J Real-Time Image Proc (2011) 6:15–22
DOI 10.1007/s11554-010-0158-5

SPECIAL ISSUE

Real time ultrasound image denoising

Fernanda Palhano Xavier de Fontes · Guillermo Andrade Barroso · Pierrick Coupé · Pierre Hellier

Received: 7 July 2009 / Accepted: 16 April 2010 / Published online: 13 May 2010

Abstract Image denoising is the process of removing the noise that perturbs image analysis methods. In some applications like segmentation or registration, denoising is intended to smooth homogeneous areas while preserving the contours. In many applications like video analysis, visual servoing or image-guided surgical interventions, real-time denoising is required. This paper presents a method for real-time denoising of ultrasound images: a modified version of the NL-means method is presented that incorporates an ultrasound dedicated noise model, as well as a GPU implementation of the algorithm. Results demonstrate that the proposed method is very efficient in terms of denoising quality and is real-time.

Keywords Image denoising · Ultrasound imaging · Non-local means · GPGPU

1 Introduction

Image denoising is a key component of image processing workflows. Denoising aims at reducing the noise in homogeneous areas while preserving the image contours. Denoising is important for postprocessing methods like segmentation, classification, object recognition, pattern analysis, registration, etc. In this context the denoising of ultrasound images is particularly challenging due to the particular texture of the ultrasound images. The noise, often referred to as "speckle", is a multiplicative signal-dependent noise. Ultrasound is a medical imaging modality suited for many applications, since the image acquisition is real-time. Therefore, ultrasound can be used for surgical guidance and robotic-assisted interventions. For surgical applications, ultrasound offers a light, real-time, inexpensive, non-ionizing capability to image the surgical field and update the preoperative planning. To fully benefit from this real-time capability, image processing methods also need to be real-time, which is also crucial for surgical applications. In this paper, a Bayesian version of the NL-means methods [3] is presented that allows to incorporate an ultrasound dedicated noise model. This denoising method was implemented using GPU technology and leads to a real-time denoising method. The paper is organized as follows: Sect. 2 presents an overview of denoising methods, Sect. 3 presents the modified NL-means method, Sect. 4 describes the GPU implementation of the algorithm, and Sect. 5 presents results in terms of denoising quality and computation time.

F. Palhano Xavier de Fontes · G. Andrade Barroso · P. Coupé · P. Hellier (✉)
INRIA Centre de Recherche Rennes Bretagne Atlantique, 35042 Rennes Cedex, France
e-mail: Pierre.Hellier@inria.fr

2 Related work

In this paper, an efficient denoising method, dedicated to ultrasound images, is implemented using GPU capabilities. In this section, we briefly introduce related methods concerning image denoising, and GPU computation.

2.1 Image denoising

Many filters have been proposed for general image denoising. Ultrasound images are corrupted by speckle, a specific noise which is associated with coherent imaging systems. Many studies have been conducted to develop specific methods dedicated to ultrasound images. Denoising is a broad area and it would be out of the paper's scope

to give an exhaustive survey. We refer the reader to [5, 6, 9, 21] for survey. Here, a short selection of denoising techniques is presented.

2.1.1 Adaptive filters

Adaptive filters are based on the assumption that speckle is essentially a multiplicative noise. The most common adaptive filters are Lee's filter [19], Frost's filter [10], and Kuan's filter [17]. The principle adopted by Lee and Kuan is very similar. Both seek to minimize the mean square error (MSE) between the true value of a pixel and its estimated value. The only difference between the two methods is that Lee utilizes a linear approximation for speckle while Kuan uses a nonlinear approximation. The Frost filter is also an MSE filter, but assumes that the image is stationary. The median filter is a spatial filter that replaces the central pixel in a window with the median of all the pixel values in that window. Loupas et al. [20] introduced a new approach to the adaptive median filter, called adaptive weighted median filter (AWM). In this case, different weights are assigned for each pixel from the local content of the image. The filters mentioned above are easy to implement and present a good compromise between complexity and denoising quality.

2.1.2 Partial differential equations filters

Some methods adapted the thermodynamic equation of heat to filter ultrasound images. Perona and Malik proposed an anisotropic filter with border detection [23]. Alvarez et al. [1] also developed a filter based on the thermodynamic equation of heat which is called curvature motion. Total variation minimization scheme (TV) was introduced by Rudin et al. [24]. The denoising problem is considered as a minimization problem with constraints, where the constraints are determined by statistical noise. In the original method, the removed noise is treated as an error. In practice, some structures and textures can be present in this error. Several groups have tried to avoid this effect [22, 25]. Unlike the aforementioned adaptive filters, these methods are iterative and keep the location and accuracy of contours while smoothing the rest of the image.

2.2 GPGPU

Modern graphic cards have high computing and programming capabilities for a reasonable monetary cost. The last generation of NVIDIA graphics card processors (GPU) have up to 240 processing units that can work in parallel with wide band memory. CUDA tools from NVIDIA allow to program graphics cards to solve intensive computation problems with a high-level language. These facts push scientists to adopt the general purpose programming on graphic processor unit (GPGPU) in aim to attempt real-time level on intensive computation problems. This is the case for image and signal processing problems. GPUCV is a port of well-known OpenCV run-time libraries in GPU devices to accelerate filters execution and image processing basics tools.

Non-local means filters computation has good characteristics to be ported on GPU: every pixel computation needs a regular analysis of same type of zones over the image input. ImageDenoise SDK CUDA [15] implements a real-time generic non-local means filter for 2D images which exploits CUDA 2D caches capabilities in textures memories into a GPU. Huhle et al. [12] implements a non-local means filter for deep data matrix of 3D points using CUDA. Recently, GPU has been used for medical image registration [18] and segmentation [4].

3 Bayesian NL-means

In this section, the original NL-means method [3] is briefly recalled, and the adaptation of the NL-means [7] to a dedicated ultrasound noise model [20].

3.1 NL-means

Here, the basic principles of the NL-means technique are presented. This method is based on a non-local paradigm, i.e., the restored intensity of a pixel is a weighted average of all pixels in the image (classically, a neighborhood around the considered pixel), weighted by the distance between the patches:

$$\mathrm{NL}(u)(x_i) = \sum_{x_j \in \Omega^{\dim}} w(x_i, x_j) u(x_j) \tag{1}$$

where $w(x_i, x_j)$ is the weight assigned to value $u(x_j)$ for restoring the pixel x_i. More precisely, the weight evaluates the similarity between the intensities of the local neighborhoods (patches) $\mathcal{N}_i$ and $\mathcal{N}_j$ centered on pixels x_i and x_j, such that $w(x_i, x_j) \in [0, 1]$ and $\sum_{x_j \in \Omega^{\dim}} w(x_i, x_j) = 1$ (see Fig. 1). The size of the local neighborhood $\mathcal{N}_i$ and $\mathcal{N}_j$ is $(2d + 1)^{\dim}$. The traditional definition of the *NL-means* filter considers that the intensity of each pixel can be linked to pixel intensities of the whole image. For practical and computational reasons, the number of pixels taken into account in the weighted average is restricted to a neighborhood, that is, a "search volume" Δ_i of size $(2M + 1)^{\dim}$, centered at the current pixel x_i.

For each pixel x_j in Δ_i, the Gaussian-weighted Euclidean distance $\|.\|^2_{2,a}$ is computed between the two image patches $\mathbf{u}(\mathcal{N}_j)$ and $\mathbf{u}(\mathcal{N}_i)$ as explained in [3]. This distance is the traditional L_2-norm convolved with a Gaussian kernel of

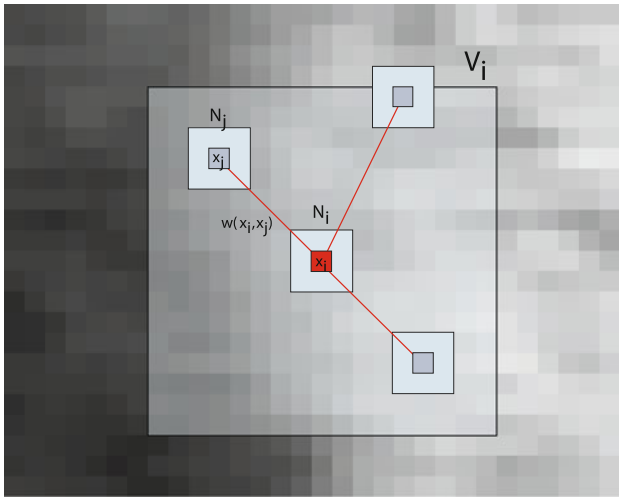

Fig. 1 Pixelwise *NL-means* filter ($d = 1$ and $M = 8$). The restored value at pixel x_i (in *red*) is the weighted average of all intensity values of pixels x_j in the search volume Δ_i. The weights are based on the similarity of the intensity neighborhoods (patches) $\mathbf{u}(\mathcal{N}_i)$ and $\mathbf{u}(\mathcal{N}_j)$

standard deviation a. The weights $w(x_i, x_j)$ are then computed as follows:

$$w(x_i, x_j) = \frac{1}{Z_i} \exp -\frac{\|\mathbf{u}(\mathcal{N}_i) - \mathbf{u}(\mathcal{N}_j)\|^2_{2,a}}{h^2} \tag{2}$$

where Z_i is a normalization constant ensuring that $\sum_{x_j \in \Omega^{\dim}} w(x_i, x_j) = 1$ and h acts as a filtering parameter controlling the decay of the exponential function.

3.2 Bayesian NL-means

The initial formulation of the NL-means filter relies on a L_2-norm between two patches, which relies on the assumption of an additive white Gaussian noise model. Unfortunately, the noise of ultrasound images cannot be considered as an additive white Gaussian noise. Here, a Bayesian formulation of the NL-means filter is used to incorporate an ultrasound dedicated noise model. We refer the reader to [7] for more details about this method.

The distribution of noise in ultrasound images has been largely studied in the literature so far and many models have been proposed, up to the utmost complexity. Among them, the Loupas noise model [20] has been successfully used in many studies. It reads as

$$u(x) = v(x) + v^{\gamma}(x)\eta(x) \tag{3}$$

where $v(x)$ is the original image, $u(x)$ is the observed image, and $\eta(x) \backsim \mathcal{N}(0, \sigma^2)$ is a zero-mean Gaussian noise. This model is more flexible and less restrictive than the usual RF model and is able to capture reliably image statistics since the factor γ depends on ultrasound devices and additional processing related to image formation. Contrary to additive white Gaussian noise model, the noise component in (3) is image-dependent. In [20], based on the experimental estimation of the mean versus the standard deviation in Log-compressed images, Loupas et al. have shown that $\gamma = 0.5$ model fits better to data than the multiplicative model or the Rayleigh model. Since, this model has been used successfully in many studies [2, 11, 16, 29]. Clearly, this model is relevant since it is confirmed that the speckle is higher in regions of high intensities versus regions of low intensities [16, 26].

In [14], a Bayesian formulation of the *NL-means* filter was proposed. Equivalent to the conditional mean estimator, it has been shown that an empirical estimator $\widehat{\mathbf{v}}(B_{i_k})$ of a block B_{i_k} can be defined as (see reference [7])

$$\widehat{\mathbf{v}}(B_{i_k}) = \frac{\frac{1}{|\Delta_{i_k}|}\sum_{j=1}^{|\Delta_{i_k}|} \mathbf{v}(B_j) p(\mathbf{u}(B_{i_k})|\mathbf{v}(B_j))}{\frac{1}{|\Delta_{i_k}|}\sum_{j=1}^{|\Delta_{i_k}|} p(\mathbf{u}(B_{i_k})|\mathbf{v}(B_j))} \tag{4}$$

where $p(\mathbf{u}(B_{i_k})|\mathbf{v}(B_j))$ denotes the probability density function (pdf) of $\mathbf{u}(B_{i_k})$ given the noise-free and unknown patches $\mathbf{v}(B_j)$.

Considering the Bayesian formulation and the Loupas noise model, a new formulation of the NL-means is proposed [7]. Assuming that

$$u(x)|v(x) \backsim \mathcal{N}(v(x), v(x)^{2\gamma}\sigma^2) \tag{5}$$

leads to

$$p(u(x)|v(x)) \propto \exp -\frac{(u(x) - v(x))^2}{2v(x)^{2\gamma}\sigma^2}. \tag{6}$$

Finally, this amounts to substituting the traditional L_2-norm by the Pearson distance. We refer the reader to [7] for extensive details about this approach.

4 GPU implementation

Due to the high complexity of the NL-Means algorithm and the recent use of GPUs for massively parallel computation, we decided to use the GPU GeForce GTX 280 hardware version 1.3 to run our NL-Means filter adapted to ultrasound images. The large computational burden is mainly due to the computation of distances between each patch, which is used to later calculate the weight of a pixel. Since these distances are independent of each other, the algorithm is intrinsically parallel and particularly suited for such implementation. To develop the software, the NVIDIA CUDA was used. CUDA is a general-purpose parallel computing architecture that uses a high-level language compatible with C/C++.

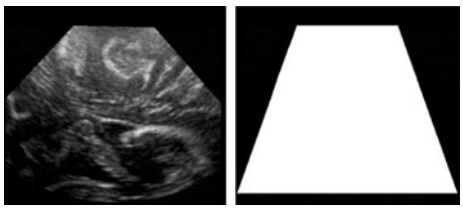

Fig. 2 Ultrasound image and corresponding mask. To reduce the computation time, only points belonging to the mask are denoised

Through CUDA, we can define functions, called kernels, that, when called, are executed N times in parallel by N different CUDA threads. These threads can be grouped into blocks. As the filtered images are 2*D* images, we chose 2*D* blocks for our implementation. The size of the blocks is an important parameter to computational time. Empirically, the fastest results were achieved partitioning each block into a 16 × 16 set of threads.

CUDA architecture has fast local memory called shared memory. This memory is faster than the GPU global memory, but is limited. For the GTX 280 there are 16,384 bytes/block which are simultaneously used by shared memory and registers. Because of this limitation, we should select the most important data to copy into shared memory.

First, an image patch and its neighborhood are copied into shared memory. Second, the search areas of each pixel are also copied into memory. This measure increases the performance, reducing runtime because of the reduction of global memory access. In ultrasound images, the area of interest containing image information is smaller than the size of the entire image. Thus, in order to further reduce the computational time, only points belonging to a given input mask were denoised, as illustrated in Fig. 2.

The program is composed of eight kernels, four main kernels for NL-Means computation and another four auxiliary kernels to compute the mean and variance of the image. Also, a function implemented in C++ is used to compute the smoothing parameter (h). Even though the program has eight kernels, only three of them will be executed to filter an image. Two input parameters, determined by the user, specify how the kernels are called. There are two different implementations of the NL-means filter, a generic one [3] and a filter specifically adapted for ultrasound images [7]. Another input parameter which determines which kernels will be executed is the neighborhood size N_i. Because of limitations on shared memory size only two options were implemented: a neighborhood of 3 × 3 pixels or a neighborhood of 5 × 5 pixels.

The pseudocode in Algorithm 1 shows the main instructions that are executed when the programs runs. First of all, a preprocessing computation is performed. Two images are created: a map of local means and a map of local variances. These images will be used to select the relevant pixels on the input image. Once the preprocessing is done, the main kernel that implements the NL-means filter, will be executed. As mentioned earlier, an area of the image will be copied into shared memory. It is very important to synchronize after the copy to ensure that all threads are completed before proceeding to the next step which is computing the new value for each pixel using the NL-means algorithm.

Algorithm 1 NL-means Algorithm

```
for all pixel x_i in the image do
  compute mean(x_i)
  compute variance(x_i)
end for
for each block do
  copy patch of the input image into shared memory
  syncthreads();
  for each pixel x_i in the block do
    copy search area Δ_i into shared memory
    syncthreads();
    for all pixel x_j ∈ Δ_i do
      N_i − N_j // compute distances
      w(x_i, x_j)
    end for
    NL(u)(x_i) = Σ w(x_i, x_j)u(x_j)
    copy pixel x_i into global memory
  end for
end for
```

5 Experiments

In this section, results are presented, both in terms of denoising quality and computation time. Three methods are compared in what follows:

- A C++ implementation of the NL-means method;
- The CUDA SDK NL-means filter called imageDenoising [15];
- Our CUDA implementation of the Bayesian NL-means, called FUID (Fast Ultrasound Image Denoising).

5.1 Denoising quality

5.1.1 Qualitative evaluation

The denoising methods were first tested on real intraoperative ultrasound images acquired during a neurosurgical

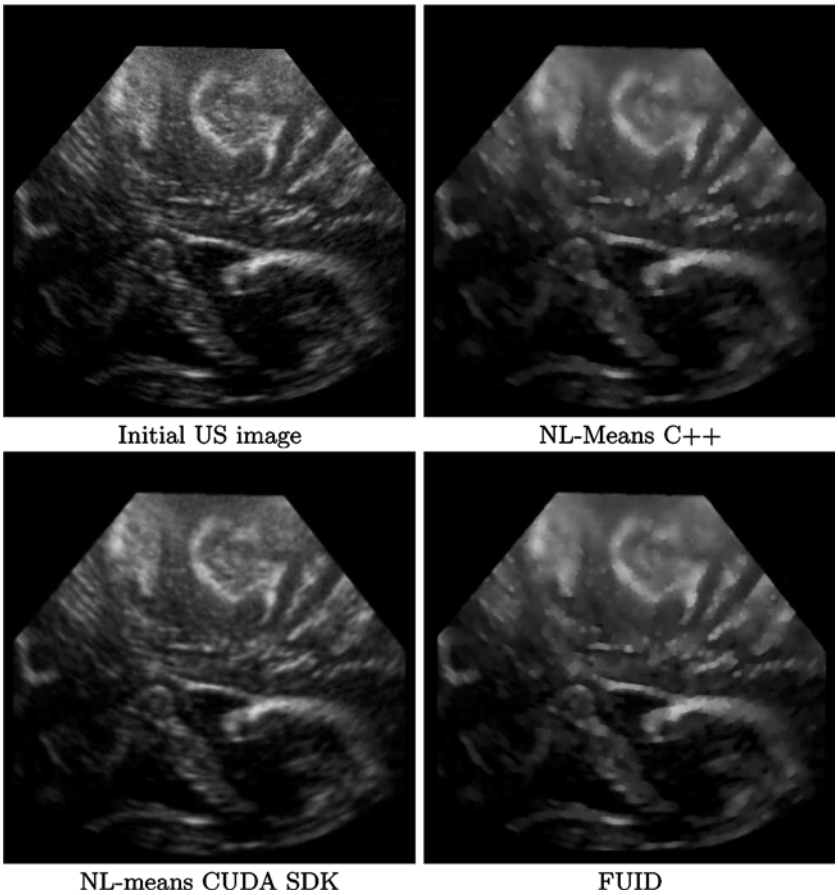

Fig. 3 Qualitative denoising results. A real intraoperative ultrasound was denoised with the three methods. Results show that the C++ implementation and FUID lead to comparable results, exhibiting the desired properties of the NL-mens technique: homogeneous areas are smoothed and edges are preserved. The CUDA SDK implementation is visually less satisfactory

procedure. The value of parameter h was computed using an analysis of variance described in [8]. In this context, ultrasound images are acquired to be registered toward pre-operative MR images to update the surgical planning. Results are presented in Fig. 3 and show that the C++ implementation and FUID lead to comparable results, exhibiting the desired properties of the NL-mens technique: homogeneous areas are smoothed and edges are preserved. The CUDA SDK implementation is visually less satisfactory. In a previously published paper [7], it has been shown objectively that the adapted NL-means performs better than the classical NL-means to denoise ultrasound images.

5.1.2 Quantitative validation

In order to evaluate the denoising filters with a relevant simulation of speckle noise, the validation framework proposed in [27, 28] was chosen for objective comparisons. This framework is based on Field II simulation [13]. Field II enables to generate realistic ultrasound images out of a echogenicity map. As a result, the input geometry of the images is perfectly known and can be used for quantitative validation. The image test Cyst is composed of three

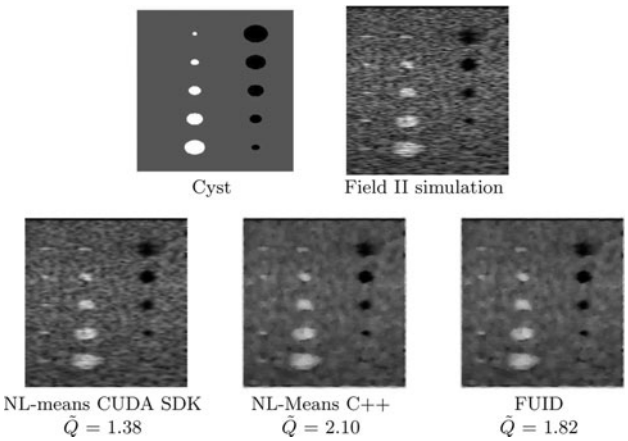

Fig. 4 Results obtained with optimal set of parameters for $\tilde{Q}$

constant classes C_r presented in Fig. 4. The result of the Field II simulation is converted to an 8-bit image of size 420 × 315 pixels. The geometry of the image is known, but not the true value of the image without speckle. Therefore, the authors introduced the ultrasound despeckling assessment index ($\tilde{Q}$) defined as

$$Q = \frac{\sum_{r \neq l} (\mu_{C_r} - \mu_{C_l})^2}{\sum_r \sigma^2_{C_r}} \quad (7)$$

Let us denote μ_{C_r} as the mean and $\sigma^2_{C_r}$ as the variance of class C_r after denoising. To avoid the sensitivity to image resolution, Q is normalized by Q_{id}. The new index $\tilde{Q} = Q/Q_{id}$ is high if the applied filter is able to produce an image with well-separated classes and small variances for each class. For these experiments we compared the FUID filter, the C++ implementation of NL-means filter, and the CUDA implementation of NL-means presented in the CUDA SDK (project named imageDenoising).

According to [27, 28], in this evaluation framework, the denoising quality is better when the $\tilde{Q}$ is high. FUID and C++ implementation present similar $\tilde{Q}$. This result was expected and validates the FUID which is based on C++ implementation.

In this experiment, the objects to be removed are composed of several pixels; thus the patch size is increased to evaluate the restoration performance of each object. The threshold (μ_1) was also modified in order to evaluate its influence. The parameters used for the test are given in Table 1.

The denoised images and the quantitative results are given in Fig. 4. Compared with the NLMmeans CUDA SDK implementation, the proposed adaptations for ultrasound images improved the $\tilde{Q}$ of the denoised image.

5.2 Computation time

5.2.1 Initial NL-means

For the experiments, five ultrasound images were used. The size of the images ranged from 360 × 288 to 1,080 × 864 pixels. We compared the computational time of FUID, C++ implementation of NL-means and CUDA SDK implementation of NL-means. Both FUID and imageDenoising were run using a 240-core NVIDIA GTX 280 GPU. The C++ implementation was performed on single core of a DualCore Intel Pentium CPU. The set of parameters of the three filters were the same. For a direct comparison with the NL-means implemented by imageDenoising, we used a classical version of FUID and C++ implementation (not adapted to ultrasound images).

The filter parameters included a neighborhood of 5 × 5 pixels and a search area of 11 × 11 pixels. The smoothing parameter was automatically computed for both the FUID and C++ implementation. The results are given below (see Fig. 5 and Tables 2, 3).

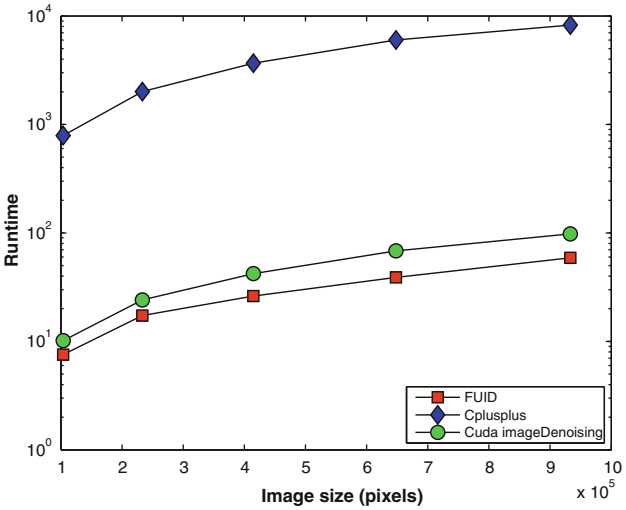

Fig. 5 Comparison: runtimes of FUID, C++ implementation of NL-means and CUDA SDK implementation of NL-means. To draw the three methods on the same plot, a logarithmic scale was used for the computation time. Results indicate that the computation time increases in a similar way for all methods. Our implementation is faster in all cases

5.2.2 Bayesian NL-means adapted to ultrasound images

We also compared the FUID with C++ implementation of NL-means both adapted for ultrasound images. Results are presented Fig. 6. We observe that FUID leads to the lowest computational time in both versions. Concerning the speedup of FUID, we observe in Fig. 7 a minimum speedup of 104.5 and a maximum speedup of 154.6 comparing with the C++ implementation.

6 Discussion and conclusion

In this paper, a modified version of the NL-means algorithm, adapted to a dedicated ultrasound noise model, was presented. The proposed algorithm was implemented using GPU technology and was compared with the C++ implementation, as well as to the NL-means approach implemented in the CUDA SDK. Results demonstrated that the proposed filter is very efficient in terms of denoising quality compared with state-of-the-art approaches.

Table 1 Set of parameters used for computational time tests

Filter	Smoothing parameter	Threshold μ_1	Neighborhood size	Search area
FUID	9.13	0.6	5 × 5	11 × 11
NL-means C++	9.13	0.6	5 × 5	11 × 11
NL-means CUDA	–	–	5 × 5	11 × 11

Table 2 Set of parameters used for computational time tests

Filter	Smoothing parameter	Threshold μ_1	Neighborhood size	Search area
FUID	9.13	0.95	5 × 5	11 × 11
NL-means C++	9.13	0.95	5 × 5	11 × 11
NL-means CUDA	9.13	–	5 × 5	11 × 11

The threshold does not apply to the NL-means CUDA implementation since the adaptive dictionary technique described in [8] was not used

Table 3 Computational time (ms) for different images size and improvement in percentage compared with the C++ implementation

Image size (pixels)	Computation time (ms)		
	NL-means's C++ implementation	NL-means's CUDA SDK implementation	FUID
360 × 288	790	10.17 (77%)	7.56 (103%)
540 × 432	2,010	24.14 (82%)	17.31 (118%)
720 × 576	3,670	42.17 (86%)	26.19 (140%)
900 × 720	6,010	68.34 (86%)	38.88 (153%)
1,080 × 864	8,240	98.06 (83%)	58.94 (141%)

Results indicate that the speedup of CUDA implementations is approximately 100 compared with the C++ implementation. In addition, our implementation is faster compared with the CUDA SDK one, with speedup factors ranging from 34% to 66% depending on the image size

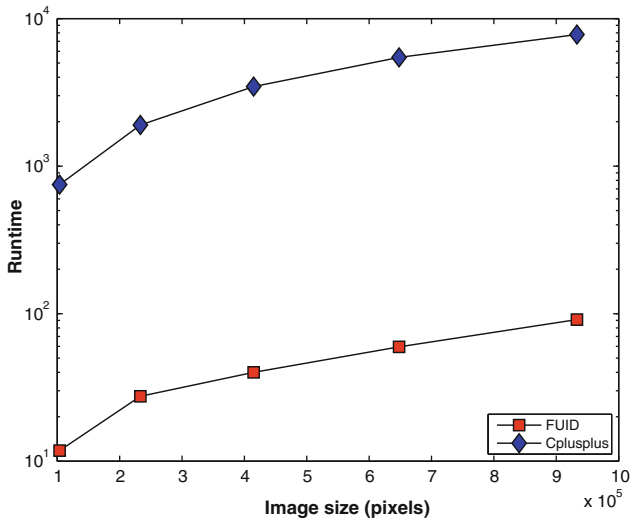

Fig. 6 Comparison: runtimes of FUID, C++ implementation of NL-means. The *y*-axis is in logarithmic scale

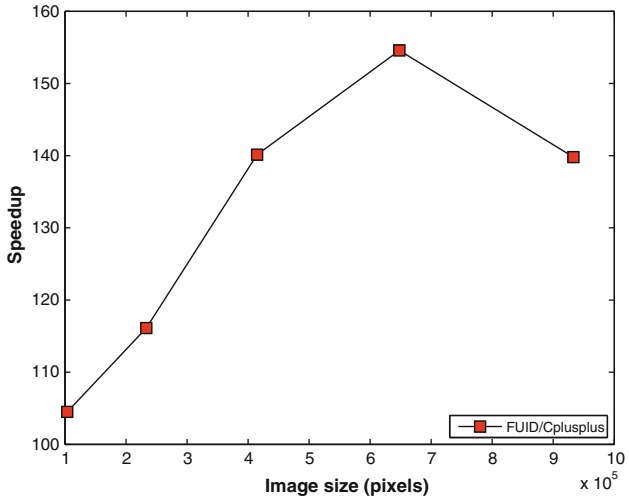

Fig. 7 Speedup of FUID/C++ implementation of NL-means

In terms of computation time, the proposed method is real-time: for medium sized images, denoising was performed in less than 20 *ms*, which is compatible with the vast majority of image processing workflows. In addition, our CUDA implementation was faster than the CUDA SDK implementation, with decrease varying between 34% to 66% depending on the image size.

Further work will investigate the use of the NL-means framework for additional image processing tasks. The patch-based framework is very general and can be declined for image super-resolution, image classification, and registration. These NL-means based algorithms will also be particularly suited for GPU implementation.

References

1. Alvarez, L., Lions, P.L., Morel, J.M.: Image selective smoothing and edge detection by nonlinear diffusion, II. SIAM J. Numer. Anal. **29**, 845–866 (1992)
2. Argenti, F., Torricelli, G.: Speckle suppression in ultrasonic images based on undecimated wavelets. EURASIP J. Adv. Signal Process. **2003**(5), 470–478 (2003)

3. Buades, A., Coll, B., Morel, J.M.: A review of image denoising algorithms, with a new one. Multiscale Model. Simul. **4**(2), 490–530 (2005)
4. Cates, J.E., Lefohn, A.E., Whitaker, R.T.: Gist: an interactive, gpu-based level set segmentation tool for 3D medical images. Med. Image Anal. **8**(3), 217–231 (2004)
5. Chan, T.F., Osher, S., Shen, J.: The digital TV filter and nonlinear denoising. IEEE Trans. Image Process. **10**(2), 231–241 (2001)
6. Chen, Y., Vemuri, B.C., Wang, L.: Image denoising and segmentation via nonlinear diffusion. Comput. Math. Appl. **39**(5–6), 131–150 (2000)
7. Coupé, P., Hellier, P., Kervrann, C., Barillot, C.: Nonlocal means-based speckle filtering for ultrasound images. IEEE Trans. Image Process. **18**(10), 2221–2229 (2009)
8. Coupé, P., Yger, P., Prima, S., Hellier, P., Kervrann, C., Barillot, C.: An optimized blockwise non local means denoising filter for 3D magnetic resonance images. IEEE Trans. Med. Imaging **27**(4), 425–441 (2008)
9. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE Trans. Image Process. **15**(12), 3736–3745 (2006)
10. Frost, V.S., Stiles, J.A., Shanmugan, K.S., Holtzman, J.C.: A model for radar images and its application to adaptive digital filtering of multiplicative noise. IEEE Trans. Pattern Anal. Mach. Intell. **2**, 157–165 (1982)
11. Hao, X., Gao, S., Gao, X.: A novel multiscale nonlinear thresholding method for ultrasonic speckle suppressing. IEEE Trans. Med. Imaging **18**(9), 787–794 (1999)
12. Huhle, B., Schairer, T., Jenke, P., Straßer, W.: Robust non-local denoising of colored depth data. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPR Workshops 2008, pp. 1–7 (2008)
13. Jensen, J.A.: Field: A program for simulating ultrasound systems. Med. Biol. Eng. Comput. **34**, 351–353 (1996)
14. Kervrann, C., Boulanger, J., Coupé, P.: Bayesian non-local means filter, image redundancy and adaptive dictionaries for noise removal. In: Proceedings of Conference on Scale-Space and Variational Methods (SSVM' 07), pp. 520–532, Ischia, Italy, June 2007
15. Kharlamov, A., Podlozhnyuk, V.: Image Denoising. http://developer.download.nvidia.com/compute/cuda/1_1/Website/projects/imageDenoising/doc/imageDenoising.pdf (2007)
16. Krissian, K., Vosburgh, K., Kikinis, R., Westin, C.-F.: Speckle-constrained anisotropic diffusion for ultrasound images. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2005)
17. Kuan, D.T., Sawchuck, A.A., Strand, T.C., Chavel, P.: Adaptive noise smoothing filter for images with signal-dependent noise. IEEE Trans. Pattern Anal. Mach. Intell. **7**(2), 165–177 (1985)
18. Kubias, A., Deinzer, F., Feldmann, T., Paulus, D., Schreiber, B., Brunner, T.: 2D/3D image registration on the gpu. Pattern Recognit. Image Anal. **18**(3), 381–389 (2008)
19. Lee, J.S.: Digital image enhancement and noise filtering by use of local statistics. IEEE Trans. Pattern Anal. Mach. Intell. **2**, 165–168 (1980)
20. Loupas, T., McDicken, W.N., Allan, P.L.: An adaptive weighted median filter for speckle suppression in medical ultrasound image. IEEE T. Circ. Syst. **36**, 129–135 (1989)
21. Motwani, M.C., Gadiya, M.C., Motwani, R.C., Harris, F.C. Jr.: Survey of image denoising techniques. In: Proceedings of GSPx, pp. 27–30. Citeseer (2004)
22. Osher, S., Burger, M., Goldfarb, D., Xu, J., Yin, W.: An iterative regularization method for total variation-based image restoration. Multiscale Model. Simul. **4**(2), 460–489 (2005)
23. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Pattern Anal. Mach. Intell. **12**(7), 629–639 (1990)
24. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D **60**, 259–268 (1992)
25. Tadmor, E., Nezzar, S., Vese, L.: A multiscale image representation using hierarchical (BV, L^2) decompositions. Multiscale Model. Simul. **2**(4), 554–579 (2004)
26. Tao, Z., Tagare, H.D., Beaty, J.D.: Evaluation of four probability distribution models for speckle in clinical cardiac ultrasound images. IEEE Trans. Med. Imaging **25**(11), 1483–1491 (2006)
27. Tay, P.C., Acton, S.T., Hossack, J.A.: A stochastic approach to ultrasound despeckling. In: Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium, pp. 221–224 (2006)
28. Tay, P.C., Acton, S.T., Hossack, J.A.: Ultrasound despeckling using an adaptive window stochastic approach. In: IEEE International Conference on Image Processing, pp. 2549–2552 (2006)
29. Wachowiak, M.P., Elmaghraby, A.S., Smolíkova, R., Zurada, J.M.: Classification and estimation of ultrasound speckle noise with neural networks. In: IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE'00), pp. 245–252 (2000)

Author Biography

Pierre Hellier graduated from "Supareo" in 1996 and obtained a master degree in applied mathematics from Toulouse University in 1996. He obtained his PhD on medical image registration in 2000. After 1 year as a postdoc at the Image Science Institute in Utrecht, the Netherlands, he was appointed as an Inria research scientist in 2001. In 2006, he was visiting professor at the Montreal Neurological Institute. He was program co-chair of the international conference "Medical Image Computing and Computer-Assisted Intervention" in 2004.