear Image Filtering

Linear image filtering consists of modifying an image and it can be used for feature detection, such as finding edges, and for noise suppression. The target pixel in the image I at location (u, v) is updated according to

$$g(u, v) = \sum_{k,l} w(k, l)I(u + k, v + l) \qquad (1)$$

with (u; v) and (k; l) being the image and the mask indices, respectively.

Sobel operator is one common edge detector, which can be defined by its horizontal and vertical components G_x and G_y, defined as:

$$G_x = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} G_y = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad (2)$$

Alternatively, noise reduction can be achieved using a discrete approximation of the 2-D Gaussian function, defined by a $N \times N$ matrix and the standard deviation of the Gaussian function, σ, which measures the filter intensity.

2.2. Image Segmentation

Image segmentation can be performed based on discontinuity, with the use of edge detection algorithms such as the Sobel operator and the Canny edge detector, and similarity of intensity values, such as the image oversegmentation using superpixels.

Superpixels are cluster of pixels that share perceptually similar features such as color or grayscale intensity levels. SLIC decomposes the image into several clustering using a 5-D distance definition, that accounts for both the color similarity in the $L^*a^*b^*$ color space and the 2-D euclidean distance.

2.3. Region-Based Descriptors

Region-based descriptors translate shape properties of a given region into a numeric feature. Major and

minor axis, correspond the maximum and minimum distances between the utmost pixels in a given region. These descriptors can be derived from other shapes, such as ellipses, that share the same second central moment as the considered binary object. Additionally, the orientation of the shape, given by the angle between the major axis and the horizontal axis can also be computed using this ellipses (Fig. 1).

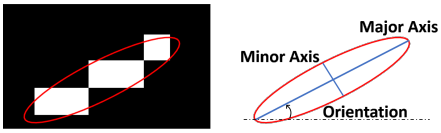


Figure 1: Major and minor axis and orientation descriptors of the region (left) can be obtained from the ellipses with the same second central moment (right).

The Euler number (E) of a binary image is the difference between the number of the connected (C) elements and the number of holes (H):

$$E = C - H \tag{3}$$

3. Theoretical Background: Deep Learning

3.1. Learning and Computer Vision

Deep learning has been a popular strategy among many scientific applications such as medical imaging diagnosis [9].

Being verified as an effective tool to solve many computer vision tasks, deep learning requires a large data set of examples to train the models. Regarding this matter, one of the most important contributors is the ImageNet, an image database containing over 14 million images, where many state of the art visual recognition models have been trained on. Many of these models have been used for transfer learning (TL) which consists in using a pretrained model that has already been trained over a large-scale dataset, and retrained in a different domain.

3.2. Convolutional Neural Networks

Convolutional neural networks (CNN) are a type of deep learning models that have been extensively used in many visual recognition tasks such as image classification and object detection. These models can be divided into two sets: a feature extraction and a classification set. The feature extraction set is composed by sets of convolutional, activation function and pooling layers, which ultimately extract a the feature vector from the input image. This feature vector is then used by the classification set to obtain the model's estimation.

One example of a CNN is the GoogLeNet, a model which is composed by several inception modules.

4. Railway Dataset

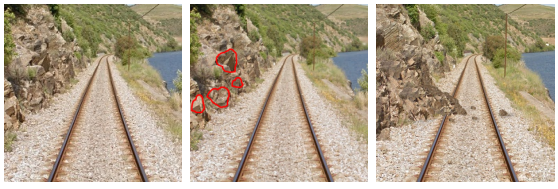
The railway dataset is composed by railway track images with and without obstacles.

The *No Obstacle* dataset was assembled from two sources. The first source is a compilation of several images obtained from *Google TrainView*, and retrieved by Pinho and Santos in [10]. The second source was compiled from a collection of Youtube videos where the frames were collected.

Since there is no large dataset of railway tracks images with obstacles, the *Obstacle* dataset had to be artificially generated.

The obstacle dataset images were generated following two different approaches. The first approach uses the algorithm developed by Pinho and Santos in [10] that automatically generates obstacle instances by placing the obstacle images (*.png* images of trees and rocks) in the original image.

Alternatively, in the second method, the obstacle instances were manually generated using image manipulation software (*GIMP*), by outlining specific contours in the target image, that could represent obstacles in a real scenario. The obstacle instances were generated by placing the contours in such a manner to simulate rock slides or landslides (Fig. 2).



(a) Target image. (b) Contour outlining (c) Obstacle instances.

Figure 2: Process of obtaining an obstacle instance with GIMP.

Table 1 describes the number of images in the complete dataset (no obstacles and obstacles).

Table 1: Data set dimension

Dataset Dimension		
No Obstacle	Obstacle	Total:
1572	747	2319

5. Obstacle Detection - Computer Vision Algorithm (*DetectCV*)

The obstacle detection based computer vision algorithm can be divided into the 3 main phases: a preprocessing, a segmentation and an obstacle detection phase.

5.1. Preprocessing Phase

First, the input image is subject to an histogram equalization to enhance the overall image contrast,

followed by a Gaussian filter is applied to diminish the presence of high frequency details that can later be detected during the edge detection phase. Then, a Region of Interest (ROI) is defined to limit the search area where the edge detection are performed upon.

The ROI is obtained by defining a binary mask, with the desired region represented by foreground pixels and it is obtained by first considering a subregion A whose frontiers are defined by:

$$\begin{cases} y & < 0.6 \cdot height_{Image} \\ x & < 0.5 \cdot width_{Image} \\ y & < 2x \end{cases} \quad (4)$$

where $y = 2x$ is the equation of the line defined by the points $p1 = (0, 0)$ and $p2 = (0, 3 \cdot widht_{Image}, width_{Image}, 0.6 \cdot heightImage)$ (Fig. 3). Then, it is mirrored around the image's vertical bissector to obtain the ROI.

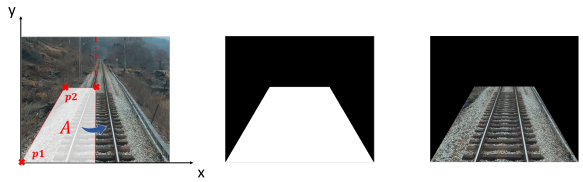


Figure 3: Subregion A defined by equations (4) (left), ROI (middle) and resulting image (right).

5.2. Segmentation Phase

In this phase, the rail edges are extracted from the image obtained after the ROI extraction (Fig. 3 (right)). Two edge detection algorithms are tested: the Sobel operator and the Canny edge detector, with the Sobel's horizontal component (2), leading to an edge map with fewer unwanted details and an accurate representation of the rail edges.

This edge map is then subject to a 3 phase reconstruction process. First, the edge map is subject to closing operation using a square 10×10 SE, with the intent of partially reconstructing every BLOB (rail or non rail). Then, the selection of the true rail BLOBs from the remaining ones are done using a orientation threshold, defined by computing the orientation of each rail (left and right) over 10 examples. The left and right rail's orientation threshold's, θ_{left} and θ_{right}, lower and upper limits are the lowest and highest orientations, obtained over the 10 examples:

$$\begin{aligned} \theta_{left\ rail} &= [61.1, 79.2]\ \ deg \\ \theta_{right\ rail} &= [-82.9, -64.1]\ \ deg \end{aligned} \quad (5)$$

which can be used to obtained the global orientation threshold:

$$\theta_{global} = \theta_{left\ rail} \cup |\theta_{right\ rail}| = [61, 83]\ \ deg \quad (6)$$

Therefore, a BLOB is classified as true rail BLOB if $|\theta_{BLOB}| \in \theta_{global}$.

Having obtained the partially reconstructed rail BLOBs, a closing operation using a square 50×50 SE, is perform upon each individual rail BLOB to obtained the final binary representation of the rails.

5.3. Obstacle Detection Phase

The obstacle detection phase is divided into two modules: the first is responsible for detecting the presence of direct obstacles obstructing the rails and second is only concern to detect large obstacles in the vicinity of the railway tracks.

Direct obstacles can be detected by computing the vertical distance $VertDist$ of each rail. After considering two possible definitions for this measurement, by either evaluating the major axis length and orientation of each rail (Fig. 4 middle) or by accessing the height of the corresponding bounding box (Fig. 4 right).

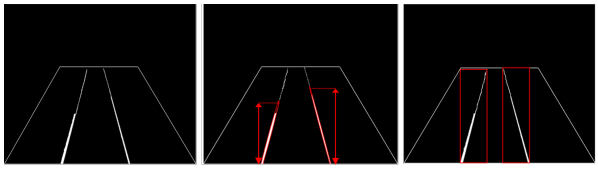


Figure 4: Binary representation of the rails (left) and $VertDist$ definition using major axis and orientation(middle) and bounding box height (right).

A rail is classified as being obstructed if $VertDist < Threshold \cdot height_{ROI}$. The value of $Threshold$ is defined by selecting 10 examples of images representing with and without obstructions, and for each rail, the corresponding value for the ratio $\frac{VertDist}{height_{ROI}}$. For the cases with no obstruction, the ratio had a minimum value of 0.85, and maximum value of 0.79 over the obstructed cases. Considering that, a $Threshold = 0,8$ is accepted. Fig. 5 (left) shows a case where no obstructions are detected and Fig. 5 (right) a case where the left rail is obstructed, according to $VertDist < 0.8 \cdot height_{ROI}$. Both cases were not used in for the estimation of the $Threshold$.

If no obstruction is found, the algorithm must look for the surrounding areas of the rails.

To obtain a binary image for each area (left, middle and right), the ROI image is subject to a superpixel over segmentation using the SLIC algorithm. Then, a clustering k-means algorithm, k=3, is used to decompose the image into 3 colored regions. Since the background pixels (represented by

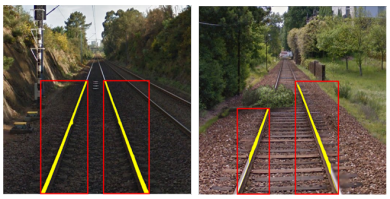


Figure 5: No obstruction detected (left) right rail obstructed (right).

the black group) occupy the majority of the image's area (this can be proven by computing the background area in Fig. 3 (middle)), the binary representation of the surrounding areas can be isolated by selecting the colored group with the second highest pixel count (Fig. 6).

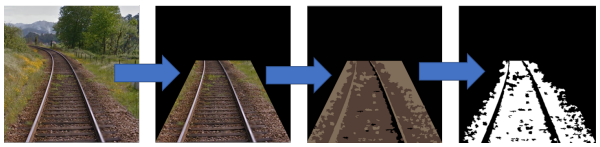


Figure 6: Process of obtaining the surrounding regions' binary representation.

With the binary representation of the surrounding region in a single image, the next step consists of separating it into 3 separate binary images, corresponding to the left, middle and right regions. Using the orientation θ and centroid coordinates $[x_{Cent}, y_{Cent}]$ of both rail BLOBs, it is possible to obtain the parametric equation of two lines, y_{left} and y_{right}, that separate the three regions. The left line is defined as:

$$y_{left} = m_{left}x + b_{left} \tag{7}$$

where $b_{esq} = y_{Cent_{left}} - m_{left}x_{Cent_{left}}$ and $m_{left} = \tan(\theta_{left})$.

The same reasoning is applied to the right line.

The left region (Fig. 7 third image) is composed by the foreground pixels with coordinates (x, y) that satisfy:

$$y > m_{left}\ x + b_{left} \tag{8}$$

the middle region (Fig. 7 fourth image) by pixel such that satisfy

$$\begin{aligned} y &< m_{left}\ x + b_{left} \\ y &< m_{right}\ x + b_{right} \end{aligned} \tag{9}$$

and the right region (Fig. 7 fifth image) by pixels that satisfy

$$y > m_{right}\ x + b_{right} \tag{10}$$

Each individual binary image is subject to a closing and opening operation. The decision whether an

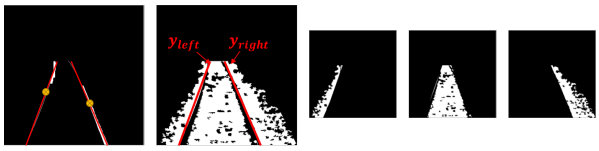


Figure 7: Separation of the surrounding area (second column) into the three regions (third to last column) using the lines y_{left} and y_{right} (red lines).

obstacle is identified is made by computing the euler number of each binary region image (after closing and opening). To illustrate this, an example is shown in Fig. 8.

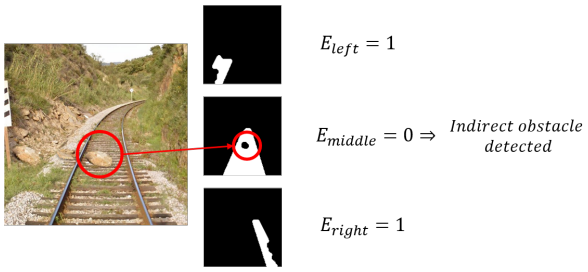


Figure 8: Middle region's euler number different of 0, an obstacle is detected.

Large objects are seen as threats one represented in the binary mask as large holes or gaps, which, and due to their threatening dimensions, are not fully closed during the closing operation and are later expanded after the opening operation. The decision of criteria of the detection is based on the euler number of each region's binary representation. If one region is split into two regions (caused by a large object) the corresponding euler number is equal to 2. If there is a hole in one of the regions (caused by large rock that does not divide one region, however, it is large enough to be considered as a threat), the corresponding euler number is equal to 0. If a region is clear of any indirect obstacle, it is represented by a single foreground with no holes and the corresponding euler number is equal to 1. Therefore, the algorithm detects the presence of an indirect obstacle if there is at least one region with an euler number different from 1.

Finally, the detection algorithm only predicts the complete safety of the railway if, and only if, both detection modules do not detect any obstacle in the railway track.

6. Obstacle Detection - Deep Learning Algorithm (*DetectDL*)

6.1. Data Augmentation and Dataset Spliting

In order to have two classification classes with an equal number of images, the Obstacle class images are subject to data augmentation techniques to generate more cases.

The used data augmentation routine produces 3 new images from an already existing one. The first image (Fig. 9 b) is obtained by flipping the original image (Fig. 9 a) around the vertical bisector axis. The second and the third (Fig. 9 c and d) are obtained by decreasing and increasing the image's contrast up to of 30%. Then one of these two images is randomly selected to be flipped around the vertical bisector axis, similarly to the first augmented image (Fig. 9 b).

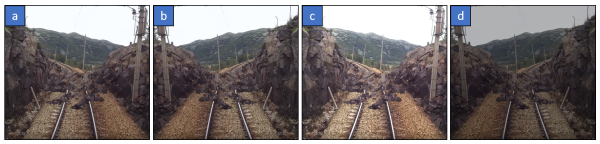


Figure 9: Data augmentation routine. (a) Original Image, (b) flipped image, (c) image with increased contrast and (d) flipped image with decreased contrast.

From the total 1572 No Obstacle images, a proportion of 70%-15%-15% is selected for the training,validation and test sets, corresponding to an amount of 1100, 236 and 236 No Obstacle images, respectively. From the 747 distinct Obstacle images, 236 are selected for the test set. To avoid having images in the validation set generated by data augmentation techniques from training examples, from the remaining 511, 236 are randomly selected as a validation set. The remaining 275 are subject to data augmentation techniques, accounting for a total of 1100 Obstacle images to be used as training set.

6.2. Hyperparameter Selection

In order to estimate the best model configuration, several hyperparameters combinations were tested using a grid search algorithm by (1) varying the number of frozen Inception modules, (2) the initial learning rate (ILR), and respective learning schedule and (3) dropout probability (p). Regarding the number of frozen *Inception* modules, the terminology adopted is the following: baseline GoogLeNet refers to the model with no frozen modules, GoogLeNet *freeze1* has 1 frozen *Inception* module frozen, and so forth.

Each hyperparameter combination is assessed by training and validating using a hold-out set. In each training phase, both training and validation sets are randomly selected from the remaining images apart from the testing images.

The baseline parameter combination tested was: zero *Inception* modules frozen, constant $LR = 0.001$ and dropout value $p = 0.4$. After accessing all the hyperparameter combination, the one that led to the best performance curves was: zero *Inception* modules, an $ILR = 0.0025$ with a decay factor of 0.1 ever 2 training epochs and a dropout value $p = 0.4$. The corresponding performance curves for the accuracy (%) and loss (cross-entropy) for both training and validation are shown in Fig. 10.

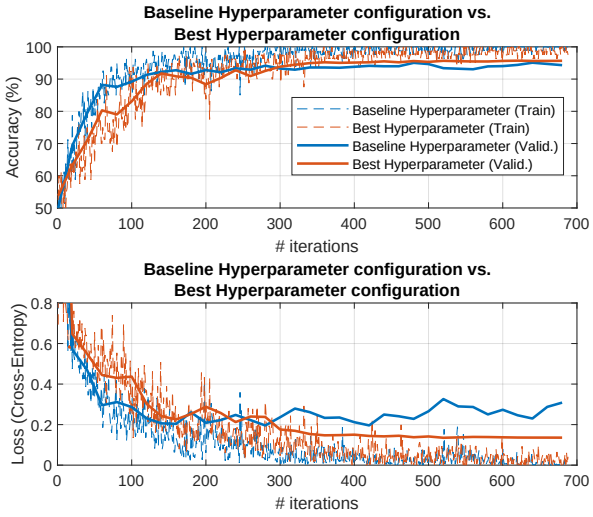


Figure 10: Performance curves obtained from the baseline hyperparameter combination (blue) and best hyperparameter combination estimation (orange).

7. Results

7.1. DetectCV

The confusion matrix for the DetectCV algorithm in the test set is represented by table 2. *DetectCV* has a reasonable capacity in correctly detecting obstacles(Sensitivity= 95.3%) and about 68.0% of the images of railway tracks classified as having an obstacle are correctly classified. *DetectCV* correctly identifies 130 of the total of 236 images as free of any obstacle (Specificity= 55.1%). Overall, the algorithm assigns the correct label (*Obstacle* or *No Obstacle*) in 75.2% of the cases.

Table 2: Confusion Matrix for the *DetectCV* algorithm applied to a 472 images test set.

		Ground truth	
		Positive	Negative
Pred.	Positive	225	106
	Negative	11	130

The respective classification metrics are $Sensitivity = 95.3\%$, $Specificity = 55.1\%$, $Precision = 68.0\%$ $NPV = 92.2\%$and $Accuracy = 75.2\%$.

7.1.1 Railway Tracks with Obstacles

DetectCV has a reasonable capacity in correctly detecting obstacles (Sensitivity = 95.3%), with only 11 false negative incidences. Fig. 11 shows some cases of the prediction of *DetectCV*.

Figure 11: True positive examples (left) and (middle) and a false negative example (right).

While Fig. 11 (left) and (middle) are correctly classified (true positive), Fig. 11 (right) represents one the false negative incidences. The cause of false positives is related to the threshold defined for the vertical distance ($VertDist$). For a higher threshold, Fig. 11 (right) would require a higher $VertDist$ so that the rails would be classified as obstructed. Therefore, with a larger threshold, the rails would be correctly classified as being obstructed.

7.1.2 Railway Tracks without Obstacles

Fig. 12 shows two examples where *DetectCV* correctly classifies the railway track image with no obstacle: railway rails have the necessary $VertDist$ to be classified as unobstructed and the euler number of each region is 1. The algorithm is also able to ignore the square electrical boxes found in between the rails (Fig. 12 bottom row) after the closing and opening operations perform upon the second image in the bottom row.

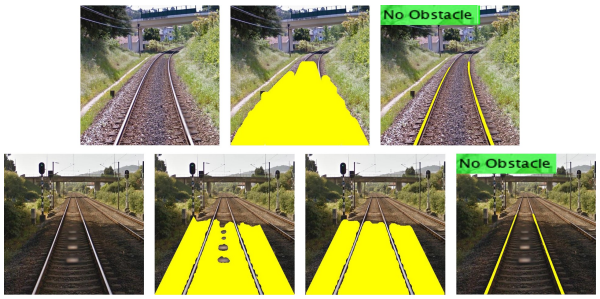


Figure 12: Ground truth: No Obstacle. Output: No Obstacle.

Situations where no obstructions (detected by the first module) can be found near the rails, the clustering algorithm is forced to segment the image into 3 different colored groups, even when the difference in the surrounding's color are perceptually irrelevant. Consequently, in some cases, the surrounding regions are represented by a non uniform binary mask, making the respective Euler number different from 1 (indicating the presence of an obstacle) (Fig. 13 (a)). Another source of error can be traced to the slightly worse quality (high blurring) of some images tested, which makes the edge detection unable to detect the rail edges accurately (Fig. 13 (b)).

(a) Output: Obstacle. $Euler_{middle} \neq 1$, leading to the miss detection of an obstacle in between the rails.

(b) Output: Obstacle. Miss detection of edges due to poor image quality.

Figure 13: Ground truth: No Obstacle. Predicted: Obstacle.

7.2. DetectDL

The confusion matrix for the DetectDL algorithm in the test set is represented by table 3. *DetectDL* has a reasonable capacity in correctly detecting obstacles (Sensitivity= 97.0%) and about 94.6% of the images of railway tracks classified as having an obstacle are correctly classified. *DetectDL* correctly identifies 223 of the total of 236 images as free of any obstacle (Specificity= 94.5%). Overall, the detection algorithm assigns the correct label (*Obstacle* or *No Obstacle*) in 95.8% of the cases. Futhermore, the *DetectDL* algorithm achieved an AUROC of 98.3% over the same test set.

Table 3: Confusion Matrix for the *DetectDL* algorithm.

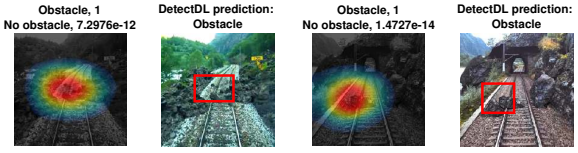
		Ground truth	
		Positive	Negative
Pred.	Positive	229	13
	Negative	7	223

The respective classification metrics are $Sensitivity = 97.0\%$, $Specificity = 94.5,\%$ $Precision = 94.6\%, NPV = 97.0\%$ and $Accuracy = 95.8\%$

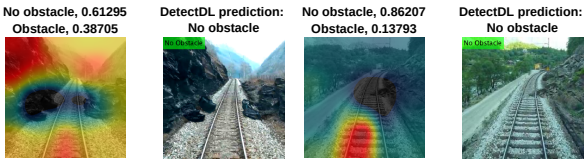
7.2.1 Railway Tracks with Obstacles

Fig. 14 shows some images with ground truth *Obstacle*. The class activation map (CAM), represented by the heat map, for the predicted class and the classification scores are represented on the left, while the model prediction is presented on the top of the right image.

Inspecting the true positive cases represented in the top row of Fig. 14, it is possible to conclude that



(a) Predicted: Obstacle. (b) Predicted: Obstacle.



(c) Predicted: No Obstacle. (d) Predicted: No Obstacle.

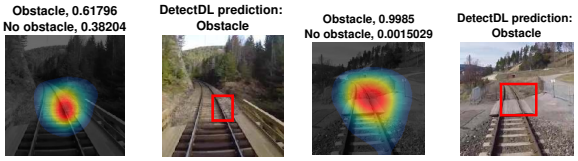
Figure 14: Ground truth: Obstacle. Top row - true positive. Bottom row - false negative.

the discriminative regions, given by the CAM, with the most relevancy to the model's prediction are the portions of the image containing rail discontinuities. In the bottom row examples, representing the false negative examples, the CAM for the misclassified *No Obstacle* class has a tendency of avoiding the image portion region containing the true obstacle.

7.2.2 Railway Tracks without Obstacles

Fig. 15 shows some images with no obstacles.

(a) Predicted: No Obstacle. (b) Predicted: No Obstacle.



(c) Predicted: Obstacle. (d) Predicted: Obstacle.

Figure 15: Ground truth: Obstacle. Top row - true positive. Bottom row - false negative.

The top row images represented in Fig. 15 depict some cases where the detection algorithm correctly classifies the image as a No Obstacle situation (true positives). The CAM for the classification label *No Obstacle* encompasses a wider discriminative region of theimage, referring to the railway rails and the sleepers between them.

The bottom row images represent situations where the detection algorithm wrongly detects the presence of an obstacle (false negatives). While the reason behind the misclassification of the image in Fig. 15 (c), might not be clear, a more coherent justification can be given for the case depicted in Fig. 15 (d). The level crossing platform creates an illusion of a rail discontinuity, misleading the model's prediction. From the total of 13 false positives incidences, 5 are related to these infrastructures.

7.3. *DetectCV vs DetectDL* - Specific cases

Comparing both detection algorithms on low quality images (high blurring effect), *DetectDL* has a better performance than *DetectCV*, whose performance depends on how well the rail edges are detected, which can be hard when subject to low quality images.

Fig. 16 shows two examples of low quality railway track images and the respective output of the computer vision and the deep learning algorithms.

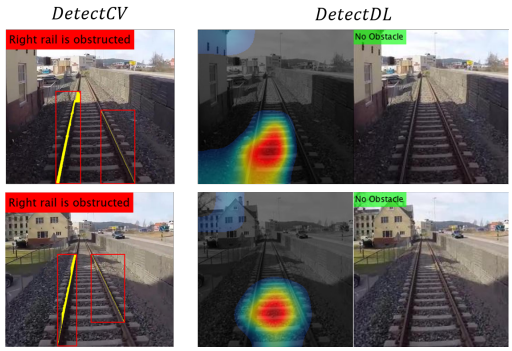


Figure 16: Comparison of both *DetectCV* (left) and *DetectDL* (right) when subject to low quality images. *DetectCV* failed in the classification, while *DetectDL* succeeded.

Both algorithms' performance were also tested using images with shadows projections onto the railway tracks. Fig. 17 shows two examples of railway track images with a shadow projected into the rails.

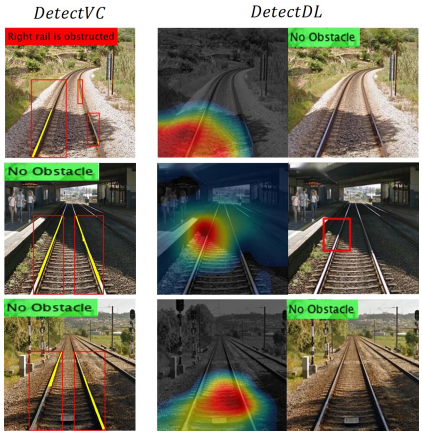


Figure 17: Images with shadows where all the cases are ground truth *No Obstacle*. First column refers to the *DetectCV* algorithm and the second to the DetectDL algorithm.

DetectCV fails when the shadow causes a discontinuity in the rail edges (Fig. 17 top row). However, the algorithm performs well when facing cases where the shadow only changes the rail's surface color, keeping the rail edges unbroken (Fig. 17 middle and bottom row).

Regarding *DetectDL*, it wrongly detects an obstacle in Fig. 17 (middle) where the shadow projection leads to a drastic change in the rail's surface color (Fig. 17 middle). In the other images (Fig.17 top and bottom), the rails retain an uniform surface color even after being covered by the shadow, therefore, not misleading the detection algorithm.

7.4. Scale Model

Both algorithms were tested using a scale model, comprised of 1/87 scale track model and a comercial *webcam* HP 2300HD (Fig. 18).

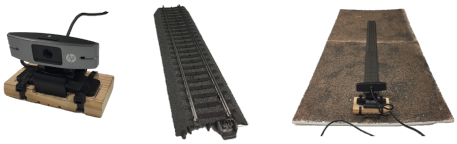
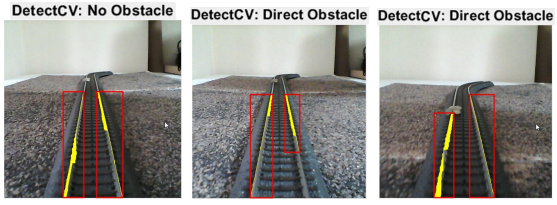


Figure 18: Scale model.

The hardware used was a laptop with built-in 2.50 GHz Intel Core i7-4710HQ, 64 bits Windows 10 operating system and an AMD Radeon R9 M265X. The Software used was MATLAB 2018b.

Fig. 19 shows some processed frames by the computer vision algorithm (*DetectCV*) using the scale model.

In the first and second frame (Fig. 19 (a) and (b)) the *webcam* was situated in the same position. In these situations, the obstacle (white rocks) are not encompassed inside the ROI. The detection algorithm correctly classifies the first frame without obstacles (true negative) (Fig. 19 (a)). However, frame 2 represents a false positive example, where the right rail is not fully detected by the algorithm, therefore being classified as obstructed (Fig. 19 (b)). As the model approaches the obstacle making it enclosed by the ROI, the algorithm correctly detects it (Fig. 19 (c)).



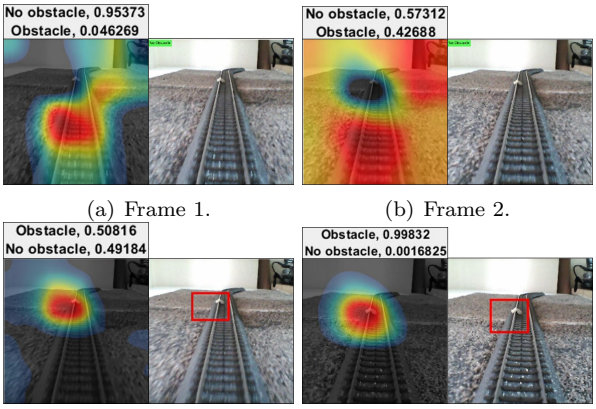
(a) Frame 1. (b) Frame 2. (c) Frame 3.

Figure 19: Frames processed by *DetectionCV* using the scale model.

In 600 captured frames, the algorithm took on average 0.35 seconds to process a single frame. The detection range achieved was $23,0$ cm, measured in the model.

Fig. 20 shows some processed frames by the computer vision algorithm (*DetectDL*) using the scale model. In the first frame (Fig. 20 (a)), the model does not yet detect the obstacle. In frames 2 and 3 (Fig. 20(b) and (c)), the model was located in the same position. At this location, the model's prediction oscillated from *No Obstacle* and *Obstacle*. For the *No Obstacle* class, the respective CAM completely ignores (by representing with a darker shade in the heat map) the discriminative location within the image where the obstacle stands. Contrary, the CAM for the *Obstacle* class, fully encloses the obstacle once ignored by the CAM represented in Fig. 20 (c).

Finally, in the fourth frame, the model correctly detects the obstacle (white rock) (Fig. 20 (d)).



(a) Frame 1. (b) Frame 2.

(c) Frame 3. (d) Frame 4.

Figure 20: Frames processed by *DetectionDL* using the scale model.

In 600 captured frames, it took the algorithm an average of 0.20 seconds to analyze a single frame. The detection range achieved was $32,0$ cm, measured in the model.

Assuming that the vehicle is traveling at a constant speed, its acceleration is constant during the breaking and it stops from a safety distance of 2 meters from the obstacle, in a situation where the obstacle is detected at its maximum distance from the vehicle, an estimation of the vehicle's speed can be obtained. Having obtained the processing rate for each frame and the respective ranges in real world dimensions (using the 1/87 scale) the estimated traveling speed is 24.6 and 30.1km/h for *DetectCV* and *DetectDL*, respectively.

8. Conclusions and Future Work

The main objectives of this work were the development, evaluation and comparison of computer vision and deep learning approaches for the automatic detection of obstacles in railway tracks, while limited to the analysis of RGB railway track images

captured by a single monocular camera.

To address this topic, for which no available dataset of images of railway tracks containing obstacles is known to exist, the first effort was focused on the creation of a novel dataset from existing images of railway tracks without obstacles, retrieved from multiple sources.

Regarding the computer vision algorithm *DetectVC*, some remarks can be drawn, which, fundamentally represent the paradigm of any computer vision system: the importance of establishing controlled working conditions for the algorithms to operate. In this work, the working conditions were defined from the beginning, by considering only images with similar orientation and by defining a ROI. However, it was later verified that some blurred images were presented where the algorithm had difficulties to achieve its task. Moreover, the selection of parameters in many computer vision systems are often based on heuristics, with no definitive guarantee that one solution fits every case. Nevertheless, the algorithm presented a good detection ability.

A deep learning approach was also explored. Deep learning models are data driven models that require minimal human supervision, with an associated cost of requiring a significantly amount of dataset to extract features from raw data. As such, their performance is highly dependent of the available data used for training and less on human expertise, contrary to computer vision algorithms. Despite the immense efforts dedicated to assemble the dataset necessary for this task, the *DetectDL* algorithm would benefit from more images. Nevertheless, with the use of transfer learning, the algorithm was able to achieved even better results than the computer vision-based detection algorithm.

On a finishing note, all the objectives defined in this work were fulfilled. Nevertheless, some recommendations are provided.

The first suggestion is the use of supervised object detection algorithms, such as the Faster R-CNN. Secondly, the deep learning network could be trained with images with the ROI, instead of using the whole image (railway tracks + background). If saved in memory, the images must be saved in a lossless format such as Portable Network Graphics (.png) to avoid the presence of compression noise. Finally, and since there is lack of sufficient data dedicated to the railway track inspection research, the expansion of the dataset is a work worth exploring.

References

[1] D. Soukup and R. Huber-Mörk, "Convolutional neural networks for steel surface defect detection from photometric stereo images," in *International Symposium on Visual Computing*, pp. 668–677, Springer, 2014.

[2] Y. Santur, M. Karaköse, and E. Akin, "An adaptive fault diagnosis approach using pipeline implementation for railway inspection," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 26, no. 2, pp. 987–998, 2018.

[3] F. Maire and A. Bigdeli, "Obstacle-free range determination for rail track maintenance vehicles," in *2010 11th International Conference on Control Automation Robotics & Vision*, pp. 2172–2178, IEEE, 2010.

[4] L. F. Rodriguez, J. A. Uribe, and J. V. Bonilla, "Obstacle detection over rails using hough transform," in *2012 XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA)*, pp. 317–322, IEEE, 2012.

[5] H. Wang, X. Zhang, L. Damiani, P. Giribone, R. Revetria, and G. Ronchetti, "Video analysis for improving transportation safety: obstacles and collision detection applied to railways and roads," in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2017*, pp. 909–915, 2017.

[6] M. Ukai, B. Nassu, N. Nagamine, M. Watanabe, and T. Inaba, "Obstacle detection on railway track by fusing radar and image sensor," in *9th World Congress on Railway Research (WCRR-2011), Lille, France*, 2011.

[7] M. C. Nakhaee, D. Hiemstra, M. Stoelinga, and M. van Noort, "The recent applications of machine learning in rail track maintenance: A survey," in *International Conference on Reliability, Safety, and Security of Railway Systems*, pp. 91–105, Springer, 2019.

[8] G. Kano, T. Andrade, and A. Moutinho, "Automatic detection of obstacles in railway tracks using monocular camera.," in *Computer Vision Systems* (D. Tzovaras, D. Giakoumis, M. Vincze, and A. Argyros, eds.), 11754, Springer International Publishing, 2019.

[9] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, "Brain tumor segmentation using convolutional neural networks in mri images," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1240–1251, 2016.

[10] J. Pinho and J. Santos, "Technical report. automatic detection of anomalies in railway tracks. instituto superior técnico. universidade de lisboa." Project developed in Computer Vision lecture.