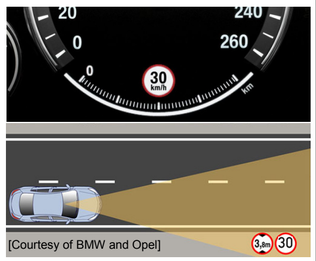
Speed-limit Sign Recognition with GPU Computing

Pınar Muyan-Özçelik*, Vladimir Glavtchev*§, Jeff Ota†, John D. Owens*
*University of California, Davis, †BMW Group Technology Office in Palo Alto, §NVIDIA Corporation in Santa Clara

Goal



We investigate the use of different GPU-based implementations for performing **real-time** speed limit sign recognition on a **resource-constrained** embedded system.

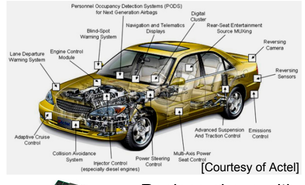
We juxtapose these alternative approaches in terms of:
✓success rate
✓run-time
✓how well it can be mapped to the GPU.

Also we determine:
✓best parameter combinations (recognition rate vs performance)
✓advantages of using a GPU instead of CPU
✓conditions which make a speed-limit sign easy/hard to recognize

Motivation

Automotive tasks can be grouped as:
✓computer vision (most tasks)
✓signal processing
✓graphics
✓networking

GPUs are good fit for all the above except the networking tasks.



[Courtesy of Actel]

Replace above with the GPU, which:
✓simplifies design
✓is cheap
✓is programmable

Hence, this study serves as a **proof of concept** for the use of GPU computing in automotive tasks.

Implementation and Results

Feature-based Approach

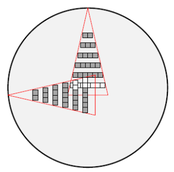
Pipeline:

New video frame

Preprocessing: Sobel edge detect with thresholding

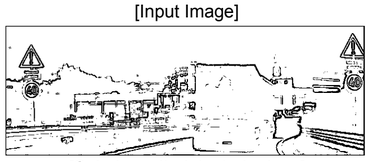
Detection Step 1: Radial symmetry voting: outputs an intensity map

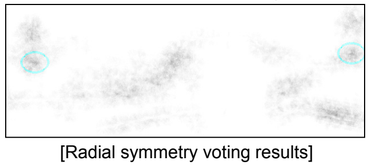
Detection Step 2: Maximum reduction to find top speed limit sign candidate locations



Radial symmetry[1] voting: Pixels vote in the direction of their gradient. Votes accumulate as they overlap to form an overall voting image.


[Input Image]


[Sobel result: detected edges]


[Radial symmetry voting results]


[Detected speed-limit signs]

Template-based Approach

(extension of Javidi et al.'s approach [2])

Pipeline:

New video frame

Preprocessing: CLAHE to enhance contrast

Detection: FFT correlations to output candidate location and size

Classification: FFT correlations to find the speed limit in the candidate location

Temporal Integration: Majority voting to display speed limit in the last few frames

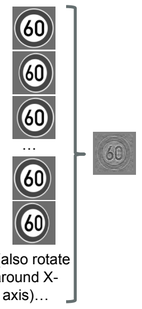
classification composite filters w/ different in-plane rotations

detection composite filters w/ different sizes

Templates:

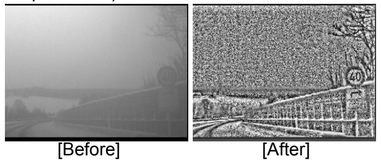
Composite Filter:

(kth-law nonlinear MACE filter generated offline w/ Matlab)



CLAHE:

(Contrast Limited Adaptive Histogram Equalization)


[Before] [After]

CLAHE improves recognition by enhancing the contrast of the scene.

Processor	Core Config.	Detection Runtime	Execution Rate	Speedup
Dual-Core Intel Atom 230 @ 1.6GHz	2	235 ms	4.25 fps	--
Intel Core2 6300 @ 1.86GHz	2	130 ms	7.4 fps	~1.8X
NVIDIA GeForce 9200M GS 128MB	1 : 8	65 ms	15 fps	~3.6X
NVIDIA GeForce 8600 GTS 256MB	4 : 32	23 ms	43.5 fps	~10.2X

SIFT-based Approach

(SIFT: Scale Invariant Feature Transform [3])

Pipeline:

New video frame

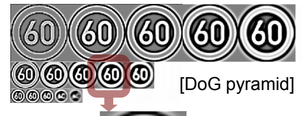
Recognition:
- Extract SIFT features
- Perform SIFT matching to find speed limit in the current frame
- **Temporal Integration:** Majority voting to display speed limit in the last few frames

templates represented by their SIFT features (generated offline)

Extracting SIFT features:
- Detect scale-space extrema to find potential points
- Select stable ones as "keypoints". Assign pos & size
- Assign one or more orientations to each keypoint
- Generate descriptor (feature vector) for each keypoint to provide invariance to viewpoint and illumination change

Detecting extrema:

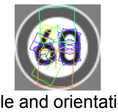

[Gaussian pyramid]


[DoG pyramid]

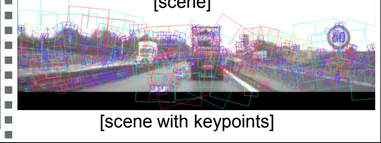

[extrema: maxima, minima]

SIFT keypoints:


[location]


[scale and orientation]


[scene]


[scene with keypoints]

Scene examples

Template-based 👍 *SIFT-based* 👍


E1: country 80
80, 80

E2: snowy 60
60, 60

E3: no-U-turn
none, none


E4: end of 50
none, none


E5: stained 60
60, 60

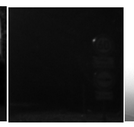
Template-based 👍 *SIFT-based* 👍

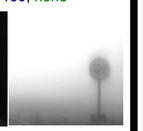

E6: 2m width
none, 20

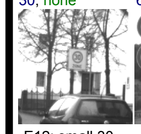

E7: 12t weight
none, 120


E8: motion blur 100
100, none


E9: part shade 30
30, none


E10: sun is behind 60
60, none


E11: light beam effect 120
120, none

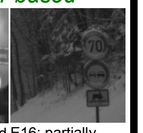

E12: small 30
30, none


E13: large viewpoint change 100
100, 130

Template-based 👎 *SIFT-based* 👍


E14: initially big 60
none, 60


E15: very rotated 70
none, 70


E16: partially occluded 70
none, 70

Template-based 👎 *SIFT-based* 👎


E17: very small 30
none, none


E18: big missing part 60
none, none

Approach	Detection Runtime	Classification Runtime	Total Runtime♦	Success Rate♦	Mapping to GPU
Feature-based	65 ms† (15 fps)	N/A	65ms	80% detection	CUDA is used to parallelize Sobel filtering, maximum-reduction, and vertices for voting patterns. The graphics pipeline (OpenGL) is used to accelerate radial symmetry voting.
Template-based	65 ms‡	40 ms‡	125 ms‡ (8 fps)	90% recognition with NO true negatives and NO false positives	Mapping the pipeline to CUDA kernels was straightforward since all stages involve data-parallel computation. CUFFT library is utilized.
SIFT-based	N/A	N/A	300 ms‡ (3 fps)	75% recognition with 4 true negatives and 9 false positives	siftGPU[4] is used. Almost all stages which perform SIFT feature extraction are performed on the GPU. Matching is also done on the GPU.

† On Intel Core2 6300 @ 1.86GHz 1.7GB RAM and 128MB GeForce Quadro NVS 150M (1 SM)
‡ On Intel Core2 Duo T8300 @ 2.4GHz 4GM RAM and 128MB GeForce 8400M GS (2 SM)
♦Includes preprocessing time
♦Success rate on 18 minutes of footage with 121 EU speed limit signs

Conclusions

Feature-based approach provides faster detection. However, it can only detect circular EU speed limit signs.

Template-based and SIFT-based approaches can recognize both EU and rectangular US speed limit signs.

In feature-based approach, increasing the number of examined candidates improves success rate, but increases the runtime.

CLAHE improves template-based success rate from 66% to 90%. Also, it eliminates all true negatives and false positives.

In template-based approach, having more composite filters with additional sizes and in-plane rotations improves success rate, but increases runtime. Success rate can also be improved by covering more out-of-plane rotations. This would also require additional filters (and hence increase runtime), since having more out-of-plane rotations in a single filter hurts its distinctiveness.

To improve the success rate of SIFT-based approach, image size is doubled, initial Gaussian blur is reduced, and DoG treshold is decreased (first two help recognizing small signs, third helps recognizing low contrast signs). However, success rate is still low since:
✓speed-limit signs lack complexity (small, no texture, simple shapes and constant color regions) → small number of features are extracted → often same number of matches are returned by different templates
✓we can't use CLAHE (creates too much noise that SIFT cannot handle)

SIFT-based approach has the slowest runtime since it consists of different stages which are computationally intensive and require large GPU memory.

Template-based and SIFT-based approaches are good/bad at:

	Template-based 👍	Template-based 👎
SIFT-based 👍	✓different weather and road conditions (E1, E2) ✓rejecting signs that have a dominant difference (E3, E4) ✓recognizing signs with insignificant modification (E5)	✓initially big signs (E14) ✓very rotated signs (E15) ✓partially occluded signs (E16)
SIFT-based 👎	✓small signs (E12). avoids failure due to viewpoint change when sign gets closer (E13) ✓lots of noise (E8,E9). makes CLAHE work (E10, E11) ✓recognition as a whole (E6,E7). bad when part of sign is occluded (E16)	✓very small signs (E17) ✓big part of the sign is missing (E18) ✓video is flickering (all white scene every other frame)

It is possible to combine different stages of alternative approaches and generate hybrid pipelines.

Overall speed-limit sign recognition algorithms are mostly data-parallel. Thus, GPU computing can be used to improve the runtimes. This allows real-time processing even on resource-constrained systems which cannot be achieved by CPU-based implementations.

References

[1] N. Barnes and A. Zelinsky. *Real-time radial symmetry for speed sign detection.* In IEEE Intelligent Vehicles Symposium (IV), pp. 566-571, June 2004.

[2] B. Javidi, M.-A. Castro, S. Kishk, and E. Perez. *Automated detection and analysis of speed limit signs.* Technical Report JHR 02-285, University of Connecticut, 2002.

[3] D. G. Lowe. *Distinctive image features from scale-invariant keypoints.* International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

[4] C. Wu. *SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT)*, http://cs.unc.edu/~ccwu/siftgpu, 2007.