doi: 10.7250/tcc.2014.004

2014 / 15

Image Pre-processing Methods for Traffic Sign Recognition

Artjoms Suponenkovs[1], Aleksandrs Glazs[2], [1, 2]*Riga Technical University*

***Abstract*–The presented paper investigates the problems of image pre-processing methods for traffic sign recognition. It describes different methods and algorithms that allow to make Traffic Sign Recognition (TSR) systems adaptable for real-life environment and to convert the input information (from the camera) to a usable format for analyzing information about a traffic sign. In the experimental part of the paper the most important aspect regarding the comparison of image pre-processing algorithms is illustrated.**

***Keywords*–Adaptive binarization, computer vision, image pre-processing, segmentation, traffic sign recognition.**

I. Introduction

The topic “Development of Traffic Sign Recognition System” is very popular due to the increasing number of cars and road accidents. Therefore, a lot of well-known companies like BMW, Volkswagen, Saab, Google Car [1]–[3] – have been using the Traffic Sign Recognition system. The TSR system can improve the attentive behavior of drivers and makes it possible to create autonomous cars. These cars, with TSR system, have a camera for getting video information. For the reason that TSR system works in real–life environment, input information from the camera is affected by various factors, such as weather conditions, illumination and others – which complicate the recognition process. For this reason we must use image pre–processing which makes image filtration and converts input information into usable format for further analysis. A lot of well-known approaches to TSR Development [4]–[7] have been using computer vision libraries (for example: OpenCV) and Multi-Layer Artificial Neural Networks. However, these approaches need huge amount of resource and high performance. As well, usually universal algorithms from computer vision libraries are not effective enough for traffic sign recognition. Thus, this paper describes effective approaches to image pre–processing development for traffic sign recognition.

II. Problem Statement

There are descriptions of different methods and algorithms which allow to adapt the Traffic Sign Recognition (TSR) system for real–life environment and to convert the input information (from camera) to usable format for analyzing information about traffic signs. The image pre–processing consists of four parts:

Part 1
- a) RGB/RAW to HSV conversion;
- b) image binarization (marked with blue and red color).

Part 2
- a) “macro” segmentation;
- b) edge detection (signature).

Part 3
- a) brightness adaptation;
- b) color (of the pixel) corrections;
- c) “micro” segmentation.

Part 4
- a) sign image normalizer (rotation).

III. Proposed Approach

Image pre-processing consists of many algorithms and methods. The total structure of TSR system is shown in Fig. 1 where the image pre-processing blocks are marked blue. Image pre-processing simplifies shape recognition, color analysis and sign content analysis. Before the shape recognition, traffic sign edge is marked by signature [8, Ch. 11]. Prior to making color analysis, color of the pixel is corrected by brightness adaptation and additional information algorithms. Before analysis of traffic sign content, sign image is scaled and rotated [9].

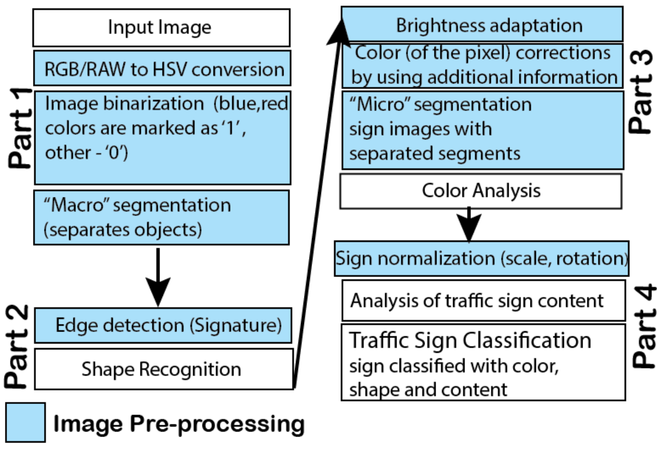

Input Image
RGB/RAW to HSV conversion
Image binarization (blue,red colors are marked as '1', other - '0')
"Macro" segmentation (separates objects)
Edge detection (Signature)
Shape Recognition
Part 1
Part 2
Brightness adaptation
Color (of the pixel) corrections by using additional information
"Micro" segmentation sign images with separated segments
Color Analysis
Sign normalization (scale, rotation)
Analysis of traffic sign content
Traffic Sign Classification sign classified with color, shape and content
Part 3
Part 4
Image Pre-processing

Fig. 1. TSR flowchart.

A. HSV Conversion

At the beginning TSR system takes the picture from camera (Fig. 1, Input Image). Usually the picture color model is in RGB or digital photography format RAW. The RGB or RAW is affected by real–life environmental factors [13, Ch.1]: lightness, weather conditions and others – therefore all color coordinate (R, G, B) values are changed. We have to find the color model that is least affected by real-life environmental factors. The Table I shows red color coordinate dependence on real–life environmental factors.

TABLE I

THE RED COLOR DIAPASONS DEPENDENCE ON REAL-LIFE ENVIRONMENTAL FACTORS

Color model	**Clear day**	**Rain / Humidity**	**Evening**	**Night**
RGB	129<R<255 0<G<71 0<B<96	129<R<255 0<G<71 0<B<96	156<R<255 0<G<98 0<B<44	100<R<255 0<G<98 0<B<20
YCbCr	40<Y<120 100<Cb<128 140<Cr<200	30<Y<120 100<Cb<128 150<Cr<200	60<Y<170 77<Cb<115 170<Cr<206	27<Y<180 70<Cb<120 156<Cr<207
Lab	30<L<100 15<A<81 0<B<40	15<L<100 15<A<81 0<B<70	15<L<100 15<A<81 0<B<40	15<L<100 15<A<81 0<B<60
HSV	**300<H<360** **0.7<S<1** **0.5<V<1**	**300<H<360** **0.7<S<1** **0.5<V<1**	**300<H<360** **0.7<S<1** **0.5<V<1**	**22<H<360** **0.7<S<1** **0.5<V<1**

The HSV model consists of hue, saturation and value (1). The hue color coordinate makes it possible to determine the color of pixels. The hue does not depend on brightness because of the value coordinate – brightness level. The S-saturation is a very important value because of the traffic sign blue and red colors. These colors are very pure, therefore, the saturation value of the traffic sign red and blue colors is close to the maximum. The HSV model separates information about color hue, brightness level and saturation [11].

RGB to HSV conversion:

$$V = Max(R,G,B)$$

$$S = \left\{ \frac{255 * V - \min(R,G,B)}{V} \right.$$

$$H = \begin{cases} 60(G-B)/S, \ if V = R \\ 120 + 60(B-R)/S, \ if V = G \\ 240 + 60(R-G)/S, if V = B \end{cases} \qquad (1)$$

$$* \text{comment}$$

$$\text{if } V = 0 \Rightarrow S = 0$$

$$\text{if } H < 0 \Rightarrow H = H + 360$$

where

R-red, G-green, B-blue color coordinates of image input pixels (the color coordinates range from 0 to 255);

H	hue (range is from 0 to 360);
S	saturation (range is from 0 to 255);
V	value (range is from 0 to 255).

B. Image Binarization

After HSV conversion we can identify pixel colors. The important colors of traffic signs are blue and red. The blue and the red colors are marked by binary information. If color coordinates of image pixel are in the range of red and blue colors (Table II), then we can write ‘1’, else ‘0’. As a result we obtain a binary image (Fig. 2).

TABLE II

THE DIAPASONS OF COLORS

HSV	**Red**	**Blue**	**Yellow**	**Black**	**White**
H	from 324 to 360 and from 0 to 30	from 190 to 236	from 40 to 60	from 0 to 360	from 0 to 360
S	from 130 to 255	from 130 to 255	from 100 to 255	from 0 to 255	from 0 to 60
V	from 50 to 255	from 50 to 255	from 100 to 255	from 0 to 90	from 150 to 255

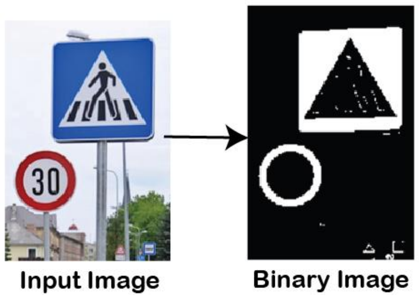


Fig. 2. Image binarization.

C. “Macro” Segmentation

We need the segmentation [12, Ch. 3] for image object separation. After red and blue color binarization, we have traffic sign borders. We can mark these borders separately. It means that each border has its own index. Therefore each traffic sign has one border and one index. After the segmentation we can analyse each traffic sign separately (Fig. 3).

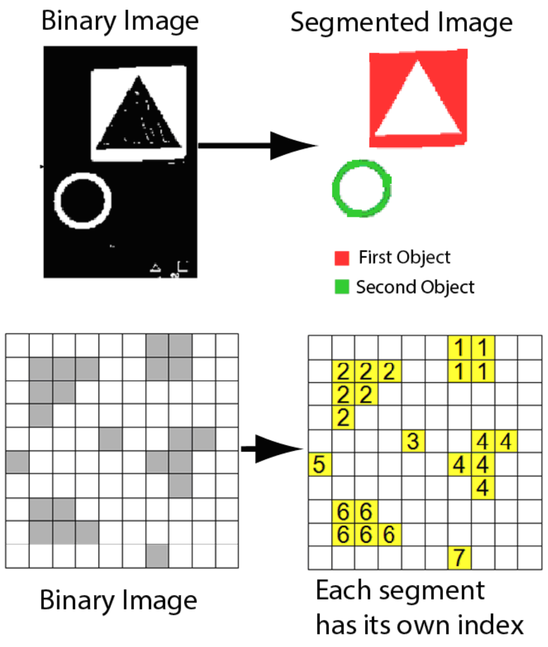


Fig. 3. Segmentation.

D. Edge Detection (Signature)

Traffic sign shape depends on the edge. After "macro" segmentation we can analyse each traffic sign edge separately. Before the shape recognition we must convert information about the edge to the favourite format. The edge format in Fig. 4 is the function of the corner theta ($r(\theta)$=Radius). The number of function peaks is dependent on the traffic sign shape. The radius is the destination between the centrode (2) (3) and the outside contour of the segment (traffic sign border) [8, Ch. 11].

Binary segment centrode calculation:

$$\bar{y} = \frac{\sum_{y_i=1}^{Hight}\sum_{x_i=1}^{Width}(y_i * I(x_i, y_i))}{\sum_{y_i=1}^{Hight}\sum_{x_i=1}^{Width}(I(x_i, y_i))} \quad (2)$$

$$\bar{x} = \frac{\sum_{y_i=1}^{Hight}\sum_{x_i=1}^{Width}(x_i * I(x_i, y_i))}{\sum_{y_i=1}^{Hight}\sum_{x_i=1}^{Width}(I(x_i, y_i))} \quad (3)$$

where

Hight and *Width* segment size;

$I(x,y)$ binary image function (pixel(x,y)-white, then '1', else 0);

$\bar{y}$ and $\bar{x}$ centrode coordinates.

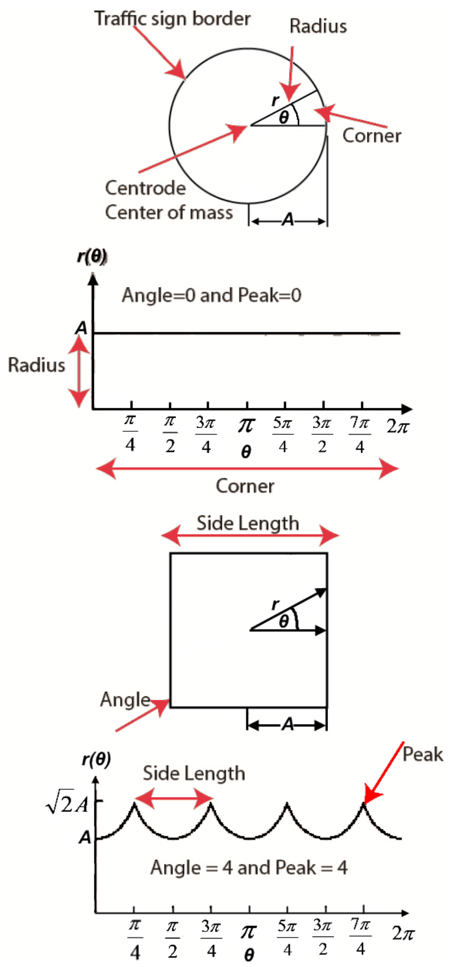


Fig. 4. Unwrapping of Shape.

E. Brightness Adaptation

If we want to make a small traffic sign segment analysis, we must make more precise determination of the color of pixels. When we use image binarization with static ranges (Table II), we have a lot of problems with black and white colors, because of different brightness. So, we must change color coordinate range according to the level of brightness. Therefore, at first we have to count region brightness level (4) (Fig. 3). Then, according to the information about the brightness level we can change the ranges of black and white. See infra brightness adaptation algorithms and results in Fig. 6.

1. Brightness level measurement

Pseudocode:

Where **REG_MASS** is array of average value of value and saturation and each element of array is separate region, **IMAGE_PIXEL** is the pixel of image, in HSV color model each pixel has 3 color coordinates: hue, saturation and value and 15 is the size of region.

```
REG_MASS_WIDTH= IMAGE.WIDTH div 15;
REG_MASS_ HEIGHT = IMAGE. HEIGHT div 15;
FOR X=0 TO IMAGE.WIDTH-1 DO
FOR Y=0 TO IMAGE.HEIGHT-1 DO
BEGIN
    REG_MASS[X div 15, Y div 15].VALUE
    = REG_MASS[X div 15, Y div    15]VALUE +
        IMAGE.PIXEL[X.Y]. VALUE;
END;
FOR X=0 TO REG_MASS_WIDTH-1 DO
FOR Y=0 TO REG_MASS_ HEIGHT-1 DO
BEGIN
    REG_MASS[X, Y].VALUE= REG_MASS[X, Y].VALUE/
        (15*15);
END;
```

Region brightness level measuring formula:

Where

$V(x, y)$	function of brightness (value from HSV);
x and y	image pixel coordinates;
$\bar{V}$	average brightness level of region;
n	region size.

$$\bar{V} = \frac{\sum_{X_i=1}^{n}\sum_{Y_i=1}^{n}V(X_i, Y_i)}{n^2} \quad (4)$$

Results:

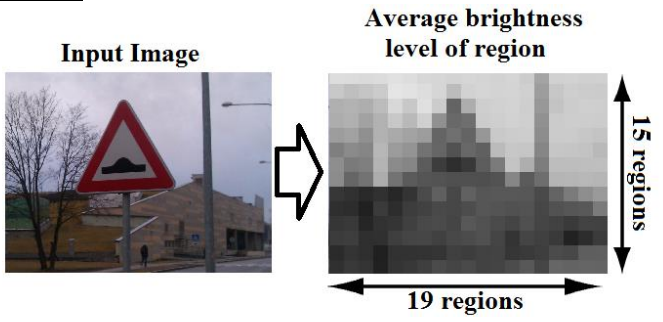


Fig. 5. Brightness adaptation.

2. Range adaptation.

When we have **REG_MASS**, we can change black and white ranges according to REG_MASS values.

Pseudocode:

Where **MAX_WHITE** and **MIN_WHITE** are limits of the white color coordinate range.

```
White color range adaptation:
MAX_WHITE.SATURATION=60;
MAX_WHITE.VALUE=255;
MIN_WHITE.SATURATION=0;
MIN_WHITE.VALUE= REG_MASS[X, Y].VALUE.

Black color range adaptation:
MAX_BLACK.VALUE= REG_MASS[X, Y].VALUE;
MAX_BLACK.SATURATION=255-
(MAX_BLACK.VALUE div 3);
MIN_ BLACK.SATURATION=0;
MIN_ BLACK.VALUE= 0.
```

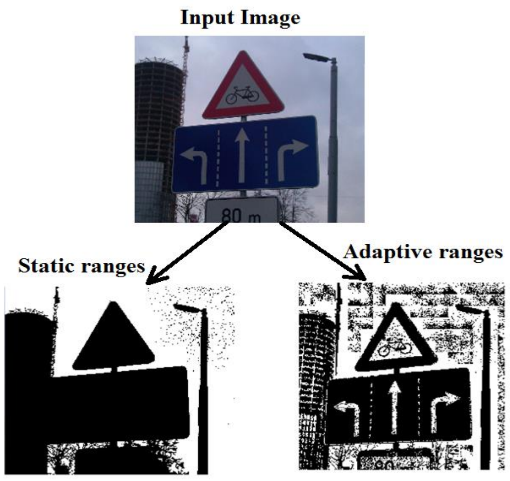


Fig. 6. Difference between the static range (Table II) and adaptive range.

F. Color (of pixel) Corrections

After brightness adaptation we have pixel color noise (Fig. 7), (Fig. 8). Therefore, we have to use color correction. We can make color correction only if we know that most of the pixels are determined correctly. Correction works more precisely, if we know additional information about the traffic sign (for example traffic sign shape).

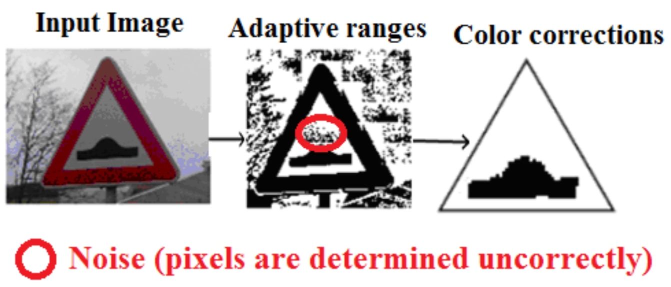


Fig. 7. Color correction results.

The most problematic colors are black and white. Therefore, black and white color correction is the most important. If we make black and white color correction, we have two centers: the center of black color and the center of white color (Fig. 8). The saturation and value diapasons are divided by 26*26 cells (Fig. 8). The cell which contains the highest number of black pixels is the center of black color and the cell which contains the highest number of white pixels is the centre of white color (Fig. 8).

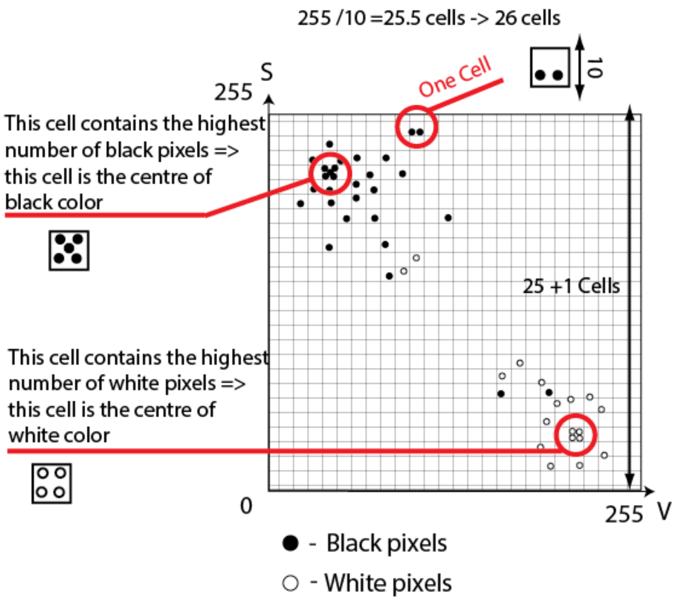

255 /10 =25.5 cells -> 26 cells
S
255
One Cell
10
This cell contains the highest number of black pixels => this cell is the centre of black color
25 +1 Cells
This cell contains the highest number of white pixels => this cell is the centre of white color
0
255
V
● - Black pixels
○ - White pixels

Fig. 8. The centers of black and white color.

After the centers are detected, we can make black and white color correction. If the distance (6) from the image pixel coordinates (*S* and *V*) to the center of white color is smaller than the distance (5) from the image pixel coordinates to the center of black color, then the pixel color is white, another pixel color is black (Fig. 9), (Fig. 10). As a result we do not have pixel color noise (Fig. 7), (Fig. 10).

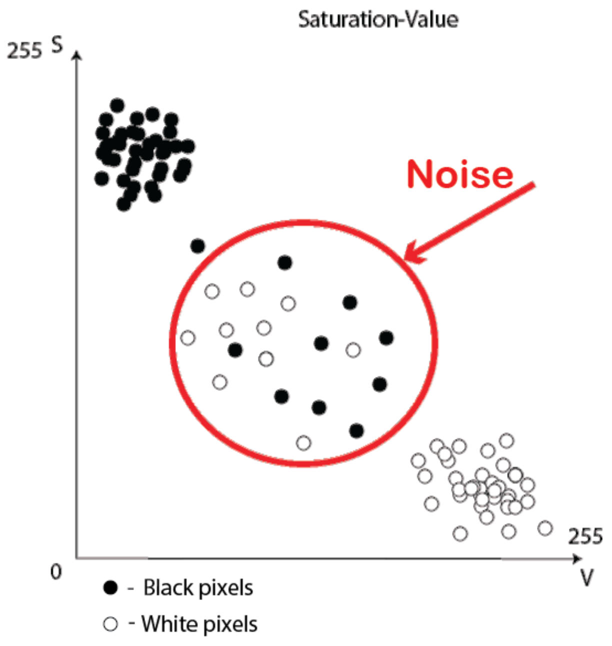


Fig. 9. Pixel noise.

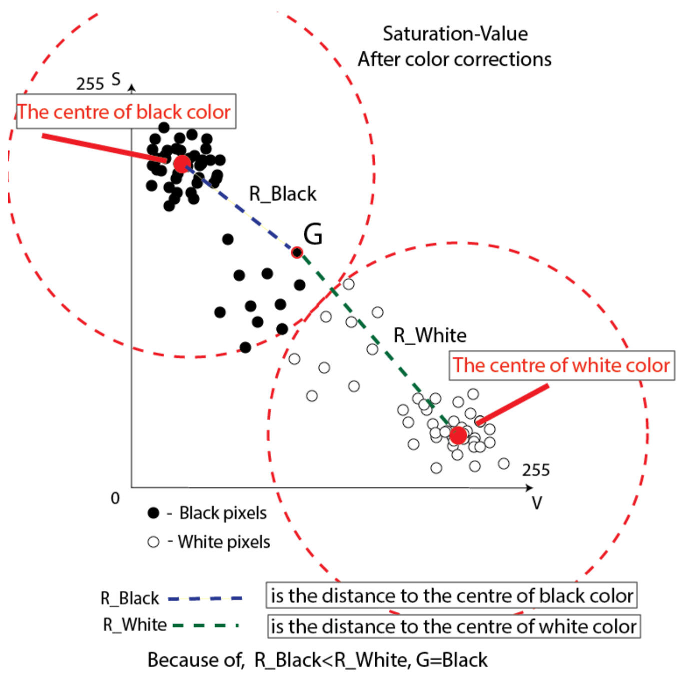


Fig. 10. Color correction algorithm.

The distance from the center of black and white color to the pixel of binary image (Fig.10):

$$R\_Black = \sqrt{\begin{matrix}(Black\_C.V - Pixel.V)^2 \\ +(Black\_C.S - Pixel.S)^2\end{matrix}} \qquad (5)$$

$$R\_White = \sqrt{\begin{matrix}(White\_C.V - Pixel.V)^2 \\ +(White\_C.S - Pixel.S)^2\end{matrix}} , \qquad (6)$$

where
R_Black distance from the centre of black color to the pixel of image;
R_White distance from the centre of white color to the pixel of image;
Pixel pixel from the image like point G in (Fig. 10);
Black_C and *White_C* centers of black and white color;
S and *V* pixel color coordinate (Saturation and Value).

G. *"Micro" Segmentation*

After color correction we can determinate the small traffic sign segments. "Micro" segmentation separates these small segments (Fig. 11). Each segment has its own index. "Micro" segmentation uses the same algorithms as "macro" segmentation, with a small difference: "micro" segmentation marks small elements.

After "micro" segmentation we can make color analysis of the segment and content analysis.

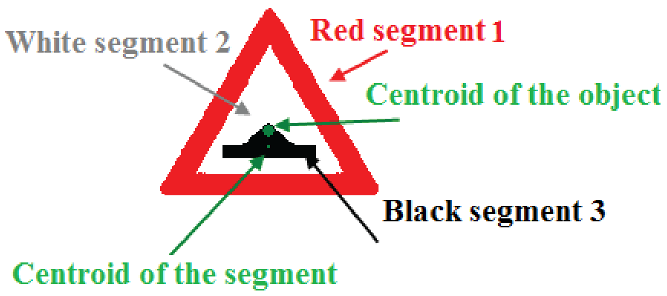


Fig. 11. Segments of traffic sign.

H. *Sign Image Normalization (Rotation)*

Usually the arrangement of a traffic sign is not quite straight. So, if we want to have a straight traffic sign, we must rotate it (Fig. 12). Traffic sign rotation is possible with rotation matrix (7), (8), [9]. Rotation matrix is dependent on θ – rotation angle. The Θ is calculated by formula (9).

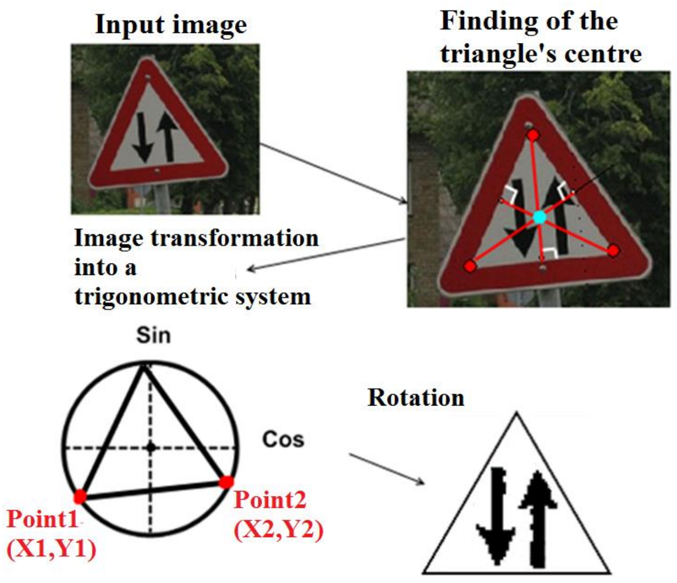

Input image
Finding of the triangle's centre
Image transformation into a trigonometric system
Sin
Cos
Point1 (X1,Y1)
Point2 (X2,Y2)
Rotation

Fig. 12. Traffic sign rotation.

Rotation Matrix:

$$RotM = \begin{bmatrix} Cos(\theta) & Sin(\theta) \\ -Sin(\theta) & Cos(\theta) \end{bmatrix} \qquad (7)$$

where θ denotes the rotation angle.

Rotation algorithm:

$$\begin{aligned}&[x, y] * \begin{bmatrix} Cos(\theta) & Sin(\theta) \\ -Sin(\theta) & Cos(\theta) \end{bmatrix} = \\ &= [x * Cos(\theta) + -y * Sin(\theta) \quad x * Sin(\theta) + y * Cos(\theta)] = \\ &= [xnew, ynew]\end{aligned} \qquad (8)$$

where *x* and *y* are the old coordinates of pixel and *xnew* and *ynew* are the new coordinates of pixel.

Θ calculation:

$$\theta = arctg\left(-\frac{Y1-Y2}{X1-X2}\right) \quad (9)$$

where *Y1, Y2,X1,X2* are coordinates of points (Fig. 12).

IV. Experimental Results

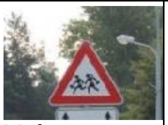
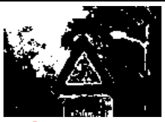
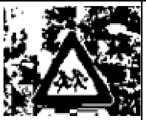
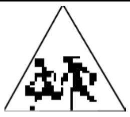
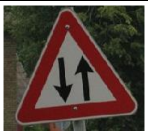
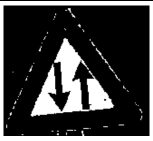
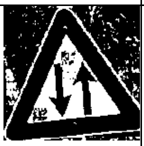
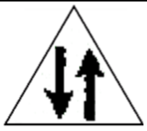
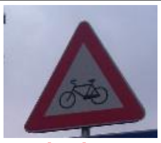
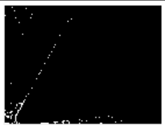
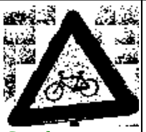
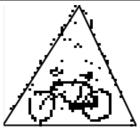
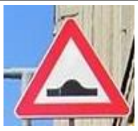
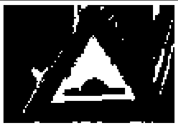
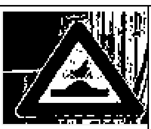
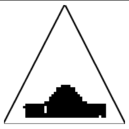
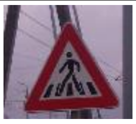
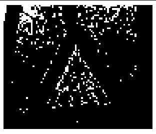
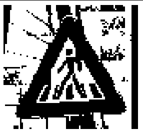
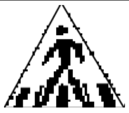
A. Experiment With Selection of Traffic Sign Content

The most difficult problem is the selection of the traffic sign content. This experiment shows the significance of methods of approach.

The experiment (Table III) consists of 4 algorithms: image binarization, brightness adaptation, color correction and sign image normalization. The experiment is evaluated by the following parameters:

Input image brightness level: high, medium and low;
Deviation angle: no angle (<3°), small angle (3° – 6.5°) and big angle (>6.5 °);
Content visibility: good, medium and bad;
Pixel noise: no noise, a bit of noise and a lot of noise;
Shape deformation: flattening, elongation and other.

TABLE III
Image Pre-processing for Traffic Sign Content

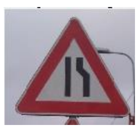
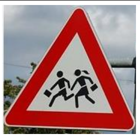
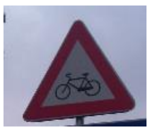
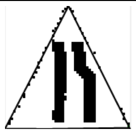
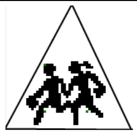
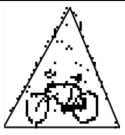
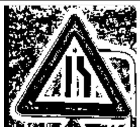
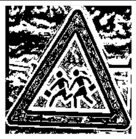
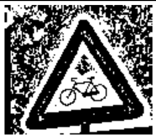
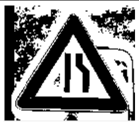
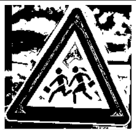
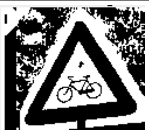
Input image	**Image binarization**	**Brightness adaptation**	**Colors corrections and sign image normalization**
 Medium brightness level	 Bad content visibility	 Good content visibility	 Good content visibility
 Medium brightness level. Big deviation angle.	 Good content visibility	 A bit of noise	 No noise. No deviation angle. Good content visibility.
 Low brightness level. Big deviation angle.	 Bad content visibility	 Good content visibility. A bit of noise.	 No noise. No deviation angle. Good content visibility.
 Flattening High brightness level	 Good content Visibility	 A lot of noise	 No noise No flattening
 Low brightness level. Small deviation angle.	 Bad content visibility	 Good content visibility	 No deviation angle

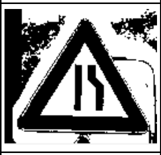
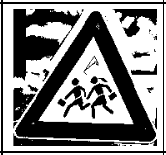
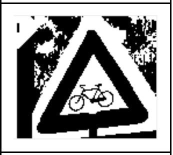
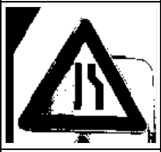
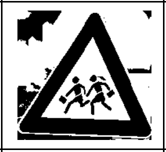
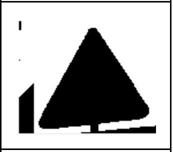
The suggested method takes away deviation angle, pixel noise, shape deformation and compensates low brightness level. As a result we will have good visibility of traffic sign content.

B. Comparison With Other Algorithms

This experiment compares the proposed method with the well–known method of adaptive binarization [10, L. 3)]. Table IV shows the difference between the results of the methods.

TABLE IV
Adaptive Binarization

Number of experiment		**1**	**2**	**3**
Input image				
The proposed method				
Time of the algorithm		< 1ms	< 1ms	< 1ms
The adaptive binarization	Radius = 5			
	Time of the algorithm	< 1ms	< 1ms	< 1ms
	Radius = 10			
	Time of the algorithm	15 ms	31 ms	15ms

Number of experiment		**1**	**2**	**3**
	Radius = 15			
	Time of the algorithm	31 ms	62 ms	16 ms
	Radius = 60			
	Time of the algorithm	313 ms	624 ms	266 ms

<u>*Comparison:*</u>

The suggested method

Advantages:

1) relatively fast method;
2) does not depend on the image size;
3) easy to use (no need to adjust).

Disadvantages:

1) consists of 3 parts.

The adaptive binarization.

Advantages:

1) easy for realization;
2) universal.

Disadvantages:

1) operates relatively slowly;
2) radius must be chosen;
3) depends on the size of the image.

V. Conclusion

The HSV conversion and image binarization allow to obtain important information from the image. “Macro” and “micro” segmentation allows to separate the image into many objects and to determine its color characteristics, form and content. The edge selection converts the information about the edge to the favourite format and it helps to analyze the traffic sign shape. Thanks to the brightness adaptation and color correction we can discern the small segments of the traffic sign, which is very important for color and content analysis.

Methods and algorithms proposed in this paper are quite effective (the selection of the traffic sign content takes less than 1ms). Experiments show that the suggested algorithm and methods are very useful. Thanks to the suggested approach, TSR system can work without multilayer neural networks and computer vision libraries.

The image pre-processing methods and algorithms allow to make TSR system adaptable for real-life environment and to convert input information (from camera) into a usable format for analyzing the information about a traffic sign.

References

[1] *BMW Automobiles* [online]. 2010. Available from: http://www.bmw.com/

[2] *Saab* [online]. 2010. Available from: http://www.saab.com/

[3] *VW Media Services* [online]. 2010. Available from: https://www.volkswagen-media-services.com/

[4] J. Hatzidimos, “Automatic Traffic Sign Recognition,” *Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics*, ICTAMI, 2004, Thessaloniki, Greece, pp. 174–184.

[5] H. Fleyeh, “Traffic and Road Sign Recognition,” July 2008, pp. 1–255.

[6] B. Hoferlin, K. Zimmermann, “Towards reliable traffic sign recognition”, 2009, p. 6.

[7] A. Lorsakul, J. Suthakorn, “Traffic Sign Recognition for Intelligent Vehicle/Driver Assistance System Using Neural Network on OpenCV,” *The 4th International Conference on Ubiquitous Robots and Ambient Intelligence*, 2007.

[8] R. C. Gonzalez, R. E. Woods, “Digital Image Processing 2ND EDITION,” 2002.

[9] D. F. Rogers, J. A. Adams, “Matematical elememnts for computer graphics,” 2001.

[10] A. Konushin, *Introduction to computer vision* [online]. 2012, Course pages. Available from: http://courses.graphicon.ru/main/vision

[11] N. A. Ibraheem, M. M. Hasan, R. Z. Khan, P. K. Mishra, “Understanding Color Models: A Review,” 2011–2012.

[12] L. G. Shapiro, G. C. Stockman, “Computer Vision,” 2006.

[13] D. A. Forsyth, J. Ponce, “Computer Vision a modern approach,” 2004.

Artjoms Suponenkovs was born in 1992. He received the degree of *B. sc. ing.* in 2014 from Riga Technical University. Presently he is Master student at the Institute of Computer Control, Automation and Computer Engineering, Riga Technical University.

E-mail: artjoke@inbox.lv

Aleksandrs Glazs was born in Riga, Latvia, April 7, 1939. He is a Professor with the Faculty of Computer Science and Information Technology, Deputy-Director of Institute of Computer Control, Automation and Computer Engineering and Head of Image Processing and Computer Graphics Division, Riga Technical University.

He received the Degree of Candidate of Technical Sciences from Riga Polytechnical Institute in 1971 and the Degree of Doctor of Technical Sciences (*Dr. habil. sc. ing.*) from the Russian Academy of Science in Moscow in 1992. He is the author of more than 100 scientific publications in different areas: pattern recognition, images processing, computer vision and computer graphics.

A. Glazs is a full member of the Baltic Informatization Academy.

Address: Meza Street 1, Riga LV-1048, Latvia.

E-mail: glaz@egle.cs.rtu.lv

Artjoms Suponenkovs, Aleksandrs Glazs. Attēla priekšapstrādes metodes ceļazīmju atpazīšanas procesā

Mūsdienās, kad automašīnu skaits strauji pieaug, rodas vairākas nopietnas problēmas. Viena no tām ir ceļu satiksmes drošība. Lai palielinātu ceļu satiksmes drošību, daudzas automašīnu ražošanas kompānijas (*BMW, Volkswagen, Saab, Google Car, Opel*) uzstāda pēdējām automašīnām (piem., *Opel Insignia*) ceļazīmju atpazīšanas sistēmas (Siemens VDO TSR, FOSTS, OpelEye….). Ceļazīmju atpazīšanas sistēmas paaugstina vadītāja uzmanību un atgādina par dažādiem ierobežojumiem. Attēla atpazīšanas uzdevums ir ļoti sarežģīts, jo dažādi faktori − apgaismojums, diennakts laiks, migla, rudens, kameras (kura saņem videoinformāciju) kvalitāte, kameras izvietojums, kameras pārvietojuma ātrums utt. ietekmē attēlu reālajā dzīvē. Rakstā tiek piedāvātas attēla priekšapstrādes metodes un algoritmi, ar kuru palīdzību ir iespējams veikt ceļazīmju atpazīšanas datorsistēmas adaptāciju reālajiem apstākļiem un vissvarīgākās ceļazīmju informācijas nodalīšanu. Piedāvātās metodes un algoritmi ļauj nodrošināt: pētāmā objekta krāsas nodalīšanu, pētāmā objekta kontūra nodalīšanu, pētāmā objekta krāsas uzlabošanu, pētāma objekta rotāciju, pētāma objekta satura nodalīšanu un vienkāršo ceļazīmju analīzes procesu. Ceļazīmju analizēšanas vienkāršošana dod iespēju neizmantot sarežģītus analizēšanas rīkus, tādus kā daudzslāņu neironu tīkli. Parasti, lai izveidotu ceļazīmes atpazīšanas datorsistēmu, tiek izmantotas ārējās bibliotēkas (piem.: OpenCV) un daudzslāņu neironu tīkli. Šai sistēmai ir vajadzīgi lieli resursi un liela veiktspēja. Arī universālie algoritmi un metodes no ārējām bibliotēkām, kurus var izmantot ar dažādiem objektiem (sejām, cipariem utt..), parasti nedod tik labus rezultātus kā specializētie algoritmi un metodes. Tāpēc šajā rakstā tiek piedāvāti specializēti algoritmi un metodes, kuri ļauj efektīvi veikt attēla priekšapstrādi ceļazīmju atpazīšanas procesā.

Артём Супоненков, Александр Глаз. Предварительная обработка изображений для распознавания дорожных знаков.

В данной статье рассмотрены проблемы предварительной обработки изображения при распознавании дорожных знаков. Описаны различные методы и алгоритмы, которые позволяют обеспечить адаптивность системы распознавания дорожных знаков к реальным условиям и отобразить получаемую информацию в удобном для дальнейшего анализа формате. На основе полученных результатов проведено сравнение с широко известными алгоритмами предварительной обработки изображения.

Проблема разработки автомобильных систем распознавания дорожных знаков является достаточно актуальной, причём эта актуальность всё время повышается в связи с стремительным ростом количества новых автомобилей и, соответственно, дорожно-транспортных происшествий. Многие известные фирмы, например: BMW, Volkswagen, Saab, Google Car - используют в новых моделях автомобилей системы распознавания дорожных знаков, которые совместно с другими системами, позволяют повысить бдительность водителей или заменить их некоторые функции функциями, которые реализуются системой распознавания дорожных знаков (интеллектуальной системой). Такие модели автомобилей оснащаются видеокамерами, с которых считывается видеоинформация для последующего анализа изображений и распознавания дорожных знаков. Такая система распознавания дорожных знаков работает в реальных условиях, поскольку на информацию, поступающую с камер, влияют различные факторы: погодные условия, освещённость и д.р., которые затрудняют процесс распознавания. Поэтому необходима предварительная обработка таких изображений, которая фильтрует получаемую информацию и отображает её в удобном для дальнейшего анализа формате. Большинство известных подходов к разработке таких систем используют готовые библиотеки (например, OpenCV) и многослойные нейронные сети. Однако такие системы требуют больших ресурсов и высокой производительности, а стандартные, универсальные алгоритмы, поставляемые внешними библиотеками, которые можно использовать при распознавание лиц, цифр, текста и т.д., обычно не позволяют распознавать дорожные знаки достаточно эффективно. Поэтому данная работа посвящена созданию эффективных алгоритмов и методов предварительной обработки изображения при распознавании дорожных знаков.