A Template-Based Approach for Real-Time Speed-Limit-Sign Recognition on an Embedded System Using GPU Computing

Pınar Muyan-Özçelik[1], Vladimir Glavtchev[1,2], Jeffrey M. Ota[2],
and John D. Owens[1]

[1] University of California, Davis, CA
[2] BMW Group Technology Office, Palo Alto, CA

Abstract. We present a template-based pipeline that performs real-time speed-limit-sign recognition using an embedded system with a low-end GPU as the main processing element. Our pipeline operates in the frequency domain, and uses nonlinear composite filters and a contrast-enhancing preprocessing step to improve its accuracy. Running at interactive rates, our system achieves 90% accuracy over 120 EU speed-limit signs on 45 minutes of video footage, superior to the 75% accuracy of a non-real-time GPU-based SIFT pipeline.

1 Introduction

As object recognition systems continue to increase in accuracy and robustness, we are beginning to see their deployment in real-time applications. In this work we target *real-time embedded systems*: systems that can interact with the real world at interactive rates using embedded processors. The low cost of embedded systems—an order of magnitude below typical CPUs—make them suitable for use in a variety of domains, including the automotive application space that we analyze here. However, meeting real-time performance requirements with the modest computational resources of these embedded processors, particularly with a parallel processing model, presents an important research challenge.

In this study, we aim to address this challenge and perform real-time speed-limit-sign recognition on an embedded platform. To achieve our goal, we leverage the inherent parallelism in the recognition task using Graphics Processing Unit (GPU) computing and construct our pipeline from modular components. The data-parallel nature of recognition tasks is an excellent fit for an embedded, low-power, parallel processor such as the low-end GPU we use in this study. In addition, the GPU offers superior price-performance and power-performance to comparable processors. Hence, in our pipeline we implement template-based recognition techniques that are well-suited to the GPU architecture. In addition, we built our approach from modular parts that can be extended or contracted. We discuss how one can make the best use of the limited resources of underlying hardware by fine-tuning the parameters of these separate components based on the tradeoff between the runtime and success rate in Section 5.

As GPUs have become programmable, they have been increasingly used for applications beyond traditional graphics that require general-purpose computation [1]. The computer vision domain is one of the domains GPUs have provided significant performance improvements (e.g. Fung et al.'s OpenVIDIA project [2]). The advent of GPU computing has also allowed researchers to revisit older and simpler, but very effective, data-parallel techniques that have fallen out of favor due to their compute demands. Template-based object recognition in the computer vision literature is one of these techniques. For instance, several other template-based road sign recognition approaches (Section 2) have leveraged template-based techniques. However, to the best of our knowledge, none of these previous studies provided real-time recognition of road signs on an embedded system. They employed commodity computers or optical devices to meet the compute demands of their approaches. Using GPU computing, we parallelize the time-consuming computation of template-based recognition and provide real-time performance on an embedded domain. In addition to being suitable to the GPU architecture, another advantage of using a template-based approach is that our pipeline can be easily modified to recognize other objects, such as US speed-limit signs or other salient road features.

In the computer vision literature, the Scale Invariant Feature Transform (SIFT) [3] is a commonly used method for performing object recognition. Hence, in order to evaluate our template-based approach, we also implement a SIFT-based speed-limit-sign recognition system on the GPU and compare these two approaches. Our results (Section 5) show that the template-based pipeline provides a higher success rate and faster runtime than the SIFT-based pipeline.

2 Previous Work

One of the approaches used for template-based road sign recognition is conventional template matching, in which cross-correlation is calculated between the template and the part of the scene of the same size to measure the match. In the literature, several studies use conventional template matching in the final classification stage of the recognition pipeline, after the candidate road signs are detected [4, 5]. Conventional template matching is not the preferred technique for sign detection because searching the candidate road sign location with this approach needs many cross-correlation computations between the templates and different parts of the scene of the same size. Since this is a convolution-type operation, it requires a long computation time. In order to reduce the search space and thus, the runtime of the conventional template matching, Betke and Makris [6] proposed using simulated annealing.

There are other template-based approaches that also involve a convolution-type operation. To detect the potential road signs, Gavrila [7] proposed matching the template and the scene using distance transforms; Cyganek [8] presented a system that operates on the Gaussian scale-space and does template matching in the log-polar domain.

Several studies in the field of optics have also investigated the template-based recognition of road signs in the frequency space [9,10,11]. They proposed systems

that perform FFT correlation between the scene and the filters generated from the templates. Drawing upon this technique, Javidi et al. [12] presented an offline system to perform speed-limit-sign recognition that is most similar in spirit to our implementation.

Some other studies in the literature make use of color information to perform template-based road sign recognition [13, 14]. In our approach we cannot utilize these techniques, since we are working with grayscale videos.

3   Approach

With our approach, we have chosen to perform template-based matching in the frequency space, since it provides a faster runtime than the approaches that perform convolution-type operations. FFT correlation involves taking the FFT of both the template and the scene, multiplying the complex conjugate of the FFT of the template with the FFT of the scene, and taking the inverse Fourier transform of the product. Hence, instead of computing many match values using a convolution, we can perform one multiplication in the frequency domain, which is more efficient. Another advantage of working in the frequency space is that it allows us to perform some operations in the Fourier domain to improve the matching performance (e.g. kth-Law, explained below).

FFT-based recognition studies in the optics literature propose correlating the scene with *composite filters* instead of the templates. Composite filters are generated from several templates and can be thought of as "combination templates". The advantage of one composite filter instead of several templates is that it reduces the number of correlations we need to perform and thus, provides faster runtime.

Synthetic discriminant functions (SDF) [15] are one of the popular techniques for generating composite filters. From the several variations of SDF filters, we have chosen to work with the MACE (Minimum Average Correlation Energy) SDF filter [16] due to its low false alarm rate. On the road, several objects may look like a speed-limit sign. The MACE filter provides a high discrimination ability against these impostor objects.

Performing a FFT correlation between the scene and the composite filter produces a correlation plane. The MACE filter minimizes the average correlation energy of the correlation plane and produces a sharp distinct peak where the object is located. We first Fourier-transform the templates we would like to include in the filter and then perform MACE filter synthesis using these transforms.

Although the MACE filter provides good discrimination, it alone does not provide sufficient accuracy. In addition, we needed to improve the *illumination invariance* of our system. To address these challenges, we extend the MACE filter by applying kth-Law nonlinearity [17], which improves the peak sharpness. The nonlinear operation raises the magnitude of the Fourier transform to the power of k, while keeping its original phase. In order to compute a FFT correlation between the scene and a kth-Law MACE filter, we apply a kth-Law nonlinear operation to the FFT of the scene before it is multiplied with the complex conjugate of the kth-Law MACE filter.

Several metrics can evaluate the goodness of the match in correlations. Since the kth-Law MACE filter is designed to suppress the sidelobes adjacent to peaks in the correlation plane, we use PSR (Peak-to-Sidelobe Ratio) [18], which measures the peak sharpness by taking into account the small area around the peak.

Kth-Law nonlinearity produces enhanced illumination invariance, but still misses many cases with low contrast. Hence, we add a *histogram equalization* preprocessing step in our system to improve the contrast of the scene. This technique involves adjusting the intensity values of the scene to equally distribute intensities throughout the whole brightness scale. However, since histogram equalization adjusts the values based on the statistics collected from the entire image, it misses some details. Speed-limit signs usually appear in small regions of the scenes. Hence, it is critical for us to bring out as much image detail as possible: we thus use "Contrast Limited Adaptive Histogram Equalization" (CLAHE) [19] to enhance the contrast of the scene. This algorithm divides the image into small tiles and performs histogram equalization on these small local regions instead of the entire image, thus bringing out more small-scale detail.

4   Implementation

Template-based speed-limit-sign recognition pipeline. The template-based pipeline has four main stages: preprocessing, detection, classification, and temporal integration. Figure 1.a shows the overview of this pipeline. We generate composite filters offline and input them to our system.

The composite filters used in the detection stage are more general than the ones used in the classification stage. They are generated from the templates 00 and 100, which helps with detecting two-digit and three-digit signs, respectively.
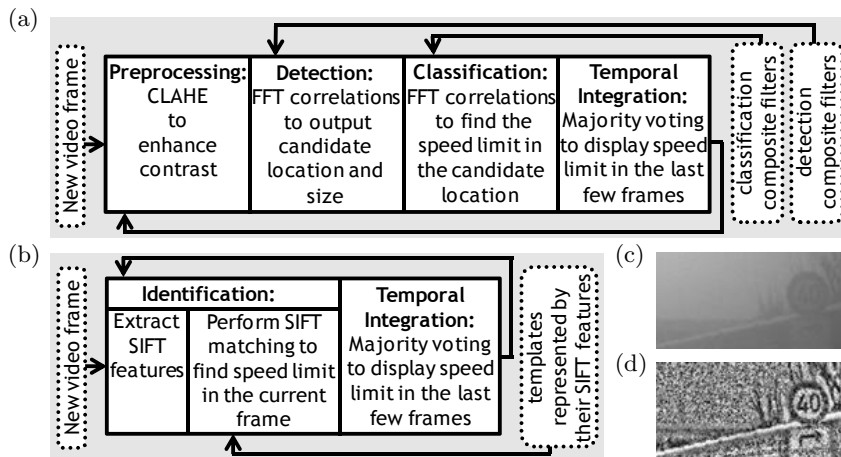

Fig. 1. (a) Template-based pipeline (b) SIFT-based pipeline (c) Before CLAHE (d) After CLAHE

On the other hand, we generate a different set of classification filters for each speed-limit sign we would like to recognize. Both detection and classification filters integrate different out-of-plane rotations of the templates. This allows the system to have rotation invariance along the X and Y axes. In addition, both type of filters have different sizes, which allows the system to have scale invariance. For each size, we have classification filters with different in-plane rotations, which provides the system with rotation invariance along the Z axis. EU speed-limit signs have circular shapes, which are somewhat insensitive to in-plane rotations when the number inside the sign is not considered. To recognize signs with shapes that are more sensitive to in-plane rotations (e.g. triangles), we can also generate in-plane rotations of detection filters to improve accuracy. We set the background of the templates to gray, which helps in recognizing signs of both dark or light backgrounds. We do not include different sizes and in-plane rotations of the templates in the creation of composite filters, because these images typically have higher changes in energy when compared to images that represent different out-of-plane rotations of the templates. Moreover, the number of images we can include in a composite filter is limited.

The effect of the preprocessing stage is shown in Figures 1.c–d. After CLAHE is applied to the scene, the speed-limit sign becomes more visible. Although the "after" scene looks fairly different than the "before" scene due to the noise introduced by CLAHE, the template-based approach we pursue works very well with these kind of preprocessed images.

In the detection stage, we first perform a FFT correlation between the scene and the detection composite filters. Then, we determine the detection filter that returns the maximum PSR. The location of the peak value in the correlation plane generated by this detection filter indicates the location of the candidate sign. Likewise, the size of this composite filter indicates the size of the candidate sign.

We start the classification stage by performing FFT correlations between the classification composite filters and the part of the scene that includes the candidate sign. We only use the classification filters that have the same size as the candidate sign. Then, we determine the classification filter that returns the maximum PSR. If this value is below a certain threshold, we conclude that there is no speed-limit signs in the scene and start processing the next frame. If not, the classification filter with the maximum PSR indicates the number displayed by the speed-limit sign in the current scene. In addition, the in-plane rotation of this filter indicates the in-plane rotation of the sign. Due to various factors (e.g. the sign is partially occluded), it is possible to misclassify the signs in the scene. Hence, providing a result that only depends on the findings of the current frame is not reliable.

In the temporal integration stage, we increase the reliability of our results by accumulating the findings from the sequence of frames. For this purpose, we employ a majority voting technique similar to the one used by Keller et al. [20]. Each frame votes for the speed-limit number indicated by its classification stage. The maximum PSR determined at this stage is used as the base vote value. If the previous frame also voted for the same number, the vote is increased

by multiplying the base value with a constant factor when one or both of the following conditions are met: 1) the size of the sign is not decreasing; 2) in-plane rotation of the sign remains the same. After all the frames in the sequence are processed, we display the speed-limit number that collected the most votes over a given threshold as the final result.

Since most operations of our implementation of a template-based approach are data-parallel, this pipeline is suitable to be implemented on the GPU. We use NVIDIA's CUFFT library to take inverse and forward FFTs. We wrote kernels to apply kth-Law nonlinearity to the FFT of the scene, to take the complex conjugate of the composite filter in frequency domain, to multiply these two FFTs, and to normalize the result of this product. In addition, we find the peak in the correlation plane with a GPU kernel that performs a reduction operation. Currently, we use optimized C code to apply CLAHE. However, this operation has several data-parallel parts and could be mapped to the GPU to further improve runtime.

SIFT-based speed-limit-sign recognition pipeline. The SIFT-based pipeline has two main stages: identification and temporal integration. Figure 1.b shows the overview of this pipeline. We extract the SIFT features of the templates offline and input them to our system. The templates we use in this pipeline are the same templates we use in the template-based pipeline to generate the composite filters.

In the identification stage, we first extract SIFT features of the scene. Then, we perform SIFT matching between the features of the scene and templates. We use SiftGPU [21], an open source GPU implementation of SIFT, for feature extraction and matching. To improve the performance, we reject a match if the orientation difference of the matched keys is larger than a certain threshold (as proposed by Kuş et al. [22]). Next, we search for a template that returns the maximum number of matches over a certain threshold to determine whether we have a speed-limit sign in the current frame. Finally, we accumulate findings from a sequence of frames in the temporal integration stage by employing similar techniques we use in the template-based pipeline.

5   Results and Discussion

Template-based versus SIFT-based pipeline. Our EU speed-limit-sign recognition test data is a collection of grayscale videos recorded in different weather (e.g. sunny, foggy, rainy, and snowy) and road (e.g. highway, in the city, and country) conditions in Europe. In total, we use footage captured from 45 minutes of driving that includes 120 EU speed-limit signs. Video size is 640x240. We have run our experiments on a laptop equipped with an Intel Core2 Duo P8600 2.4 GHz CPU and a GeForce 9600M GT, a laptop graphics card comparable in performance to next-generation embedded GPUs.

The runtime of the template-based pipeline is 18.5 fps. Since our video capture rate is slower, we are able to process all frames in real time with template-based approach. Computation time for the SIFT-based pipeline increases with the number of keypoints extracted from an image and for a frame with moderate

complexity the runtime is around 120 ms/frame (∼8 fps), which is slower than the video capture rate. Hence, the SIFT-based pipeline does not provide real-time performance. The SIFT-based approach has a slower runtime than the template-based one on our resource-constrained GPU, since it consists of different stages that are computationally intensive and require a large GPU memory.

The template-based pipeline returned a 90% success rate with no misclassifications or false positives. An offline run of the SIFT-based pipeline provided a 75% success rate with four misclassifications and nine false positives. The SIFT-based approach has a lower success rate mainly due to two reasons. First, SIFT recognition works best when objects have some complexity. However, speed-limit signs have simple shapes and constant color regions and do not have any texture. In addition, they usually appear small in the videos. Thus, often SIFT cannot extract enough distinct features from these signs and the same number of matches are returned by different templates. Secondly, we could not use CLAHE in the SIFT-based pipeline. Applying CLAHE on the template-based pipeline improved our success rate from 65% to 90% and eliminated all misclassifications and false positives. However, since SIFT cannot handle the noise introduced by CLAHE, we could not utilize this technique in the SIFT-based pipeline.

Both pipelines perform well in simple cases as well as several challenging cases. Both reject signs that have a dominant difference. Because of the thick line that crosses the whole sign, an end-of-50 sign (Figure 2.1) can be distinguished from a speed-limit sign. Both also recognize signs with insignificant modifications. Figure 2.2 depicts a speed-limit sign with a small stain in the digit 0.

We see several cases where the template-based pipeline succeeds and the SIFT-based pipeline fails:

• The template-based approach capably recognizes small signs (e.g. Figure 2.3). As the sign gets closer, we see a larger viewpoint change and thus, recognition becomes harder. As a result, in some cases the SIFT-based pipeline misclassifies the speed-limit sign, since it does not start recognizing the sign when it is far and small. In the SIFT-based pipeline, in order to recognize small signs, we doubled
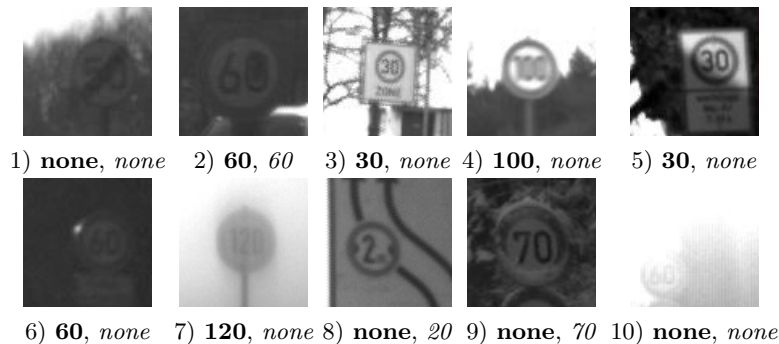

1) **none**, *none*   2) **60**, *60*   3) **30**, *none*   4) **100**, *none*   5) **30**, *none*

6) **60**, *none*   7) **120**, *none*   8) **none**, *20*   9) **none**, *70*   10) **none**, *none*

Fig. 2. Scene examples: **Template-based** and *SIFT-based* pipeline results shown in **bold** and *italic*, respectively.

the image size and reduced the initial Gaussian blur. Even if these improvements were helpful, they were not effective enough to recognize smaller signs such as the one in Figure 2.3.

• The template-based pipeline is better at handling different kind of noise introduced by effects such as motion blur (Figure 2.4) and partial shade (Figure 2.5). Our template-based approach is resistant to noise, allowing us to use CLAHE. By applying CLAHE, we can recognize hard cases such as when the sun is behind the sign (Figure 2.6) or a light beam effect (Figure 2.7). In order to recognize low contrast cases with a SIFT-based pipeline, we have decreased the threshold used to select potential keypoints. Although this change has provided some improvement, it was not effective enough for recognizing very-low-contrast cases such as Figure 2.6.

• The template-based approach is better at recognizing signs as a whole. Since the SIFT-based pipeline deals with features that are local, it can misclassify signs that look like only part of a sign. For instance, a 2-meter-width-limit sign in Figure 2.8 is misclassified as a 20 km/h speed-limit sign by the SIFT-based approach, since the digit "2" in both signs looks very similar.

We also see cases where the SIFT-based pipeline succeeds and the template-based pipeline fails:

• Although dealing with local features causes the SIFT-based approach to miss the big picture and fail in Figure 2.8, it becomes an advantage for recognizing partially occluded signs. For instance, SIFT-based pipeline performs better than template-based approach in Figure 2.9, where the sign is partially occluded by snow.

• The SIFT-based pipeline is good at recognizing signs with large rotations as well as the ones that initially appear large in the videos. In order to recognize these signs with the template-based pipeline, we need to cover larger rotations and bigger sizes with composite filters, which in turn would increase our runtime.

In some cases, both template-based and SIFT-based approaches fail:

• Although the template-based pipeline succeeds where the SIFT-based approach has problems recognizing small signs (e.g. Figure 2.3), if the signs get even smaller, template-based also starts missing them. Including an additional composite filter with smaller sizes does not help since it introduces misclassifications and false positives.

• Both pipelines perform poorly when a big part of the sign is missing. For instance, in Figure 2.10 a big part of the sign is not visible due to the bright sunshine.

Template-based pipeline parameter study. With additional computational resources, we could perform more computation and achieve higher accuracy. With an embedded system and its limited computational capabilities, however, we instead conducted a parameter study to answer the question "given a fixed amount of compute resources, what is the best way to allocate those resources to achieve maximum accuracy?" We thus varied our base configuration, which consisted of five sizes [25 30 35 40 45], three in-plane rotations $[-6° \ 0° \ 6°] = [0° \ \mp 6°]$, seven out-of-plane rotations along the Y-axis $[0° \ \mp 10° \ \mp 20° \ \mp 30°]$, and three

out-of-plane rotations along X-axis [0° ∓10°], all run on a 4-SM GPU. We found these parameters were well-suited for maximum accuracy on our base hardware platform.

Varying each of these parameters allowed us to extend or contract different modules of our template-based pipeline. Hence, this parameter study also provides insights for making recommendations to achieve optimum performance when we are given less or more compute power. Space prevents us from describing the detailed results, but we draw the following conclusions from these experiments. With less compute power available, we could choose to reduce the number of frames per second that we analyze, but instead a better option is to process only the smaller sizes of signs at a higher frame rate. With more compute power, we can achieve higher accuracy in three principal ways: increase the processing frame rate, add larger sizes of signs, and generate additional composite filters with larger out-of-plane rotations.

6   Conclusion

Our work addresses the main challenge of meeting the real-time performance requirements of speed-limit-sign recognition with limited hardware resources. To achieve this goal, we exploit the inherit parallelism in this task using GPU computing and build our pipeline from parameterized modules. Our template-based pipeline is suitable to be implemented on the GPU and can be easily modified to recognize other salient road features. Our future work includes expanding the breath of detected objects (US signs and other signs of different types), investigating other vision tasks (such as optical flow to detect potential collisions), and developing software support for our data-parallel embedded system that can run multiple tasks simultaneously (such as vision, graphics, and speech recognition) while delivering throughput and/or latency guarantees.

References

1. Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A.E., Purcell, T.: A survey of general-purpose computation on graphics hardware. Computer Graphics Forum 26(1), 80–113 (2007)
2. Fung, J., Mann, S., Aimone, C.: OpenVIDIA: Parallel GPU computer vision. In: ACM Multimedia, pp. 849–852 (2005)
3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60(2), 91–110 (2004)
4. Piccioli, G., De Micheli, E., Parodi, P., Campani, M.: Robust road sign detection and recognition from image sequences. In: IEEE IV, pp. 278–283 (1994)

5. Miura, J., Tsuyoshi, K., Shirai, Y.: An active vision system for real-time traffic sign recognition. In: IEEE ITSC, pp. 52–57 (2000)
6. Betke, M., Makris, N.C.: Fast object recognition in noisy images using simulated annealing. In: IEEE ICCV, pp. 523–530 (1995)
7. Gavrila, D.: Traffic sign recognition revisited. In: DAGM, pp. 86–93 (1999)
8. Cyganek, B.: Road signs recognition by the scale-space template matching in the log-polar domain. In: Martí, J., Benedí, J.M., Mendonça, A.M., Serrat, J. (eds.) IbPRIA. LNCS, vol. 4477, pp. 330–337. Springer, Heidelberg (2007)
9. Guibert, L., Keryer, G., Servel, A., Attia, M., MacKenzie, H., Pellat-Finet, P., de Bourgrenet de la Tocnaye, J.: On-board optical joint transform correlator for real-time road sign recognition. Opt. Eng. 34(1), 135–143 (1995)
10. Taniguchi, M., Mokuno, Y., Kintaka, K., Matsuoka, K.: Correlation filter design for classification of road sign by multiple optical correlators. In: SPIE, vol. 3804, pp. 140–147 (1999)
11. Perez, E., Javidi, B.: Nonlinear distortion-tolerant filters for detection of road signs in background noise. IEEE Transactions on Vehicular Technology 51(3), 567–576 (2002)
12. Javidi, B., Castro, M.A., Kishk, S., Perez, E.: Automated detection and analysis of speed limit signs. Technical Report JHR 02-285, University of Connecticut (2002)
13. Torresen, J., Bakke, W.J., Sekanina, L.: Efficient recognition of speed limit signs. In: IEEE ITSC, pp. 652–656 (2004)
14. Hsu, S.H., Huang, C.L.: Road sign detection and recognition using matching pursuit method. Image and Vision Computing 19(3), 119–129 (2001)
15. Casasent, D.: Unified synthetic discriminant function computational formulation. Appl. Opt. 23(10), 1620–1627 (1984)
16. Mahalanobis, A., Kumar, B.V.K.V., Casasent, D.: Minimum average correlation energy filters. Appl. Opt. 26(17), 3633–3640 (1987)
17. Javidi, B., Painchaud, D.: Distortion-invariant pattern recognition with fourier-plane nonlinear filters. Appl. Opt. 35(2), 318–331 (1996)
18. Savvides, M., Kumar, B.V., Khosla, P.: Face verification using correlation filters. In: IEEE AutoID, pp. 56–61 (2002)
19. Zuiderveld, K.: Contrast limited adaptive histogram equalization. In: Graphics Gems IV, pp. 474–485 (1994)
20. Keller, C.G., Sprunk, C., Bahlmann, C., Giebel, J., Baratoff, G.: Real-time recognition of U.S. speed signs. In: IEEE IV, pp. 518–523 (2008)
21. Wu, C.: SiftGPU: A GPU implementation of scale invariant feature transform, SIFT (2007), http://cs.unc.edu/~ccwu/siftgpu
22. Kuş, M.C., Gökmen, M., Etaner-Uyar, A.S.: Traffic sign recognition using scale invariant feature transform and color classification. In: ISCIS, pp. 1–6 (2008)