21^{st} *Computer Vision Winter Workshop*
Luka Čehovin, Rok Mandeljc, Vitomir Štruc (eds.)
Rimske Toplice, Slovenia, February 3–5, 2016

Hessian Interest Points on GPU

Jaroslav Sloup, Michal Perďoch, Štěpán Obdržálek, Jiří Matas
Center for Machine Perception
Czech Technical University Prague
sloup|perdom1|xobdrzal|matas @fel.cvut.cz

Abstract.

This paper is about interest point detection and GPU programming. We take a popular GPGPU implementation of SIFT – the de-facto standard in fast interest point detectors – SiftGPU and implement modifications that according to recent research result in better performance in terms of repeatability of the detected points. The interest points found at local extrema of the Difference of Gaussians (DoG) function in the original SIFT are replaced by the local extrema of determinant of Hessian matrix of the intensity function.

Experimentally we show that the GPU implementation of Hessian-based detector (i) surpasses in repeatability the original DoG-based implementation, (ii) gives result very close to those of a reference CPU implementation, and (iii) is significantly faster than the CPU implementation. We show what speedup is achieved for different image sizes and provide analysis of computational cost of individual steps of the algorithm.

The source code is publicly available.

1. Introduction

A viewpoint-independent representation of objects in images is one of the fundamental problems in computer vision. A popular approach is to extract a set of local measurements, known as descriptors, at a sparse set of image locations. These locations are called interest points and their purpose is (as opposed to dense image sampling) to reduce the spatial domain of further computation, hence reducing the cost to obtain, and memory requirements to store, the image representation.

It follows that for an interest point extraction process to be practical it needs to repeatedly identify the same points on object surface when the viewpoint or environment (*e.g.* illumination) change. Establishing correspondences between interest points representing an object in multiple images is a building step for a multitude of computer vision tasks, including stereo or multi-view reconstruction, object recognition, and image search and retrieval.

These are the desirable qualities for which interest point detectors are evaluated:

- **Transformation Covariance**. The detected points should correspondingly 'follow' the object as it is depicted from different viewpoints. This paper concerns similarity-covariant detectors which follow 2D image locations (objects at different positions in the image), scales (objects at different distances) and 2D orientations (in-plane rotation of the objects). Affine detectors, which additionally follow out-of-plane 3D rotations, are not considered here.
- **Repeatability** of detected interest points. The percentage of the points detected at corresponding image locations when the viewpoint changes.
- **Accuracy** with which the interest points are located and their scales and orientations are estimated.
- **Coverage** of various visually different classes of objects.
- **Robustness** under image degradation – noise, motion blur, compression, out of focus images, etc.
- **Detection Speed**, the computational cost of the interest points detection.

One of the most popular interest point detection algorithms is still the Scale-Invariant Feature Transform (SIFT) proposed by David Lowe [8] in 2004.

It consistently ranks high on benchmarks in quality of detected points, but is computationally expensive, therefore unsuitable *e.g.* for real-time video processing. Many speedier approximations and alternatives were proposed, *e.g.* SURF [2], FAST [11] and ORB [12], or CenSurE [1] and SUSurE [3], which can detect interest points significantly faster than SIFT. But often at the expense of repeatability and accuracy.

The only widely used detector that in most tests scores higher in repeatability than SIFT is the so-called Hessian detector. In SIFT, points are identified at local minima or maxima of the Difference of Gaussians function, thence in presence of blob-like local image structures. In the Hessian detector, the points are located where the determinant of the Hessian matrix (a matrix of second-order partial derivatives) attains local extrema. Which occur either for blob-like (local maxima) or for saddle-like structures (local minima). Experiments show that the extrema of the determinant of Hessian are more repeatable and accurate than the extrema of the Difference of Gaussians, and, thanks to the additional detection of saddle points, the object coverage is generally also improved. The detection speed of Hessian is similar to that of the SIFT.

Taking advantage of the recent widespread availability of programmable graphic cards, execution time of many computer vision algorithms benefits if reimplemented for GPUs. Interest point detectors are no exception, a GPGPU (general-purpose GPU) SIFT implementation is available from [15, 14, 4]. The SIFTs are detected in real-time for moderately sized videos or images on a consumer-grade GPU, therefore there is now a large group of applications for which it is no longer necessary to sacrifice detection quality for execution speed.

We build upon the available GPU SIFT implementation [15] and extend it with several contributions. The Difference of Gaussians is replaced with the determinant of the Hessian matrix as the function of which extrema indicate presence of interest points. This improves repeatability, and coverage, of the detected points, as is experimentally demonstrated below. Selection of best K points (when ordered by magnitude of the determinant) is implemented in an early stage of the algorithm. If only a specific number of points is requested, it is faster to decide which these are early, on the GPU, before orientations are determined and descriptors computed. Additionally, the feature type (saddle, dark or white blob) is now part of the GPU code output. This is useful in follow-up matching – features of different types should not be considered for a correspondence.

Some of the functionality that was available in the original CPU SIFT implementation and omitted in the GPU version was reintroduced. We add the optional capability to compute orientations and descriptors only in $\langle 0, \pi \rangle$ range instead of $\langle 0, 2\pi \rangle$ by disregarding sign of the gradients involved, which is beneficial when matching images taken under significantly different illumination (day and night). The restriction that at each image location only at most two interest point orientations are detected was lifted. And the maximal number of iterations used for sub-pixel localization of a detected point is now configurable, the original Sift-GPU code allowed only a single iteration.

In the rest of the paper we quickly describe the SIFT detector and explain the relations and differences between the Laplacian operator, the Difference of Gaussians and the determinant of the Hessian matrix (Section 2). In Section 3 we sketch the GPU implementation and analyze the computational cost of individual components. Experiments in Section 4 show that the Hessian indeed achieves better performance than original SIFT and that the GPU and CPU implementations of Hessian give very similar results.

2. Laplacian of Gaussian, Difference of Gaussians and Determinant of Hessian Matrix

Let us consider a grayscale image to be a discretized form of an underlying real-valued continuous function $f(x, y) : \mathbb{R}^2 \to \mathbb{R}$. Its *Gaussian scale-space* representation $L(x, y, t) : \mathbb{R}^3 \to \mathbb{R}$ is then defined as

$$L(x, y; t) = g(x, y, t) * f(x, y)$$

where

$$g(x, y, t) = \frac{1}{2\pi t} e^{-\frac{x^2+y^2}{2t}}$$

is a rotationally symmetric 2D Gaussian kernel parametrized by variance $t = \sigma^2$, and where $*$ denotes convolution. Partial *Gaussian derivatives* of the image at a given scale t are then written as

$$\begin{aligned} L_{x^\alpha y^\beta}(\cdot, \cdot, t) &= \partial_{x^\alpha y^\beta} L(\cdot, \cdot, t) \\ &= (\partial_{x^\alpha y^\beta} g(\cdot, \cdot, t)) * f(\cdot, \cdot). \end{aligned}$$

The Hessian matrix for a given t is a square matrix

of second-order partial derivatives

$$\mathbf{H} = \begin{pmatrix} \dfrac{\partial^2 (f * g)}{\partial x^2} & \dfrac{\partial^2 (f * g)}{\partial x \, \partial y} \\ \dfrac{\partial^2 (f * g)}{\partial x \, \partial y} & \dfrac{\partial^2 (f * g)}{\partial y^2} \end{pmatrix} = \begin{pmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{pmatrix}.$$

Let λ_1 and λ_2 denote the eigenvalues of the Hessian matrix. Laplacian (or the Laplace operator, the sum of second partial derivatives) of the Gaussian is then

$$\nabla^2 L = L_{xx} + L_{yy} = \lambda_1 + \lambda_2.$$

The Laplacian of Gaussian, appropriately normalized for different scales [7], is a basis for one of the first and also most common detector of blob-like interest points. Local scale-space extrema are detected that are maxima/minima of $\nabla^2 L$ simultaneously with respect to both space (x, y) and scale t [5]. In discrete domain, interest points are detected if the value of $\nabla^2 L$ at this point is greater/smaller than all values in its 26-neighbourhood. Locations of such points are covariant with translations, rotations and rescaling in the image domain. If a scale-space maximum is found at a point $(x_0, y_0; t_0)$ then after a rescaling of the image by a scale factor s there will be a corresponding scale-space maximum at $(sx_0, sy_0; s^2 t_0)$ [6].

The Laplacian of the Gaussian operator $\nabla^2 L(x, y, t)$ can be approximated [7] with a difference between two Gaussian-smoothed images at different scales t and $t + \Delta t$

$$\nabla^2 L(x, y; t) \approx \frac{t}{\Delta t} \left(L(x, y; t + \Delta t) - L(x, y; t) \right).$$

This approach is referred to as the Difference of Gaussians (DoG). In fashion similar to the Laplacian detector, interest points are detected as extrema in the 3D scale-space. The Difference of Gaussian is used in the SIFT algorithm [8].

Another differential interest point detector is derived from the determinant of the Hessian matrix $\mathbf{H}$

$$\det \mathbf{H} L(x, y; t) = (L_{xx} L_{yy} - L_{xy}^2) = \lambda_1 \lambda_2.$$

At image locations where the determinant is positive the image contains a blob-like structure. The Hessian matrix will there either be positive or negative definite, indicating presence of either bright or dark blobs. If the determinant of the Hessian matrix is negative, the matrix is indefinite, which indicates a saddle-like interest point [5].

The determinant of the Hessian operator has better scale selection properties under affine image transformations than the Laplacian operator or its Difference-of-Gaussians approximation [7]. It was also shown to perform significantly better for image-based matching using local SIFT-like or SURF-like image descriptors, leading to higher efficiency and precision scores [7]. In an approximation computed from Haar wavelets it is the basis for the interest point detector in SURF [2].

3. GPU Implementation and Computation Time Analysis

The GPU interest point implementation proceeds in steps shown in Figure 1. First, the input image is loaded and transferred to a GPU texture. The scale pyramid data structures, which make up the majority of the GPU memory required, are allocated once at the beginning, and reallocated only in case a bigger image is eventually processed later. The allocation typically takes several hundreds of milliseconds. Initial image upscaling by a factor of two, which is sometimes used in feature detection, is not performed. The scale space pyramid is then filled – a process that involves smoothing with Gaussian kernels with multiple std. deviations. Keypoints are detected as scale-space extrema of the determinant of the Hessian matrix and their locations are collected to a linear list. Optionally, the points are ordered by the response (the absolute value of the determinant) and only the top K points are kept for further processing. Keypoint orientations are then determined, with approximately 20% of the points ending with two or more orientations assigned. The points, now with the orientations, are again collected to a list and SIFT descriptors are computed.

Figure 2 shows the execution speed measured on three GPU cards. The photo shown on left, which represents a typical picture used in large-scale image retrieval tasks, was resized to eight different resolutions. Three CUDA-enabled graphics cards were tested: NVidia GeForce GT 730M (384 CUDA cores in 2 streaming multiprocessors, 1024MB DDR3 memory, 64-bit bus) is a representative of a common mobile/laptop GPU. NVidia GTX 750Ti (640 CUDA cores in 5 SMs, 2048MB GDDR5 memory, 128-bit bus) represents a gaming desktop card, and NVidia GTX Titan Black (2880 CUDA cores in 15xSMs, 6144MB GDDR5 memory, 384-bit bus) is a server card. Additionally, execution times of the reference

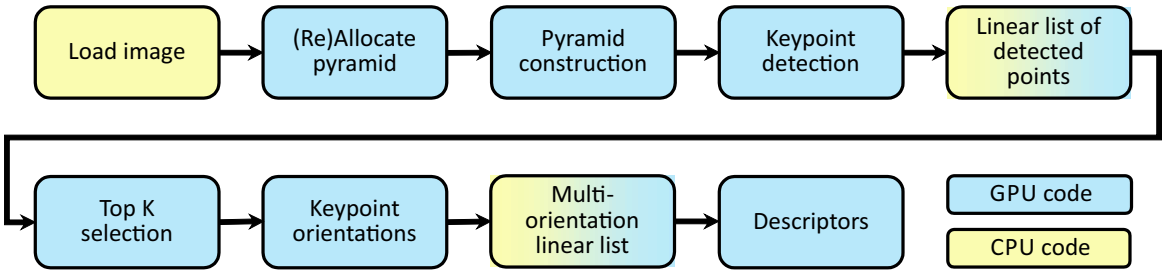
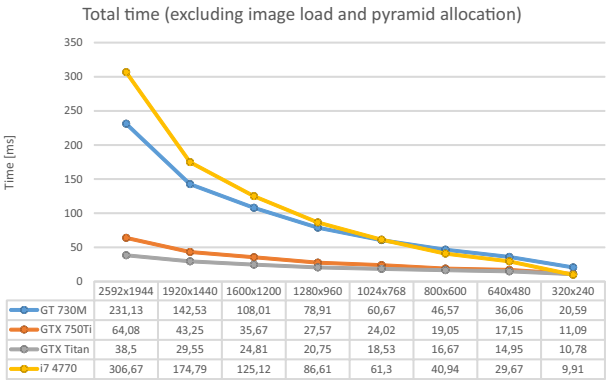


Figure 1. Block diagram of the computation pipeline. CPU code shown in yellow, GPU code in blue.



	2592x1944	1920x1440	1600x1200	1280x960	1024x768	800x600	640x480	320x240
GT 730M	231,13	142,53	108,01	78,91	60,67	46,57	36,06	20,59
GTX 750Ti	64,08	43,25	35,67	27,57	24,02	19,05	17,15	11,09
GTX Titan	38,5	29,55	24,81	20,75	18,53	16,67	14,95	10,78
i7 4770	306,67	174,79	125,12	86,61	61,3	40,94	29,67	9,91

Figure 2. Detection time for a test image (left) at eight different resolutions (right). Three GPUs were measured, together with a reference CPU implementation.

CPU implementation running on a current desktop CPU (i7 4470) are reported. While the mobile GPU is only slightly faster than the CPU, the other two GPU cards are roughly five and eight times faster.

Figure 3 shows a break down of load distribution over individual stages of the keypoint detection process (refer to Fig. 1). The analysis is shown for the desktop (left) and the mobile (right) GPUs. While the desktop card is about five times faster, the proportional distribution of the load is very similar.

Comparing the execution speed of the original Sift-GPU implementation (using the Difference of Gaussians) with our Hessian-based detector, see Fig. 4, we observe that the quality improvement demonstrated below in Experiments comes at no additional computational cost.

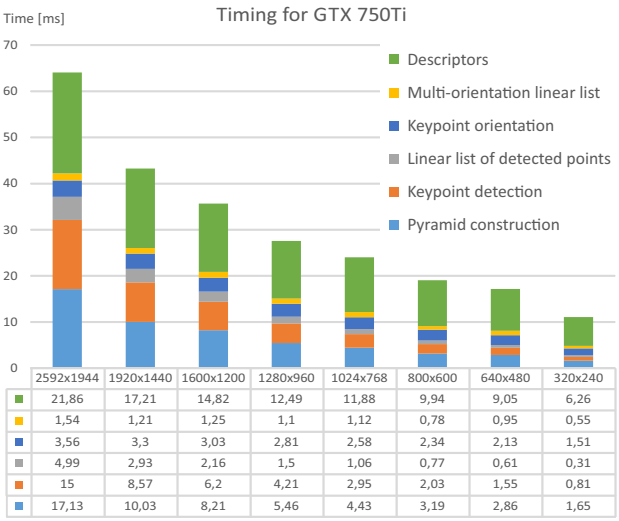
Finally in Figure 5 we show the timing when requesting only the best K keypoints. As expected, the stages preceding the top K selection are not affected. The stages following, orientation estimation and computation of the descriptor, take longer for more keypoints, although the increase is sublinear until the GPU processing power is saturated at around 8000 descriptors computed in parallel.

4. Experiments

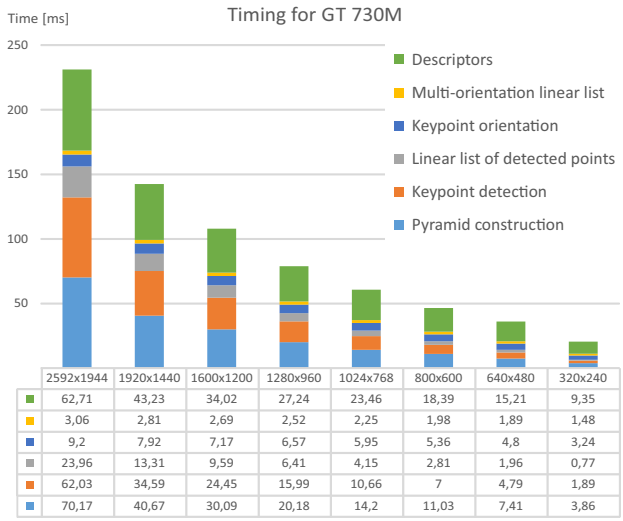
The performance of the proposed GPGPU implementation of the determinant-of-Hessian detector was compared with other publicly available detectors, based on the Difference of Gaussians, multiscale Laplacian and the determinant of the Hessian matrix. In particular, we have evaluated Lowe's[8] original version of SIFT and its VLFeat re-implementation, CPU implementation of the Hessian and the Laplacian, and the original GPU code of SiftGPU. SURF detector [2], which is based on a fast approximation of the Hessian matrix, is also included. Two sets of experiments are presented: first one evaluating transformation invariance of the detectors in terms of repeatability and the number of correspondences, second one evaluating performance in a retrieval system.

4.1. Parameter Setting

One of the advantages of the determinant of Hessian based detector is in responding to an additional type of local features – saddle points [6]. In our initial experiments on a large set of images, we observed that the number of saddle points in natural images is about the same as the number of bright and dark blobs together. Therefore the Hessian gives roughly twice as many points as the Laplacian/DoG

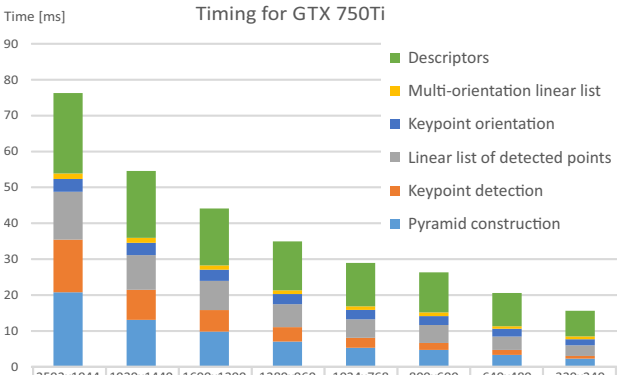


	2592x1944	1920x1440	1600x1200	1280x960	1024x768	800x600	640x480	320x240
Descriptors	21,86	17,21	14,82	12,49	11,88	9,94	9,05	6,26
Multi-orientation linear list	1,54	1,21	1,25	1,1	1,12	0,78	0,95	0,55
Keypoint orientation	3,56	3,3	3,03	2,81	2,58	2,34	2,13	1,51
Linear list of detected points	4,99	2,93	2,16	1,5	1,06	0,77	0,61	0,31
Keypoint detection	15	8,57	6,2	4,21	2,95	2,03	1,55	0,81
Pyramid construction	17,13	10,03	8,21	5,46	4,43	3,19	2,86	1,65



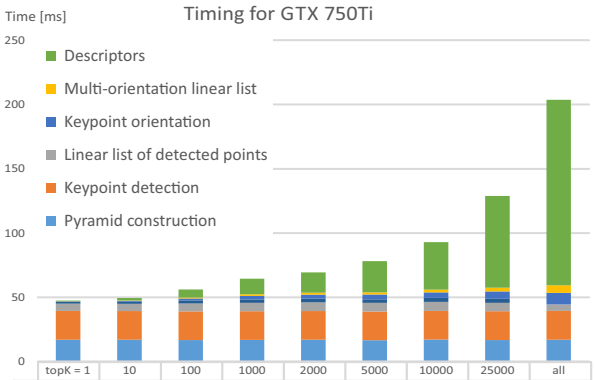
	2592x1944	1920x1440	1600x1200	1280x960	1024x768	800x600	640x480	320x240
Descriptors	62,71	43,23	34,02	27,24	23,46	18,39	15,21	9,35
Multi-orientation linear list	3,06	2,81	2,69	2,52	2,25	1,98	1,89	1,48
Keypoint orientation	9,2	7,92	7,17	6,57	5,95	5,36	4,8	3,24
Linear list of detected points	23,96	13,31	9,59	6,41	4,15	2,81	1,96	0,77
Keypoint detection	62,03	34,59	24,45	15,99	10,66	7	4,79	1,89
Pyramid construction	70,17	40,67	30,09	20,18	14,2	11,03	7,41	3,86

Figure 3. Execution time of individual stages of the computation pipeline (refer to Fig. 1), evaluated at several image resolutions, with a default threshold on the detector response. The desktop GPU is about five times faster than the mobile GPU, but the relative load distribution between individual stages is virtually identical. Also the relation of the execution speed and image resolution is similar.



	2592x1944	1920x1440	1600x1200	1280x960	1024x768	800x600	640x480	320x240
Descriptors	22,4	18,61	15,85	13,6	12,09	11,16	9,24	7,13
Multi-orientation linear list	1,53	1,4	1,18	1,03	0,98	1,06	0,74	0,78
Keypoint orientation	3,57	3,43	3,14	2,84	2,65	2,48	2,17	1,69
Linear list of detected points	13,37	9,63	8,13	6,41	5,16	4,99	3,69	2,98
Keypoint detection	14,67	8,37	5,97	4,02	2,77	1,95	1,4	0,75
Pyramid construction	20,76	13,12	9,85	7,04	5,32	4,71	3,34	2,29

Figure 4. Execution time of the original Sift-GPU code, evaluated at several image resolutions. Compare to the timing of our Hessian-based detector on the same hardware (Fig. 3 left). The improved qualitative performance, demonstrated in Section 4, comes with a negligible computational cost.



	topK = 1	10	100	1000	2000	5000	10000	25000	all
Descriptors	0,48	1,84	6,18	12,15	15,75	24,27	36,89	71,36	144,27
Multi-orientation linear list	0,11	0,29	0,8	1,23	1,5	1,69	2,17	2,82	5,98
Keypoint orientation	0,22	0,54	1,5	2,72	3,17	3,75	4,26	5,79	8,91
Linear list of detected points	1,63	1,81	2,35	2,65	2,85	2,78	3,12	3,08	0
Keypoint detection	5,62	5,68	6,15	6,48	6,76	6,72	7,01	6,63	5,07
Pyramid construction	22,33	22,29	22,32	22,34	22,35	22,34	22,32	22,3	22,46
	17,11	17,05	16,79	16,94	17,02	16,63	17,18	16,87	17,01

Figure 5. Execution times when a limited number of K best points is requested. Computed on the full size 2592x1944 image without a threshold on detector response. As expected, the processing time of the steps preceding the top K selection are not affected, while the later steps, most importantly the computation of the descriptor, scale with the number of points requested.

detectors, if detector configurations and thresholds are kept the same. To take an advantage of these additional points while keeping the representations comparable in size for the experiments, the detected points in each image were ordered by the absolute response value of the detector and the best 1000, 2000 and 4000 points were selected for evaluation. Finally, to diminish the slight differences in detection of dominant orientation, the orientations were fixed to vertical in the retrieval experiment, and were not used in the detector repeatability experiment.

4.2. Datasets and Evaluation Protocols

A standard benchmark protocol and dataset for evaluation of covariant interest point detectors was proposed by Mikolajczyk et al. [9]. It consists of eight sets, each of six images, with an increasing effect of image distortions: camera viewpoint, image scale, isotropic blur, underexposure and image

compression. We have selected one scene with each distortion. Ground truth transformations are known, relating reference images of each set to all other images in that set. The transformations are used to compute repeatability scores by considering the *overlap error* ϵ of all pairs of detected points:

$$\epsilon(R_{\mathbf{E}_1}, R_{\mathbf{E}_2}) = 1 - \frac{R_{\mathbf{E}_1} \cap R_{\mathbf{H}_{12}^\top \mathbf{E}_2 \mathbf{H}_{12}^{-1}}}{R_{\mathbf{E}_1} \cup R_{\mathbf{H}_{12}^\top \mathbf{E}_2 \mathbf{H}_{12}^{-1}}},$$

where $R_{\mathbf{E}}$ represents the elliptic region defined by $x^\top R_{\mathbf{E}} x = 1$ and $\mathbf{H}_{12}$ is the known homography between the reference 1 and the test image 2. To compensate for different sizes of regions from different detectors, a scale factor is applied such that a region $R_{\mathbf{E}_1}$ is transformed to a normalized size (equivalent to a radius of 30 pixels). Before evaluating the overlap error, region $R_{\mathbf{E}_2}$ is scaled using the same factor.

The image retrieval performance was tested using the Oxford buildings dataset and protocol defined by Philbin et al. [10]. In short, five queries are defined for each of eleven landmarks in Oxford, and a ground truth shortlist of positive examples is given. For each query an average precision (AP) is computed as the area below the precision-recall curve. Finally, a mean AP (mAP) is reported for the whole set of 55 query images.

4.3. Evaluation of detectors Repeatability

Repeatability is of one of the important properties of interest point detectors. It is a measure that approximates the probability of the point redetection given the distortion between images. The detectors should be configured to provide comparable numbers of points to make the assessment fair. The repeatability score is complemented with the absolute number of corresponding points detected – the predicted upper bound of the matching problem. Figures 6, 7 and 8 show the measured scores when the number of detected points was limited to 1000, 2000, and 4000 respectively.

We observe that all the three DoG-based detectors (original Lowe's, from VLFeat and SiftGPU) perform virtually the same, as do the two Hessian-based detectors (CPU and GPU implementations). This strongly indicates that the measured performance is indeed inherent of the methods and not of a particular implementation. We also see that the Hessian performs in most cases better than the Laplacian and its DoG approximation. The additionally detected saddle points complement the blobs well and provide valuable correspondences between the images. With the exception of image blur, the fast but approximate SURF performs slightly worse than the other methods.

4.4. Evaluation in Image Retrieval

The repeatability of a detector predicts its pairwise matching potential. To assess the discrimination ability of a coupling of a detector (DoG, Laplacian, or Hessian) with a descriptor (SIFT), a large-scale image retrieval experiment was performed. The Oxford building dataset with about 5000 images was used. Each detector was again run in three configurations, requesting at most 1000, 2000, resp. 4000 best interest points. The SIFT descriptor was computed from a local neighborhood around each point and stored. As there are no significant orientation changes in the dataset, orientation of the interest points was fixed as vertical in this experiment. The measurement region sizes – radius of the SIFT index w.r.t. the detected scale of a interest point – were kept on their default values: 6.0 for DoG detectors (Lowe, VLFeat), 5.2 for Laplacian and CPU and GPU Hessian. The reasoning behind this is that the DoG detectors return slightly smaller (5-10%) intrinsic scale, determined by the smaller of the two subtracted Gaussians.

A standard Bag of Words (BoW) approach with and without Spatial Verification (SV) was used [13, 10]. SIFT descriptors were quantized into three different vocabularies for each detector, with: 500k visual words for 1000 points/image, and 1M visual words for 2000 resp. 4000 interest points per image. The TF-IDF scoring in an efficient inverted index was used to get the BoW ranking. The spatial verification estimated a similarity transformation between the query and each of the top 1000 ranked images. Finally, images were re-ranked based on number of correspondences. The ranking for each query was evaluated using Oxford buildings protocol and an mean Average precision computed as defined in [10].

The results are summarized in Table 1. The Hessian detectors consistently outperformed both the Laplacian and the Difference of Gaussians, regardless the size of the representation. Particularly for the highest number of interest points per image (4000), where both DoG implementations were struggling to deliver this many points, their performance dropped. Thus we can conclude that the complementary saddle points detected by the Hessian detector consistently

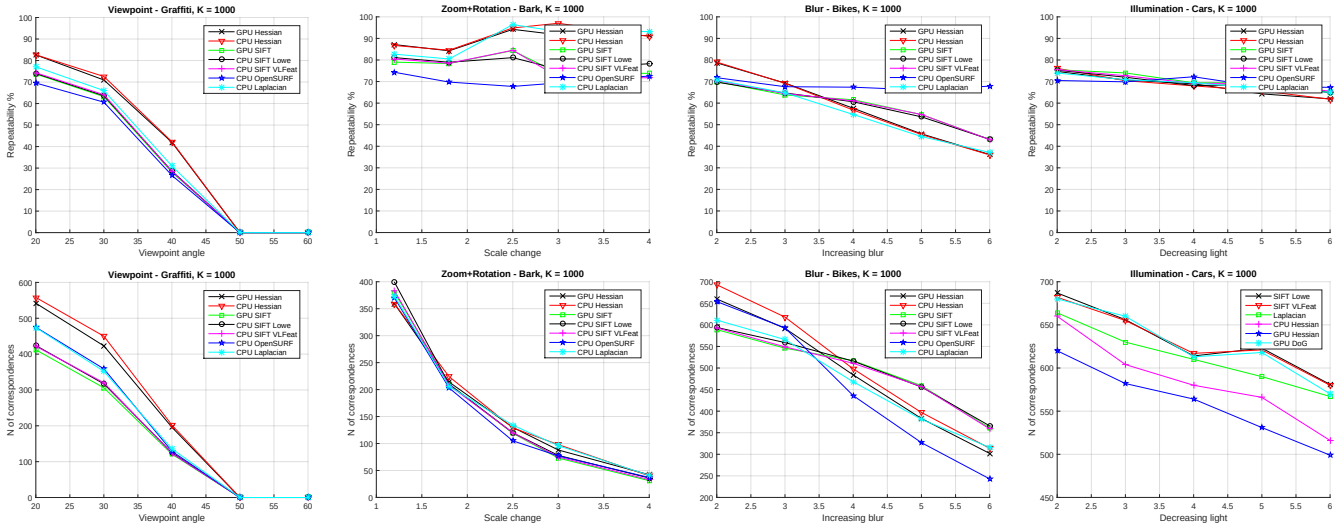


Figure 6. Repeatability score and number of correspondences on image sequences with (from left to right): a significant view angle change, scale change, image blur and exposure change. Number of features per image was limited to the best 1000 according to absolute response value.

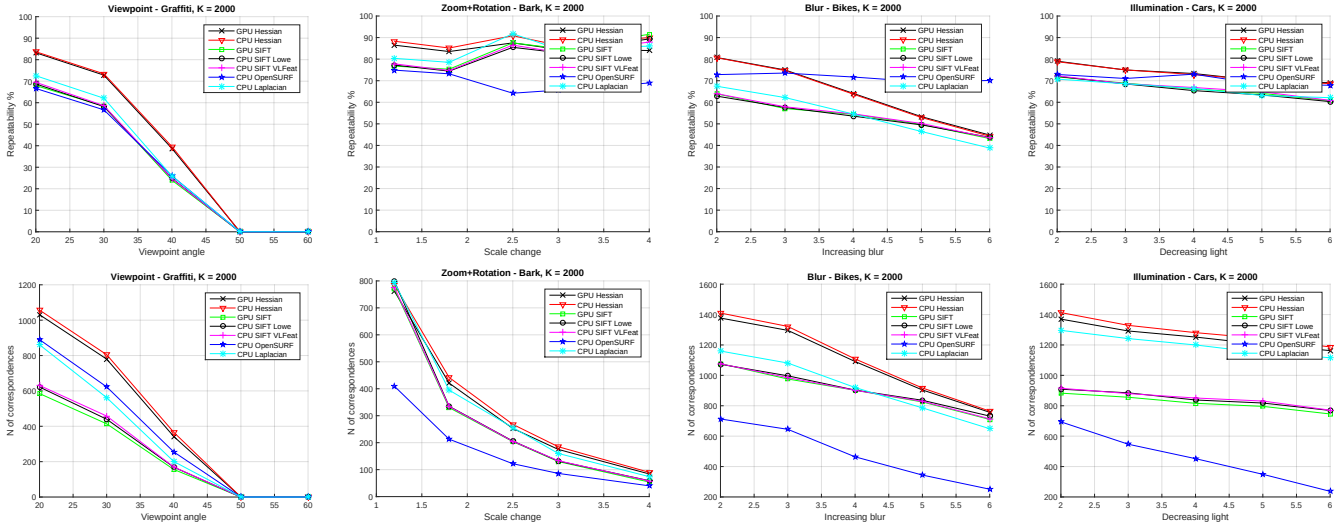


Figure 7. Repeatability score and number of correspondences on image sequences with (from left to right): a significant view angle change, scale change, image blur and exposure change. Number of features per image was limited to the best 2000 according to absolute response value.

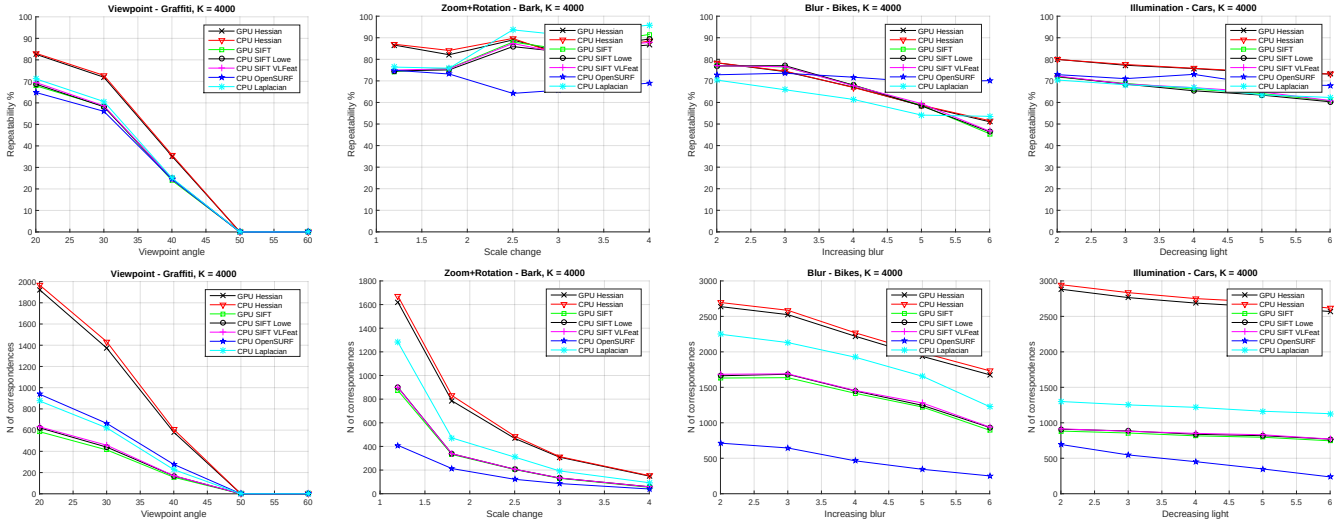


Figure 8. Repeatability score and number of correspondences on image sequences with (from left to right): a significant view angle change, scale change, image blur and exposure change. Number of features per image was limited to the best 4000 according to absolute response value.

Method	Max.feat.	Lowe DoG	VLFeat DoG	CPU Laplacian	CPU Hessian	GPU Hessian
BoW	1000	0.551	0.512	0.572	0.584	0.579
	2000	0.517	0.547	0.568	0.625	0.629
	4000	0.558	0.585	0.617	0.643	0.615
BoW+SV	1000	0.590	0.554	0.601	0.627	0.621
	2000	0.584	0.594	0.617	0.675	0.678
	4000	0.639	0.650	0.692	0.716	0.699

Table 1. Image retrieval experiment. The Bag of Words (BoW) method with and without Spatial Verification (SV) was evaluated with different interest point implementations. Features were limited to best 1000, 2000 resp. 4000 points per image based on detector's response. The values in the table are the measured mean average precisions, defined in [10].

improve the retrieval performance.

5. Conclusion

We have implemented an interest point detector based on the determinant of the Hessian matrix. Such a detector was previously shown, and the observation was confirmed in our experiments, to be superior in the quality of detected points to commonly used detectors based on the Difference of Gaussians. Starting with a publicly available GPU implementation of SIFT detector, we have implemented several modifications and experimentally verified that the performance indeed improved. The implementation, which is in CUDA for compatible NVidia graphics cards, was published and made available.

Acknowledgements

The authors were supported by Toyota Motor Europe.

References

[1] M. Agrawal, K. Konolige, and M. R. Blas. Censure: Center surround extremas for realtime feature detection and matching. In D. A. Forsyth, P. H. S. Torr, and A. Zisserman, editors, *ECCV (4)*, volume 5305 of *Lecture Notes in Computer Science*, pages 102–115. Springer, 2008. 2

[2] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, June 2008. 2, 3, 4

[3] M. Ebrahimi and W. W. Mayol-Cuevas. SUSurE: Speeded Up Surround Extrema feature detector and descriptor for realtime applications. pages 9–14, Aug. 2009. 2

[4] H. Fassold and J. Rosner. A real-time gpu implementation of the sift algorithm for large-scale video analysis tasks. In *IS&T/SPIE Electronic Imaging*, pages 940007–940007. International Society for Optics and Photonics, 2015. 2

[5] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer, 1994. 3

[6] T. Lindeberg. Feature detection with automatic scale selection. *IJCV*, 30(2):79–116, 1998. 3, 4

[7] T. Lindeberg. Image matching using generalized scale-space interest points. *Journal of Mathematical Imaging and Vision*, 52(1):3–36, 2015. 3

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 20(2):91–110, 2004. 1, 3, 4

[9] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *IJCV*, 65(1-2):43–72, 2005. 5

[10] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 6, 8

[11] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer Berlin Heidelberg, 2006. 2

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571, Nov 2011. 2

[13] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. volume 2, pages 1470–1477, 2003. 6

[14] M. Soltan Mohammadi and M. Rezaeian. Siftcu: An accelerated cuda based implementation of sift. In *Third Symposium on Computer Science and Software Engineering, Sharif University, Tehran*, volume 3, 2013. 2

[15] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007. 2