Automatic mapping of traffic signs

Bence Dusek [a], Mátyás Gede [a, *]

[a] *Institute of Cartography and Geoinformatics, Faculty of Informatics, Eötvös Loránd University, Bence Dusek- dusekbence@gmail.com, Mátyás Gede - saman@map.elte.hu*

* Corresponding author

Abstract: Nowadays, people easily can get into their cars and drive hundreds of kilometers in a few hours, but for that to work efficiently a system of rules must be applied and those rules have to be communicated transparently. This is why traffic signs are an influential part of our lives and every kind of information about each is helping the government, the community, and the drivers. This paper presents a novel and cost-efficient method for acquiring information on traffic signs, such like the category and the 3D position. The former can be gained using camera images and a Convolutional Neural Network model. The latter can be obtained using positioning devices. With the help of a GNSS device the absolute position of the vehicle can be learned and based on that a local coordinate system can be established. From the vehicle's point of view the coordinates and the orientation of the traffic sign can be acquired by applying a stereo camera and an IMU (Inertial Measurement Unit) sensor. Then, with the help of these attributes a large database can be built, maintained, and updated. This project displays that adequately precise data can easily be accessible using a few cheap devices and sensors.

Keywords: deep learning, traffic sign detection and recognition, mobile mapping system, computer vision

1. Introduction

Artificial intelligence has deeply woven itself into today's technology. One of its branches, deep learning is a really powerful tool for extending the boundaries of many distinct territories of life itself (computer science, medicine, psychology, even geography). The authors of this paper aimed to create an effective and cost-efficient application of this technology in the field of GIS, thus we started to experiment traffic sign recognition and mapping system powered by deep learning and using a variety of easily accessible tools for absolute and relative positioning (stereo camera, GPS device, IMU device). Although at the time of writing of this article some phases are still in a theoretical or preliminary shape but based on our – and other's – experiments it is completely feasible.

Our first step was understanding deep learning and artificial neural network (ANN) on a fundamental level. Then based on this knowledge creating a model that is capable of recognizing the various traffic signs. This model is a convolutional neural network (CNN), which is a special kind of ANN used mainly for image recognition and pattern detection (Rosebrock, 2017).

After training the network, we needed to create a tool that could utilize this model as intended. This meant the construction of a program that can analyze input images whether they contain traffic signs. If so, take that part of the current image and pre-process it to a format that the model can work with. Afterwards, based on how well the model was trained, it gives back a prediction value that the program prints out in a percentage format.

In the near future, we plan to integrate the aforementioned devices and a single-board computer (e.g. a Raspberry Pi) into a portable device that can be mounted into any vehicle. The stereo camera is needed for sign detection and to determine the position of the detected traffic signs from the vehicle's point of view. The GPS device provides the position of the vehicle in a global coordinate system, and the IMU (Inertial Measurement Unit) device the orientation of the vehicle.

Many studies are experimenting with mobile mapping systems supplemented with LiDAR technology, relying heavily on point clouds in the matter of positioning, detecting and identifying the traffic signs (Solián et al., 2016), (Balado et al., 2020). This way it is possible to achieve more precise coordinate values but it is extremely expensive to acquire such a device and it requires an expert to operate properly.

The key concept in this article is to make an architecture from the most affordable and easily accessible instruments that can detect, predict and position traffic signs as accurately as possible.

2. Proposed method

2.1 Traffic sign detection

The core component of the project is the sign detection. All the other parts of the procedure are based on this step so it is extremely momentous that this would operate with maximal efficiency. A fortiori, on the road where precision and vigilance are key factors for safe driving, it is inadmissible to have major inaccuracies.

Traffic signs are immensely conspicuous just for drivers to notice them in time. Consequently, there is a considerable number of methods to distinct a traffic sign from its surroundings in the field of computer vision. It is possible to capture the color, the shape, (Liu et al., 2020) or the retro-reflectivity (Solián et al., 2016) of the signs and based on the (individual or mixed) detection of these properties can we achieve the desired outcome.

Color-based detection is all about guessing the right threshold value that can separate the signs from their backgrounds (Ritter, 1992). Most often, the HSI, HSV, HSL color models are being used for this procedure, since these are easily accessible and the most up to par with the ever-changing lighting conditions but most times it's still not enough. As practice shows, it is difficult to preset the hue value of the aforementioned color spaces (Liu et al., 2020).

The retro-reflective trait of a traffic sign can primarily be exploited for LiDAR detection. From the generated point clouds the reflection of a sign can easily be segmented and processed for recognition, like in the work of Solián et al., 2016 for example.

There are countless already tested shape-based techniques like the Hough transform (González et al., 2011) or radial symmetry (Barnes et al., 2008). The one our detection algorithm is using at the time of writing of this article is a contour-based detection, mainly because it's effective and easy to implement. We are stating this, because finding the most adequate method (or methods) always takes experimenting. Contour detection is a well working technique, but it might be rewarding to mix and match it with other already existing methods.

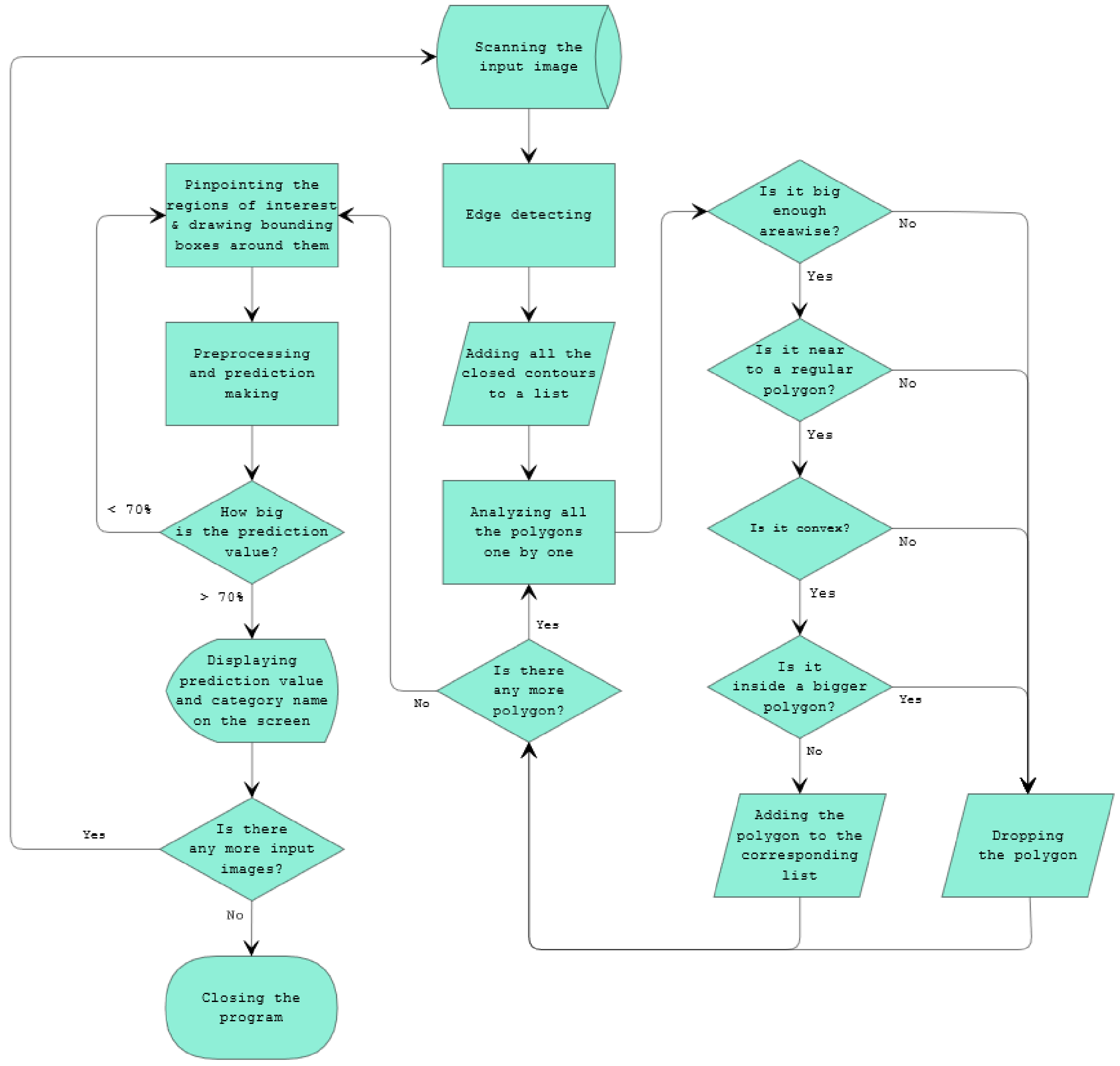
Scanning the input image
Edge detecting
Adding all the closed contours to a list
Analyzing all the polygons one by one
Is it big enough areawise?
No
Yes
Is it near to a regular polygon?
No
Yes
Is it convex?
No
Yes
Is it inside a bigger polygon?
Yes
No
Adding the polygon to the corresponding list
Dropping the polygon
Is there any more polygon?
Yes
No
Pinpointing the regions of interest & drawing bounding boxes around them
Preprocessing and prediction making
How big is the prediction value?
< 70%
> 70%
Displaying prediction value and category name on the screen
Is there any more input images?
Yes
No
Closing the program

Figure 1. Flowchart of the detection program

The basic operation of our detection and recognition software can be seen in Figure 1. With the help of a Canny edge detector, the software is able to notice all edges on the current image, tries to fit polygons to closed contours, and saves these polygons to a list. Then the program analyzes whether these polygons are large enough areawise, near regular, convex, and are not inside other bigger polygons. If a polygon doesn't meet the requirements, the program drops it, but if it does then the program – depending on how many corners it has – saves it to one of the corresponding lists. There are three lists: one for triangles, one for rectangles, and one for octagon shapes (Figure 2). Note, that the detector perceives the circle shaped signs as octagons (which is not a problem as we only need to find the regions of interest). This cycle takes as long as the original list contains polygons.

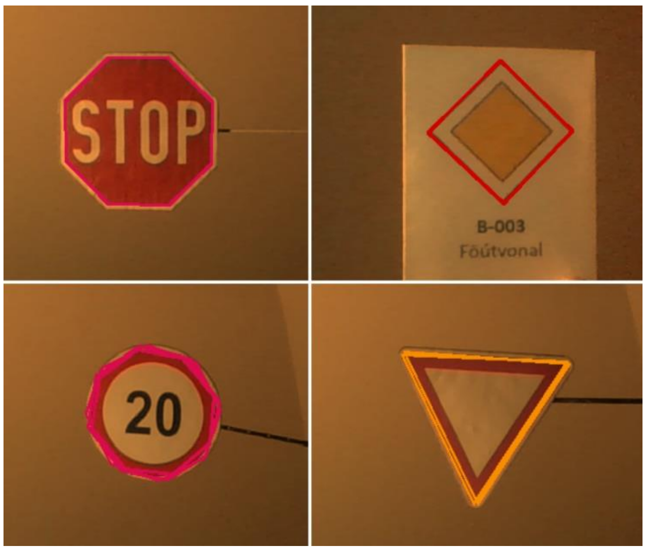

Figure 2. The three shapes the program is detecting: triangle, rectangle, octagon (the circle is perceived as an octagon)

These polygons are the “road sign candidates” on the image. (As we later see, many of these candidates are just arbitrary rectangular or circular shapes, but they include all clearly visible traffic signs.) Based on the three lists, the program creates regions of interest by drawing an expanded bounding box around polygons, and cuts them out of the current image. These images are then preprocessed in order to fit them as input of the CNN model which can make a prediction on them. If the highest prediction probability does not exceed 70% for one region, the candidate is dropped. But if it does, then the program will put out the exact prediction value and the category name (Figure 3).

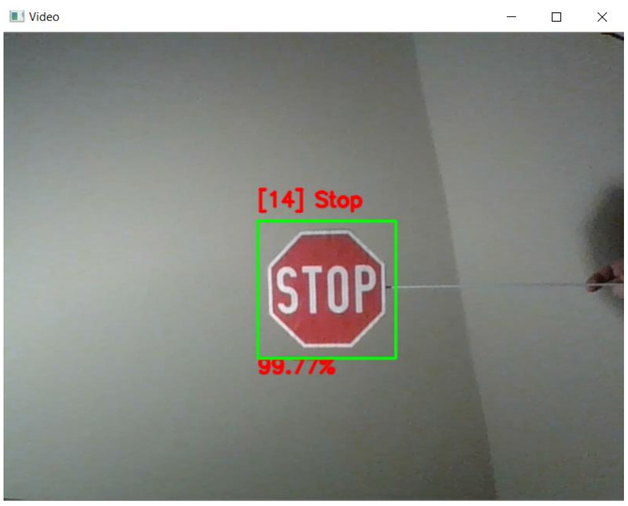

Figure 3. The final result of the program

After completing all these steps, the program examines whether there are more input images that need to go through the cycle again, or not and the program can shut down.

2.2 Traffic sign recognition

As it was stated and discussed in the previous subsection, traffic sign detection is the core element of a project like this. But also, as in real life, one can look, but not see. Without comprehension, information are just words without context. It works similarly in this case as well; the detection part of the program will perceive the traffic sign, but only the recognition part can identify the exact type of that sign, only that can tell what it indicates, and put it into context. In this project, a convolutional neural network is serving the role of this “interpreter”, with its help the system is able to classify traffic signs from images.

Our model is a deep learning-based CNN, which is a special kind of artificial neural network and it was inspired by the behavior of the human brain itself. Like the brain’s, an ANN model’s building block is the neuron which is a little node that only activates and sends the information forward to the next neuron if the input value is big enough.

The essence of an ANN model is that it can teach itself based on the given database, this method is called machine learning or deep learning. The main difference between the two is the number of layers the neurons are grouped in, the more layer the deeper the model (Rosebrock, 2017). The first two is self-explanatory: the input layer handles the models input data and passes on to the hidden layer and the output layer handles the output data of the model. The hidden layer is the main component, it is responsible for the model’s “self-teaching” property.

CNN models are widely used in the case of image recognition. In these cases, a special kind of hidden layer type is used, called convolutional layer. These layers detect the occurring patterns with the help of filters/kernels scanning through the given images.

The training program we used was based on one from the internet which is freely modifiable and accessible (Hassan, 2020). The reasoning behind this is, that many of these programs are already well parameterized, thus can teach a model to recognize a traffic sign appropriately. The database behind, is the German Traffic Sign Recognition Benchmark (or GTSRB for short), which is a highly popular and easily downloadable image collection of the traffic signs located in Germany, many developers use it for training CNN models (Stallkamp et al., 2012). At the moment, we only added little modifications to this part of the project since it’s performing adequately well, but in the near future, it will be highly crucial to have a few extensions added to the program.

2.3 Traffic sign localization

Based on all of the above, we get a solid foundation that can detect and recognize various traffic signs. But, with only this much, it certainly could not be listed under the topic of GIS. Traffic sign localization changes that.

Traffic sign localization means that the exact location of a traffic sign can easily be pinpointed either locally, globally, or both. According to our plans, we want to set up an architecture similar to the one in the article, Madeira et al., 2005. In brief, a stereo camera for the local attitude, a GPS device for the global attitude, and an IMU device for the directions is needed. While it does cost some money to purchase and install these devices, it is still much cheaper than a mobile mapping system equipped with a LiDAR sensor.

The details are not fully developed yet, but the basic operation of the system is explained in Figure 4. The GNSS and IMU sensors would most likely be a part of a small, hand-held device (i.e., mobile phone) and the stereo camera would be attachable through a cable. With the help of the first two instruments, we could get detailed 3D information about the vehicle. With the camera, it becomes possible to capture the traffic sign's position from the vehicle's point of view. And with the results of all three measuring devices combined, the coordinates, orientation, and type of the traffic sign would become available in a global coordinate system.

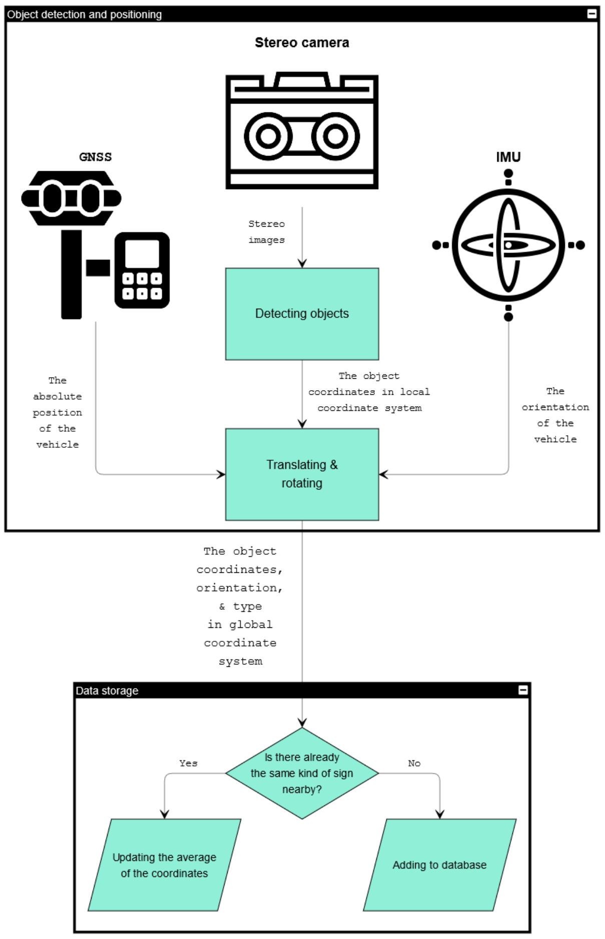
Object detection and positioning
Stereo camera
GNSS
IMU
Stereo images
Detecting objects
The absolute position of the vehicle
The object coordinates in local coordinate system
The orientation of the vehicle
Translating & rotating
The object coordinates, orientation, & type in global coordinate system
Data storage
Yes
Is there already the same kind of sign nearby?
No
Updating the average of the coordinates
Adding to database

Figure 4. Diagram of the proposed traffic sign mapping system

After these measurements, the program would examine whether a particular traffic sign already has an entry in the database or not. If so, no new entry would be added, it would only update the already existing one linked to that sign by calculating an average from the original and the new value.

3. Results and future prospects

At the time of writing of this paper, the sign detection and recognition part of the project has been already implemented. The program has been tested via two different methods: cut out cardboard signs in front of a web camera and video files recorded during a car ride. The latter is essential, because it models the intended use well enough.

During the tests, the contour-based detection rarely missed any traffic signs on the video feed. Even if a sign was not detected on a certain frame, it usually had been found on some of the previous or next frames as its position had changed on the image. Unfortunately, contour detection picked many circular or rectangular non-sign objects as well: wheels, other roadside signs, windows (Figure 5).

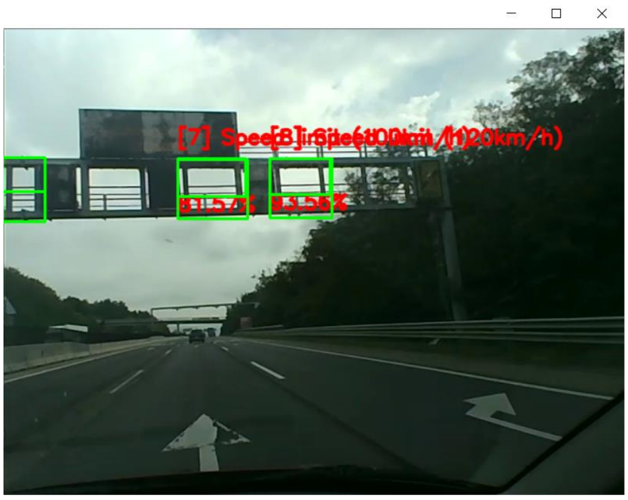

Figure 5. The problem of misdetection, in some cases the program even recognizes them as traffic signs, falsely

This also affected the success rate of sign recognition. As the model had been trained on traffic sign images only, the system always tried to sort non-sign images also into one of the sign classes. However, there are a few approaches how to remedy this problem. In the previous chapter I mentioned that we will experiment more on the detection part to decide what method is the best to use for more accurate results. An alternative way to decrease the occurrence of this issue is through extension. Hopefully, this problem of misdetection can be reduced significantly by adding one or more “not a sign” class to training data, including several sample images.

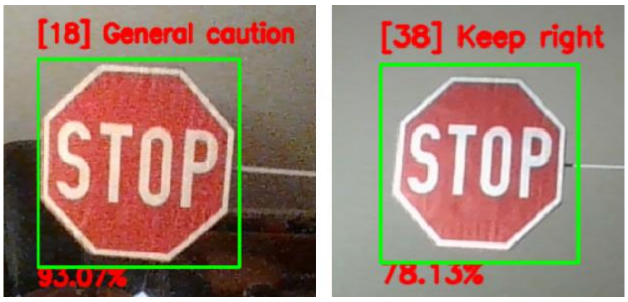

Figure 6. The problem of misrecognition

Another significant issue that needs to be corrected is the misrecognition (Figure 6). In the GTSRB there are approximately 35000 images in 43 traffic sign categories. This is an enormous amount of data which are not evenly distributed across the categories and therefore the recognition task cannot predict uniformly well on the various traffic signs. For instance, the “Speed limit (20 km/h)” folder only contains 180 images, the “Stop” folder has 690, and the “Yield” folder has 1920. As it can be seen, there is a huge variance between the numbers of the images, which evidently effects the results (Figure 7 and Figure 8). Demonstrably, this can only be solved if we expand the database with additional images and categories. Furthermore, it needs to be adjusted for one country’s traffic system (in our case it probably will be Hungary’s).

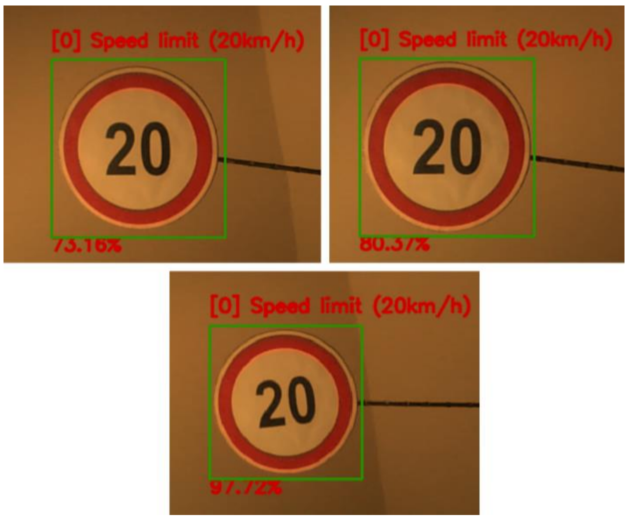

Figure 7. The fluctuation of the prediction value in the case of the “Speed limit (20 km/h)” traffic sign (First value: 73.16%; Second value: 80.37%; Third value: 97.72%)

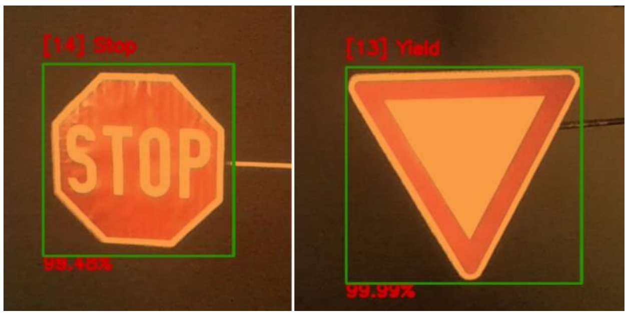

Figure 8. The prediction value of the “Stop” and “Yield” signs

One of the important aspects, that still needs to be decided, is whether the system would work in real-time or based on post-processing. In the former, the data would require no direct processing by the user’s hand and would bring faster results. On the other hand, the latter is easier for the developer to implement. Despite this, the authors plan to examine if real-time processing is possible using low cost devices, using some resource optimization if neccessary.

One of the ideas for better optimization is reducing the number of frames per second that need analyzing. By previous calculations, 10 to 15 FPS would be suitable for getting an adequate amount of information from the images. This could work as well in populated areas as in highway environments. Connected to this, if the vehicle stops, the processing of the system should be minimalized. That means that the detection part should fully halt for the time being. If there is a sufficient size buffer in the pipeline, and recognition part works asynchronously, the average speed should increase on the same hardware.

Another idea is the inverse task. It would be undoubtedly useful, if the system could be capable not just to register traffic signs, but also inspect whether an already registered sign – from a previous car travel – is at its place. If not, then register it in the respective record of the database and warn the user about it as well.

Though, a lot of grounds have already been covered, there is still much to be done. Future work will focus on the implementation of the previously discussed matters. Especially important aspects are the extension of the database, the filtering out of junk data and the improvement and fine-tuning of the CNN method. This way, much more accurate prediction values could be achievable, which would mean more reliable data to use.

4. Acknowledgements

EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies – The project is supported by the Hungarian Government and co-financed by the European Social Fund.

5. References

Balado, J., González, E., Arias, P. and Castro, D. (2020). *Novel Approach to Automatic Traffic Sign Inventory Based on Mobile Mapping System Data and Deep Learning*. DOI: 10.3390/rs12030442

Barnes, N., Zelinsky, A. and Fletcher, L. S. (2008). *Real-time speed sign detection using the radial symmetry detector*. IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 2, pp. 322–332. DOI: 10.1109/TITS.2008.922935

González, Á., Garrido, M. Á., Llorca, D. F., Gavilan, M., Fernandez, J. P., Alcantarilla, P. F., Parra, I., Herranz, F., Bergasa, L. M., Sotelo, M. Á., Revenga de T. P. (2011). *Automatic traffic signs and panels inspection system using computer vision*. IEEE Transactions on Intelligent Transportation Systems, vol. 12, no. 2, pp. 485–499. DOI: 10.1109/tits.2010.2098029

Hassan, M. (2020). *Opencv Projects – Traffic Sign Classification*, https://www.murtazahassan.com/courses/opencv-projects/ (last read: 05.04.2021.)

Liu, L., Tang, X., Xie, J., Gao, X., Zhao, W., Mo, F. and Zhang, G. (2020). *Deep-Learning and Depth-Map Based Approach for Detection and 3-D Localization of Small Traffic Signs*. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 2096-2111, April 27, 2020, DOI: 10.1109/JSTARS.2020.2966543.

Madeira, S. R., Bastos, L. C., Sousa, A. M., Sorbal, J. F. and Santos, L. P. (2005). *Automatic Traffic Signs Inventory Using a Mobile Mapping System*. GIS PLANET 2005, International Conference and Exhibition on Geographic Information

Ritter, W. (1992). *Traffic sign recognition in color image sequences*. in Proc. Intell. Vehicles'92 Symp., pp. 12–17.

Rosebrock, A. (2017). *Deep Learning for Computer Vision with Python*. 1st ed. Pyimagesearch.com. Philadelphia, Pennsylvania.

Soilán, M., Riveiro, B., Martínez-Sánchez, J. and Arias, P. (2016). *Automatic Road Sign Inventory Using Mobile Mapping Systems*. XXIII ISPRS Congress, 12–19 July, 2016, Prague, Czech Republic. DOI: 10.5194/isprs-archives-XLI-B3-717-2016

Stallkamp, J., Schlipsinga, M., Salmena, J. and Igelb, C. (2012): *Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition*. DOI: 10.1016/j.neunet.2012.02.016