Real-Time GPU Based Road Sign Detection and Classification

Roberto Ugolotti, Youssef S.G. Nashed, and Stefano Cagnoni

Department of Information Engineering, University of Parma,
Viale G.P. Usberti 181/A, 43124 Parma, Italy

Abstract. This paper presents a system for detecting and classifying road signs from video sequences in real time. A model-based approach is used in which a prototype of the sign to be detected is transformed and matched to the image using evolutionary techniques. Then, the sign detected in the previous phase is classified by a neural network. Our system makes extensive use of the parallel computing capabilities offered by modern graphics cards and the CUDA architecture for both detection and classification. We compare detection results achieved by GPU-based parallel versions of Differential Evolution and Particle Swarm Optimization, and classification results obtained by Learning Vector Quantization and Multi-layer Perceptron. The method was tested over two real sequences taken from a camera mounted on-board a car and was able to correctly detect and classify around 70% of the signs at 17.5 fps, a similar result in shorter time, compared to the best results obtained on the same sequences so far.

Keywords: Road Sign Classification, Differential Evolution, Particle Swarm Optimization, Learning Vector Quantization, Neural Networks, GPGPU.

1 Introduction

Automatic road sign detection and classification is a task that can help drivers and increase road safety. For this reason, this problem has been frequently tackled [2], up to systems mounted on recent car models, with limited functionalities.

Solutions to this problem usually include two different stages: the presence of a sign is first detected in the image, then it is classified to precisely recognize its meaning and possibly activate some driving system control. In the detection phase, the features used most frequently to recognize a sign are shape and color. Detection based on RGB color is usually fast but performance degrades when dealing with illumination changes [17] or other artifacts related to image acquisition. These problems can be reduced by performing conversion to other color-spaces like HSV/HSI [13]. Shape-based detection is generally more robust against these problems, besides allowing one to work with gray-scale images. However, it has to struggle against occlusions, different viewing angles or the presence of other artificial objects like commercial signs or buildings. For these reasons, a combination of color and shape information is usually preferred [5,15].

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 153–162, 2012.

The term "road sign classification" is not used consistently in the literature: in fact, some authors reduce the classification problem to what, in this paper, we call detection [15], i.e. distinguishing between a sign and a different object in the scene. In other papers [16], the classification problem takes into account only the classification between different categories of signs (e.g. prohibitory versus warning signs). The classification task considered here is concerned with the distinction of signs within the same category (prohibitory, warning, mandatory), as the category is implicitly determined in the detection phase.

The techniques most frequently used for classification are artificial neural networks [10,13] and support vector machines [9].

In the detection phase of our method a model representing a category of signs is rigidly transformed and then reprojected onto the image using an Inverse Perspective transform: this model-based approach shows good results against several problems that can affect the images, like partial occlusions or color unbalancing. After doing so, a fitness function is calculated to assess the degree of matching between the reprojected model and the image, turning detection into an optimization problem, in which a sign is considered to have been detected when the similarity is above a pre-defined value. To perform such an optimization we considered and compared Differential Evolution (DE) [18] and Particle Swarm Optimization (PSO) [7]. For the classification stage, we used Multi-layer Perceptrons (MLP) [4] and learning vector quantization (LVQ) [8] neural networks. In both phases we relied on GPUs to increase processing speed and to reach real-time performance, implementing our methods in CUDA-C [14], a C language extension for developing parallel routines (*kernels*) that run on GPU.

2 Sign Recognition System

A first implementation of the system, limited to the detection stage of three categories of signs (priority, prohibitory and warning), has been first presented by Mussi et al in [11]. The work described in this paper completed the system, adding the detection of mandatory signs and the classification stage. In addition, a DE-based detection stage has been developed and compared to the one based on PSO.

2.1 Sign Detection

The sign detection stage is based on a generally-applicable object detection algorithm that includes the following steps:

1. Consider some sets of key points, of known coordinates with respect to a reference position, and representative of the shape and colors of specific regions of the object to detect.
2. Translate and rotate the sets to a hypothesized position visible by the camera and project them onto the image.
3. Verify that the color histograms of the sets match those of their projection on the image to assess the presence of the object being sought.

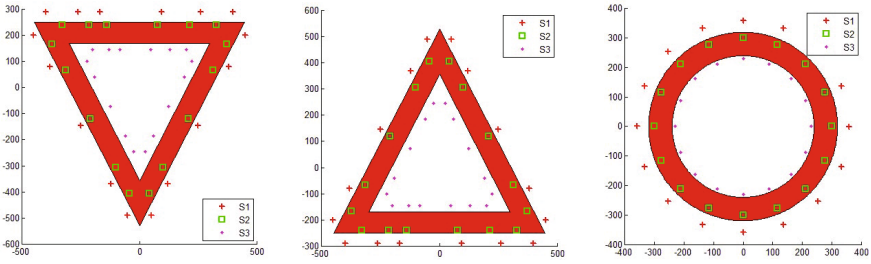

Fig. 1. The model used to recognize priority, warning and prohibitory signs. Each model consists of three sets of points. The first (S_1) lies just outside the sign; the second (S_2) on the red band, while the third (S_3) inside the sign.

In [11], PSO generates location estimates for a sign as each particle in the swarm encodes the candidate sign pose as four values: its offsets along the x, y and z axes, and its rotation around the vertical axis (yaw) in the camera reference frame. Rotation around the camera optic axis (roll) and the horizontal axis (pitch) have been ignored after some preliminary tests showed that they have little relevance. The image region located by the model is then rectified via an inverse perspective transform in order to obtain a frontal view. Then, the three sets of points (see Figure 1) are evaluated according to the fitness function described below; a sign is considered to have been detected when the fitness value is below a fixed threshold.

For every set of points, three color histograms in the HSV color space are computed. Then, the Bhattacharyya coefficient [6] $B(x, y)$, which estimates the overlap between two statistical samples, is used to compare the histograms. The fitness function can be expressed as follows:

$$f = \frac{k_0(1 - B(h_1, h_2)) + k_1(1 - B(h_2, h_3)) + k_2 B(h_1, h_r)}{k_0 + k_1 + k_2}$$

where h_i is the histogram of set S_i, and h_r is a reference histogram centered on red; k_0, k_1 and $k_2 \in \Re^+$ are used to weigh the elements of the equation. This means that a sign is detected when:

- the histogram of the set outside the sign is different from the histogram of the set located on the red band;
- the histogram of the points in the red band is as different as possible from the one computed on the inner area of the sign;
- the histogram of the points in the red band is similar to the reference histogram we defined for the red hue (a Gaussian centered on pure red).

This operation is repeated twice for every frame to permit recognition of more than one sign of the same category. This method has shown good robustness against illumination changes.

Mandatory signs (see figure 2) are detected similarly to the other three categories, although some changes have been introduced to improve performance.

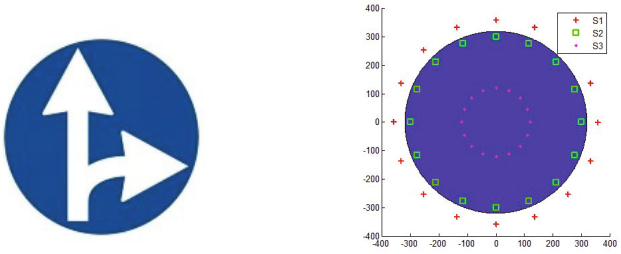

Fig. 2. An example of a mandatory sign, along with the corresponding model

We still consider three similarly arranged sets of points: the only difference in their location is that set S_3 is located closer to the center of the sign, in order to get more information about the white foreground, which usually includes the main information useful for sign recognition. The fitness function has also been changed to:

$$f = \frac{\bar{k}_0(1 - B(h_1, h_2)) + \bar{k}_1(B(h_3, h'_r)) + \bar{k}_2 B(h_2, h''_r) + \bar{k}_3(1 - B(h_2, h_r))}{\bar{k}_0 + \bar{k}_1 + \bar{k}_2 + \bar{k}_3}$$

where h'_r is a reference histogram with peaks corresponding to blue and white, h''_r a reference histogram centered on blue, and $\bar{k}_i$, $i = 1, \ldots, 4$ are positive weights.

2.2 Sign Classification

Before performing classification, the rectified image of the detected sign is pre-processed to reduce the complexity of this step. The following pre-processing steps are performed:

- the image is re-sampled to 50×50 pixels, converted to gray scale, and the pixels outside the sign are removed;
- for mandatory signs, in which the foreground is brighter than the background, gray-scale values are inverted, in order for the inputs to the final classifier to have similar contrast features;
- the histogram is calculated, average-filtered to remove isolated peaks and stretched between the mean intensities of the sign background and foreground.

In our test set we considered 27 different classes for prohibitory, 30 classes for warning, and 27 for mandatory signs. The training sets contained 9 instances of each sign, obtained by applying the pre-processing phase over synthetic noiseless images of the sign and adding some noise in terms of position and rotation. Since there is only one possible instance of priority signs, we implemented a binary classifier and created a test set which includes priority signs along with signs of different categories and other elements that are not signs (cars, trees, guard rails . . .).

3 Implementation Details

Both the detection and classification phases have been implemented on a Graphical Processing Unit (GPU) within the CUDA (Compute Unified Distributed Architecture) environment available from nVIDIA. CUDA requires that a problem be divided into groups of cooperating threads (each group is called a *thread block*), organized in one-, two- or three-dimensional grids. The threads within a block are also organized in similar grids. The performance of CUDA code largely depends on the grid configurations and memory access schemes. As in CPUs, also in GPUs, memory is arranged in a hierarchy, in which each thread has its own private local memory, thread blocks use shared memory that is visible only to the threads of the block, while all threads have access to global device, texture, and constant memory spaces [14]. To maximize efficiency, algorithms developed in CUDA-C should rely mainly on fast local and shared memory, avoiding frequent accesses to global memory locations.

The details about the parallel design and implementation of the methods used for detection and classification are discussed in the following subsections.

3.1 Parallel Particle Swarm Optimization

Our parallel version of PSO [11] is divided into three kernels: position update, fitness evaluation, and bests update. Two-dimensional thread-block grids represent different swarms (one for each type of sign) along one dimension while, along the other, every block represents a particle. Position update and fitness evaluation are performed by the threads of a block working on particle data loaded in shared memory for faster performance. Kernels follow a common procedure that: i) loads particle data into shared memory from global device memory, ii) processes the data in local and shared memory, iii) stores the results back to global memory, at the end of its execution, to make them visible to other kernels. We use four swarms (one for each sign category), each consisting of 64 particles arranged in a ring topology, that run for 250 generations. As for PSO parameters, we set C_1 and C_2 to 1.19, and the inertia factor w to 0.72.

3.2 Parallel Differential Evolution

Differential Evolution (DE) [18] is a powerful stochastic real-parameter optimization algorithm [1]. New solutions are generated by performing a crossover operation between one element of the current population (*parent*) and a *donor*, which is created by the combination of some randomly chosen solutions from the population. This new element, called *trial*, is then evaluated and can replace the parent if it has a better fitness than the parent's. The parallel implementation of DE [12] resembles parallel PSO, except that, instead of the position update kernel, DE uses a kernel to generate trial vectors for every element in the solution group, and another kernel to evaluate the fitness of the trial solutions to possibly replace the parent with them. There are several flavors of DE, depending on the method for choosing the individuals that are combined to form donor solutions,

and the type of crossover used to generate the trial solutions. In our experiments, we adopt random mutation and binomial crossover. The same PSO swarm configuration was used for the DE population, also run for 250 generations, with DE parameters: F set to 0.5 and C_r to 0.9.

3.3 Parallel Multi-Layer Perceptron

The Multi-layer Perceptron (MLP) is the most commonly used artificial neural network [4], in which only feedforward connections between neurons are allowed and a supervised training algorithm (backpropagation) is used.

The number of operations needed to perform the classification using MLP is very high. For instance, the MLP we use for the classification of warning signs has four fully-connected layers whose sizes are respectively 2500, 180, 90, 30. This means that, for the first layer, the total number of products that has to be computed is 2500×180, as each of the 2500 inputs is multiplied by the weight of the connections between it and all neurons of the next layer, and must then be summed. These operations amount to a product between a 2500-elements vector and a 180×2500 matrix. In our parallel implementation the computation of each layer of the network is a CUDA kernel. The operations performed by each kernel are:

1. loading layer inputs in shared memory;
2. computing, partly in parallel and partly sequentially, the products (as suggested by [3]);
3. summing the products by means of a parallel reduction;
4. computing the activation function over the sum of the products.

3.4 Parallel Learning Vector Quantization

Learning Vector Quantization (LVQ) is a supervised classification algorithm developed by Kohonen [8]. The basic idea is to find a small set (called *codebook*) of prototypes (called *codebook vectors*) representative of all possible inputs. New data are then classified according to the most similar codebook vector. The training phase is performed by iterating over the examples in a training set: if an example is correctly classified, the codebook vector closest to the data sample is attracted towards it, otherwise the codebook vector is moved in the opposite direction. The CUDA implementation of LVQ is straightforward: a kernel computes the distance between the input vector and the codebook vectors that compose the network. Then, a parallel reduction is performed to compute the classification result, that is the label corresponding to the most similar codebook vector. Table 1 shows the topologies and sizes of the networks that have obtained the best results during our experiments.

4 Experimental Results

In this section we will describe the data and the experiments performed to evaluate our system, for both detection and classification.

Table 1. MLP and LVQ structures for the four categories of signs

Category	MLP Topology	LVQ size
Prohibitory	2500×180×90×27	176
Warning	2500×180×90×30	246
Mandatory	2500×100×50×27	128
Priority	2500×120×16×2	128

4.1 Detection

The benchmark used to evaluate the results in a real environment is composed of two sequences [10]. The first one, which includes 10000 frames at a resolution of 750×480 pixels, was acquired at 7.5 fps in Parma on a sunny day. The sequence contains images featuring all possible light orientations. The second sequence is about 5000 frames long and was acquired at 7.5 fps in Turin on a cloudy day. Images in this sequence feature more constant lighting but lower contrast.

We compared the results of DE and PSO in terms of correct/incorrect detections of the signs. DE was able to detect more signs (scoring more true and false positives); this suggests that DE has a greater exploitation ability, and is able to refine solutions better than PSO. However, if fitness values are compared, PSO has a better average, with a lower standard deviation, showing a more consistent behavior. Table 2 reports the best and worst detection results obtained over 10 runs on each sequence.

4.2 Classification

The evaluation of the classification system was firstly performed over a test set that comprises synthetic, good quality and noisy or deformed images of signs, as described in [10], then on the two real sequences. The first rows of table 3 show the percentage of correct classification for the four categories on the test set. The same table (second and third row) shows the results of the classification of all signs detected in the experiments reported in table 2. We take all detections into account: this means that the same sign can be detected and classified in more than one frame.

Table 2. Results of the detection phase (min-max) for the four categories of signs (detections): worst and best result in 10 independent runs

		Parma Sequence			Turin Sequence		
		Total	False Positives	Detections	Total	False Positives	Detections
Warning	DE	51	0-1	27-31	53	0-2	39-43
	PSO	51	0-0	27-30	53	0-1	35-40
Prohibitory	DE	44	2-6	26-30	47	2-4	39-42
	PSO	44	0-1	22-27	47	0-1	39-40
Priority	DE	30	5-11	18-22	15	2-4	13-15
	PSO	30	0-2	15-18	15	0-1	7-12
Mandatory	DE	62	2-4	40-41	39	0-1	27-29
	PSO	62	0-1	35-39	39	0-1	24-27

Table 3. Percentage of correct sign classifications in the test set and in the two sequences for the four categories of signs

Set		Warning	Prohibitory	Priority	Mandatory	Total
Test Set	LVQ	74.3	78.7	98.7	89.3	81.6
	MLP	71.5	80.5	99.4	94.3	83.1
Parma Sequence	LVQ	91.2	81.9	97.0	99.4	92.4
	MLP	69.0	69.2	95.4	99.4	83.2
Turin Sequence	LVQ	75.3	94.8	100	100	92.5
	MLP	77.1	77.8	98.9	99.6	88.4

The table shows that the two methods have similar performance (with slightly better performances for the MLP) on the test set, but LVQ achieves better results over the real sequences. This probably happens because the test set contains several deformed or noisy images, while the images produced by the detection are usually good in terms of quality and positioning. The conclusion can be that LVQ is able to yield better results when operating on good quality images, while MLPs have better generalization ability. Finally, in table 4, we present the results of the entire system using DE and LVQ. Since, in the actual implementation, there is no way to track signs during the flow of the sequence (and, consequently, each sign can be detected multiple times), we consider a sign to have been correctly classified by evaluating off-line if at least half of its classifications are correct. Figure 3 shows some examples of correct classifications, correct detections with wrong classification and wrong detections.

4.3 Computational Efficiency

Experiments were run on a PC equipped with a 64-bit Intel® Core(TM) i7 CPU running at 2.67 GHz, combined with a Quadro FX5800 graphics card by nVIDIA, having 4Gb of video RAM and 240 processing cores. All the operations performed on each frame (two repetitions of detection plus classification for each sign category) require an average time of 57 ms, which corresponds to a frame rate of 17.5 fps. A sequential version of the same algorithm could reach only a frame rate of 4-5 fps [11], a processing speed which is not acceptable for this kind of application.

Table 4. Final results of the system using DE and LVQ

	Parma Sequence		Turin Sequence	
	Total	Correct	Total	Correct
Warning	51	25-28	53	37-43
Prohibitory	44	24-27	47	38-41
Priority	30	17-20	15	11-15
Mandatory	62	38-41	39	26-29

Fig. 3. Some results of the system. The first three columns show signs that have been correctly detected and classified, the fourth shows misclassified signs, while the last one shows wrong detections. Our method is robust against differences in light conditions and partial occlusions.

5 Conclusions

We presented a system for detecting and classifying road signs. Detection is performed using a method in which a model of the sign to detect is translated and projected onto the image. Candidate solutions are created by means of swarm intelligence techniques. Differential Evolution and Particle Swarm Optimization have been compared in this phase, showing that PSO has better average results than DE, while DE exhibits a better exploitation ability, which produces a larger number of detections. Classification have been performed using Learning Vector Quantization and Multi-layer Perceptrons. The results showed that LVQ has better performance when working on good quality images, while MLPs have greater generalization ability. The system has been implemented on GPU using CUDA and is able to correctly detect and classify around 70% of the signs at 17.5 fps, a similar result in shorter time, compared to the best results obtained on the same sequences so far [10].

Acknowledgments. Youssef S. G. Nashed is funded by the European Commission (MIBISOC Marie Curie Initial Training Network, FP7 PEOPLE-ITN-2008, GA n. 238819). Roberto Ugolotti is funded by Compagnia di San Paolo and Fondazione Cariparma. The authors wish to thank Luca Donati for his help in the development of the system and his support regarding the library used for the training of MLP[1], and VisLab for providing the sequences.

[1] Libcudann is freely available at `http://sourceforge.net/projects/libcudann`

References

1. Das, S., Suganthan, P.: Differential Evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15(1), 4–31 (2011)
2. Escalera, S., Baró, X., Pujol, O., Vitrià, J., Radeva, P.: Background on traffic sign detection and recognition. In: Traffic-Sign Recognition Systems. SpringerBriefs in Computer Science, pp. 5–13. Springer, London (2011)
3. Harris, M.: Optimizing parallel reduction in CUDA. NVIDIA Developer Technology (2008)
4. Haykin, S.: Neural Networks: a comprehensive foundation. Prentice Hall (1999)
5. Jiang, Y., Zhou, S., Jiang, Y., Gong, J., Xiong, G., Chen, H.: Traffic sign recognition using ridge regression and Otsu method. In: IEEE Intelligent Vehicles Symposium (IV), pp. 613–618 (2011)
6. Kailath, T.: The divergence and Bhattacharyya distance measures in signal selection. IEEE Transactions on Communication Technology 15(1), 52–60 (1967)
7. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
8. Kohonen, T.: Learning Vector Quantization. Neural Networks 1(supplement 1) (1988)
9. Maldonado-Bascon, S., Lafuente-Arroyo, S., Gil-Jimenez, P., Gomez-Moreno, H., Lopez-Ferreras, F.: Road-sign detection and recognition based on support vector machines. IEEE Trans. on Intelligent Transportation Systems 8(2), 264–278 (2007)
10. Medici, P., Caraffi, C., Cardarelli, E., Porta, P., Ghisio, G.: Real time road signs classification. In: IEEE International Conference on Vehicular Electronics and Safety, ICVES 2008, pp. 253–258 (2008)
11. Mussi, L., Cagnoni, S., Cardarelli, E., Daolio, F., Medici, P., Porta, P.: GPU implementation of a road sign detector based on Particle Swarm Optimization. Evolutionary Intelligence 3(3), 155–169 (2010)
12. Nashed, Y.S., Ugolotti, R., Mesejo, P., Cagnoni, S.: libCudaOptimize: an open source library of GPU-based metaheuristics. In: Proc. Genetic and Evolutionary Computation Conference, GECCO 2012 (2012)
13. Nguwi, Y.Y., Kouzani, A.: Detection and classification of road signs in natural environments. Neural Computing & Applications 17(3), 265–289 (2008)
14. nVIDIA Corporation: nVIDIA CUDA programming guide v. 4.0 (May 2011)
15. Ohara, H., Nishikawa, I., Miki, S., Yabuki, N.: Detection and recognition of road signs using simple layered neural networks. In: Proceedings of the 9th International Conference on Neural Information Processing, ICONIP 2002, vol. 2, pp. 626–630 (2002)
16. Paulo, C., Correia, P.: Automatic detection and classification of traffic signs. In: Workshop on Image Analysis for Multimedia Interactive Services (2007)
17. Prieto, M.S., Allen, A.R.: Using self-organising maps in the detection and recognition of road signs. Image and Vision Computing 27(6), 673–683 (2009)
18. Storn, R., Price, K.: Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute (1995)