ACTA UNIVERSITATIS AGRICULTURAE ET SILVICULTURAE MENDELIANAE BRUNENSIS

Volume 63 240 Number 6, 2015
http://dx.doi.org/10.11118/actaun201563062197

ASSISTANCE SYSTEM FOR TRAFFIC SIGNS INVENTORY

Karel Zídek[1], Tomáš Koubek[1], David Procházka[1], Marcel Vytečka[1]

[1] Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic

Abstract

ZÍDEK KAREL, KOUBEK TOMÁŠ, PROCHÁZKA DAVID, VYTEČKA MARCEL. 2015. Assistance System for Traffic Signs Inventory. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, 63(6): 2197–2204.

We can see arising trend in the automotive industry in last years – autonomous cars that are driven just by on-board computers. The traffic signs tracking system must deal with real conditions with data that are frequently obtained in poor light conditions, fog, heavy rain or are otherwise disturbed. Completely same problem is solved by mapping companies that are producing geospatial data for different information systems, navigations, etc. They are frequently using cars equipped with a wide range of measuring instruments including panoramic cameras. These measurements are frequently done during early morning hours when the traffic conditions are acceptable. However, in this time, the sun position is usually not optimal for the photography. Most of the traffic signs and other street objects are heavily underexposed. Hence, it is difficult to find an automatic approach that can identify them reliably. In this article, we focus on methods designed to deal with the described conditions. An overview of the state-of-the-art methods is outlined. Further, where it is possible, we outline an implementation of the described methods using well-known Open Computer Vision library. Finally, emphasis is placed on the methods that can deal with low light conditions, fog or other situations that complicate the detection process.

Keywords: OpenCV, traffic signs, image processing, object recognition, road inventory, machine learning, Viola-Jones detector, support vector machines

INTRODUCTION

It is possible to find a substantial amount of articles that deals with a road sign detection (see Loraskul *et al.* (2007) or Salhi *et al.* (2014)). The described methods vary from pixel correlation calculations to neural networks. In addition, most of the articles do not discuss the ability to work in adverse weather conditions or other disturbing factors (heterogeneous backgrounds, fog, low light conditions etc.). However, there are many companies working in the mapping industry that must deal with such conditions. Used camera equipment has naturally also its limitations; hence, the detection applications must deal with chroma and luminance noise (see Fig. 1, left image), optical distortions and other hardware-based shortcomings. On a suitable sunny day, there is not a significant difference between an image obtained by a state-of-the-art camera and an older one. However, when the mentioned problems (hardware limitations and environmental conditions) are multiplied, the result images can be below the algorithm recognition abilities. Fig. 1 (middle image) represents a situation with low light conditions, additional image distortion caused by raindrops (see Fig. 1 right image) and a shape distortion caused by lens.

This article is focused on researchers or developers that consider development of a traffic sign recognition system. To simplify the development, we refer to an implementation of the described methods that is provided by Open Computer Vision Library (OpenCV, 2014). In spite of that, the overall recognition process is very complex. For this reason, the article deals just with the first part of this process – detection of traffic sign position within the RGB picture. The other part of the process, identification of a precise kind of the sign (e.g. Stop sign), is described just briefly. Nevertheless, even this step is supplemented by a possible implementation.

1: *Examples of images distorted by hardware limitations of the camera (left image), light and weather conditions (middle image) or raindrops (right image)*

Image Preprocessing Methods

The pre-processing phase can substantially simplify the sign detection process. Firstly, it is frequently recommended to reduce the resolution of input data. Further, it is possible to crop the input image in the way that areas with possible sign appearance are left only. This step has no significant influence on the results; it can however substantially speed-up the detection process.

Now, basic filtering can be applied. The frequently used are, e.g., blur filter that can reduce the image noise or colour equalisation. In a case of underexposure of the image, histogram equalisation can be done. Its influence is described, e.g., in Bahlmann *et al.* (2005). Histogram correction must be done accordingly to the used detection method and colour model. For instance, in case of HSL model, hue and saturation values should not be corrected using this method. Representation value of the image can be harmed.

The output of this phase is an image with reduced size and reduced distortions. Nevertheless, all these filters and adjustments, in general, must be applied cautiously. Inappropriate application can have a negative influence on the detection process. Additionally, particular pre-processing filter must be chosen accordingly to the used detection method. Therefore, the selected pre-processing methods are in detail described in the following section that outlines the recognition part of the process.

Road Sign Recognition Methods

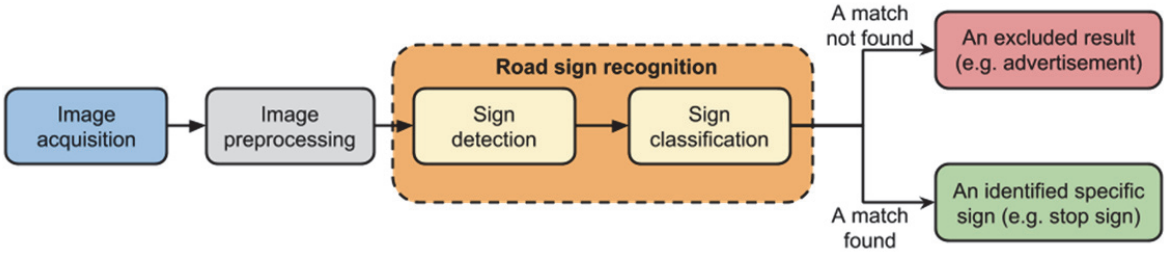
All traffic signs must fulfil conventions of Inland Transport Committee (1968). This agreement substantially simplifies their recognition. Naturally, each country issues a decree that describes an application of these rules to the national legislation. For instance, there are five key categories of traffic signs in the Czech Republic (more details in the Regulation of Ministry of Transport (2001)). It is similar, with some exceptions, to road signs systems in other parts of the world. For instance, the first group consists of warning signs. A particular pattern is a triangular shape with a bold red edge.

A general sign recognition process is outlined in Fig. 2. It naturally starts with an image acquisition. Once the image is obtained, it must be cleared of distortions and unnecessary information. This part of the process is known as the image pre-processing. Further, a location where potential sign is placed is usually calculated. This step can be described as a sign detection. The potential sign is then classified. Hence, the specific kind of sign is identified, or a potential sign is excluded from the results. The outlined steps can be implemented using many different methods.

The purpose of the detection and image quality are decisive factors for selection of an appropriate method. Real-time processing methods (e.g. methods for car systems) must be adjusted for high processing speed. On the other hand, this is obviously not a case of an offline system (for instance a road sign inventory). In this article, we aim at offline systems. In addition, different methods have to be used for recognition of data in orthogonal projection, different for data with deformed perspective, etc.

Road Sign Detection Process

As been already mentioned, the purpose of this step is to identify a possible road sign position



2: *Overview of road sign recognition process*

in the image. The following section presents selected frequently used methods. The key issue is usually to choose an appropriate combination of these methods. Success rate of the described methods can be in some cases improved by their proper combination, e.g. the colour segmentation and morphological operations can be used before a descriptor detection. That depends on the particular application.

Colour Segmentation

The goal of colour segmentation is to split the image into so-called regions of interest (ROI) that possibly contain a road sign. This approach is based on the assumption that the sign colours are strictly defined by an appropriate national decree. Hence, image areas that contain these colours are possible sign positions. For colour segmentation, it is possible to use the OpenCV function *inRange(inputImage, lowerBoundary, upperBoundary, segmentedImage)*. The method *inRange()* filters *inputImage* and stores the result into *segmentedImage*. The parameters *lowerBoundary* and *upperBoundary* specify the range of segmentation.

Some authors find this method unsatisfactory because of the problems with the colour temperature, flares and other variables that change with a daytime, weather, etc. As a possible solution is mentioned usage of colourless data. Such an approach is described, e.g., in Bahlmann *et al.* (2005). A colour and colourless data methods are compared in Bonaci *et al.* (2011). The result is that the colour data detection provides generally better results. However, success rate depends on the distance from an observer. In the case of distant signs, the colourless data method was better. The principal reason is that colour segmentation is not suitable for data that contains lesser amount of colour information. This is the case of very distant, hence small, signs.

In the case of colour data, it is desirable to minimize these problems. It is possible to use standard functions (included in OpenCV) like median filter or bilateral filter. Indeed, in the real-time applications, some of these filters can cause problems. There is a problem to achieve a proper ratio between desirable results and acceptable speed. The median filter is used to smooth the images on the basis of an average value of selected squared area of neighbour pixels. In OpenCV, method *medianBlur(inputNoisyImage, blurredImage, sizeOfKernel)* can be called. The *sizeOfKernel* parameter describes the size in pixels of the squared area that is considered during the average pixel value calculation; therefore its value must be an odd positive number.

The median filter has one major disadvantage for the sign detection. Too high values could cause blurring of object edges, and that could make sign detection more difficult. For this reason, for an image with higher signal/noise ratio bilateral filter that preserves most of the edges in the image is frequently used. The bilateral filter is also included in the OpenCV library and can be used as follows: *bilateralFilter(inputNoisyImage, outputImage, diameter, sigmaColor, sigmaSpace)*. The *diameter* parameter represents a diameter of the neighbouring pixels. The *sigmaColor* parameter states how much the pixels from the neighbourhood will influence the resulting colour. The *sigmaSpace* defines how much the pixels will affect each other when their colours are similar.

As been described, the dependence on the light conditions belongs among the principal disadvantages of the colour segmentation. Higher resistance can be found in some methods, e.g. filtration in colour spaces HSL or HSV. These methods are partially luminance invariant, and this property provides that some of the unwanted effects are filtered out (see Tran (2013)). The methods of HSL or HSV are based on hue component which filter out the particular colour tones. Similarly, the colour space CIELUV can be used (see Mathias *et al.*, 2013). Using the OpenCV library, colour model conversions can be done using the function: *cvtColor(inputImage, convertedModelImage, conversionCode)*. The conversionCode parameter selects the required conversion method. For the purposes of our application, we use a conversion from RGB to HSL; the parameter is in this case CV_BGR2HLS. Other conversion methods are named similarly.

Morphology Based Operations

Although morphology-based methods are not widely used in the road sign detection, they are in some cases used as auxiliary tools. These methods can be applied after the colour segmentation phase primarily for the identification of ROI (sometimes also called BOI – blobs of interest). Each ROI contains one object that is given e.g. by its convex hull. The convex hull represents the smallest convex region that contains all given points. The convex hull is useful especially when shape of the sign is distorted. This application is described in Maldonado-Bascón *et al.* (2007). OpenCV provides for the convex hull calculation a function *convexHull(InputArrayOfPoints, outputArrayOfHullPoints)*. This function implements Sklansky (1972) algorithm.

Despite that, in some cases, these morphological operations can be used even before the colour segmentation phase for e.g. colour noise reduction (erode and dilate operations). The dilation method convolutes a source image by a chosen kernel. This kernel is moved over the source image, the maximal pixel value overlapped by the kernel is calculated and the image pixel in the anchor point position (usually the centre of the kernel) is replaced with this new value. For instance, dark lines on a light background are narrower after the application of this method. The erode operation is the complete opposite. The minimal value under kernel is chosen; therefore, the dark areas of the image are enhanced.

Hence, a dark line on a white background is thicker after the application. The first step is to create a structuring element (convolution kernel) using *getStructuringElement(shape, size, anchorPoint)*.

The *shape* parameter determines a shape of the element (e.g. MORPH_RECT). The *size* parameter declares the size of the element and *anchorPoint* defines the place where the calculated new value will be stored. When the structuring element is defined, the erode and dilate operations can be called.

Machine Learning Methods

Methods based on artificial neural networks are widely used in computer vision applications. Several articles mention usage of artificial neural networks with Viola-Jones descriptor (frequently used for face detection). Neural networks can provide shape pre-classification and exclude false positive results of the shape detection phase (described below). According to Bonaci *et al.* (2011), false positive results decrease is from 65% to 7%, so these methods can be used especially for simplification of the classification phase.

The most frequently used machine learning method is the Viola-Jones detector that is mentioned above (see Viola and Jones (2009)). This approach is used in many projects – Mathias *et al.* (2013), Bonaci *et al.* (2011) or Bahlmann *et al.* (2005). Grayscale images are usually used as the input for the method. However, even the colour data are acceptable. In such case, the search is usually done just within a specific colour channel.

The Viola-Jones algorithm is based on so-called Haar-like features. The Haar-like feature could be perceived as an area detector. The value of a Haar-like feature is, in the simplest case, computed as a difference of luminance values of two neighbouring areas. This feature can be scaled and positioned in the image. The shape and orientation of each feature help to achieve detection of different kind of object. The basic set of features defined by Viola and Jones (see Viola and Jones (2009)) was significantly improved by R. Lienhart and J. Maydt (see Lienhart and Maydt (2002)). They extended the basic set of features with rotated variants of basic features. The Haar-like features are moved over the input image and rank the actual position. The classifiers are sorted into cascades.

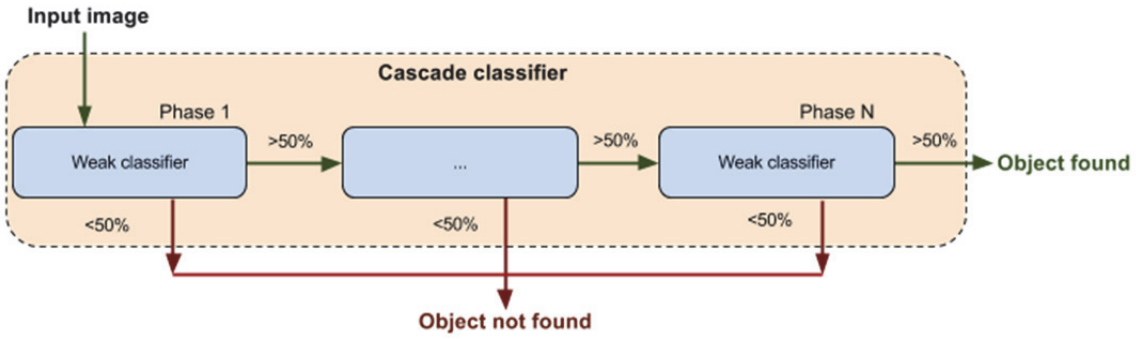
The cascade classifier is a set of weak classifiers. The sequence of weak classifiers can provide classification strength of a strong classifier. Each weak classifier has only a limited evaluation ability (e.g. the precision of correct classification is only above 50%). Nonetheless, by chaining more of the weak classifiers into a sequence (cascade), most of the undesired samples can be eliminated (see Fig. 3). Therefore, it is possible to classify the input samples with a high success rate.

Hence, even in our case, the algorithm goes through this cascade of classifiers and calculates which conditions are fulfilled. Generally, a substantial advantage is that with a sufficiently large training set, they are invariant to perspective deformation. A drawback of this approach is the computational complexity. Therefore, it has been improved.

Further, the adaptive boosting algorithm is used for improvement of this classification approach. The algorithm evaluates particular classifiers and assigns their weights. Through the process of classification, the weights are continuously recalculated, taking into account the previous results. It improves the final results for complex situations; as described in Rojas (2009).

In OpenCV, it is possible to use implementations based on Haar features (Viola-Jones algorithm) or Local Binary Patterns (LBP). Both training and detection using LBP are substantially faster than with the Haar features; despite that, the Haar classifier is generally more precise. However, quality of the training set is an essential issue. As is explained in OpenCV (2014), it is possible to train an LBP-based classifier that will provide almost the same quality of results as the Haar-based one. The library also provides several implementations of the described AdaBoost algorithm. The implementation is described in detail on the OpenCV portal.

The OpenCV library provides two basic functions for cascade classifier training: opencv_haartraining and opencv_traincascade. The latter is a newer implementation that supports both HAAR and LBP features. In our implementation, the newer version is exclusively used. A positive training set must be



3: *Cascade classification principle*

prepared to produce an XML file with the cascade. OpenCV provides a function for this purpose: it is *opencv_createsamples*. For the detection process itself method *CascadeClassifier::detectMultiScale(Input, Output, scaleFactor, minNeighbors, minSize, maxSize)* can be used. The input is an image in which we want to find the object. The results are stored into a vector of rectangles called the output. Each found object is placed in one rectangle. The parameter ScaleFactor specifies a reduction of image size at each image scale. The minNeighbors determines how many neighbour rectangles must be found to retain a candidate rectangle. The parameters minSize and maxSize determine the maximum and minimum size of the detected object.

Further advanced method for such kind of object detection is the Integral Channel Features Classifier (ICF), described by Mathias *et al.* (2013). This method was initially used for pedestrian detection in car safety systems. ICF is basically an extension of Viola-jones detector. The input of this method is not only grayscale versions of source images, but also various kinds of heterogeneous sources of information. In Dollár *et al.* (2009), the following different sources are used: grayscale images, separated colour channels, histograms of oriented gradients, results of linear (convolution) and nonlinear transformation (Canny edge detector) or threshold operation. ICF also uses AdaBoost for speeding-up of the object classification. Compared to the Viola-Jones detector is ICF more robust. Even more, the method has not very high computational complexity. However, this approach is not implemented in the OpenCV library.

Shape-based Detection Methods

The shape-based methods are focused on a selected property of the detected object (e.g. colour, shape etc.). A substantial limitation of this approach is that this particular property can suffer from a damage in source data (due to weather conditions or any other external influence). Among these methods also belongs well-known Canny edge detector used, e.g., in Garcia-Garrido *et al.* (2006). With the intention to simplify the computation, closed or almost closed edges are processed only. In such an image produced by Canny edge detector, the lines are identified using Hough transformations. For instance, for the triangle shaped signs, this method goes through the intersections of groups of three lines that forms the angle of 60 degrees. Nevertheless, this method has two significant disadvantages: it is not perspectively invariant, and rising amount of lines has a negative influence on the performance. The OpenCV library again provides both algorithms. Their explanation and usage examples can be found in OpenCV (2014).

Road Sign Detection Implementation Results

As been already described, traffic signs have specific shapes and graphical content. However, many objects with similar shape or appearance can be found in a typical environment. Precise identification of a particular traffic sign is, for this reason, a non-trivial problem. We designed an identification process composed of two phases, due to these facts. Usage of more than one method brings a substantial improvement of the process. For the first phase, we choose Viola-Jones descriptor described above. This algorithm is more suitable for this case than the colour segmentation methods because source images that were shot in bad weather conditions contain obvious colour deformations. The colour segmentation methods are therefore less suitable for this kind of system. Our preliminary tests confirmed this assumption. We used this method for the first step of the traffic sign detection in the scene. For each group of signs (on the basis of the shape) we create a detection cascade.

There are two basic approaches to prepare a training set. The first one is based on the deformation of a single picture with a desired object. The picture is rotated around all axes at specified angles and resampled to different resolutions. The training images based on this method can be prepared using this function:

```
opencv_createsamples.exe -img
inputImage.jpg -w 24 -h 24 -num 1000
-vec output.vec
```

Where *-img* specifies the path to the input image with an object, *-w* and *-h* specify the resolution of the created samples, *-num* determines the number of output samples and *-vec* specifies a .vec output file with the images in the binary format. The deformation angles of the images are left default in this case.

The other approach is based on a preparation of positive samples manually. It is necessary to prepare a text file that contains information about the picture and the object on each line. The following line is an example of a single record from the file:

```
positives\positive0.png 1 50 50 220 220
```

The first column is a path to the image, the second specifies the number of objects in this image, the next two numbers are x and y coordinates of the left upper corner of the rectangle that specifies the position of the object. The last two numbers determines the size of the rectangle. With such a configuration file, it is possible to create a training image using the function:

```
opencv_createsamples.exe -info
positives.txt -w 24 -h 24 -vec output.
vec
```

Where *-info* specifies a path to the text file in the required format. The other parameters have the same meaning as described above.

The training process also requires negative samples, where the desired object is not present. Their list must be again prepared in a text file.

Each line of the text file contains a full path to one negative sample. The training can be realized using the following function:

```
opencv_traincascade.exe -data folder
  -vec output.vec -bg negatives.txt
-numPos 100 -numNeg 50 -numStages 12 -w
              24 -h 24
```

Where *-data* specifies a folder for temporary files, *-vec* is a dataset of positive images created in the previous step, *-bg* is the text file with the paths to the negative images, *-numPos* and *-numNeg* specify the number of positive and negative samples used for every training stage, *-numStages* determine the number of training stages. The parameters *-w* and *-h* specify the resolution of the positive samples. Output is an XML file that contains the cascade.

The critical factor for a training process is the number of the negative and positive samples. In general, it is recommended to have approximately one thousand of positive samples and two thousand negative samples. However, it always depends on the specific application. For the detection process itself, OpenCV provides the function *detectMultiScale*. The output is a vector of rectangles where each rectangle contains the detected object. In our application, the output vector of rectangles contains positions and sizes of regions that match the traffic signs cascade (see Fig. 4 A). All traffic signs detected by a particular cascade are of one type (shape). In this stage, the number of false positive regions without any sign is also detected (see Fig. 4 B). It is necessary to examine all these regions and discard all of the false positive matches in the following step.

A substantial improvement can be removing of the particular image regions because a few regions in each sample have obviously a zero chance of a traffic sign occurrence (see Fig. 5 left image). Further, it is possible to set maximum and minimum size of a detected region. These restrictions can lower the false positive rate and also could reduce

4: *Correctly detected regions – A, False positive regions – B*

I: *Results of the first phase detection without any correction of a position or size of a ROI*

Number of test pictures	Quality of test pictures	Correct detection	False positive detection	False negative detection
285 (city)	poor (rainy)	6	265	1

5: *A false positive region on the left picture (top right) can be eliminated due to its position. On the right image is shown a distortion from raindrops*

the computation load for the first phase. We aimed at off-line system, hence the application of these rules were not necessary. The false positive records can be easily disposed in the identification step.

The results of our implementation of the detection phase are in Tab. I. The testing data contained samples from a standard city environment (Brno, Czech Republic) obtained during a rainy weather by a mapping car. Raindrops on the camera glass substantially distorted many samples. The samples contained seven traffic signs of the "right of way" type. Six of them were detected. The detection algorithm missed one sign due to these raindrops on the glass (see Fig. 5 right image). This single miss caused approx. 16% decrease of the detection precision on the value 84%; however, in case of larger data set, the success rate will be significantly higher. Indeed, higher accuracy of the detection process can be also achieved with more positive and negative samples.

DISCUSSION

The next step after detection of the potential traffic signs is a classification of its particular kind. Hence, the output of this step is an explicit specification of a sign in the source data image. As well as during the sign detection, machine-learning methods are used in sign classification process. Among frequently used approaches, artificial neural networks described by Garcia-Garrido *et al.* (2006) and Support Vector Machines (SVM) used by Mathias *et al.* (2013), Bonaci *et al.* (2011) or Maldonado-Bascón *et al.* (2007) belongs.

Support vector machines is a classifier that can transform nonlinear classification task to a linear one. An SVM-based method divides the training dataset into two groups and creates a hyperplane, which separates them. This algorithm is based on supervised learning principle. The main part of the algorithm is a kernel based transformation that allows to find an optimal path dividing the hyperplane. The SVM is a binary classifier; e.g. binary decision tree is used in Bonaci *et al.* (2011). OpenCV library includes support for this kind of classification. SVM implementation (OpenCV, 2014) uses two primary functions for SVM: the *CvSVM::train* method for the classifier training and the *CvSVM::predict* method for the application of the classifier on the real data. The algorithm must be combined with some feature descriptor that can be classified. A commonly used descriptor is HOG (Histogram of Gradients) or pyramids of HOGs, e.g. used by Mathias *et al.* (2013) or Bonaci *et al.* (2011). The comparison of a classification done by SVM and neural networks is outlined in Bonaci *et al.* (2011). Due to this research, SVM shows approximately 10% higher success in classification than ANN.

CONCLUSION

As been already explained at the beginning of the article, there is no optimal set of methods for traffic sign recognition. A suitable combination depends on environmental conditions during the image acquisition, hardware limitations, etc. The influence of all these factors can be limited by selection of appropriate parameters of the described methods. Even more, a different method combination must be chosen for real-time systems used, e.g. in cars.

On the basis of the previous research, we proposed a solution for a dataset that is affected by poor weather conditions and/or perspective deformations. This dataset is processed in an off-line system for traffic sign inventory. As been already described, road inventory is a time-consuming and expensive process that involves a significant amount of manual work. Such off-line detection systems can reduce time-consuming manual work of an operator.

Our solution is based on two-step processing. Firstly, the general kind of a traffic sign is identified on the basis of the shape (restrictions, warnings…). For this step, we used the Viola-Jones detector from the machine learning methods. The result is a set of regions that potentially contain a traffic sign from the particular group.

We have been able to find desired traffic signs with 84% precision. Particularly, we detected 6 of 7 signs that can be found in the data. The single missed sign was heavily distorted by raindrops; and it was difficult to identify it even for a human observer. The important result is that, with exception of this single issue, none of recognizable signs was missed.

The number of false positive samples can be significantly reduced by wider collection of training data. Moreover, these false hits can be easily filtered-out by a human operator. The time required for such task is negligible. On the other hand, search for missing sign is extremely time consuming. Therefore, we adjusted our solution to maximise the sign detection probability; even at the cost of the high false positive rate.

The second step is the identification of the precise type of traffic sign (e.g. an elk warning or train warning). For this second step, we recommend support vector machines (SVM) approach. One of the key advantages of the proposed approach is that in the first step is precisely identified the traffic sign class; therefore, the SVM in the second step is just testing different signs from this particular category. This solution allows a better adjustment of the methods and reduces the time necessary for the whole detection process.

Acknowledgement
Data used in this article were provided by GEODIS BRNO, spol. s. r. o.

REFERENCES

BAHLMANN, C., ZHU, Y., VISVANATHAN, R. et al. 2005. A System for Traffic Sign Detection, Tracking, and Recognition Using Color, Shape, and Motion Information. In: *Proceedings of Intelligent Vehicles Symposium*. 6–8 June 2005. IEEE. 255–260. DOI: 10.1109/IVS.2005.1505111.

BONACI, I., KUSALIC, I., KOVACEK, I. et al. 2011. Addressing false alarms and localization inaccuracy in traffic sign detection and recognition. In: *Proceeding of 16th Computer Vision Winter Workshop.* Mitterberg, February 2–4. Available at: http://www.zemris.fer.hr/~ssegvic/pubs/bonaci11cvww.pdf. [Accessed: 26. 11. 2015].

DOLLÁR, P., TU, Z., PERONA, P. et al. 2009. Integral channel features. In: *British Machine Vision Conference, BMVC 2009 – Proceedings*. London, 7–10 September. British Machine Vision Association, BMVA. [Online]. Available at: http://pages.ucsd.edu/~ztu/publication/dollarBMVC09ChnFtrs_0.pdf. [Accessed: 26. 11. 2015]

ECONOMIC COMMISSION FOR EUROPE – INLAND TRANSPORT COMMITEE. 1968. *Convention on Road Signs and Signals.*

GARCIA-GARRIDO, M. A., SOTELO, M. A., MARTIN-GOROSTIZA, E. 2006. Fast Traffic Sign Detection and Recognition Under Changing Lighting Conditions. In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. Toronto, 17–20. September 2006. 811–816.

LIENHART, R., MAYDT, J. 2002. An Extended Set of Haar-like Features for Rapid Object Detection. In: *International Conference on Image Processing (ICIP'02), Proceedings*. Rochester, United States, 22–25 September.

LORASKUL, A., SUTHAKORN J. 2007. Traffic Sign Recognition for Intelligent Vehicle/Driver Assistance System Using Neural Network on OpenCV. In: *The 4th International Conference on Ubiquitous Robots and Ambient Intelligence, Proceedings.* [Online]. Available at: http://bartlab.org/Dr.%20Jackrit's%20Papers/ney/3.KRS036_Final_Submission.pdf. [Accessed: 26. 11. 2015].

SALHI, A., MINAUI, B., FAKIR, M. 2014. Robust Automatic Traffic Signs Recognition Using Fast Polygonal Approximation of Digital Curves and Neural Network. *International Journal of Advanced Computer Science and Applications*. Special Issue on Advances in Vehicular Ad Hoc Networking and Applications. DOI: 10.14569/SpecialIssue.2014.040201.

MALDONADO-BASCÓN, S., LAFUENTE-ARROYO, S., GIL-JIMÉNEZ, P. et al. 2007. Road-Sign Detection and Recognition Based on Support Vector Machines. *IEEE Transactions on Intelligent Transportation Systems,* 8(2): 264–278.

MATHIAS, M., TIMOFTE, R., BENENSON, R. et al. 2013. Traffic Sign Recognition – How far are we from the solution? *Proceedings of IEEE International Joint Conference on Neural Networks*. Dallas, 4–9 August. DOI: 10.1109/IJCNN.2013.6707049.

MDČR. 2001. *Regulation 30/2001 of Ministry of Transport.* [Online]. Available at: http://www.mdcr.cz/NR/rdonlyres/DFE87D07-9E39-467D-95F9-35D3AB525369/0/MicrosoftWord30.pdf. [Accessed: 2015-01-02]

ROJAS, R. 2009. *AdaBoost and the Super Bowl of Classifiers A Tutorial Introduction to Adaptive Boosting*. Berlin: Freie University.

SKLANSKY, J. 1972. Measuring Concavity on a Rectangular Mosaic *IEEE Transactions on Computing,* C-21(12): 1355–1364. DOI: 10.1109/T-C.1972.223507.

TRAN, H. S. 2013. *Traffic Sign Recognition system on Android devices*. Massey University.

VIOLA, P., JONES, M. 2001. Rapid Object Detection using a Boosted Cascade of Simple Features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Kauai, HI; United States, 8–14 December. [Online]. Available at: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf. [Accessed: 26. 11. 2015].

Contact information

Karel Zídek: karel.zidek@mendelu.cz
Tomáš Koubek: tomas.koubek@mendelu.cz
David Procházka: david.prochazka@mendelu.cz
Marcel Vytečka: marcel.vytecka@mendelu.cz