Műszaki Tudományos Közlemények vol. 9. (2018) 39–42.
https://doi.org/10.33894/mtk-2018.09.05
Hungarian: https://doi.org/10.33895/mtk-2018.09.05,
http://eda.eme.ro/handle/10598/30334

THE IMPLEMENTATION OF AN OPENCV-BASED TRAFFIC SIGN IDENTIFIER VIDEOANALYST SOFTWARE

Viktor BALÁZS[1], László SZILÁGYI[2], Antal APAGYI[3], Timotei István ERDEI[4]

[1, 2] *University of Debrecen, Faculty of Informatics, Debrecen, Hungary*
[1] *groolue@gmail.com*
[2] *sziszi610@gmail.com*
[3, 4] *University of Debrecen, Faculty of Engineering, Department of Mechatronics, Debrecen, Hungary*
[3] *apagyi.toni@gmail.com*
[4] *timoteierdei@gmail.com*

Abstract
Nowdays, accidents tend to happen because our attention is being split up by the ever-growing influx of information, losing the focus from the driving, traffic signs, and other signals. The consequences of these minor or major accidents weight down on our shoulders. During our project, we tried to eliminate, or help this issue, using present technology, improving upon that, trying to avoid these accidents. Our task consisted on implementing a software, that could identify traffic signs from any video streams.

Keywords: *OpenCV, C++, traffic signs, video, Computer Vision, Industry 4.0.*

1. Introduction

In the last decade image and video processing has made great leaps in development. The basic premise was a shape detecting software, which could recognize triangular, circular and polygonal shapes on pictures and indicate them with different colours, and - in anticipation of further developments – distinguish between the red triangle and red circle.

In this project, this software was further developed and transformed to be able to detect traffic signs on videos.

During the program's development it served as an inspiration that Tesla, Google, Uber, and other greater car manufacturer companies are designing and testing self-driving cars, some of which are already in circulation. Google's own self driving car, called Waymo, has already reached more than 3 million miles **[1]**.

2. Planning and principle of operation

Initially, we wanted to use all parts of our previous work, but during planning, we added new features and altered parts of the initial program.

Some of the current image analysis systems use machine learning, others search for objects seen in pictures in a pre-made database **[3]**. During software development, human nervous system and brain processing ability are often taken into consideration **[4]**.

The program processes the individual frames of the given video file; analysing single images one-by-one rapidly. On these pictures, the triangle, circle and quadrilateral recognition functions run parallel.

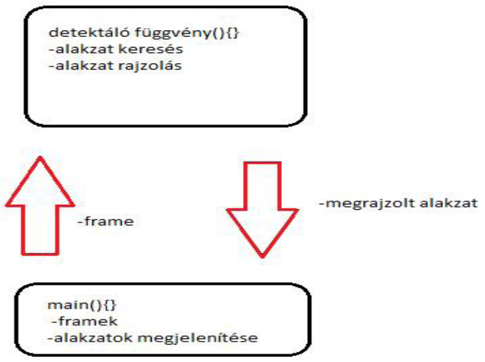


Figure 1. *Principle of operation*

Polygon recognition functions work with the contours that can be found on the image, while circle recognition is achieved with the help of the Hough-transformation. In order to recognize shapes, images first must be converted to binary images, on which shape detection functions can work.

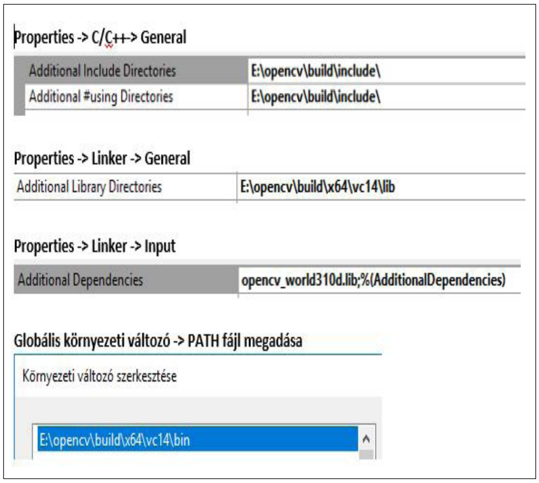
Videos were converted to 640x360 resolution and 30 fps format using Freemake Video Converter **[5]**.

3. Hardware and software

Due to the high computing power requirement, the project was developed on an ASUS X550C notebook. The computer has Intel (R) Core (TM) i5-3337U @ 1.80GHz CPU, 8GB DDR3 1600MHz memory and NVIDIA GeForce GT 720M video card and can achieve 1366x768 resolution. VGA based image analysis is key to accomplishing this task.

The computer runs a 64-bit Microsoft Windows 10 Home operating system. The program was written in C++ with the OpenCV image processing library in Visual Studio 2015 IDE **[6]**. Several video conversion software packages were used to achieve the correct video input. OpenCV must be imported to every new project. This happens in the project > properties menu, where we can provide the path to the folder that contains the OpenCV files, in the C/C++ menu. In the linker menu we can decide which library file should be used.

Additionally, the full path of the OpenCV folder must be added to the PATH global environment variable.



Properties -> C/C++> General

Additional Include Directories	E:\opencv\build\include\
Additional #using Directories	E:\opencv\build\include\

Properties -> Linker -> General

Additional Library Directories	E:\opencv\build\x64\vc14\lib

Properties -> Linker -> Input

Additional Dependencies	opencv_world310d.lib;%(AdditionalDependencies)

Globális környezeti változó -> PATH fájl megadása

Környezeti változó szerkesztése

E:\opencv\build\x64\vc14\bin

Figure 2. *Importing OpenCV*

4. Operation of the main program

After scanning the appropriate header files, the task of the main program is to split the input video into frames for further processing. For video analysis and image processing we used OpenCV's cv.hpp core.hpp, highgui.hpp and video.hpp header files.

The video file is stored in the cvCapture variable, which is OpenCV's data type for handling videos.

The video is scanned into the variable using the cvCreateFileCapture() function by giving it the path of the video. The frames are cut by the cvQueryFrame() method. The frames are processed in a loop, here it is important to mention that the computer has relatively strong hardware, but is still not suitable for processing all the frames, so only every 10th frame is analysed.

Images taken from the video are then resized, so the unnecessary parts of the image can be excluded from processing. This is possible because traffic signs are usually on the right side of the road, so it is sufficient to analyse only the right-hand part of the image. This is done by using the cvSetImageROI() function (Region of Interest). Frames are stored in IplImage structure, which corresponds to the Intel Image Processing Library format. One frame is stored in 3 structures of this type, so that during parallel processing one of the analysing functions can be called on each image separately. In the main program, in the loop, the given function is called on every thread and the returned image is displayed in a window. Parallelism can be achieved by using the <omp.h> header file. The program only displays the window if it finds a shape, thus improving performance. Windows are displayed using the cvShowWindow method, while using cvMoveWindow we can move it to the x,y coordinates given as parameters.

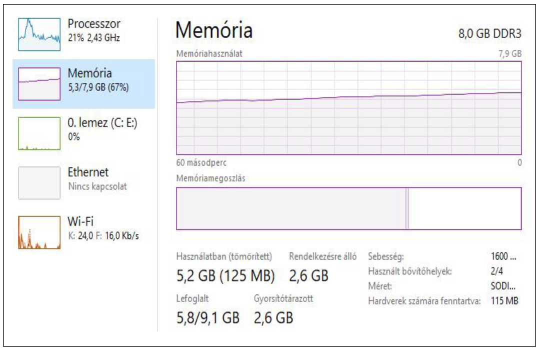
At the end of the run, these windows can be closed and the reserved memory freed by pressing a button.

5. Performance and function

The program was tested on a 5 minutes long video, during which the different traffic signs were detected with small errors. Despite the low quality of the video and the fact that images are dismantled continuously, the program used more than 2GB memory. In addition, the CPU utilisation did not rise above 25% (2.50GHz). The memory requirement can increase with the length and resolution of the video.

During its run, the program recognized and contoured traffic signs with about 70% accuracy (69.1%). This is a statistically good ratio, which can be further increased by using additional algorithms and higher resolution videos. However, analysing videos recorded at high speed might reduce this ratio, because of blurring and reflections.

During the analysis, the video is played in a separate window, and there are three different windows in which the circle, triangle and rectangle shaped signs are displayed. These are called Circle, Triangle and Rectangle.



Processzor 21% 2,43 GHz
Memória 5,3/7,9 GB (67%)
0. lemez (C: E:) 0%
Ethernet Nincs kapcsolat
Wi-Fi K: 24,0 F: 16,0 Kb/s

Memória 8,0 GB DDR3
Memóriahasználat 7,9 GB
60 másodperc 0
Memóriamegoszlás

Használatban (tömörített) 5,2 GB (125 MB)
Rendelkezésre álló 2,6 GB
Lefoglalt 5,8/9,1 GB
Gyorsítótárazott 2,6 GB
Sebesség: 1600 ...
Használt bővítőhelyek: 2/4
Méret: SODI...
Hardverek számára fenntartva: 115 MB

Figure 3. *Performance at the end of the run*

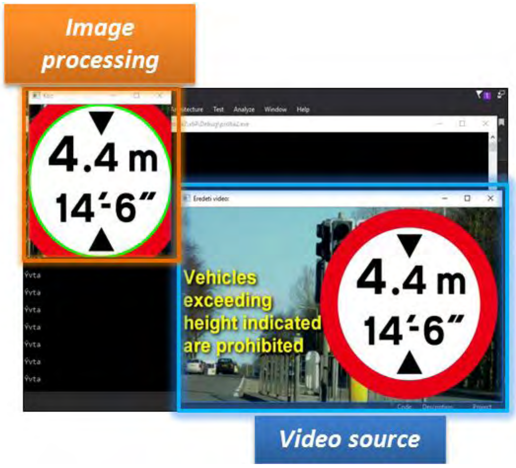


Figure 4. *Test I.*

6. Conclusions

Our endeavour seems to be fruitful, as our traffic sign detecting software was completed. The program analyses the video in the given path, it recognises and displays the traffic signs on the video. After the run, the program closes the displayed windows and itself. During planning it was conceived that current generation hardware has high computing capacity, so a Raspberry PI 3 may also be able to perform image analysis with „reduced" parameters.

Another option would be to further develop the program to a point where it could be part of a self-driving vehicle.

However, it can be stated that performance may be different when implementing the program on another language or using a different operating system. OpenCV is available for Java, Python, C++ and C and supports Windows, Linux, Mac OS, iOS and Android operating systems.

Moreover, the program can be transposed into a more powerful work environment, where analysing higher quality videos would be possible, and possibly it could be developed further to lower resource requirements.

Acknowledgments

The work/publication is supported by the EFOP-3.6.1-16-2016-00022 project. The project is cofinanced by the European Union and the European Social Fund.

References

[1] Waymo, *On the Road, 2017*.
https://waymo.com/ontheroad/
(accessed 2017, May 18)

[2] Kichun Jo, Junsoo Kim, Dongchul Kim, Chulhoon Jang, Myoungho Sunwoo: *Development of Autonomous Car—Part I: Distributed System Architecture and Development Process*. IEEE Transactions on Industrial Electronics 61/12. (2017) 7131-7140.
https://www.doi.org/10.1109/TIE.2014.2321342

[3] M. Russell, S. Fischaber: *OpenCV based road sign recognition on Zynq*. Industrial Informatics (INDIN), 11th IEEE International Conference, 29-31 July 2013.
https://www.doi.org/10.1109/INDIN.2013.6622951

[4] C. Y. Fang, C. S. Fuh, P. S. Yen, S. Cherng, and S. W. Chen: *An automatic road sign recognition system based on a of human recognition processing*. Computer Vision and Image Understanding 96/2 (2004), 237–268.
https://doi.org/10.1016/j.cviu.2004.02.007

[5] Freemake, *Freemake Video Converter*, 2017.
http://www.freemake.com/hu/downloads/
(accessed 2017, May 14).

[6] OpenCV, *Releases - OpenCV library*, 2017.
http://opencv.org/releases.html
(accessed 2017, May 14).

[7] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, T. Koehler: *A System for Traffic Sign Detection, Tracking, and Recognition Using Color, Shape, and Motion Information*. In: IEEE Proceedings. Intelligent Vehicles Symposium, 6-8 June 2005, Las Vegas, USA.
https://doi.org/10.1109/IVS.2005.1505111

[8] H. X. Liu, B. Ran: *Vision-Based Stop Sign Detection and Recognition System for Intelligent Vehicle*. In: Transportation Research Record Journal of the Transportation Research Board 1748/1 (2001) 161-166.
https://doi.org/10.3141/1748-20

[9] Rapsberry PI, *Raspberry Pi 3 Model B specifications*, 2017.
https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
(accessed 2017, May 14).

[10] BoofCV, *Overall Runtime Speed*, 2017.
http://boofcv.org/notwiki/images/benchmark_surf/overall_all_speed.gif
(accessed 2017, May 17).

[11] G. Husi, T. I. Erdei, Zs. Molnár: *A Novel Design of an Augmented Reality Based Navigation System & its Industrial Applications*, 15. IMEKO TC10 – Technical Diagnostics in Cyber-Physical Era Budapest, 6–7.06.2017.

[12] G. Husi: *Minőségmenedzsment-rendszerek módszereinek alkalmazása a Magyar Köztársaság Rendőrségénél*, Phd thezis, 2006, 120.

[13] A. Husam, A. S. Adila, Zs. Molnár, T. I. Erdei, G. Husi: *Reviewing the notable progress of effective techniques in the development of stroke hand rehabilitation*. In: A XXII. Fiatal műszakiak tudományos ülésszak előadásai. Proceedings of the 22th international scientific conference of youngth engineers, Kolozsvár/Cluj, Kolozsvár, Románia, Műszaki Tudományos Közlemények 7. (2017) 63–66.
https://eda.eme.ro/handle/10598/29814

[14] N. C. Obinna, T. I. Erdei, Zs. Molnár, G. Husi: *LabVIEW Motion Planning and Tracking of an Industrial Robotic Manipulator (KUKA KR5 arc): Design, Modelling, and Simulating the Robot's Controller Unit*. In: A XXII. Fiatal műszakiak tudományos ülésszak előadásai. Proceedings of the 22th international scientific conference of youngth engineers, Kolozsvár/Cluj, Kolozsvár, Románia, Műszaki Tudományos Közlemények 7. (2017) 331–334.
https://eda.eme.ro/handle/10598/29838

[15] T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi: *AGV cyber physical navigation system*. In: A XXII. Fiatal műszakiak tudományos ülésszak előadásai. Proceedings of the 22th international scientific conference of youngth engineers, Kolozsvár/Cluj, Kolozsvár, Románia, Műszaki Tudományos Közlemények 7. (2017) 135–138.
https://eda.eme.ro/handle/10598/29834