Traffic Sign Detection on GPU using Color Shape Regular Expressions

Artem Nikonorov[1], Maksim Petrov[2] and Pavel Yakimov[2]

[1]Image Processing Systems Institute of Russian Academy of Science,
Molodogvardeyskaya st. 151, 443001, Samara, Russia
[2]Samara State Aerospace University,
Moskovskoe shosse, 34, 443086 Samara, Russia

Abstract. Regular expression matching on GPU is state-of-the-art technique for the processing of a huge amount of the text information, for example, in network activity analyzers and intrusion prevention systems. This paper proposes a fast and robust algorithm for traffic sign detection, which is based on color shape regular expressions matching through the image pixels. We consider a fast massively-parallel GPU implementation of the color shape regular expression, using deterministic finite automaton and special case of non-deterministic finite automaton. The performance of the proposed localization algorithm is compared with the Hough transform, which is commonly used for traffic sign recognition.

1 Introduction

This paper is devoted to the problem of traffic sign recognition, which becomes more and more acute. The number of cars in the world is constantly increasing as well as the number of road accidents. Therefore, closer attention is paid to the road system of intelligent information processing and decision making. Engineers around the world have developed a number of active safety systems for cars, such as ABS (anti-lock braking system), EBD (electronic brake force distribution), ESP (electronic stability control), and many others. Among such systems, traffic sign recognition system (TSR) is considered to be one of the most advanced.

The traffic sign recognition system is designed to provide the driver with actual information about the traffic conditions. There are several examples of such system: Opel Eye', 'Speed Limit Assist' by Mercedes-Benz, 'Traffic Sign Recognition' by Ford, etc. Most of them aim to detect and recognize only speed limit traffic signs, these restrictions apply to the algorithms of traffic sign detection because of several reasons [1].

The task of traffic signs recognition is computationally complex, and the performance of recent mobile computers is not always enough to provide real-time traffic signs recognition. Moreover, most of such recognition techniques are based on Hough transform, which allows detecting parameterized curves in the image. This method is

Nikonorov A., Petrov M. and Yakimov P..
Traffic Sign Detection on GPU using Color Shape Regular Expressions.
DOI: 10.5220/0004393900630072
In *Proceedings of the 4th International Workshop on Image Mining. Theory and Applications* (IMTA-4-2013), pages 63-72
ISBN: 978-989-8565-50-1

suitable when searching for circular traffic signs, but the detection of triangular traffic signs using Hough transform causes a certain problem.

The traffic sign recognition usually consists of two main steps: the detection of signs and then the recognition of the identified object. There are a lot of different recognition techniques, and it is actually not problematic to recognize the detected object of a small size, having samples or templates of possible traffic signs. In this paper, we will introduce our new method of detection traffic signs, which is based on the use of regular expressions.

2 Traffic Sign Detection

There are several methods to detect a traffic sign in the image, and one the most common is the technique that uses the sign color information. The color feature makes it possible to distinguish the sign from other surrounding road conditions. The first step of traffic sign recognition can be implemented on the basis of color information [6] It is necessary to mention that the first papers about the recognition of traffic signs appeared in the 1960s; the first working software solutions emerged in the 1990s [7].

In many studies, some restrictions on the? RGB component of images are used to detect the signs in the background of some road conditions, similar to those in [6]:

$$\begin{aligned} r / g &> \alpha_{red} \\ r / b &> \beta_{red} \\ g / b &> \gamma_{red} \end{aligned} \quad . \tag{1}$$

In particular, the color filtering in the RGB space is used in [8] to detect speed limit signs. In some works, HSI space or HSB are considered more suitable for recognition tasks [6], [9].

The approach to the detection of signs that is based on the color filtering does not require a lot of computing resources. This explains the popularity of this approach in the early 1990s until 2006 [8]. However, the accuracy of this approach is insufficient / too low. A good example of one of the three signs which is "not red enough" in terms of using color filtering, is shown in Figure 1.

Fig. 1. Error detection in sign detection based on color information.

The current performance of embedded computing systems allows using more sophisticated algorithms of detecting signs in real-time. In addition to color information, all of these systems detect the signs, using the information about the objects' shape [10].

One of the most common methods that use the information about shape is Hough Transform [11]. The main idea of Hough transform detection method is using a special Hough space, which is actually an accumulator image with voices for the sought-for objects in each pixelHaving transformed the original image, the Hough space image contains one or several brightest points with the largest amount of voices for the objects coordinates [12]. For example, in paper [13], a Hough-like transform is used to detect the U.S. speed limit signs. The main advantage of this method is robustness and resistance to different distortions like noise. However, this method is hard to implement in real time when processing full HD frames from a video sequence. Thus, we propose a new effective and fast method of traffic sign recognition based on color filtering.

3 Detecting Signs using Color Shape Regular Expressions

The proposed in this paper approach lies in the fact that the procedure of color filtering and pattern matching is modified: we use regular expressions like a scheme for colors of image lines pixels in image line processing. We will call this approach *color shape regular expressions*. These regular expressions are implemented using the *Deterministic Finite Automaton* (DFA) or Non-*Deterministic Finite Automaton (NFA)* [2]. To describe it, let us consider the detection of the class of red traffic signs. Further, according to the properties of DFA, the approach can be generalized to an arbitrary set of signs.

The steps of our algorithm are as follows.

1. The coordinates of 'white point' are identified in the analyzed frame retrieved from a camera.
2. A saturation analysis is performed according to (1), to make a decision about the point color. The threshold values are determined based on the coordinates of the white point. The threshold is performed in accordance with the threshold values (1).
3. The colors are divided into three main classes: red (class R), white (class W), and class C with all the colors which are not represented in R and W.
4. Passing through the image pixels line by line, each area is compared to the possible line of the sought-for sign.
5. The following comparison is made. For a set of signs, it is possible to select the most distinctive line with certain sequences of pixels with colors from the mentioned classes. So, for many of the prohibition signs and the signs of priority (for example, 'passing is prohibited', 'give way', 'parking is prohibited'), the typical sequence of colors is coded as:

$$R\{b_0,b_1\}[WRG]\{w_0,w_1\}R\{b_0,b_1\} \quad (2)$$

The sequence (2) is treated as a regular expression written for the alphabet of symbols for color classes red, white and grey - R, W, G. Here the constants $\{b_0, b_1\}$

denote a valid range limits of the width of the red sign border line,; $\{w_0, w_1\}$ specify the range of the width of the inner area of the sign.

Formally, the regular expressions, dealing with image lines, can be defined in a similar way as the regular expressions that processes the text strings [3]. Table 1 shows the main elements of color shape regular expressions introduced by analogy with the text regular expressions. A symbol means the color of a pixel. The positive result of the processing of some segment of the image line with the help of color shape regular expression means a positive localization decision.

Table 1. Color shape regular expressions elements.

Name	Regular expression	Destination
Epsilon	E	{""}
Character	A	For some character α, i.e. the color of pixel form class α.
Concatenation	RS	Denoting the set {αβ|α in R and β in S} e.g., {”ab”}{”d”, ”ef”} = {”abd”, ”abef”}
Alternation	R|S	Denoting the set union of R and S. e.g., {”ab”}|{”ab”,”d”,”ef”} = {”ab”,”d”,”ef”}.
Kleene star	A*	Denoting the smallest super-set of R that contains ε and is closed under string concatenation. This is the set of all strings that can be made by concatenating zero or more strings in R, e.g., {”ab”, ”c”}* = {ϱ,”ab”,”c”,”abab”,”abc”,”cab”,”ababab”, ...}. Where R is a set of symbols..
Kleene cross		This is the set of all strings that can be made by concatenating one or more strings in R

Regular expressions can be implemented in the form of DFA and NFA. The DFA implementation complexity is *O(N)*, where *N* is a string size [2].

DFA implementing expression (2) is shown in Fig. 2, symbols R, W, and G means colors which is shown in (2). In this figure we use the following notation for the DFA states. State 0 is a state before the object, state 1 is the object beginning, N means detection failure, Y means positive detection. States Y and N are final states of DFA. States from A to H is calculated from expression (2) by the following rules:

$A = b_0$, $B = b_1$, $C = b_1 + 1$, $D = b_1 + w_0$,

$E = b_1 + w_1$, $F = b_1 + w_1 + 1$, $G = b_1 + w_1 + b_0$, $H = b_1 + w_1 + b_1$.

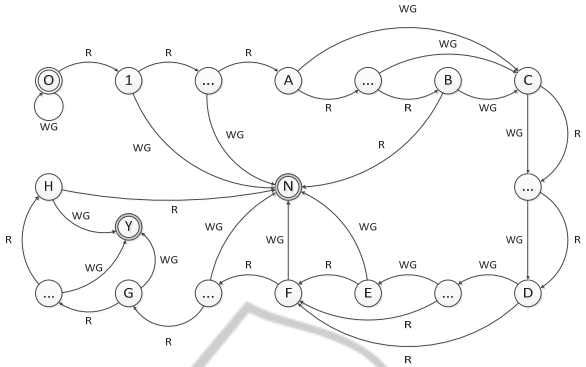

Fig. 2. Deterministic finite automation implementing a certain expression.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G	G	R	R	R	R	R	G	G	G	R	R	R	R	R

Fig. 3. An example of Image line for processing using DFA.

The disadvantage of such definition is the fact that the color shape regular expression matching, implemented as a DFA, can lead to missing objects in the image. Suppose the image line in Figure 3 begins with the desired object pixel 11, but expression (2) will begin processing with pixel 3. When the value $b_0 < 5$, the true position of the object will be treated by pattern (2) as an inside part of the object (this part is shown as states from C to H in Fig. 2. Thus, the true position of the object will be misdetected.

Non-deterministic finite automaton (NFA) implementation, which allows multiple switches between the states, is devoid of this disadvantage and does not fail to localize the object. NFA regular expressions are implemented in (flex, grep, pcre) and in iNFAnt for GPU processors [4].

We can show that in the case of color shape regular expressions processing with NFA, we can obtain the following modification of the scheme from Figure 2. Starting from state C to state H, any red pixel is regarded as an opportunity to branch to a new DFA of the same structure. This simplified scheme can keep the complexity of the performance of each branch of the algorithm *O(N)*, but nevertheless it realizing NFA. In more detail, the features of such scheme implementation for the GPU are described in the next section.

Finally, regular expression (2) could be generalized for other classes of traffic signs, using the alternation operation from Table 1. For example, we can add blue sign support to expression (2):

$$R\{b_0,b_1\}[WRCB]\ \{w_0,w_1\}R\{b_0,b_1\}|\ B\{b_0,b_1\}[WRCB]\{w_0,w_1\}B\{b_0,b_1\} \quad (3)$$

where B is a symbol for the blue color class, other symbols are the same as in expression (2). To use this regular expression, we should include blue pixels detection in the color segmentation step. After a positive localization result of the expressions for some segment α of the image line, the same localization will be process in the orthogonal direction to the segment. If the search in the orthogonal direction gives a

positive result for orthogonal segment β, then the rectangle made by segments α and β is selected as boundary of the required object. Finally, we denote a localization rule for the image segments that build boundary rectangle of the object:

$$R(\alpha_i) \wedge R(\beta_i) = true, \quad \alpha_i \perp \beta_i, \quad \alpha_i \wedge \beta_i \neq \varnothing \tag{4}$$

where α_i and β_i are intersected segments of the image line and the column, and R is some color shape regular expression (2) or (3). In general, the regular expression in (4) may differ for the rows and columns of the image, but in case of the traffic sign detection they are the same. Consequently, the boundary rectangle S for the object is defined by the intersection of the rectangles $S_i(\alpha_i, \beta_i)$, which are based on the crossing of segments α_i and β_i.

$$S = \bigcap_i S_i(\alpha_i, \beta_i) \tag{5}$$

The composition of rules (2) – (5) provides the overall localization rule. Such composition that build the boundary rectangle of the object is based on the principle of enhancement of weak classifiers (boosting) which are successfully used in AdaBoost [5] and Viola Jones detectors [14].

4 GPU Implementation based on Finite Automaton

Parallel processing is very important for such embedded systems as driver assistant systems. In this section, we will describe massively parallel implementation of color shape regular expressions for GPU. Then, we will compare the proposed method with well-known Hough transformation.

In commonly used approaches to pattern or regular expression, matching parallelization is made using data decomposition. It means that each parallel thread processes an independent portion of data. For regular expression matching in network traffic, each thread processes its own packet; for template matching in image analysis, each thread processes one row of the image.

We will discuss three approaches of GPU implementation for color shape regular expressions. Firstly, we will consider a naïve "plain" implementation of pattern matching, using expression (2) as a template. Secondly, we will study DFA implementation of the color shape regular expressions, and finally, we will consider the implementation based on NFA of a special kind. In all of these approaches, image lines are processed independently.

The plain implementation uses conditions, loops and accumulator variables to implement regular expression search like usual pattern matching. The code in this case contains many conditions and branching. It works for CPU processing, but as GPU architecture is close to SIMD [16], branching is too expensive for GPU.

On a modern GPU, a number of threads (usually 8 or 16) execute one instruction at a time. This number of threads is called *warp* [16]. If only one thread of the warp satisfies a certain condition, other threads must wait for this thread. This fact leads to more than 8 or 16 times decrease in overall computation speed.

There are many branching conditions for the naïve implementation of color regular expressions. For example, if one thread of the warp finds pixels matching to the object's part, and the other threads do not, then they must wait. This feature strongly limits GPU parallelization speedup of naïve implementation of color shape regular expressions. In our realization, the GPU/CPU speedup was 3.3 as shown in Table 2.

In traffic analysis systems and IPS [15], the speedup of GPU implementation of regular expression matching is more than 15. In these systems, DFA is usually used, and state-of-the-art systems use NFA [17].

DFA implementation for the color shape regular expression is quite similar to DFA implementation of the text regular expression. The listing 1 shows this DFA code. In this listing array `statesTable` implements states of DFA, function `storeObject()` is called when object detected. NFA implementing by last string of the for-loop when function `startNewBranch()` is called.

Listing 1. NFA implementation of the color shape regular expressions.

```
for(i = 0, i<N, i++){
  symbol = symbolFromImageLine(img, i);
  state = statesTable[state][symbol];
  if(state == 1) objectStart = i;
  if(state == STATE_N) objectStart = 0;
  if(state == STATE_Y) storeObject(objectStart, i);
  if(state >= STATE_C & symbol == RED)
    startNewBranch(i);
}
```


In our case, we need only simple special case of NFA. The main DFA processes the line until state C. It is the state when the first border of the object is passed. Starting from this state, every red pixel is treated as a possible starting point for another object location. So, every state from C is threated as two way decision in terms of NFA [2]. In each state with two way decision we must start new DFA processing when R symbol come. The listing in Fig. 5 shows the principal code of this special case of the NFA. Ideally in each "backward point" we need a new thread to process, really we reserve fixed size tread pool for processing the line, using the described NFA. The size of the pool is a parameter for experimental estimation.

5 Results

We compared the proposed method with Hough transform that is commonly used for traffic signs detection. We use standard GPU implementation of Hough transform from OpenCV library [18]. This realization has one disadvantage: it detects only circle shaped signs, but with high accuracy. Our regular expression matching algorithm detects all the signs with square, circle and triangle shapes. Unfortunately, the algorithm has a possibility of false positive detection. Such kind of error should be filtered later during the sign recognition. Consequently, it does not seem possible to make a precise comparison of the accuracy of Hough detector and color shape regular expressions. As shown in many papers the accuracy of Hough transform is 90-95%

for traffic sign detection. Accuracy of our detection algorithm is quite less, but all the errors are false positive detection which can be eliminated in recognition stage.

There are comparisons results of the performance of the described algorithms are shown in Table we show. We used Nvidia GeForce 560 as GPU, and Intel i5 2430 as CPU. It is shown in Table DFA implementation has no effect for performance of CPU realization but significantly increasing GPU performance. Implementation of NFA is slower than DFA but it reduces false positive detection errors by 31% in comparison with DFA. As shown in table color shape regular expressions are significantly faster than Hough transform for traffic sign detection.

Table 2. Performance comparison between Hough transform and color shape regular expressions.

	CPU, ms	GPU, ms
Hough transform	90	30
Color shape regular expressions, naïve implementation	15	4.5
Color shape regular expressions, DFA	16	2.5
Color shape regular expressions, NFA	20	3

Finally, examples of traffic signs detection and recognition by Hough transform and by color shape regular expression is shown in Fig. 4.

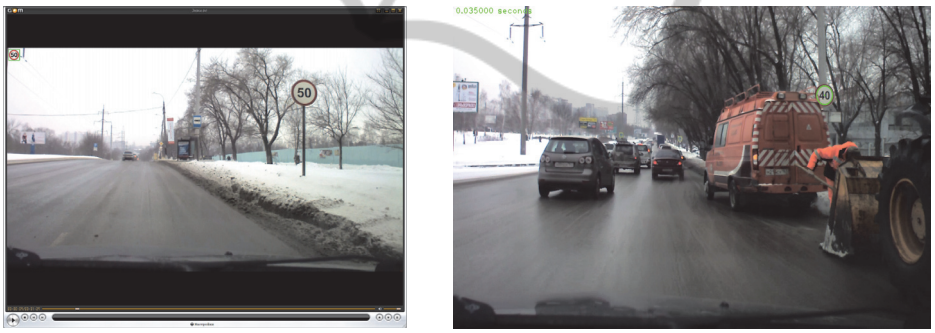

Fig. 4. Detection and recognition of the traffic sign.

6 Conclusions

Regular expression matching on GPU is state-of-the-art technique for the processing of a huge amount of the text information, for example, in network activity analyzers and intrusion prevention systems. This paper proposes a fast and robust algorithm for traffic sign detection, which is based on color shape regular expressions matching through the image pixels. The special case of non-deterministic finite automaton used for GPU implementation allows reaching of more than 15 times speedup in comparison with CPU version.

Unfortunately, the proposed algorithm has a possibility of false positive detection. Such kind of error should be filtered later during the sign recognition. Consequently,

it does not seem possible to make a precise comparison of the accuracy of Hough detector and color shape regular expressions. As shown in many papers the accuracy of Hough transform is 90-95% for traffic sign detection. Accuracy of our detection algorithm is quite less, but all the errors are false positive detections which can be eliminated in recognition stage. Detailed accuracy comparison of Hough transform and color shape regular expressions is a challenge for further research.

Acknowledgements

Authors are grateful to Professor Vladimir Fursov for many helpful discussions and the constructive criticism concerning the evaluations. This work was partially supported by Russian Foundation of Basic Research (Project No. 11-07-12051-ofi-m, 12-07-00581-a, 12-07-31208).

References

1. Shneier, M.: Road sign detection and recognition. Proc. IEEE Computer Society Int. Conf. on Computer Vision and Pattern Recognition. (2005) 215 – 222.
2. Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. (2001). Introduction to Automata Theory, Languages, and Computation (2 ed.). Addison Wesley. ISBN 0-201-44124-1. 551 p.
3. Aho, Alfred V. Algorithms for finding patterns in strings, In van Leeuwen, Jan. Handbook of Theoretical Computer Science, volume A: Algorithms and Complexity. The MIT Press. pp. 255–300, 1990.
4. Cascarano N., Rolando P., Risso F., R. Sisto. 2010. iNFAnt: NFA pattern matching on GPGPU devices. SIGCOMM Comput. Commun. Rev. 40, 5, (2010). 20-26.
5. Freund Y., Schapire R.E. A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. 1995,34pp.
6. Shneier, M.: Road sign detection and recognition. Proc. IEEE Computer Society Int. Conf. on Computer Vision and Pattern Recognition. (2005) 215 – 222.
7. Ruta A., Y. Li, X. Liu: Towards Real-Time Traffic Sign Recognition by Class-Specific Discriminative Features. British Machine Vision Conf. (2007).
8. Torresen, J. A., W. Bakke, Y. Yang: Camera Based Speed Limit Sign Recognition System. 13th ITS World Congress and Exhibition. (2006) 115 – 129.
9. Ren, F., J. Huang, R. Jiang, R. Klette: General traffic sign recognition by feature matching. IEEE 24th Int. Conf. Image and Vision Computing. (2009) 409 – 414.
10. Oh, J.-T., H.-W. Kwak, Y.-H. Sohn, W.-H. Kim: Segmentation and Recognition of Traffic Signs Using Shape Information. Lecture Notes in Computer Science, Springer Berlin/Heidelberg. Vol. 3804. (2005) 519 – 526.
11. Ballard, D.: Computer vision. Englewood Cliffs, N.J., USA. (1982).
12. Hardzeyeu, V., Klefenz, F.: On using the hough transform for driving assistance applications. 4th International Conference on Intelligent Computer Communication and Processing, (2008) 91-98.
13. Christoph Gustav Keller, Christoph Sprunk, Claus Bahlmann, Jan Giebeland, Gregory Baratoff: Real-time Recognition of U.S. Speed Signs. Intelligent Vehicles Symposium, IEEE. (2008) 518-523.

14 Viola P., Jones M., Robust Real-time Object Detection, Second International Workshop On Statistical And Computational Theories Of Vision – Modeling, Learning, Computing, And Sampling, (2001) 25.
15 Vasiliadis G., Polychronakis M., Ioannidis S., Parallelization and Characterization of Pattern Matching using GPUs, IISWC'2011. (2011).
16 NVIDIA CUDA C Best Practices Guide, Santa Clara, (2010) 75p.
17 Zu Y., Yang M., Xu Z., Wang L., Tian X., Peng K., Dong Q., GPU-based NFA implementation for memory efficient high speed regular expression matching, Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming, (2012) 129-140.
18 http://opencv.willowgarage.com/wiki/