Development of a software for automatic identification of the rolling stock at marshaling yards using a neural network

Nataliya Moskat[1*], *Olesya* Ignatieva[1], and *Andrey* Shulzhenko[2]

[1] Rostov State Transport University, 344000 Rostov-on-Don, Russia
[2] Rostov Branch of JSC NIIAS, 344000 Rostov-on-Don, Russia

Abstract. The research paper presents the author's software for automatic identification the rolling stock at marshaling yards using a neural network. Methods for processing images from several cameras, recognizing cuts, creating a fleet model and displaying this model for further display in the Russian Railway information systems have been implemented. The software was developed in the Python 3.6 programming language, using the OpenCV and Torch libraries. Based on the obtained results, a sys-tem for monitoring the track availability of the fleet formation was implemented. A method for neural network training is demonstrated using the example of recognizing cuts of a marshaling yard.

1 Introduction

The issues for programming automatic identification in railway transport have been developed for several decades. One of the significant achievements in the researched field was the Automatic Rolling Stock Identification System (ARSIS "Palma"), which operates on the principle of reading an electronic number recorded in an on-board code sensor on a locomotive or wagon. This system made it possible to re-place the manual writing off of numbers and ensure the efficiency and reliability of the information about moving objects. However, this system involves both the installation of the specialized sensors for each unit of the rolling stock, and the sets placement of receiving and recording devices. Recently, the information technology has been developed rapidly, and there are new opportunities for modern system programming of the rolling stock identification [1,2,3]. The neural network use supports to identify an object under various conditions. Neural networks are resistant to input data noise and they are adaptable to changes (adding new functionality, changing the interface, etc.), fault-tolerant, and have ultra-high speed. It should also be noted that the methods of synthesizing neural networks are invariant on the dimension of the space condition, the possibility of choosing the structure of the neural networks in a significant range of parameters depending on the complexity and objective specifics to be solved in order to achieve high quality solution [4,5].

* Corresponding author: nancy-rostov@yandex.ru

The aim of the paper is: the development of a rolling stock recognition technique using a neural network based on the affine transformation, convolutional neural networks; implementation of the software identification; model training neural net-work.

2 Setting up target in the module system development for monitoring the track availability of the marshaling yard

As shown by the analysis of ready-made solutions, marshaling yards and railway networks are well automated and have been controlled over passing vehicles. But the fleet formation remains poorly automated, and it is still necessary to conduct the track and gaps occupancy monitoring between the moving units with human activity. In the systems described above, the track occupancy is monitored only observing last car, while other parameters such as speed, acceleration, and thrust speed are calculated in a virtual model based on statistical data obtained during the cut moving to the 3rd brake position. The application of the approach presented by ARSIS "Palma" will be very costly and difficult to implement as, it will be necessary to place the RNG at a short distance for a complete fleet analysis, and the total length of the fleet tracks on average exceeds 50 km.

In this case, there is a need to develop a solution capable of providing reliable information about the cuts condition on the track of the fleet formation without resorting to the use of standard sensors. And such a solution can be a computer vision system that will process data from a video surveillance camera installed on lighting towers in Figure 1. Thereby it provides the most complete and accurate information throughout the fleet.

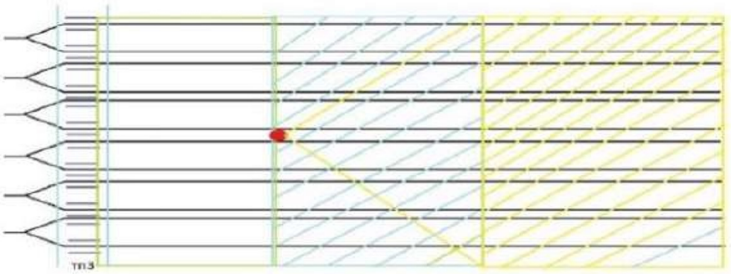

Fig. 1. Scheme of the camera visibility zone

Figure 2 shows the developed algorithm of the program.

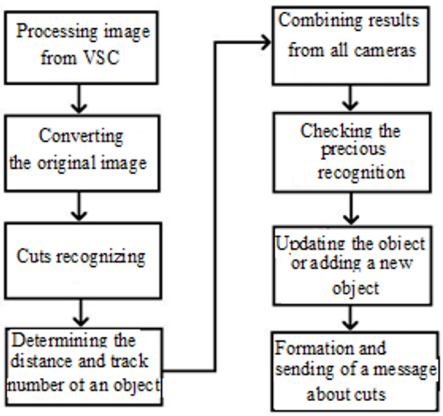

Fig. 2. The program algorithm

3 The rolling stock recognition method based on the neural network

The rolling stock recognition method is based on a number of actions [6]. It is including:

- Pre-processing an image to improve the precision of determining the distance and track number of an object in the image by transforming the original image using affine transformation methods.
- Application of convolutional neural networks in the process of recognizing moving units in a digital image.
- Application of the camera calibration formula to calculate the distance to the detected rolling stock.

To simplify the objective of determining the distance to the recognized object and determining the number of the track where it is located, it is also necessary to make some changes to the images received from the camera. Moreover, the railway tracks were located vertically. It is also necessary to correct the distortion as it is presented due to the fisheye cameras application.

The first is corrected with the help of an affine transformation; it is a mapping of a plane or space into itself, when the parallel lines turn into parallel lines, intersecting lines turn into intersecting ones, crossing lines turn into crossing ones. It looks as formula:

$$f(x) = M * x + v,$$

where M is a nonsingular matrix and $v \epsilon R^n$.

At the same time, for each camera it is necessary to find its own transformation matrix like a rotation, stretching, compression, reflection, translation, which will look like a combination of the above matrix.

To identify objects in images, we use a convolutional neural network (CNN). The specific detector is a two-dimensional (2-D) file of significances that represents a portion of the image. Although they may vary in size, the filter size is typically 3x3 matrix; it also determines the receptive space size. The filter is then applied to a region of the image, and the dot product is calculated between the input pixels and the filter, which is then passed to the output significance. After that, the filter is gradually moved, repeating the process until the hardcore covers the entire image. The final result of a series of dot products on the input and filter is known as a feature map, activation map, or folded object. After each convolution operation, the CNN applies a Rectified Linear Unit (ReLU) [7] transformation to the feature map, introducing non-linearity into the model. The structure of a CNN is hierarchical because later layers can see pixels within the input spaces of the previous layers.

To determine the distance to the analyzed object in meters through the distance on the image in pixels, the camera calibration formula from Shubnikov I.S. and Palagut K.A., "Analysis of methods and algorithms for determining the parameters of an object and the distance to it in the image" can be used:

$$D = \frac{L*K}{\frac{W}{x}-1+K},$$

where D is the searching distance to the object, m;

L is the path length, m;

W is the path length in pixels;

x is the distance from start of the path to the analyzed point on the image in pixels;

K is the camera obliquing factor, calculated by the formula:

$$K = \frac{W-M}{M},$$

where M is the distance from the start to the middle of the path in pixels.

4 Implementation of a for automatic identification of the rolling stock at marshalling yards using a neural network

The preliminary pre-processing includes the creation of working folders for easy use, which store all the necessary modules, scripts, data for training the neural network and information for calibration.

4.1 Implementation of the cameras` connection

When the program is run, the parameters (address, number of the observed sector) of the video surveillance cameras are read from the configuration file. The connection to the cameras is implemented using the RTSP protocol (real time streaming protocol). In the main process for each camera, its own executable process is created, its objective is to process the image, recognize cuts and determine the distance to each found object in Figure 3.

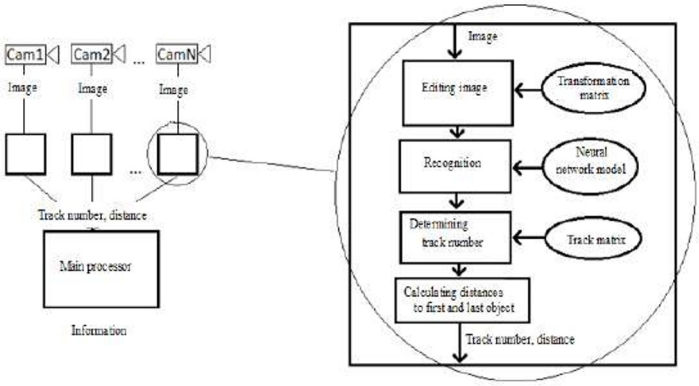

Fig. 3. Image processing algorithm

4.2 Converting the original image from CCTV cameras

Before the image from the surveillance cameras enters the recognition module of moving units, it goes through several editing steps:

- demapping of the segment of interest and rotation (if necessary);
- distortion correction;
- adding empty fields (if necessary);
- warping.

The image segment of interest is cropped and rotated so that the rails are vertical. Next, the optical distortion of space is corrected. For some images, empty fields are also added so that during subsequent deformations, important information is not lost and / or distorted. The last step is to perform an affine and perspective conversion using the transformation matrix shown in Figure 4.

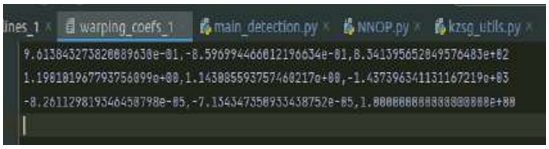

Fig. 4. Matrix of transformation

4.3 Cuts recognition

The prepared image enters the recognition module, where it goes through resolution standardization, color space change from BGR to RGB, and conversion from the standard image representation to a tensor.

After that, cuts are detected on the image. If the objects were found, then a list of parameters is returned in Figure 5, represented as the coordinates of the upper left and lower right corners of the rectangle the supposed object location.

Fig. 5. List of the recognized objects

During the visualization process, neural network predictions are displayed in a separate window in Figure 6.

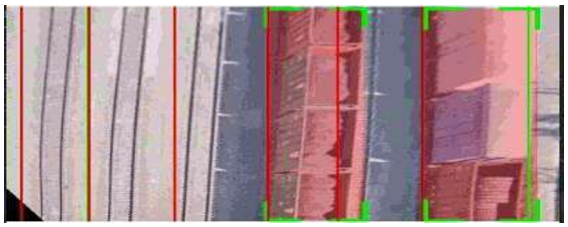

Fig. 6. Visualization of predictions

4.4 Determining the path number of the predicted object

Subsequently, the predicted object is checked for belonging to one of the available paths. The check is carried out according to the degree of occupancy of the space, limited by vertical lines to the left and right of the railway tracks. If the area on the path is occupied by more than 60% percent, then the intended object is assigned the number of this track in Figure 7.

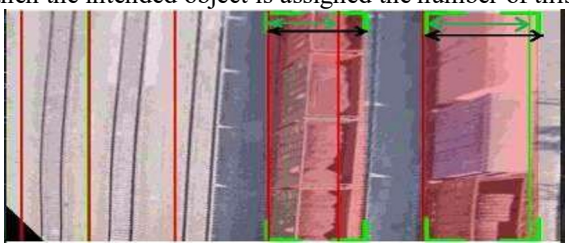

Fig. 7. Assigning a track number to a recognized cut

4.5 Determining the distance to the start and end of an object

After determining the track number, the distances are calculated through the distance between pixels. So in Figure 8, a recognized cut is shown, which falls into the area 23 of the track.

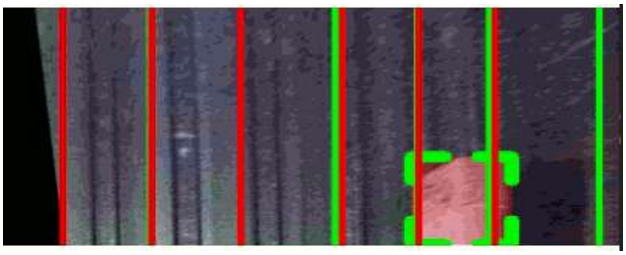

Fig. 8. Recognized cutter on 23 track

In Figure 9, the first line contains a list of bounding box coordinates, and the second line contains a list of objects with their parameters (track number and calculated distances to their start and end).

```
[[296 148 367 235]]
[{'way': 23, 'start': 92, 'finish': 127}]
```

Fig. 9. The result of calculating the distance to the beginning and end of the object

4.6 Combining predictions from all cameras

Further, the recognized objects from all cameras are combined.

In Figure 10, you can see that the cutter falls into the visibility zone of cameras 2 and 3, respectively. Its Information is divided into two sectors and it must be combined.

Fig. 10. The process of combining images

In Figure 11, the top line of the console shows the recognition results in the form of a list of tuples. The first element of the tuple is the camera number; the second is an array with a list of dictionaries containing the parameters of recognized cuts (track number, start and end of the cut). The second line shows that the object from the 2nd sector on the 20th path with the parameters: 80-96 merged with the object from the 3rd sector with the parameters 96-200, forming a cut on the 20th path, the end car is located at a distance of 80 meters, and the first one is at a distance of 200 meters.

```
[(1, None), (2, [{'way': 20, 'start': 88, 'finish': 96}]), (3, [{'way': 20, 'start': 94, 'finish': 200}])]
1 str(1): (w-20 (80:200) v(None))
```

Fig. 11. Consolidation of the recognition results

4.7 Prediction Filtering

In Figure 12, the first line of the console shows the infiltrated result of the merge that was performed. The second line shows the result after several filtrations, which allow us to take away false predictions and correct erroneous ones.

Fig. 12. Filtering Predictions

4.8 Model Interaction

The following figures show possible cases in processing new predictions. Figure 13 is an update with new data.

Fig. 13. Updating object parameters based on new data

Here, for each cut in the model, an analogue was found in the list of predictions. Based on these predictions, the objects have been updated. Figure 14 shows a new object in the model.

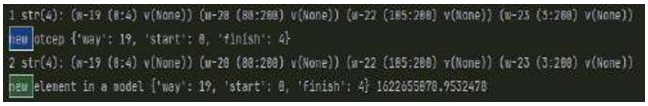

Fig. 14. Adding a new object to the model

Here, a cut was found on track 19, which is not yet in the model. Therefore, a new object was added to the model, passing the necessary checks. Figure 15 has been updated based on the old settings.

Fig. 15. Updating an object in the model which doesn`t get new data

Here is a situation where there is no recognition for a cut on track 19 and it is updated based on the previous state of the object.

4.9 Formation of a message about the state of the cuts in the model

After the model update is completed, sending message is created that includes all the cuts in the model and their location: distance to the first and last car as well as the moving direction, if it has any in Figure 16.

Fig. 16. Model Cut Status Messages

4.10 Visualization of information about cuts in the fleet

Figure 17 shows the window of the AWS KZSP application, which displays cuts and their states. So on the 21st path you can see two moving cutters, in the direction from the hill, the distance to each can be seen on the ruler and calculate the distance between them

Fig. 17. Data display in AWP KZSP

5 Neural network training

There are a number of classical methods for training a neural network [8,9]. The methodology presented in [10] is interesting.

To assess the quality of the algorithm, the metrics precision and recall are introduced, the formulas:

$$precision = \frac{TP}{TP+FP};$$
$$recall = \frac{TP}{TP+FN},$$

where

TP is the true positive objects;

FP is the false positive objects, that the classifier called true, but they were false;

FN is the false negative objects that the classifier could not recognize.

Precision can be interpreted as the proportion of objects that are called positive by the classifier and are actually positive, and recall shows the proportion of all positive objects of all positive classes, the algorithm has found.

A good result can be considered models trained with precision and recall above 90%.

5.1 Data pre-processing

Data pre-processing is a markup of images that contain objects of interest. The markup process is a demapping of an object in the image with a rectangular area. A separate application has been developed for this purpose. Bounding rectangles are drawn using two coordinates of opposite corners, which are set by pressing the left mouse button on the image in Figure 18.

Fig. 18. Markup process

In the process of saving data, the coordinates are converted into a standardized format: the coordinates of the center of the quadrilateral x and y, the width and height w and h in Figure 19.

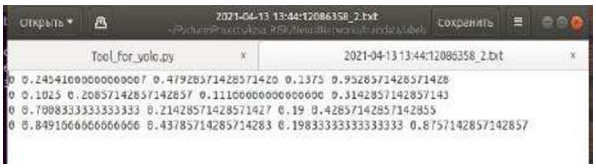

Figure 19. Bounding box options

5.2 Learning process

After pre-processing the data and setting up the architecture, you need to change the hyper parameters and start learning through the script in Figure 20, where it is necessary to specify:

- img 416 image size;
- batch 32 batch size;
- epochs 700 number of epochs;
- data is the name of the file that stores the name of folders with training data and the number of classes;
- cfg is the name of the file containing the structural model of the network;
- weigths file name of the weights that need to be retrained;
- name is the name for the trained scales;
- nosave flag to cancel the saving of weights at the last training iteration;
- cache flag for unloading the entire data set into the video memory, not by batches.

Fig. 20. Script to start learning

Figure 21 shows graphs showing recognition precision at the end of training.

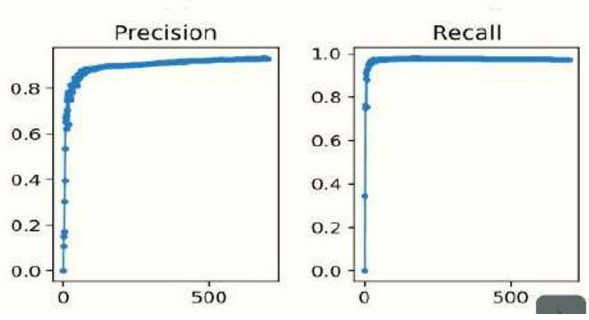

Fig. 21. Recognition precision

Checking the model on a test set in Figure 22 shows excellent results, which indicates successful network training.

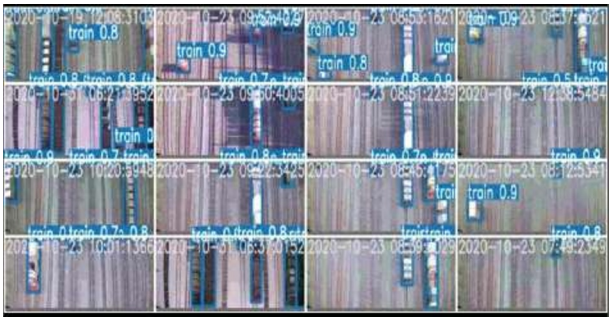

Fig. 22. Model testing

6 Conclusion

This development can either serve as a replacement for obsolete morally and technically track occupancy control systems, or be used as an independent system. Nowadays, the advantages in comparison with current systems, are: obtaining more complete information about cuts in the fleet in real time; more precious determination of the distance to moving units; definition of windows between cuts.

The specified system was put into operation at the railway station “Inskoy”. Negotiations are proceeding on implementation this system to other marshaling yards.

References

1. A. Chernov, M. Butakova, A. Guda, P. Shevchuk, European Dependable Computing Conference **1279,** 33-43 (2020) https://doi.10.1007/978-3-030-58462-7_3
2. A.N. Shabelnikov, I.A. Olgeizer, A.V. Sukhanov, IOP Conference Series: Materials Science and Engineering **918(1),** 012067 (2020)
3. A.E. Khatlamadzhiyan, S.M. Kovalev, V.B. Tarassov, Lecture Notes in Networks and Systems **330,** 513–526 (2022)
4. M.A. Butakova, A.V. Chernov, S.M. Kovalev, A.V. Sukhanov, S. Zajaczek, Lecture Notes in Electrical Engineering **554** (2020)
5. G. Peyré, M. Cuturi, Foundations and Trends in Machine Learning **11**(5-6), 355-607 (2019)
6. Implementation of intelegent monitoring for the marshaling yard. https://elibrary.ru/item.asp?id=41651687&, last accessed 2022/05/15.
7. Rectified Linear Units, the linear correction unit activation function. https://programmersought.com/article/97782232355/, last accessed 2022/05/15.
8. Ch. A. Naesseth, F. Lindsten, Th.B. Schön, Foundations and Trends® in Machine Learning **12(3),** 307-392 (2019)
9. L. Stanković, D. Mandic, M. Daković, M. Brajović, B. Scalzo, Sh. Li, A. G. Constantinides, Foundations and Trends® in Machine Learning **13(1),** 1-157 (2020)
10. J. Zhang, J.Yang, Jun Yu, J. Fan, International journal of intelligent systems **37(5),** 3117-3141 (2022)