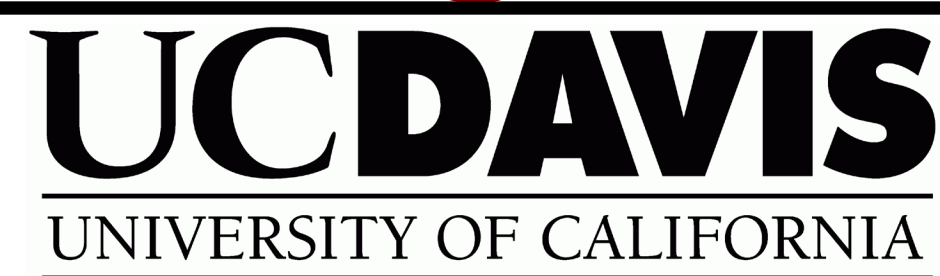


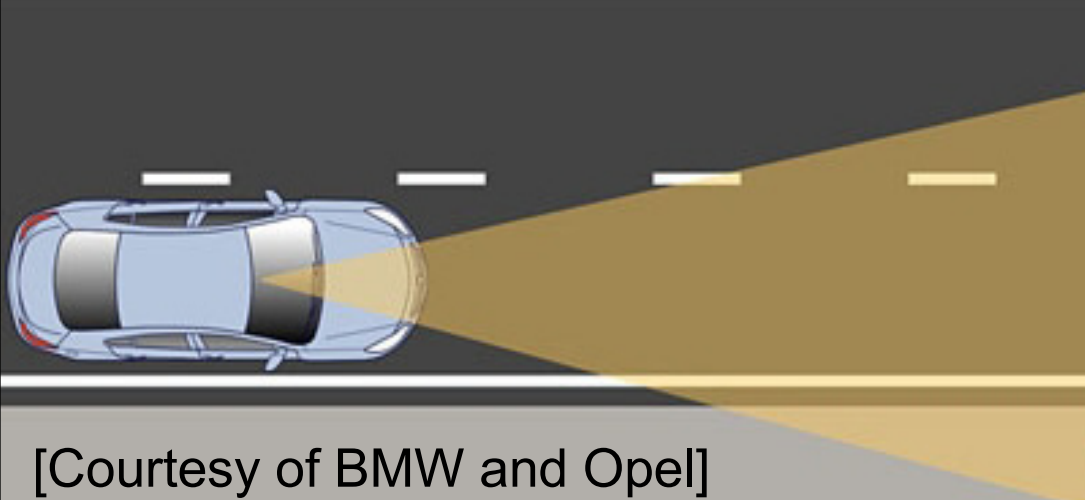
A Template-Based Approach for Real-Time Speed-Limit-Sign Recognition on an Embedded System Using GPU Computing



Pinar Muyan-Özçelik¹, Vladimir Glavtchev^{1,2}, Jeffrey M. Ota², John D. Owens¹
¹University of California, Davis, ²BMW Group Technology Office in Palo Alto



Goal



We perform **real-time** speed-limit-sign recognition on a **resource-constrained** embedded system.



Speed Limit: 60 km/h

Approach

To achieve our goal, we:

- ✓ use a **low-end GPU** (Graphics Processing Unit) as the main processing element
- ✓ pursue a **template-based** pipeline
- ✓ construct our pipeline from **modular components**

Why GPU?

- ✓ data-parallel nature of recognition task is an excellent fit for parallel GPU architecture
- ✓ offers superior price-performance and power-performance to comparable processors

- ✓ GPU computing allows us to revisit older but effective data-parallel techniques that have fallen out of favor due to their compute demands such as template-based object recognition

Why template-based recognition?

- ✓ using GPU we can parallelize its computation to provide real-time performance on an embedded domain
- ✓ it can be modified to recognize other objects such as other salient road features

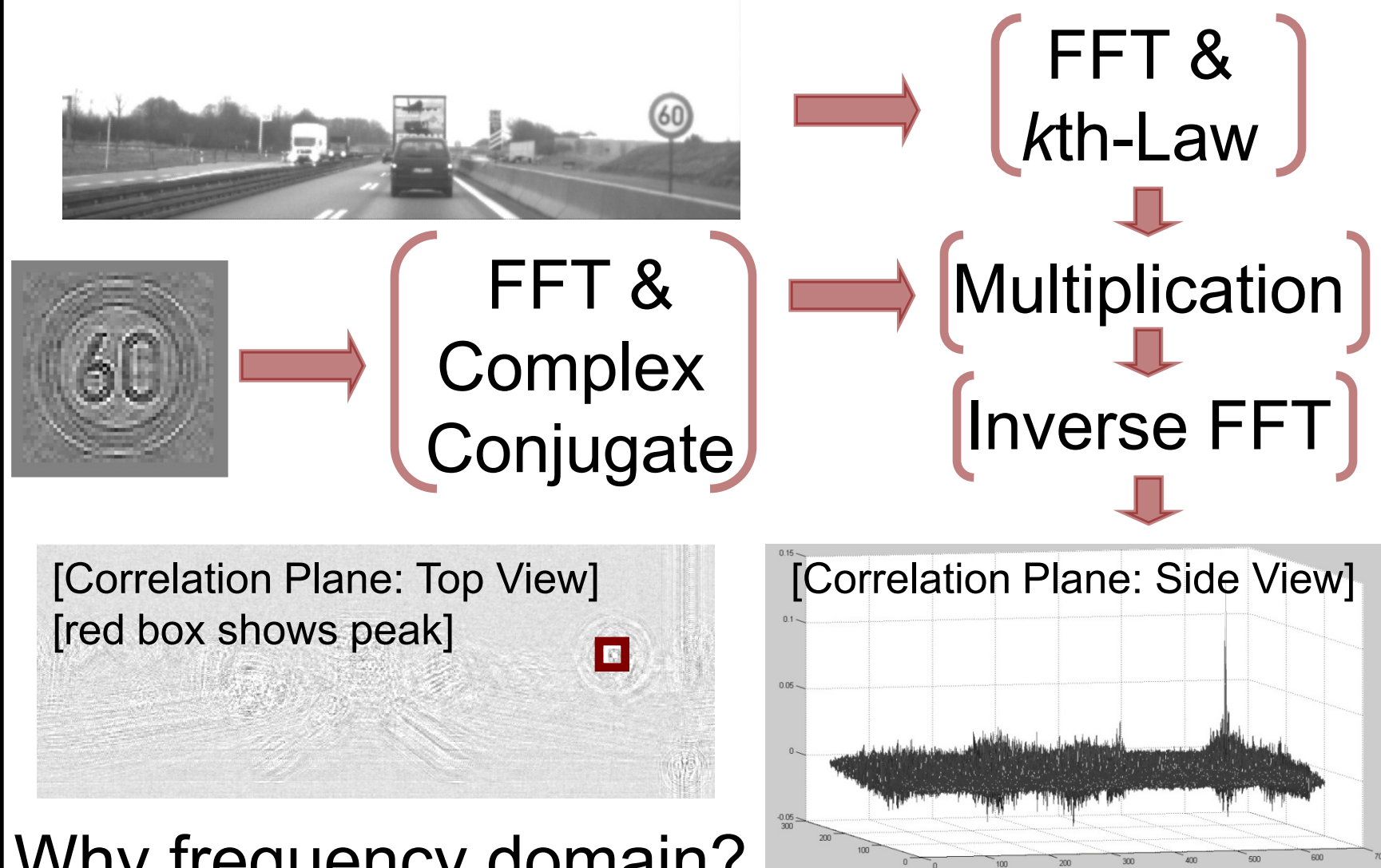
Why modular components?

- ✓ we can make the best use of the limited resources by fine-tuning the parameters of separate components based on the tradeoff between the runtime and success rate

Methods and Implementation

Template-based matching in frequency domain:

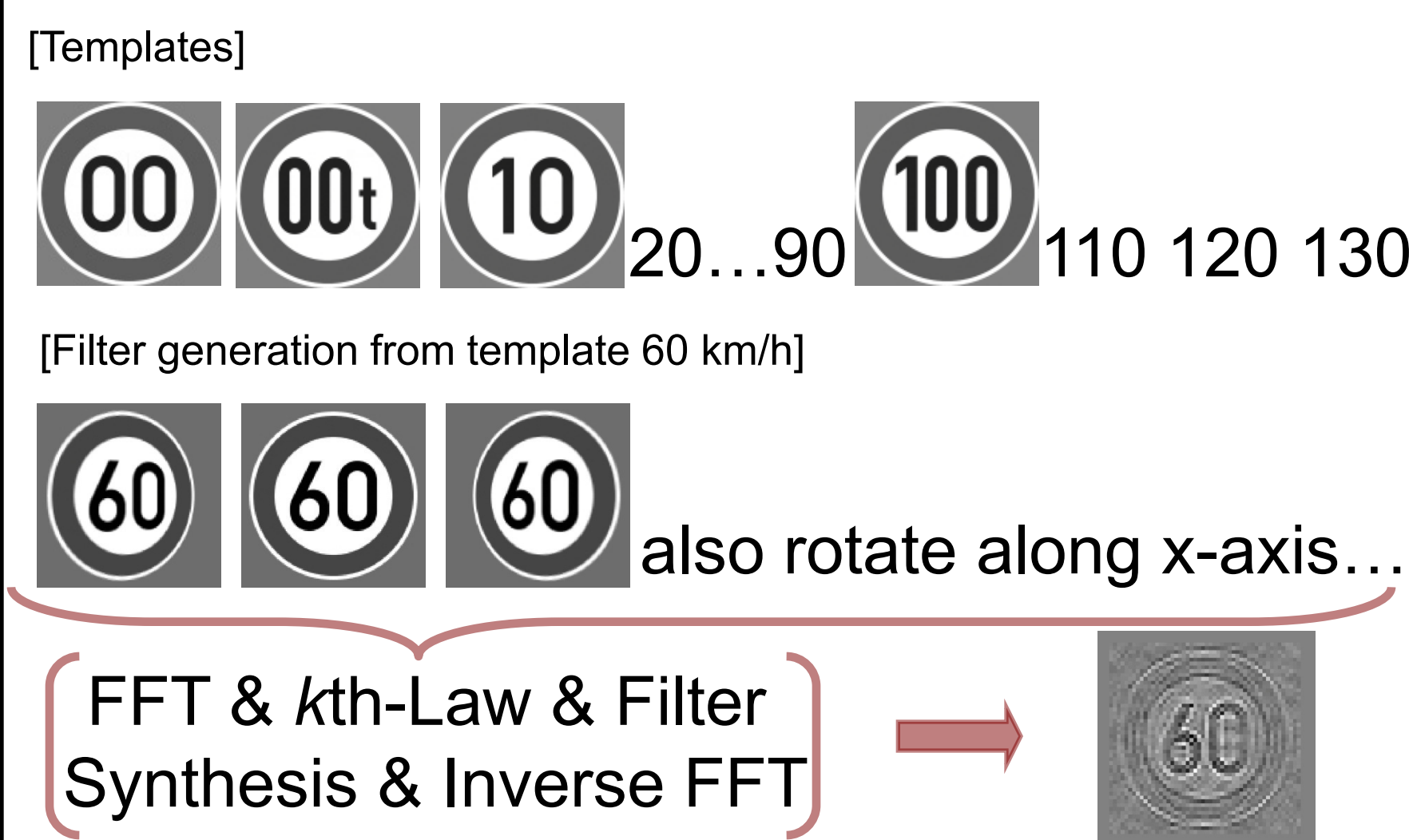
We compute **FFT correlations** between the scene and a k th-Law MACE (Minimum Average Correlation Energy [1]) composite filters:



Why frequency domain?

- ✓ provides a faster runtime than the approaches that perform convolution-type operations
- ✓ allows performing operations in the Fourier domain to improve the accuracy (e.g. k th-Law)

Generating k th-Law MACE composite filters from templates^[2]:



Why composite filters?

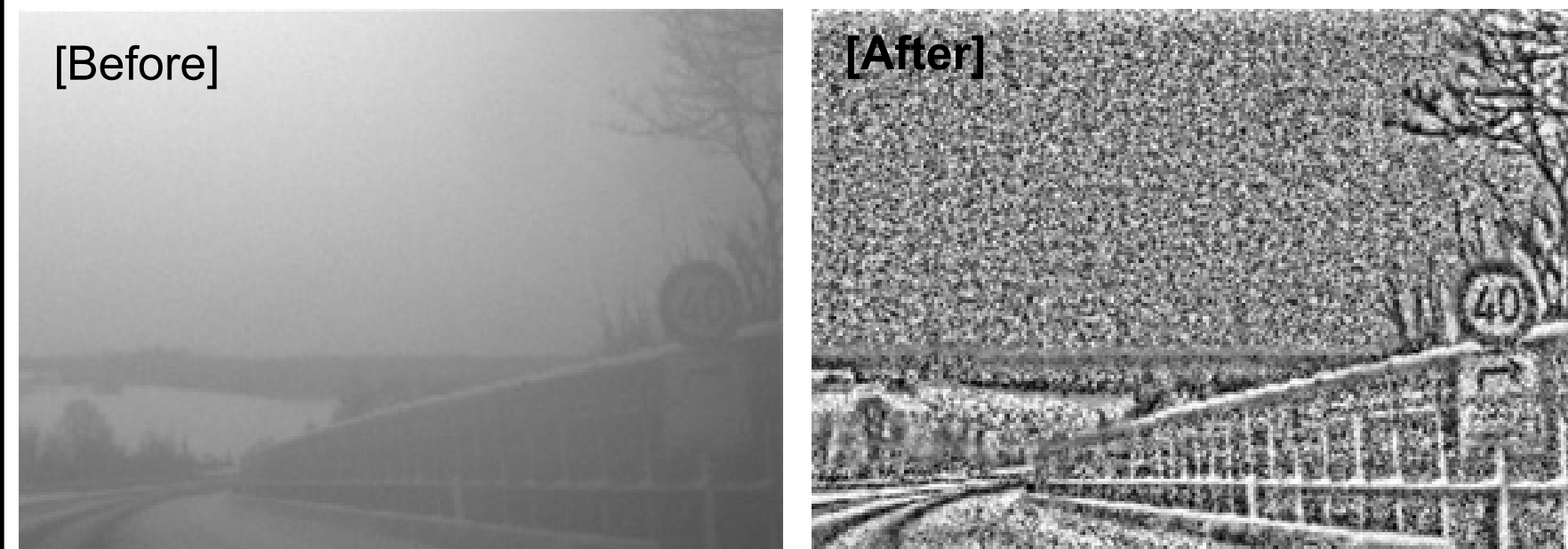
- ✓ reduces the number of correlations we need to perform and thus, provides faster runtime

Why k th-Law MACE composite filters?

- ✓ MACE filter provides a high discrimination ability against impostor objects which may look like a speed-limit sign
- ✓ k th-Law improves discrimination capability and illumination invariance of MACE filters

Applying CLAHE: (Contrast Limited Adaptive Histogram Equalization [3])

CLAHE improves recognition by enhancing the contrast of the scene.



Temporal integration with majority voting:

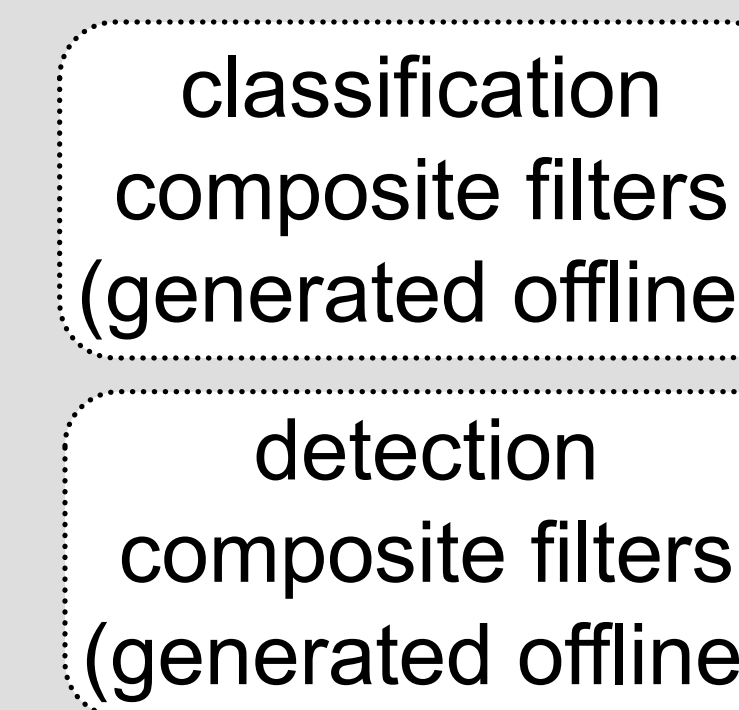
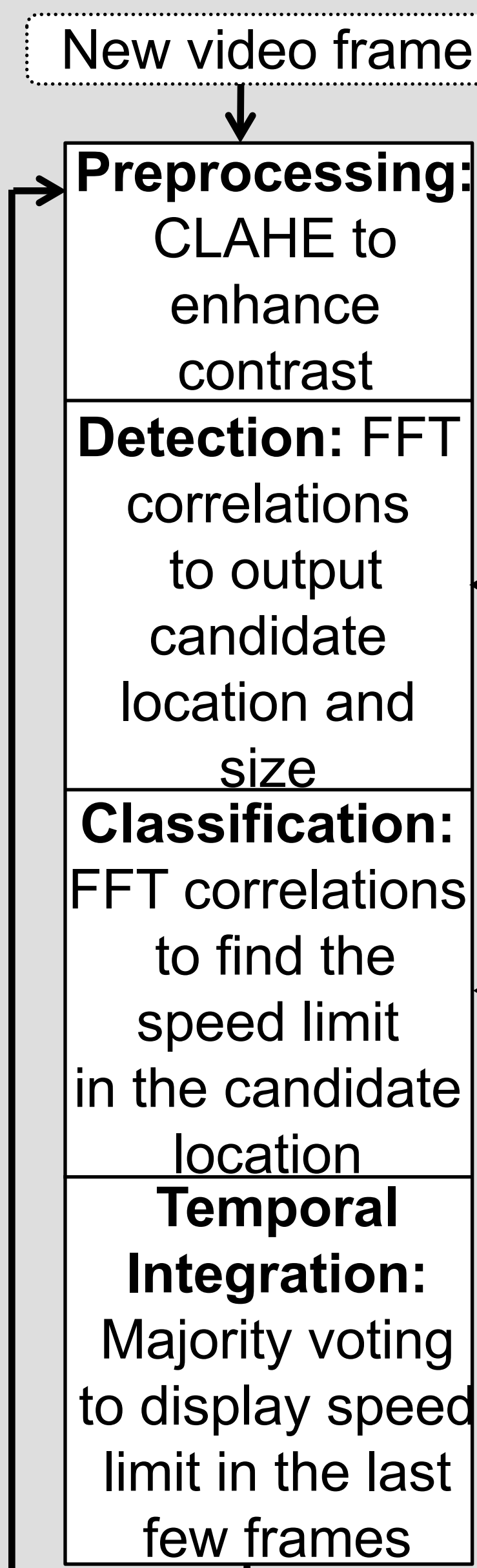
It increases the reliability of our results by accumulating the findings from the sequence of frames.

Frame	n	n+1	n+2	...	n+9	n+10
Sign	30	80	30		80	80
Confidence	9	10	9.5		14	13.5
Size	25	25	30		45	45
Rotation	-6°	0°	+6°		0°	0°
Total Vote	9	10	30.3		70	93.6

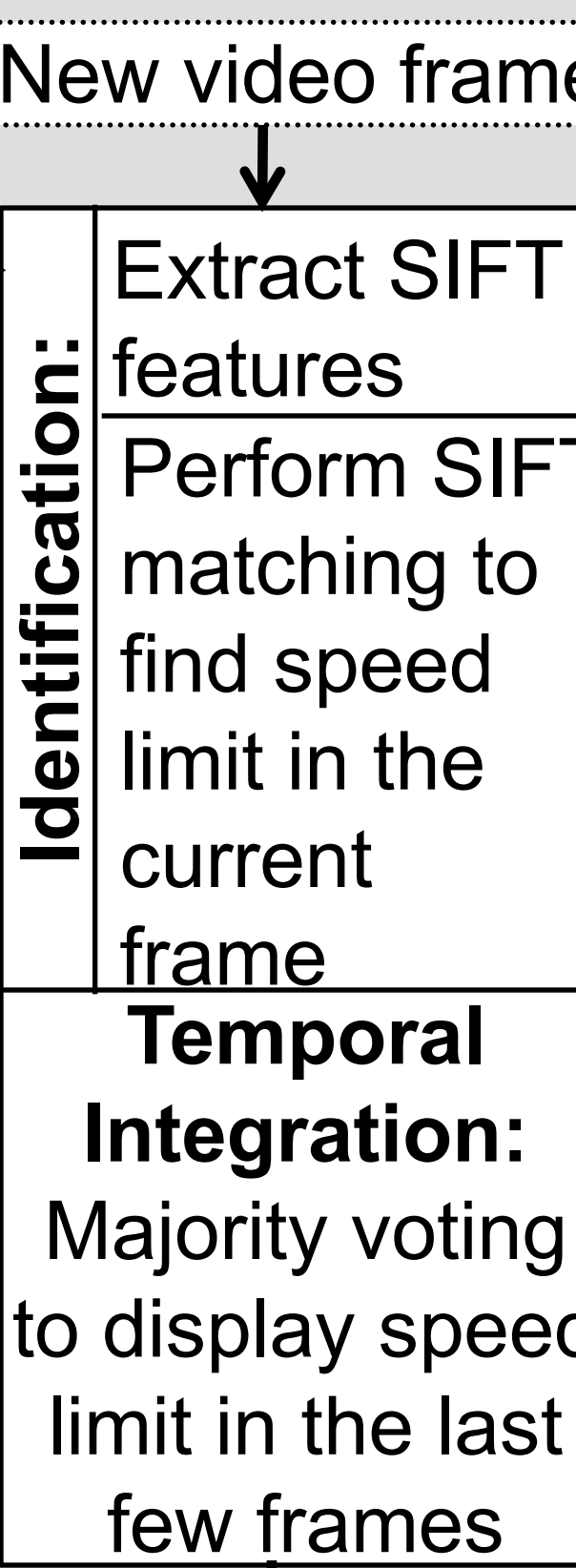
[We use a similar majority voting technique presented by Keller et al. [4]. The vote of the sign is increased by multiplying the confidence with a constant factor, when size is not decreasing or rotation stays the same]

Pipelines:

Template-based



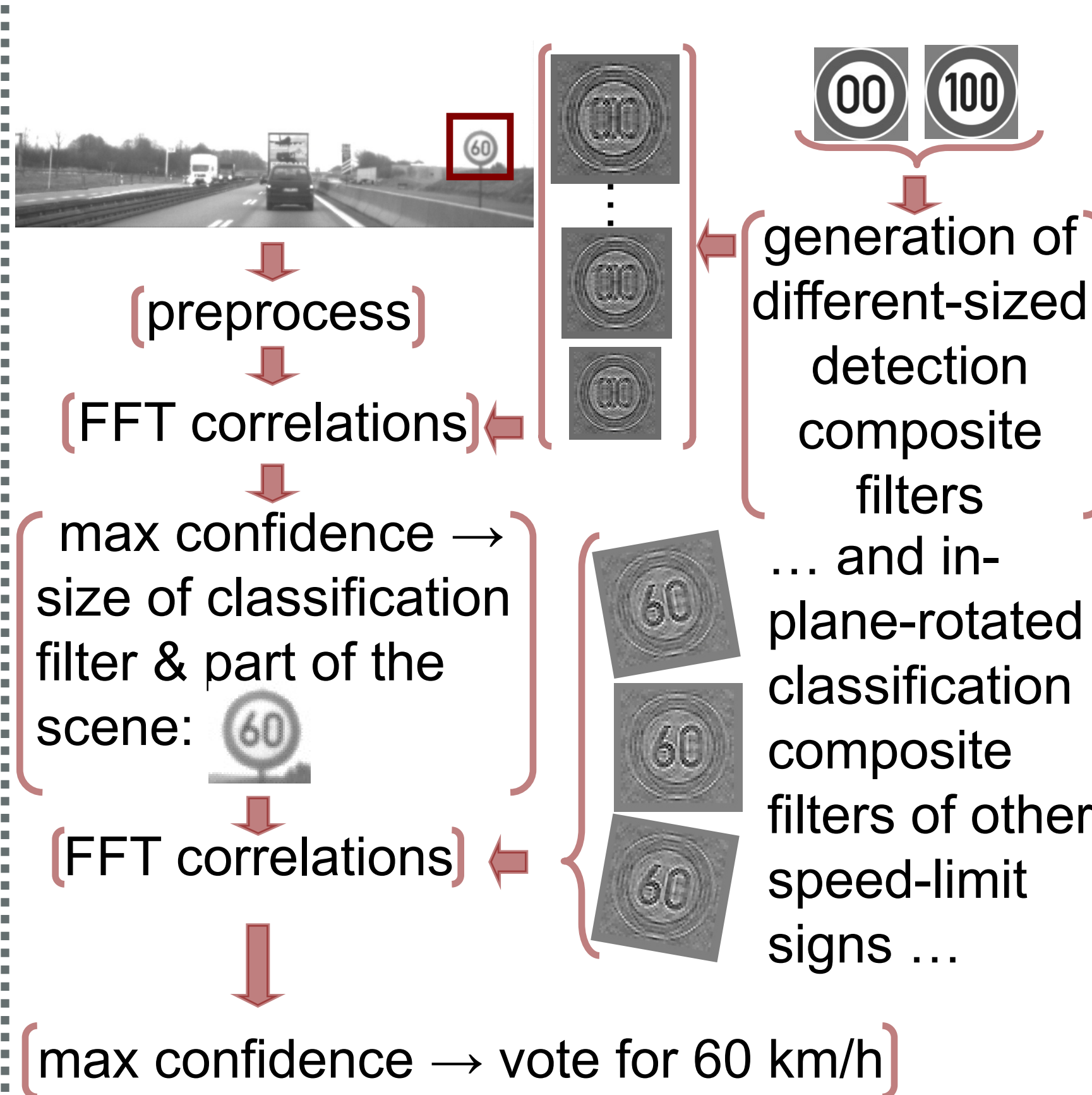
SIFT-based (Scale Invariant Feature Transform [5])



Why SIFT?

- ✓ Since SIFT is a commonly used method for performing object recognition, its GPU implementation provides a good basis for evaluating our approach.

Overview of template-based approach:



[Javidi et al.'s offline approach [2] is similar in spirit to the detection and classification stages of our template-based approach]

GPU mapping of the implementations:

For the template-based pipeline, we use:

- ✓ CUFFT library to take inverse/forward FFTs
- ✓ CUDA kernels to apply k th-Law nonlinearity to the FFT of the scene, to take the complex conjugate of the composite filter in the frequency domain, to multiply these two FFTs, and to normalize the result of this product
- ✓ CUDA kernel to find the peak in the correlation plane by performing a reduction operation

For the SIFT-based pipeline, we use:

- ✓ SiftGPU^[6], an open source GPU implementation of SIFT, for feature extraction and matching

Results

Approach	Runtime [†]	Success Rate [§]
Template-based	18.5 fps	90% with no MC & FP
SIFT-based	~ 8 fps (offline run)	75% with 4 MC & 9 FP

MC = Misclassifications FP = False Positives

[†] On Intel Core2 Duo P8600 2.4 GHz CPU and a GeForce 9600M GT GPU

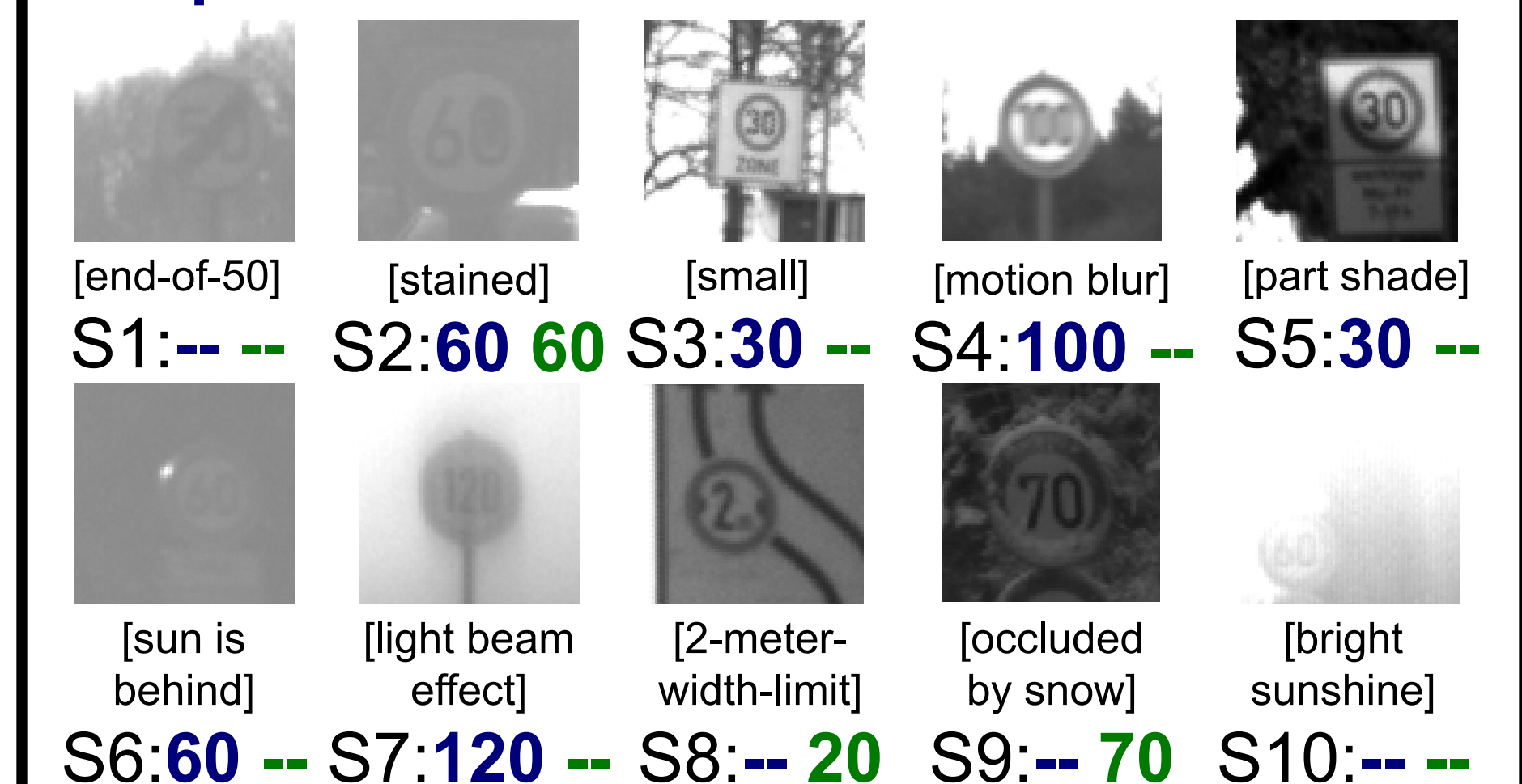
[§] Test data include grayscale videos recorded in different weather/road conditions in Europe. The footage captures 45 minutes of driving that includes 120 EU speed-limit signs

Why does SIFT have lower performance?

- ✓ it has computationally expensive stages
- ✓ it works best when objects have some complexity, but speed-limit signs are simple
- ✓ cannot handle noise introduced by CLAHE (in the template-based approach, CLAHE increased success rate from 65% to 90% and eliminated all MC and FP)

Scene Examples:

Template-based and SIFT-based results:



	Template (good at)	Template (bad at)
SIFT (good at)	<ul style="list-style-type: none"> ✓ rejecting signs that have a dominant difference (S1) ✓ recognizing signs with insignificant modification (S2) 	<ul style="list-style-type: none"> ✓ initially big or very rotated signs ✓ partially occluded signs (S9)
SIFT (bad at)	<ul style="list-style-type: none"> ✓ small signs (S3) ✓ lots of noise (S4-S5), allows CLAHE to work (S6-S7) ✓ recognition as a whole (S8) 	<ul style="list-style-type: none"> ✓ very small signs ✓ big part of the sign is missing (S10)

Tuning modules with available compute:

In template-based approach, if we have less compute power, we can:

- ✓ process only the smaller sizes of signs and keep the frame rate high

With more compute power, we can:

- ✓ increase the frame rate
- ✓ add larger sizes of signs
- ✓ generate additional composite filters with larger out-of-plane rotations

References

- [1] A. Mahalanobis, B.V.K. V. Kumar, and D. Casasent. *Minimum average correlation energy filters*. Applied Optics, 26(17), pp. 3633-3640, 1987.
- [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] K. Zuiderveld. *Contrast limited adaptive histogram equalization*. Graphics Gems IV, pp. 474-485, 1994.
- [4] C.G. Keller, C. Sprunk, C. Bahlmann, J. Giebel, G. Barattoff. *Real-time recognition of U.S. speed signs*. IEEE Intelligent Vehicles Symposium, pp. 518-523, 2008.
- [5] D. G. Lowe. *Distinctive image features from scale-invariant keypoints*. International Journal of Computer Vision, 60(2), pp. 91-110, 2004.
- [6] C. Wu. *SiftGPU: A GPU Implementation of Scale Invariant Feature Transform*, http://cs.unc.edu/~ccwu/siftgpu, 2007.