On the Choice of Tensor Estimation for Corner Detection, Optical Flow and Denoising

Freddie Åström[1,2] and Michael Felsberg[1,2]

[1] Computer Vision Laboratory, Linköping University, Sweden
[2] Center for Medical Image Science and Visualization (CMIV), Linköping University

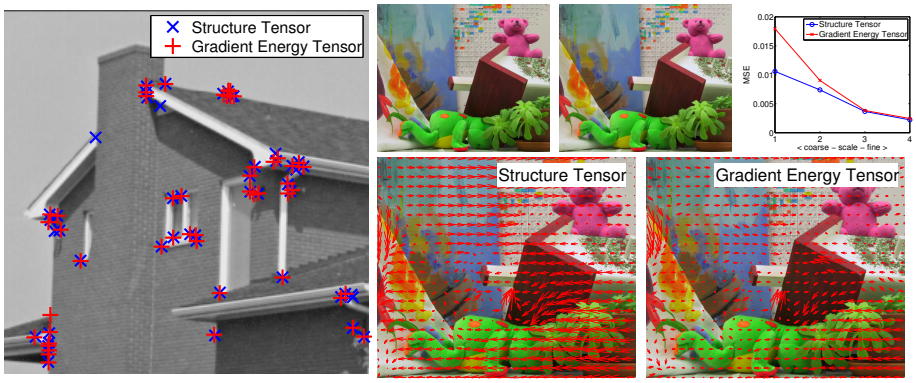
Abstract. Many image processing methods such as corner detection, optical flow and iterative enhancement make use of image tensors. Generally, these tensors are estimated using the structure tensor. In this work we show that the gradient energy tensor can be used as an alternative to the structure tensor in several cases. We apply the gradient energy tensor to common image problem applications such as corner detection, optical flow and image enhancement. Our experimental results suggest that the gradient energy tensor enables real-time tensor-based image enhancement using the graphical processing unit (GPU) and we obtain 40% increase of frame rate without loss of image quality.

1 Introduction

A drawback of many current state of the art image processing methods is the high computation requirements necessary to achieve high-quality results. As a consequence, the computational constraints limit the methods applicability as useful tools in real-time applications, implying that processing-pipelines operate on suboptimal image data. Specifically, the structure tensor introduced by Förstner and Gülch [1] and Bigün and Granlund [2] is an integral part of many image processing applications, such as corner detection [3, 4], optical flow [5] and tensor-based image denoising [6].

In this paper we propose to replace the structure tensor with an alternative tensor, the gradient energy tensor [7] that does not (necessarily) require a post-convolution of its tensor-components to form a rank-2 tensor. The principal difference to the structure tensor is that the gradient energy tensor use higher-order derivatives to capture the orientation in a neighbourhood. In figure 1 we have used the tensors for corner detection and dense optical flow and as visualized the two tensors produce very similar results. As a major contribution of this work we present an adaptive tensor-based image denoising method implemented using Nvidia CUDA programming language. Our approach is superior to those based on the structure tensor with regards to computational performance, without loss of image quality.

The structure tensor is defined as the outer product of the image gradient directions and in the case of two-dimensional images, the tensor is at most of rank-2 and determines the local energy distribution of a neighbourhood. In practice,



Corner Detection
Good Features to Track [4]

Optical flow
Dense Lucas-Kanade [5]

Fig. 1. An illustration that the Gradient Energy Tensor performs comparably to the Structure tensor in the two applications of corner detection and optical flow estimation (sequence Teddy frames 10 and 11 [9]). The gradient energy tensor does ***not*** (necessarily) require a post-convolution of its tensor-components, which is the case in the case of corner detection. See text for details.

to enforce robustness to noise and to form a rank-2 tensor, the structure tensor components are averaged using a post-smoothing of the tensor-components. Furthermore, without the post-convolution operation the tenor is of rank-1 and thus it cannot be used to describe corners and junctions in the image structure. The averaged structure tensor can also be viewed as a second-moment matrix, which estimates the local variance of the image data [8].

The gradient energy tensor is particularly suitable when considering graphical processing units (GPU). A GPU is a high-performance graphics card and is designed for massive parallelization of data-processing tasks. The GPU architecture is most suitable for problems with high spatial locality in the image plane and is therefore very fitting for the solution of partial differential equations (PDE). In this work we present a novel tensor-based PDE, which uses the gradient energy tensor for image enhancement and utilize the GPU architecture to enable real-time image enhancement.

The standard approach to image filtering using PDE's is defined using the structure tensor, a rank-2 tensor which is transformed such that it describes the direction parallel to the image structure. This approach does not allow for real-time computations of high-resolution colour images since the tensor is defined using several convolutions of the tensor components. In contrast, the gradient energy tensor does not requires any convolution to form a rank-2 tensor but yet performs equally well, or better in denoising, when considering the resulting peak signal to noise ratio (PSNR) and structural similarity index (SSIM) [10].

1.1 Contributions

In this work we present a lesser known tensor: the gradient energy tensor as an alternative to the commonly used structure tensor. Our main contribution is a novel PDE-based denoising scheme which utilize the gradient energy tensor to drive an image enhancement process. In section 3.1 we adopt the Good Features to Track [4] framework to our tensor formulation and show comparable performance to the structure tensor using a repeatability test for different viewpoint angles. We also demonstrate that it is possible to solve the Lucas-Kanade [5] optical flow formulation using the gradient energy tensor. In section 3.2 we illustrate the approach on sequences from the Middleburry optical flow dataset [9].

Finally, in section 4 we do an exhaustive evaluation on high-resolution colour images and describe the GPU implementaion using Nvidias CUDA programming language. We show that by using the gradient energy tensor we significantly boost the achieved frames per second (fps) compared to the structure tensor to enable real-time image denoising.

2 Estimating directional information

Many image processing algorithms contain an estimate of local orientation as an integral part of the methods. Often the directional information is computed using the so called structure tensor [1, 2]. The tensor is the outer product of the image gradients whereafter the components are averaged in a local neighbourhood. If the averaging operator, $w(x)$ is circular (*i.e.* Gaussian) then the structure tensor is isotropic in homogeneous regions. Below, we define the structure tensor and the gradient energy tensor which we show in the experimental part outperforms the structure tensor with regards to computation efficiency without compromising the accuracy of the final result.

2.1 Structure tensor

The structure tensor, $T \in \mathbb{R}^{2\times 2}$, is symmetric and positive semi-definite [1, 2]. The tensor is defined as the outer product of the image gradients followed by post-convolutions, one for each component, *i.e.*

$$T(u_x, u_y) = \begin{pmatrix} \int w(\xi) u_x(x-\xi)^2 d\xi & \int w(\xi) u_x(x-\xi) u_y(x-\xi) d\xi \\ \int w(\xi) u_x(x-\xi) u_y(x-\xi) d\xi & \int w(\xi) u_y(x-\xi)^2 d\xi \end{pmatrix} \quad (1)$$

The effect of the post-convolution operator is that T will have two non-zero eigenvalues, thus the operator can be used to estimate the orientation of image structures. In the case the convolution is given by the identity *i.e.* $w(x) = 1$ for $x = 0$ and $w(x) = 0$ otherwise, then the eigenvalues are given by the trace of T and 0.

2.2 Gradient energy tensor

The gradient energy tensor (GET) was first introduced by Felsberg and Köthe [7]. Let $Hu = \nabla\nabla^t u$ be the Hessian and $\nabla\Delta u = \nabla\nabla^t\nabla u$, then GET is

$$GET(\nabla u) = HuHu - \frac{1}{2}\Big(\nabla u[\nabla\Delta u]^t + [\nabla\Delta u]\nabla^t u\Big) \tag{2}$$

In contrast to the structure tensor (1), the gradient energy tensor (2) does *not* (necessarily) require a convolution operator to form a rank 2 tensor. The response from the two tensors are illustrated in figure 2. The presence of second and third-order derivatives in GET does makes it more sensitive to noise than the structure tensor, however, it allows us to capture orientation of structures not possible to detect using the structure tensor.

The second difference to the structure tensor is that the gradient energy tensor is not necessarily positive semi-definite. In applications where it is required to have a positive semi-definite tensor it is straightforward to define the eigenvalues of GET to be positive. That is, simply compute the eigenvaluedecomposition of GET and use the alternative definition

$$GET^+(\nabla u) = vv^t|\mu_1| + ww^t|\mu_2| \tag{3}$$

where v, w are the eigenvectors and μ_1, μ_2 eigenvalues of GET respectively. From (3) the tensor's orientation information is made explicit, the eigenvectors describe the local orientation of the neighbourhood and the eigenvalues describe the magnitude.

Since the positivity of the tensor is reflected in the sign of its eigenvalues the presence of negative eigenvalues can be determined on beforehand if the condition $\text{tr}\,(HuHu) - \nabla^t u\nabla\Delta u \geq \sqrt{l}$ is not satisfied where $l = (\text{tr}\,GET)^2 - 4\det(GET) \geq 0$. Since GET is symmetric it has real eigenvalues. Thus by its eigenvalue decomposition it is sufficient to show that $\text{tr}\,GET \geq \sqrt{l}$ in order for GET to be positive semi-definite. l is necessarily positive since $l = (a-c)^2+4b^2 \geq 0$ where a, b and c are the components $GET(\nabla u) = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$ and

$$a = u_{xx}^2 + u_{xy}^2 - u_x(u_{xxx} + u_{xyy}) \tag{4}$$

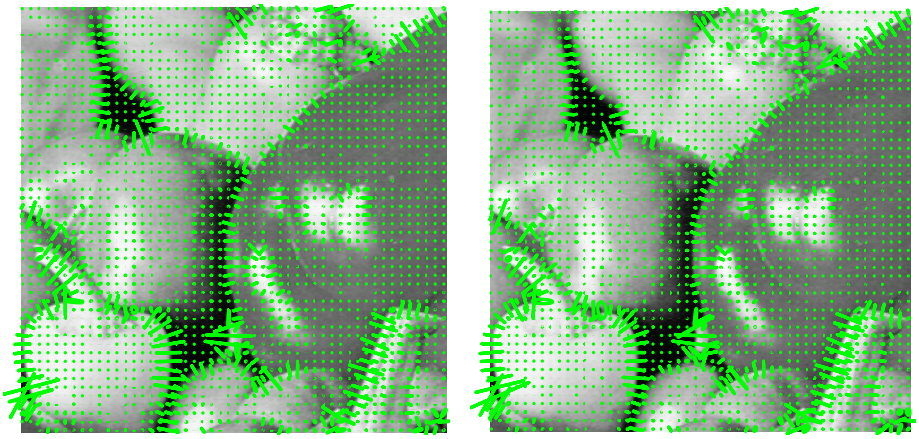
$$b = u_{xx}u_{xy} + u_{yx}u_{yy} - \frac{1}{2}(u_x(u_{yxx} + u_{yyy}) + u_y(u_{xxx} + u_{xyy})) \tag{5}$$

$$c = u_{yy}^2 + u_{yx}^2 - u_y(u_{yxx} + u_{yyy}) \tag{6}$$

An analysis of the positivity of the 1-dimensional energy operator was done in [11]. In the remainder of the paper we apply the GET for corner detection, more specifically the good features to track approach, and GET^+ is used to compute the Lucas-Kanade optical flow and tensor-based image denoising.

2.3 Eigendecomposition of 2×2 tensors

In the previous section we have shown that both the structure and gradient energy tensor can be used to compute the local orientation. By factorizing the



Structure Tensor (1) *Gradient Energy Tensor* (2)

Fig. 2. Illustration of the resulting tensor-fields. The structure tensor require additional post-smoothing to form a rank-2 tensor compared to the gradient energy tensor which is defined without post-smoothing. Note that the size of the ellipses has been scaled for improved visualization.

tensors into their eigendecomposition the directional change and its magnitude is made explicit. Specifically, a matrix $S \in \mathbb{R}^{2\times 2}$ can be decomposed into its eigenvalues $\mu_{1,2}$ and eigenvectors v, w representation such that

$$S = vv^t\mu_1 + ww^t\mu_2 \tag{7}$$

where v, w are two orthonormal vectors. The eigenvectors describe the orientation and the eigenvalues the magnitude of the directional change in a neighbourhood. The eigenvalues can be computed by solving the characteristic polynomial $\det(S - \mu I) = 0$ and the solution is given by

$$\mu_{1,2} = \frac{1}{2}\left(\operatorname{tr} S \pm \sqrt{(\operatorname{tr} S)^2 - 4\det S}\right) \tag{8}$$

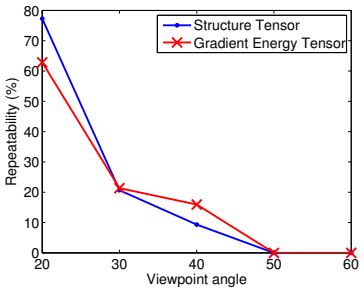
For the applications presented in this work we are not required to compute the explicit eigendecompistion (7), rather we are primarily interested in the eigenvalues. Thus, by observing that $vv^t + ww^t = I \iff ww^t = I - vv^t$, then S can be expressed as [12],

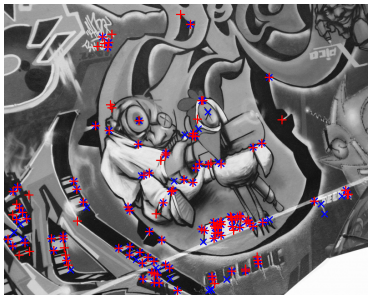
$$S = (\mu_1 - \mu_2)vv^t + I\mu_2 \tag{9}$$

and we compute the eigenvector-product vv^t as

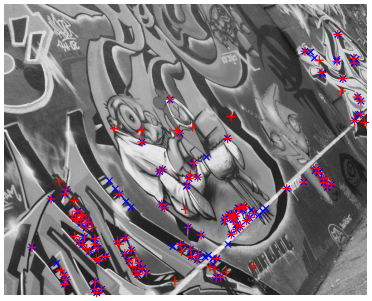
$$vv^t = \frac{1}{(\mu_1 - \mu_2)}(S - I\mu_2) \tag{10}$$

Reference





20 degrees viewpoint angle



40 degrees viewpoint angle

Fig. 3. Examples of corner detection for the structure tensor (with post-smoothing) and the gradient energy tensor (without post-smoothing) using Good Features to Track. Both tensors detect corners accurately but not always the same corners.

3 GET corner detection and optical flow

3.1 Corner detection

The first application we consider is corner detection using Good Features to Track [4]. The problem of corner detection is part of many image processing pipelines such as interest point detection and sparse optical flow. The Good Features to Track framework detects corners by considering the eigenvalues of the structure tensor. If the structure tensor has two non-zero eigenvalues $\mu_{1,2}$, both larger than some threshold μ then the orientation tensor is necessarily invertible, *i.e.* the tensor is of rank 2. If $\min(\mu_1, \mu_2) > \mu$ where μ is often set to a fraction of the largest minimum eigenvalue then the neighbourhood contains a corner point.

In this work we set $\mu = 0.01$ and figure 3 shows the 128 strongest detected corners for the structure tensor and the gradient energy tensor. We use a Gaussian kernel with standard deviation 1 for post-smoothing of the structure tensor components. We also computed the repeatability measure [13] at 40% overlap (see figure 3) for the *viewpoint* dataset where the viewpoint angle has been changed from 20-60 degrees from the reference image [14]. The repeatability

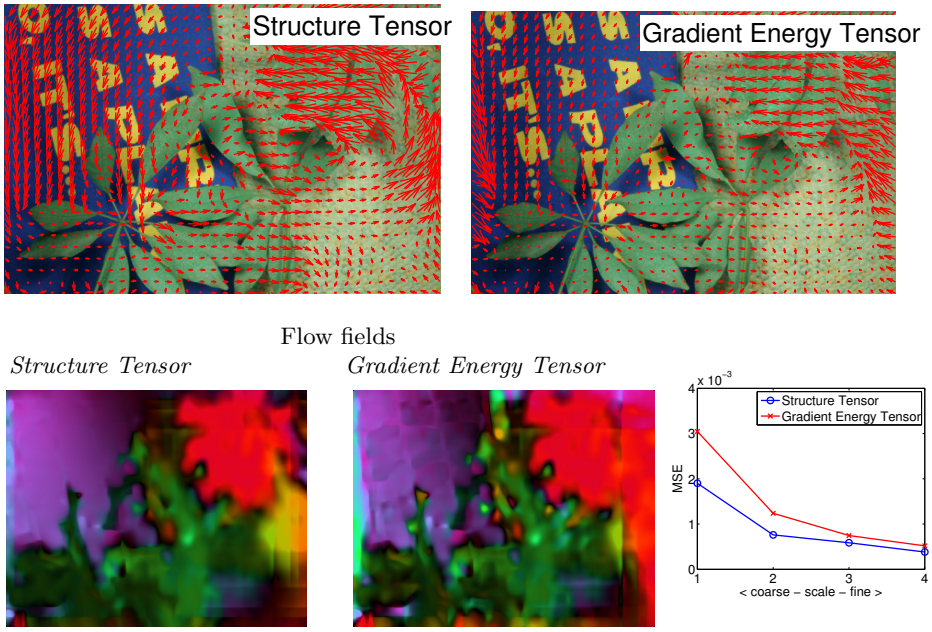


Fig. 4. Optical flow estimated from frames 7 and 8 of the Schefflera sequence in the Middleburry optical flow dataset [9]. Top row illustrate the vector field. On the bottom we show the obtained flow fields where the direction is colour coded and intensity corresponds to the magnitude. The graph to the right show the mean squared error between the warped image $J(x+d)$ and the reference image $I(x)$ in (11).

measure is similar for the two tensors. Note that the gradient energy tensor, in this example, does not contain a post-smoothing of the tensor components.

3.2 Optical flow

Our second application is to apply the gradient energy tensor to the original Lucas-Kanade optical flow formulation [5] to compute a dense motion field. By minimizing the below energy functional the structure tensor appear as part of the minimizer

$$E(u) = \int_{\Omega} [J(x+d) - I(x)]^2 w(x)\, dx \tag{11}$$

where J and I are two images of size Ω with some unknown displacement vector d. The minimizer of (11) is obtained by solving (see [15])

$$\left[\int_{\Omega} [\nabla J(x) \nabla^t J(x)] w(x)\, dx\right] d = \left[\int_{\Omega} [(J(x) - I(x)) \nabla J(x)] w(x)\, dx\right] \tag{12}$$

The bracket in the left hand side of (12) is the structure tensor, T in (1). Figure 4 shows the results when we solve (12) with the structure tensor and when

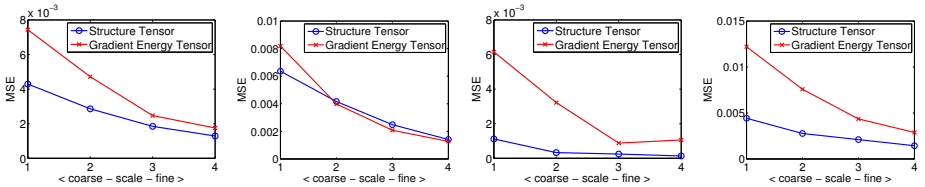


Fig. 5. Mean squared error (MSE) graphs between the first and second image after warping the first image with the estimated motion field. The sequences are part of the Middlebury dataset [9]. The estimate of the motion field in the Wooden sequence for the gradient energy tensor diverge on the finest scale, but in the other sequences the MSE yield comparable final displacement estimate.

we interchanged the structure tensor with the gradient energy tensor with positive eigenvalues, *i.e.* GET^+ in (3). Due to the large displacement between the image pairs we are required to have a post-convolution of the GET^+ components similarly to the structure tensor in order to capture the motion.

For both tensors we solve the normal equation (12) explicitly using the pseudo-inverse on multiple scales. The solution we obtain at a coarse scale is propagated to a finer scale and after each scale we apply a median-filter to the estimated motion field [16]. Furthermore, we found that the magnitude scaling of the gradient energy tensor eigenvalues resulted in a poor motion estimation. Therefore, we scale the eigenvalues of the gradient energy tensor using the factor $\sigma i/I$ where $i \in I = \{1, 2, 3, 4\}$ are the scales, $i = 1$ is the fines scale, and σ is the standard deviation of the Gaussian filter $w(x)$. As for the structure tensor, the selection of σ is dependent on the absolute motion present within the frames, *i.e.* if the displacement is large then a larger σ is required. In figure 5 we show the mean squared error between $J(x+d)$ and $I(x)$ for some additional sequences from the Middlebury optical flow dataset [9].

The optical flow formulation (11) does (obviously) not give state of the art results, however the approach illustrates that the gradient energy tensor is a possible alternative to the structure tensor. We expect that many interesting further results can be derived from this approach.

4 Iterative tensor-based PDE denoising

Image enhancement methods based on partial differential equations (PDE) are often considered to be too computationally expensive for practical applications. The main bulk of computations is the iterative update scheme and the calculation of the post-convolution of the structure tensor components [6].

It is in this application that the gradient energy tensor really excels over the structure tensor. In this section we implement the proposed iterative denoising scheme on the GPU. We show how to utilize the highly parallelizable nature of iterative PDE-based denoising schemes and benefit from the locality of the

gradient energy tensor. The implementation is done using Nvidias CUDA programming language with OpenGL support, the GPU we use is the GTX 670 with 4Gb on card memory and the workstation is equipped with an Intel Xeon CPU at 3 GHz and 8 Gb of installed RAM memory. Even though the hardware specification is in the middle segment of the consumer-market we show that by using the gradient energy tensor, the proposed iterative tensor-based PDE denoising scheme can reach near maximum PSNR at 60 frames per second (fps) for a three channel colour 1280×720 pixel image (HD720p). This is a significant improvement over the structure tensor running at 30 fps while achieving similar peak signal to noise ration (PSNR) and structural similarity [11] (SSIM) values.

4.1 The proposed filtering scheme

The standard formulation of tensor-based anisotropic diffusion [6] using the structure tensor is given in the PDE below with Neumann boundary condition

$$\begin{cases} u - u^0 - \beta \operatorname{div}\left(D(\nabla u)\nabla u\right) = 0 \text{ in } \Omega \\ \qquad\qquad\qquad\qquad \boldsymbol{n} \cdot \nabla u = 0 \text{ on } \partial\Omega \end{cases} \tag{13}$$

where β is a stepsize parameter which controls the smoothness of the solution u that minmizes the PDE. In (13), $D(\nabla u)$ is the *diffusion tensor* computed as

$$D(T(\nabla u)) = vv^t g(\mu_1) + ww^t g(\mu_2) \tag{14}$$

where v, w and $\mu_{1,2}$ are the eigenvectors and eigenvalues of the structure tensor T in (1). g is the diffusivity function and here we set it as $g(s) = \exp(-s/k)$ where k is the edge-stopping parameter determining the adaptivity to the image structure. Instead of using the diffusion tensor in the PDE (13) we propose to use the gradient energy tensor with positive eigenvalues, GET^+ in (3), as the tensor controlling the orientation estimate of the image structures, *i.e.* we define the following PDE

$$\begin{cases} u - u^0 - \beta \operatorname{div}\left(D^+(\nabla u)\nabla u\right) = 0 \text{ in } \Omega \\ \qquad\qquad\qquad\qquad \boldsymbol{n} \cdot \nabla u = 0 \text{ on } \partial\Omega \end{cases} \tag{15}$$

The computation of the eigenvalues are done according (9), *i.e.*

$$D^+(GET^+(\nabla u)) = \left(\frac{g(\lambda_1) - g(\lambda_2)}{\lambda_1 - \lambda_2}\right)(GET^+ - I\lambda_2) + Ig(\lambda_2) \tag{16}$$

where $\lambda_{1,2}$ are the eigenvalues of GET^+ computed according to (8) where we set $S = GET^+$. In practice we compute (14) using (16) with $S = T$.

In order to solve the PDEs (13) and (15), we use a forward Euler iterative scheme with finite differences to approximate the image derivatives. For a discussion on the numerical stability of iterative scheme see [17].

Anisotropic Diffusion [6]

```
for i=0 to maxint do {
 convolution_rows()
 convolution_cols()
 compute_structure_tensor()
 filter_update()
}
```

```
compute_structure_tensor() {
 gradient_products(a,b,c)
 convolution_rows(a)
 convolution_cols(a)
 convolution_rows(b)
 convolution_cols(b)
 convolution_rows(c)
 convolution_cols(c)
 remapp_eigenvalues(a,b,c)
}
```

Gradient Energy Diffusion

```
for i=0 to maxint do {
 convolution_rows()
 convolution_cols()
 compute_energy_tensor()
 filter_update()
}
```

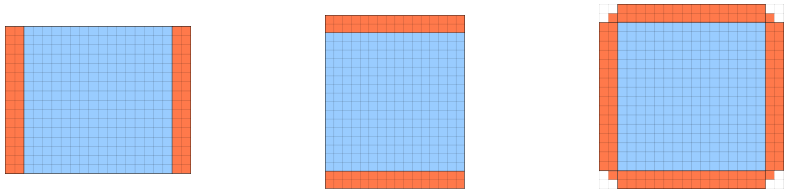
```
compute_energy_tensor() {
 a = (4)
 b = (5)
 c = (6)
 remapp_eigenvalues(a,b,c)
}
```

Table 1. Algorithm pseudo-code for the main CUDA kernels. Left: Standard approach to adaptive image diffusion using the structure tensor. Right: adaptive image diffusion using the gradient energy tensor. The convolutions are separable Gaussian functions of size 5×5 with standard deviation of 1.

4.2 Implementation details

A GPU implementation is about how to efficiently utilize the parallelism of the graphics card architecture. Using CUDA terminology, the parallelism is achieved by dividing the image data into blocks, each block contains the threads that are to be executed in parallel on a streaming multiprocessor. Today's GPU architectures offer many memory types (global, texture, shared, local ...) and our implementation is focused on utilizing the high-performance shared memory. We achieve this by coalescing memory access when streaming data from global memory to shared memory. We considered using texture memory due to its automatic handling of Neumann boundary conditions and memory-access optimization for localized reads, however texture memory is read-only and we require dynamic updates of intermediate results. Also, shared memory is on-chip, and therefore read-access requires less clock-cycles than the texture memory. These differences in memory-latency have a significant impact on runtime performance, for example in convolutions [18].

Since we are interested in processing images with three colour channels, we simplify our implementation by defining a container describing the three colour channels red, green and blue as well as the alpha channel (required for visualization using OpenGL). The image data is read and written using 24-bits but during the filter process we use single precision float. There are primarily three steps involved in the iterative filtering scheme, pre-filtering for regular-



(a) Convolution rows (b) Convolution columns (c) Gradient energy tensor

Fig. 6. We use tiles of size 16×16 (blue regions) with padding of 2 pixels (red region) for the convolution and computation of the gradient energy tensor. We use corresponding tile layouts for computing the image gradient and filter update.

Fig. 7. Colour test images and the corresponding image sizes in pixels that are used in the evaluation.

ization of the image derivatives, orientation estimation and filter update. The steps are illustrated in table 1 and highlights the primary difference between the two implementations: the computation of the structure tensor requires three full (separable) convolutions of the image data whereas the energy tensor does not require any post-convolution of the tensor-components, which is key to the gain in computation speed. Figure 6 shows the memory layout of the shared memory that we use in the CUDA kernels shown in table 1. We use tile sizes of 16×16 and note that each entry in the tile consists of four entries *i.e.* red, green, blue and alpha channels. This approach is convenient since it both simplifies the code and facilitates efficient memory access. The padding of the tiles (the red region in figure 6) is done with two pixels in the case of the separable convolution since we use a Gaussian filter of size 5×5 with standard deviation of 1 for smoothing the image and tensor-components. The coefficients of the Gaussian filters is set using constant memory. The shared memory associated with the gradient energy tensor require a padding of 2 pixels in horizontal and vertical direction since the support of the third order finite difference derivative is 5 pixels. Lastly, since we only require first order diagonal derivatives, it is sufficient to read the closest corner-pixel into the shared memory, further simplifying the global memory access pattern.

In order to measure the resulting computation times, we are required to compute the resulting execution speed of the filtering methods. We do this by using the standard sdkStartTimer() and sdkStopTimer() available in CUDA.

```
void display() {
 sdkStartTimer(&timer);
 unsigned int *dResult;
 // Initialize OpenGL for visualization
 diffusionFilterRGBA(dResult, ...);
 // Map dResult to OpenGL resources and draw image on display
 sdkStopTimer(&timer);
}
```

Table 2. Main function of the tensor-based image denoising method. The timer values reported in section 4.3 are timed including the OpenGL rendering.

We have chosen to include the OpenGL rendering in the timing of the filter performance as shown in table 2 since it more accurately reflects the expected real-time capabilities of video denoising where each frame in a video sequence would be visualized.

4.3 Results

The focus of our evaluation is to show that the gradient energy tensor does not compromise the denoising quality compared to the structure tensor while achieving a faster runtime. Our measures include the peak signal to noise ratio (PSNR) and the structure similarity index [10] (SSIM) for the image quality. With regards to the total runtime (measured as shown in table 2) we report achieved frames per seconds (fps) for each method. For each of the measures a higher value is better than a lower value.

Figure 7 show the colour test image *pippin_Florida0002.bmp* from the McGill colour image database [19] with the original resolution 2560×1920 pixels. The image was downsampled (using bicubic interpolation) to 1920×1080 (HD1080) and 1280×720 (HD720), two common image resolutions in high-definition (HD) video. We corrupt each image with 20 standard deviations of normal distributed noise. The filtering scheme is iterated 10 times with a fixed update step of size 0.20 (we refer to [17] for a discussion on convergence results for iterated solutions of PDEs). The diffusivity constant (see g in (16)) is set to $k/10$ where we compute $k = (e^1 - 1)/(e^1 - 2)\sigma^2$ [20], this yields $k/10 = 0.0015$ when $\sigma = 20/255$ and each colour channel is quantized using an 8-bit representation.

Figure 8 show the obtained PSNR (a) and SSIM (b) values compared to the iteration number, as expected the two methods are comparable for the obtained error measures. The best error values are in agreement between the two methods but obtained at different iterations numbers, it is not a discrepancy in method performance but an issue of parameter tuning. In figure 8 (c) and (d) we show the fps that we obtain for each iteration. After four iterations, in (c), the filter using the gradient energy tensor is stable at 60 fps whereas the standard diffusion scheme using the structure tensor has dropped to 30 fps for the smallest image resolution. In (d) we show the tradeoff between PSNR and obtained fps for *both*

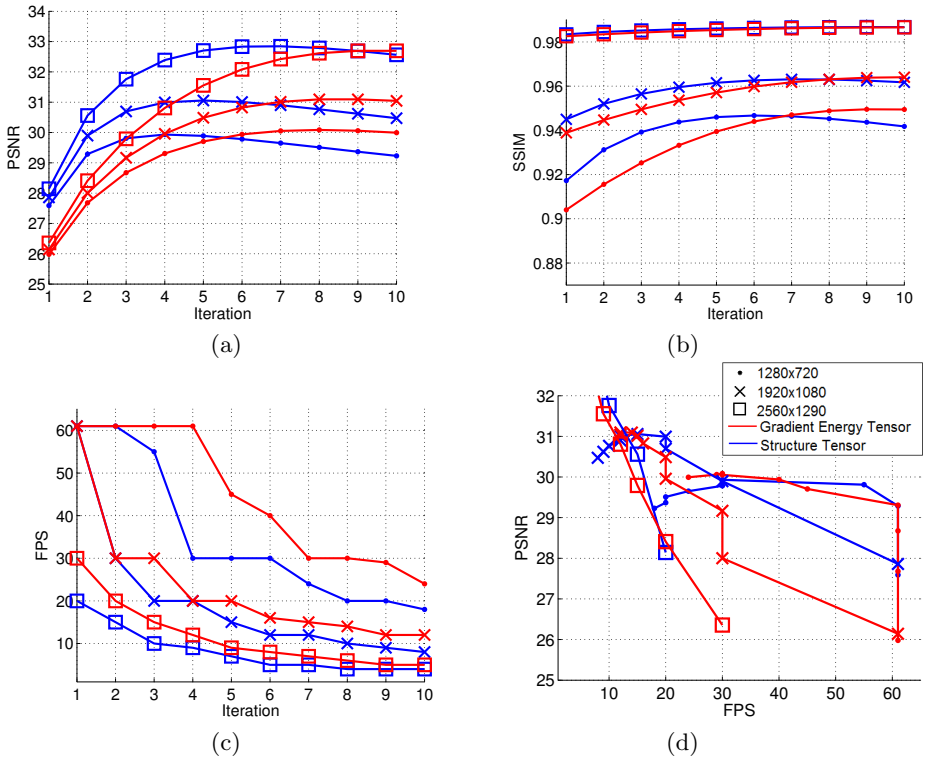


Fig. 8. Obtained PSNR (a), SSIM (b) and fps (c) for the considered image sizes. The PSNR and SSIM values are similar for the two tensors but the obtained fps is significantly higher for the gradient energy tensor.

tensors: a higher fps result in a lower PSNR value. Note that the fps count is *independent* of the image content. Future work will include a more comprehensive study of the GET orientation estimation compared to the structure tensor for other noise levels and image types than considered in this work.

In figure 9 we illustrate an example of the final denoised result, the zoomed images were cropped from the 2560×1920 resolution image and the original image was corrupted by 20 standard deviations of normal distributed additive noise. The images illustrate the denoised result after 10 iterations and note that the gradient energy tensor fps-count is 25% higher than the structure tensor but with indiscernible image quality and PSNR (cmp. figure 8 (a) and (c)). In table 3 we give the ratio of the gradient energy tensor error measures over the structure tensor error measures averaged over the iterations, *i.e.* if the ratio is identical to 1.0 there is no difference in method performance, if the value is larger than 1.0 then the ratio indicate that the gradient energy tensor performs better. The SSIM value difference is less than 10^{-3} whereas the PSNR shows a 3.2% difference in the favour of the structure tensor, however considering the final

Fig. 9. Two patches from the 2560 × 1920 resolution image and the corresponding denoised images at iteration 10. The applied noise was 20 standard deviation of normal distributed additive noise. Note that the difference in visual appearance is not discernible but the fps count is improved by 25% with the gradient energy tensor.

Table 3. Ratios are computed as the measure obtained by the gradient energy tensor divided by the measure obtained by the structure tensor and averaged over iterations. A ratio of 1.0 show that the performance is identical. The SSIM and PSNR ratios are nearly identical but at up to 40% higher fps values in the largest image resolution, the PSNR show a marginal loss in image quality for the gradient energy tensor.

Ratio	PSNR	SSIM	fps
1280 × 720	0.991	0.995	1.35
1920 × 1080	0.982	0.997	1.27
2560 × 1920	0.968	0.999	1.40

fps-count the gradient energy tensor is up to 40% more computationally efficient for the largest image size.

5 Conclusion

In this work we have presented the gradient energy tensor as an alternative to the structure tensor for local orientation. We have considered three applications: corner detection, optical flow and adaptive image enhancement. Due to the absence of a post-convolution of the gradient energy tensor components the tensor is highly suitable for efficient implementation on the GPU, and we showed that the tensor yield significant improvement in obtained frames per second compared to the structure tensor without compromising PSNR and SSIM error values.

Acknowledgement. This research has received funding from the Swedish Research Council through grants for the projects Visualization-adaptive Iterative Denoising of Images (VIDI) and Extended Target Tracking (ETT), within the Linnaeus environment CADICS and the excellence network ELLIIT.

References

1. Förstner, W., Gülch, E.: A fast operator for detection and precise location of distinct points, corners and centres of circular features. In: ISPRS Intercommission, Workshop, Interlaken, pp. 149-155. (1987)
2. Bigun, J., Granlund, G.H.: Optimal Orientation Detection of Linear Symmetry. In: Proceedings of the IEEE First ICCV. (1987) 433–438
3. Harris, C., Stephens, M.: A combined corner and edge detector. In: In Proc. of Fourth Alvey Vision Conference. (1988) 147–151
4. Shi, J., Tomasi, C.: Good features to track. In: CVPR '94. (1994) 593–600
5. Lucas, B.D., Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2. IJCAI'81, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1981) 674–679
6. Weickert, J.: Anisotropic Diffusion in Image Processing. Teubner-Verlag, Stuttgart, Germany (1998)
7. Felsberg, M., Köthe, U.: GET: The connection between monogenic scale-space and Gaussian derivatives. In: Scale Space and PDE Methods in Computer Vision. Volume 3459 of LNCS. (2005) 192–203
8. Lindeberg, T.: Scale-Space Theory in Computer Vision. Kluwer international series in engineering and computer science: Robotics: Vision, manipulation and sensors. Springer (1993)
9. Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A Database and Evaluation Methodology for Optical Flow. Int. J. Comput. Vision **92** (2011) 1–31
10. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Processing **13** (2004) 600–612
11. Bovik, A.C., Maragos, P.: Conditions for positivity of an energy operator. Signal Processing, IEEE Transactions on **42** (1994) 469–471
12. Granlund, G.H., Knutsson, H.: Signal processing for computer vision. Kluwer (1995)
13. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. Pattern Analysis and Machine Intelligence, IEEE Transactions on **27** (2005) 1615–1630
14. Mikolajczyk, K.: Implementation, http://www.robots.ox.ac.uk/~vgg/research/affine (2014)
15. Tomasi, C., Kanade., T.: Detection and Tracking of Point Features. Technical report, Carnegie Mellon University Technical Report CMU-CS-91-132 (1991)
16. Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: CVPR2010. (2010) 2432–2439
17. Scherzer, O., Weickert, J.: Relations Between Regularization and Diffusion Filtering. Journal of Mathematical Imaging and Vision **12** (2000) 43–63
18. Podlozhnyuk, V.: Image convolution with CUDA, NVIDIA Corporation white paper, v1.0 (2007)
19. Olmos, A., Kingdom, F.A.A.: A biologically inspired algorithm for the recovery of shading and reflectance images. Perception **33** (2004) 1463–1473
20. Felsberg, M.: Autocorrelation-Driven Diffusion Filtering. Image Processing, IEEE Transactions on **20** (2011) 1797–1806