Real-time embedded video denoiser prototype

Andrea Petreto, Thomas Romera, Florian Lemaitre, Manuel Bouyer, Boris Gaillard, Patrice Menard, Quentin Meunier, Lionel Lacassagne

REAL-TIME AMBEDDED VIDEO DENOISER PROTOTYPE

9TH INTERNATIONAL SYMPOSIUM – OPTRO2020
OPTRONICS IN DEFENCE AND SECURITY
OECD CONFERENCE CENTER, PARIS, FRANCE /28 – 30 JANUARY 2020

Andrea Petreto(1,2), Thomas Romera(1,2), Florian Lemaitre(2), Manuel Bouyer(2), Boris Gaillard(1), Patrice Menard(1), Quentin Meunier(2), Lionel Lacassagne(2)

(1) LHERITIER, 10 rue de l'entreprise, 95 862 Cergy, apetreto@lheritier-alcen.com
(2)Sorbonne Université, CNRS, LIP6, F-75005 Paris, France, firstname.lastname@lip6.fr

KEYWORDS: Embedded systems, Video denoising, Real-time, Image processing

ABSTRACT:

Low light or other poor visibility conditions often generate noise on any vision system. However, video denoising requires a lot of computational effort and most of the state-of-the-art algorithms cannot be run in real-time at camera framerate. Noisy video is thus a major issue especially for embedded systems that provide low computational power. This article presents a new real-time video denoising algorithm for embedded platforms called RTE-VD [1]. We first compare its denoising capabilities with other online and offline algorithms. We show that RTE-VD can achieve real-time performance (25 frames per second) for qHD video (960x540 pixels) on embedded CPUs with an output image quality comparable to state-of-the-art algorithms. In order to reach real-time denoising, we applied several high-level transforms and optimizations. We study the relation between computation time and power consumption on several embedded CPUs and show that it is possible to determine find out frequency and core configurations in order to minimize either the computation time or the energy. Finally, we introduce VIRTANS our embedded real-time video denoiser based on RTE-VD.

1. INTRODUCTION

Forces, either from Police or Army are in great need for enhanced surveillance systems. One recurrent issue with those systems is the noise apparition in poor conditions. More particularly in low light situations or with fog, video can become so noisy that the range of detection, recognition and identification (DRI) is significantly reduced. Two main kind of video surveillance technology are commonly used: infrared (IR) and visible domain (VIS) cameras. The IR technology does not suffer from noise due to low light conditions as the VIS technology. However, if IR performs very well in terms of detection, it does not provide a lot of information for recognition. This is why VIS technology is still needed for many surveillance applications. Therefore, we need to provide solutions to improve video quality for VIS systems in order to push back the DRI limitation.

Noise reduction can be achieved in three different ways. We can improve the optic or the sensor, but we can also add a software-based video processing to enhance the video quality. The two first solutions are pretty expansive to apply and have their limitations. Especially the sensor improvement since modern sensors are able to achieve really low read noise level, so the preponderant noise is now the photon noise which can't be avoided. Software denoising is also a challenging task if we need to perform at real-time.

In this work we focus on real-time embedded video denoising. Single image denoising is a well-known research area and a lot of algorithms have been developed. They can be either based on pixel-wise filters [2-4], patch-based methods [5-6] or Convolutional Neural Network [7-9]. Compared to simple image denoising, video denoising is a much less explored domain. It brings however the additional temporal dimension. This dimension can be used to retrieve more information and perform better denoising.

Denoising algorithms can be classified into two criteria: the computation time and the denoising efficiency. In this work we consider heavily noisy video processed in real-time: 25 frames per second (fps) on embedded systems.

Some video denoising algorithms are able to deal with heavy noise situations but are too slow to be used for real-time denoising [10-15]. Moreover, they have a high latency as they use many frames to process the current one (typically 5-7). Some other algorithms are designed for real-time denoising [16] on embedded architectures [17], but they only suit to light noise like compression artifacts. As far as we know there is no real-time embedded video denoising algorithm able to deal with very noisy video as in this work.

Table 1. PSNR results for 7 sequences of the Derf's Test Media Collection on state-of-the-art methods and on our algorithm (RTE-VD)

Noise	Method	Crowd	Park_joy	pedestrian	Station	Sunflower	Touchdown	Tractor	overall
σ = 20	STMKF	26.25	25.29	28.34	26.66	26.97	28.87	25.37	26.70
	RTE-VD	26.38	25.65	30.58	30.98	32.51	30.17	29.38	28.73
	VBM3D	28.75	27.89	35.49	34.19	35.48	32.85	31.44	31.34
	VBM4D	28.43	27.11	35.91	35.00	35.97	32.73	31.65	31.11
σ = 40	STMKF	20.80	20.75	20.70	20.41	20.70	20.86	19.80	20.56
	RTE-VD	22.25	21.64	25.72	27.76	27.87	27.05	25.99	24.85
	VBM3D	24.81	23.78	30.65	30.62	30.21	30.21	27.82	27.43
	VBM4D	24.65	23.22	31.32	31.53	30.09	30.09	28.09	27.35

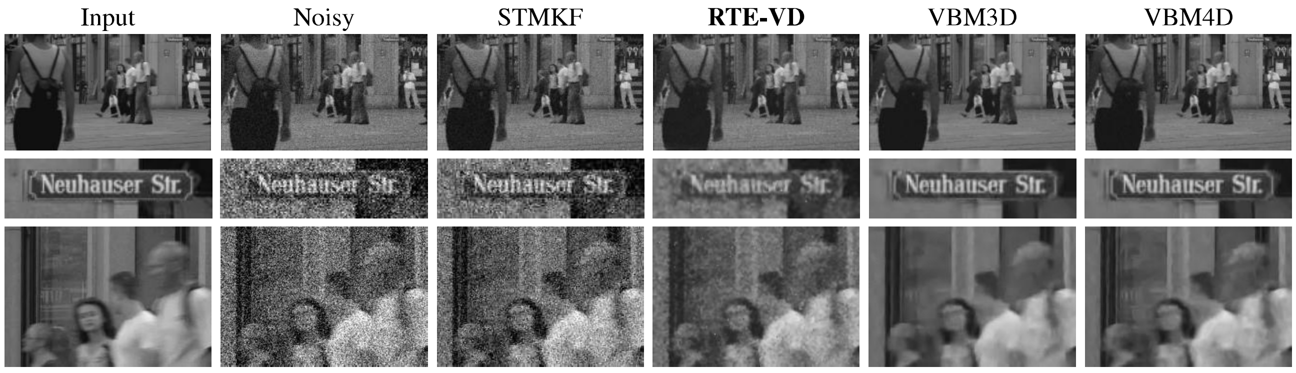

Figure 1. Visual comparison on the pedestrians sequence with a standard deviation noise of 40 (PSNR in Tab.1)

In this paper we present a new Real-Time Embedded Video Denoising (RTE-VD) algorithm in section 2. Then, we compare our method with other state-of-the-art algorithms in terms of denoising efficiency in section 3. We then present the main optimizations made to achieve real-time performances in section 4. In section 5, we compare RET-VD with other state of the art methods in terms of computation speed and we also provide a time versus energy consumption study for various embedded platforms. Finally, we introduce our embedded video denoiser prototype called VIRTANS in section 6.

2. DENOISING ALGORITHM

Most state-of-the-art video denoising algorithms rely on patch-based filtering methods [11] [14] [15]. These methods are able to provide a very effective noise reduction but are very time consuming. Their memory access pattern also generates a lot cache misses so it makes it difficult to optimize. For these reasons we did not consider patch search for a real-time implementation.

In the case of video denoising, we need to maintain a temporal coherence between frames. Usually it is ensured by the patch search. Since we do not consider this option, we must find another way. We decided to use an optical flow estimation [12] [18]. Optical flow estimation methods are also time consuming but are more sensitive to code transformations so they can be highly accelerated [19-23]. Many optical flow algorithms exist [24]. In order to perform a robust video denoising, the optical flow estimation must be dense, robust to noise and handle discontinuities within the flow [18]. Therefore, we chose the TV-L1 algorithm to compute the optical flow since it satisfies all these constraints [25].

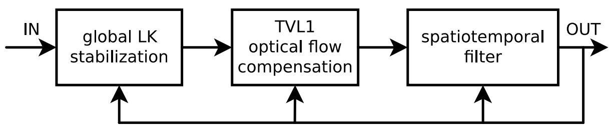

Figure 2. Main steps of the denoising chain RTE-VD

Our denoising chain RTE-VD is composed of 3 main steps visible in Fig. 2. First a stabilization step is applied using global one-pass Lucas-Kanade approach [26]. This step is used to compensate the camera movement. It eases the next step which is the computation of the optical flow using the TV-L1 algorithm. The flow enables to match past frame with the current one so we can apply the last filtering step without generating motion blur. This filtering step uses a 3D spatial-temporal bilateral filter approach [27].

3. DENOISING EFFICIENCY

In this section we compare RTE-VD performances with other state of the art algorithms in terms of denoising efficiency. To do so, we reproduced the comparison made in [28]. The video sequences are originally in colored 1080p format. We downscaled them to 960*540 grey-scale pixels. A Gaussian noise is added to each image with standard deviations of 20 and 40. All the video sequences are from the *Derf's Test Media Collection* [29].

We compared the PSNR value and visuals results of RTE-VD with two very popular offline methods: VBM3D [13] and VBM4D [14] and one state of the art method for real time denoising: STMKF [16]. Every time, we used the source code provided on the authors website with the default parameters depending on the noise intensity.

shows the PSNR results for all the considered methods and video sequences. Overall RTE-VD is less than 3 dB below VBM3D/4D while it is 2dB above STMKF for a noise deviation of 20 and more than 4dB above, for a noise deviation of 40. Given these results it appears that RTE-VD performs between VBM3D/4D and STMKF being closer from the 2 offline methods on very noisy situations.

Some visual results are presented in Figure 1. These results tend to confirm the PSNR results. We can see that RTE-VD brings a clear improvement compare to the noisy input. VBM3D and 4D are usually visually better. On the other hand, STMKF is much less effective that other methods. If RTE-VD provides an overall more detailed rendering than STMKF, it has some difficulties on static scenes. This may due to poor approximation of the optical flow on very small movements. Approximation errors are in this case indeed proportionally more important so it may generate a blur effect. This issue should be resolved by filtering the optical flow. This solution is considered for our future works.

4. ALGORITHM OPTIMIZATIONS

In order to perform real-time performances, RTE-VD had to be optimized and some trade off were made. Some transformations are specific to each step of the denoising chain. Some others described in [30] can be applied to every step. These high-level transformations are:

- SIMDization: handcrafted Neon instructions.
- Multi-thread parallelization (with OpenMP).
- Operator fusion: to reduce memory accesses.
- Operator pipeline: to increase memory locality.
- Modular memory allocation: to reduce memory footprint and enforce memory locality.
-

The efficiency of these type of optimizations on both speed and energy consumption has been shown in [31] and [32].

4.1. Global Lucas-Kanade stabilization

To stabilize the video stream, we use a modified version of the Lucas-Kanade method [26] with a global approach. The input image is convolved with a gate function, (a function that is 0 outside a specified interval and 1 inside it) in order to compute its average over a neighborhood. As the gate size depends on the largest movement to compensate, the convolution kernels can be very large. To speed the convolution up, integral images (also known as *summed area table* [33]) are used. The convolution is computed by multiple threads, each one processing a strip. Consequently, each thread only computes the partial integral image it needs. We then apply the Lucas-Kanade flow estimation to a neighborhood of the same size as the input images.

4.2. TV-L1 Dense optical flow estimation

The TV-L1 algorithm is the main step to optimize since it takes more than 90% of the total computation time. Since this step is so critical, we provided a more specific study of its optimizations in [19]. In this previous work we showed that the transformations we made result in a processing 5 times faster while being 6 times less power consuming on embedded ARM Cortex A57 CPU.

Also, since TV-L1 is an iterative method, we fixed the iterations to a given number, so the computation time is no more data dependent and remain constant. Important design choices were made for the TV-L1 computation. A lot of parameters have indeed a major impact on the optical flow precision versus speed tradeoff. In the end we used a 3 scales configuration with an unbalanced number of iterations and warp per scale. We compute 80 iterations with 4 warps at the most zoomed-out scale, 20 iterations with 2 warps at the medium scale and 3 iterations with 1 warp at the largest scale.

Optimization of optical flow estimation methods is crucial for many other embedded applications other than denoising. Our work on optical flow is also used for embedded meteors detection in a nanosatellite project [34-35].

4.3. Spatial-temporal filter

To actually filter out the noise, we compose a spatial bilateral filter [26] with a unilateral filter to handle the temporal dimension and speed the processing up. Thus, we are able to decorrelate the strength of the filter in the spatial domain from the temporal domain. The filter is defined in Eqs 1-3 where I_f is the filtered image, I_p is the previous compensated filtered image and I is the current image. x is the coordinates of the current pixel. Ω is the filter kernel domain while σ_i, σ_d, σ_t are the smoothing parameters of filter respectively of the spatial intensity difference, the distance and the temporal intensity difference between pixels.

Eq.1 $I_f(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I_t(x_i) e^{\frac{-[I_t(x_i) - I_t(x)]^2}{2\sigma_i^2}} \times e^{\frac{-[x_i - x]^2}{2\sigma_d^2}}$

Eq.2 $W_p = \sum_{x_{i \in \Omega}} e^{\frac{-[I_t(x_i) - I_t(x)]^2}{2\sigma_i^2}} \times e^{\frac{-[x_i - x]^2}{2\sigma_d^2}}$

Eq.3 $$I_t(x) = I_p(x)e^{\frac{-[I_p(x_i)-I(x)]^2}{2\sigma_t^2}} + I(x)\left(1 - e^{\frac{-[I_p(x_i)-I(x)]^2}{2\sigma_t^2}}\right)$$

In order to accelerate the computation, we approximate the bilateral filter as a separable filter. It has been shown in [36] that even if the bilateral is not separable, this remains a good approximation. We also used a fast approximation of the exponential function which manipulates the bit representation of floats as defined in the IEEE-754 standard. Such an approximation is described in [37] and is accurate enough for our application.

5. TIME & ENERGY CONSUMPTION

In this section we first compare the execution time of RTE-VD to the algorithms previously introduced. We then briefly study the impact of our optimizations on computation speed. Finally, we evaluate the performances of RTE-VD on various embedded CPUs and frequencies.

We consider 4 different platforms. The first one is an Intel Xeon Silver 4114 2×10C/20T@2.20GHz. The 3 others are the latest Nvidia Jetson embedded platforms; for those platforms, we only consider the ARM CPUs and not the GPUs. Their names and specifications are given in Tab.2.

Table 2 Technical specifications of the target embedded boards

Board	Process	CPU	Fmax (GHz)	Idle Power (W)
TX2	16nm	4xA57 + 2xDenver	2.00	2.0
AGX	12nm	8xCarmel	2.27	6.3
NANO	12nm	4xA57	1.43	1.2

5.1. Processing time and analysis

We compared the computation time of RTE-VD with the other methods that we considered in section 3. We ran RTE-VD, STMKF, VBM3D and VBM4D on the Xeon platform. We also tested RTE-VD and STMKF on the AGX platform. The results for 960x540 pixels images are presented int Tab.3. On the same platform RTE-VD is more than 200 times faster than VBM3D and more than 4600 times faster than VBM3D. On the AGX platform, RTE-VD is 2.5 times slower than STMKF but still achieves real-time processing with 26.7 frames per second.

Table 3 Denoising time depending on the used method and platform for 960x540 pixel images

Algorithm	Time (s)	Platform
STMKF	0.0045	Xeon
RTE-VD	0.0097	Xeon
VBM3D	2.0	Xeon
VBM4D	45	Xeon
STMKF	0.015	AGX
RTE-VD	0.037	AGX

To exhibit the efficiency of our optimizations, we compared our *Fast* implementation to a *Slow* straightforward implementation. The integral image computation is hard to parallelize without major transformations. Those transformations are only applied to the *Fast* version. In order to provide fair results, we compare the *Fast* and *Slow* versions in mono-thread and then analyze only the *Fast* version in multi-thread. The images used for the experiments are square images, with a width varying from 200 to 1500 pixels. The size of the image has negligible impact on the processing speed per pixel. Thus, 1000x1000 images are considered for the following experiments.

The results for all algorithms involved in RTE-VD on the AGX are presented in Tab.4. It shows that in mono-thread, we have an overall speedup of x18, and in multi-thread, a speedup of x70 using the 8 cores of the AGX. We can also observe that the major part of the speedup is obtained on the filtering, mainly due to its approximation with a separable filter. As a consequence, the optical flow estimation now takes almost 98% of the total time of the *Fast* version, while filtering was the most time-consuming part of the processing chain without the optimizations. Since the optical flow estimation is, in the end, the most consuming step, we produced a more detailed study in [19].
In this previous work the impact of each optimization is discussed. Both time and power consumption are considered.

Table 4 Execution time (ms) and speedup of RTE-VD on AGX CPU

Algorithm	*Slow* 1C	*Fast* 1C	*Fast* 8C	speedup
LK	6.66	1.59	0.37	x18
Flow	260.73	107.93	27.59	x10
Filter	1717.85	1.39	0.25	x6871
Total	1985.24	110.90	28.21	**x70**

5.2. Time & energy efficiency of RTE-VD

Since we target embedded systems, we have to consider not only the computation speed but also the energy consumption. Therefore, we ran RTE-VD at various frequencies on the three Nvidia embedded systems. The frequencies have been taken among the available frequencies of each board and the external memory frequency has been set to its maximum. For the AGX and NANO, we have used a multi-threaded version on all the physical cores, respectively 8 and 4. For the TX2, we have first used only the two Denver cores, then only the four A57 cores and finally all 6 cores. The cooling system of each target board has been set to the maximum and the energy saving policies of the Operating System have been deactivated. We have simultaneously measured time and power consumption for various frequencies and image sizes.

In order to perform simple and reproducible power measurements, the electrical consumption of the entire system has been measured. A board was

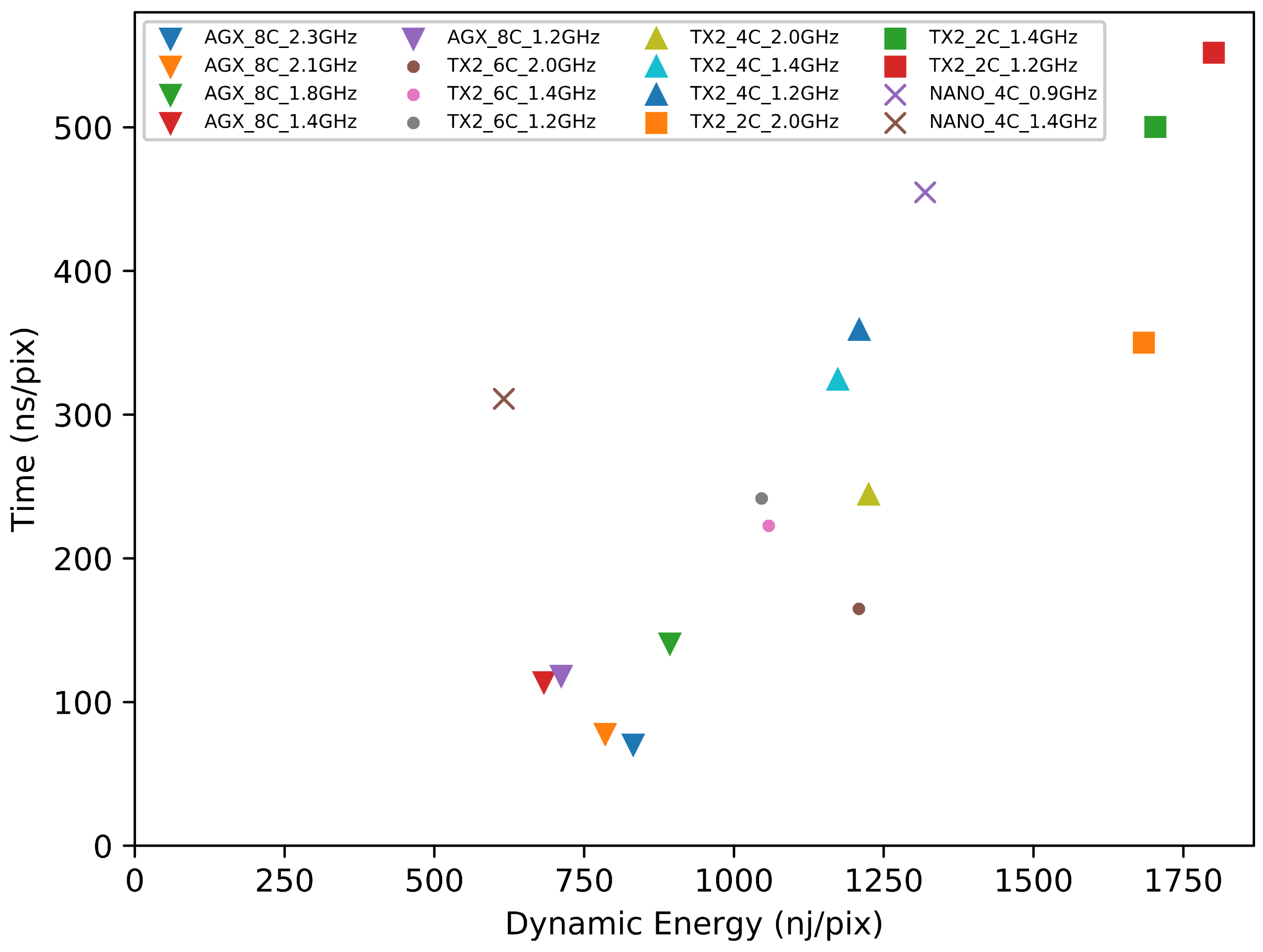

Figure 3 Speed and energy efficiency of RTE-VD depending on CPU architectures and frequencies

developed to this effect and has been inserted between the power source and the target system. The board samples both voltage and current at 5 kHz. Measurements and code executions are synchronized using GPIOs.

We can define four different metrics related to power consumption:

- ***Static energy:*** the energy associated to the static power when the system is idle (here, when the power consumption comes from the leakage and running the operating system)
- ***Dynamic energy:*** the energy associated to the dynamic power: that is the extra energy consumed by the computation: the total energy minus the static energy.
- ***Compute energy:*** the energy consumed by the whole system, that is the sum of static and dynamic energy.
- ***Period energy:*** the energy consumed by the whole system between two executions starts, including the waiting time.

The period energy is more relevant if we consider a complete system for which we know all of its applications and behaviors. Since this is not yet the case here, we only consider compute and dynamic energies in the following.

Our measurements show that considering the compute energy, the maximum frequency is always the most efficient for all configurations, being both the fastest and the least energy consuming. Considering dynamic energy, results show that there is a possible tradeoff between speed and energy consumption by selecting different frequencies values.

Fig.3 represents the computation time in nanosecond per pixel and the dynamic energy consumption in nanojoule per pixel for all the boards and frequencies. The NANO is the least energy consuming; however, it is possible to be 2.7x faster while consuming almost as little energy using the AGX. The AGX is the fastest since it is 2.3x faster than the fastest configuration of the TX2 and 4.4x faster than the fastest NANO configuration. Due to its etching process (16nm), the TX2 is often less energy efficient and slower than the two others (12nm). This is especially visible if we compare the NANO to the TX2 using only its A57 quad core at equivalent frequency. Since the NANO also possesses a quad core A57, it is able to compute as fast as the TX2 while consuming 1.6x less. However, the maximum frequency of the TX2 being higher, it is able to be faster than the NANO even using only the A57 cores. Therefore, it would be interesting to have a TX2 like architecture processed in 12nm. Since Nvidia announced for early 2020 a new embedded architecture with a computation power between the Nano and the AGX, the gap between the two platforms should be filled.

The Tab. 5 synthesizes the best configurations minimizing either energy consumption or computation time for each platform. It gives for each configuration the largest image size possibly processed at 25 frames per second.

Table 5 Best configurations for real-time denoising at 25 frame per second

Configuration	Energy (nJ/pix)	Time (ns/pix)	Max size (#pix)	Freq (GHz)
NANO min energy NANO min time	616	311	358	1.4
TX2 min energy	1046	242	406	1.2
TX2 min time	1209	165	492	2.0
AGX min energy	683	114	592	1.4
AGX min time	832	70	754	2.3

6. VIRTANS: EMBEDDED DENOISER

In this section we introduce the VIRTA family and its first born: the VIRTANS. VIRTA stands for Video Real-Time Algorithm and NS for Noise Suppression. VIRTA aims to provide a new embedded platform to perform heavy real-time video processing algorithms. It is currently in its early development stage and the first algorithm to be deployed on, is RTE-VD. With RTE-VD, VIRTA becomes VIRTANS. It is based on the industrial version of the TX2 architecture and comes in a pretty small form factor (cf Fig. 4).

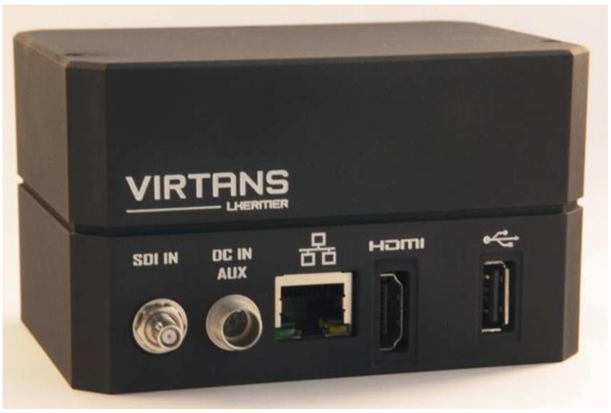

Figure 4 Current prototype of the VIRTANS platform

For now, video processed at 25 fps by VIRTANS, are 480x270 pixel large. We only use the 4 A57 CPU cores of the platform to run RTE-VD. As we have seen in Tab.5 and Fig.3 it is possible to have better performances using also the 2 Denver cores. In order to increase the size of the denoised image size we also plan to provide a GPU implementation. Our previous work on optical flow in [19] showed that for the Horn & Schunck optical flow estimation algorithm [38], GPUs are faster and more energy efficient than the CPU.

Our partial results on TV-L1 confirm that the TX2 GPU is 2.2x faster than the CPU. Since the GPU implementation is currently an ongoing study, we are not able to provide more results for now. As future works, we plan to provide a complete GPU implementation of RTE-VD. The goal is actually to integrate a hybrid CPU/GPU implementation into VIRTANS to benefit from the best of both architectures.

VIRTANS is compatible with the Camera from LHERITIER. It receives an SDI video flow and is able to communicate with the camera. This communication can be used to retrieve useful information from the camera sensors. For example, the optical sensor to get information about the noise intensity, or a gyroscopic sensor to get information about the camera movements. The denoised video is live displayed on any screen via an HDMI port.

7. CONCLUSION

In this article, we present a novel real-time embedded video denoising chain called RTE-VD, which is able to restore details on very noisy video while achieving high performance on embedded systems.

In order to achieve real-time processing, we applied several code transformations (like SIMDization, multi-threading, operator fusion and pipelining) and were able to get an implementation 70x faster than a naive one. The PSNR and visual results from our experiments validates our approach.

We thus compared RTE-VD to other state-of-the-art algorithms: VBM3D, VBM4D and STMKF. RTE-VD always denoises better than STMKF while being less effective than the more costly algorithms VBM3D and VBM4D. On heavy noise situations (σ=40), RTE-VD achieves an overall PSNR more than 4dB above STMKF.

While RTE-VD is 2.5x slower than STMKF, it is still able to process 960x540 pixels video at 25 fps on a Nvidia Tegra AGX. Given these results, we believe that RTE-VD is a denoising algorithm particularly well positioned for speed/accuracy tradeoff.

Since we are targeting embedded systems, we also studied the link between time and energy consumption on various embedded CPUs. Within the tested platforms, it appears that the Nano is the least power consuming platform while the AGX is the fastest. We were also able to determine for each platform multiple efficient frequencies minimizing either computation time or energy consumption.

Finally, we introduced VIRTANS: our first real-time embedded video denoiser prototype based on the TX2 architecture and on RTE-VD. VIRTANS comes in a small form factor and can be plugged to a LHERITIER camera to perform live denoising.

As future work, we plan to increase the whole performance by balancing the load on the CPU and the GPU and by studying what parts of the algorithm can be hybridized with 32 and 16-bit computation. 16-bit computation has shown good speed versus accuracy results on simpler algorithms including optical flow estimation [39-42][44]. We have also leads to shrink further the form factor of VIRTANS. Finally some step from the denoising chain will be used for the other VIRTA applications like turbulences rejection or scene pre-segmentation to focus operator attention [43].

8. ACKNOWLEDGMENT

The authors would like to thank the DGA for the support they provide to this project through a “Thèse Cifre Défense”.

9. REFERENCES

[1] A. Petreto, T. Romera, F. Lemaitre, I. Masliah, B. Gaillard, M. Bouyer, Q. L. Meunier et L. Lacassagne, «A New Real-Time Embedded Video Denoising Algorithm,» chez *IEEE International Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 2019.

[2] M. Zhang et B. K. Gunturk, «Multiresolution bilateral filtering for image denoising,» *IEEE Transactions on image processing,* vol. 17, pp. 2324-2333, 2008.

[3] T. Chen, K.-K. Ma et L.-H. Chen, «Tri-state median filter for image denoising,» *IEEE Transactions on Image processing,* vol. 8, pp. 1834-1838, 1999.

[4] A. Buades, B. Coll et J.-M. Morel, «A review of image denoising algorithms, with a new one,» *Multiscale Modeling & Simulation,* vol. 4, pp. 490-530, 2005.

[5] A. Buades, B. Coll et J.-M. Morel, «A non-local algorithm for image denoising,» chez *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[6] M. Lebrun, A. Buades et J.-M. Morel, «A nonlocal Bayesian image denoising algorithm,» *SIAM Journal on Imaging Sciences,* vol. 6, pp. 1665-1688, 2013.

[7] H. C. Burger, C. J. Schuler et S. Harmeling, «Image denoising: Can plain neural networks compete with BM3D?,» chez *2012 IEEE conference on computer vision and pattern recognition*, 2012.

[8] V. Jain et S. Seung, «Natural image denoising with convolutional networks,» chez *Advances in neural information processing systems*, 2009.

[9] K. Zhang, W. Zuo, Y. Chen, D. Meng et L. Zhang, «Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising,» *IEEE Transactions on Image Processing,* vol. 26, pp. 3142-3155, 2017.

[10] C. Zuo, Y. Liu, X. Tan, W. Wang et M. Zhang, «Video denoising based on a spatiotemporal Kalman-bilateral mixture model,» *The Scientific World Journal,* vol. 2013, 2013.

[11] P. Arias, G. Facciolo et J.-M. Morel, «A Comparison of Patch-Based Models in Video Denoising,» chez *2018 IEEE 13th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, 2018.

[12] A. Buades et J.-L. Lisani, «Video Denoising with Optical Flow Estimation,» *Image Processing On Line,* vol. 8, pp. 142-166, 2018.

[13] K. Dabov, A. Foi et K. Egiazarian, «Video denoising by sparse 3D transform-domain collaborative filtering,» chez *2007 15th European Signal Processing Conference*, 2007.

[14] M. Maggioni, G. Boracchi, A. Foi et K. Egiazarian, «Video denoising using separable 4D nonlocal spatiotemporal transforms,» chez *Image Processing: Algorithms and Systems IX*, 2011.

[15] P. Arias et J.-M. Morel, «Towards a bayesian video denoising method,» chez *International Conference on Advanced Concepts for Intelligent Vision Systems*, 2015.

[16] S. G. Pfleger, P. D. M. Plentz, R. C. O. Rocha, A. D. Pereira et M. Castro, «Real-time video denoising on multicores and GPUs with Kalman-based and Bilateral filters fusion,» *Journal of Real-Time Image Processing,* pp. 1-14, 2017.

[17] J. Ehmann, L.-C. Chu, S.-F. Tsai et C.-K. Liang, «Real-Time Video Denoising on Mobile Phones,» chez *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018.

[18] C. Liu et W. T. Freeman, «A high-quality video denoising algorithm based on reliable motion estimation,» chez *European Conference on Computer Vision*, 2010.

[19] A. Petreto, A. Hennequin, T. Koehler, T. Romera, Y. Fargeix, B. Gaillard, M. Bouyer, Q. L. Meunier et L. Lacassagne, «Energy and Execution Time Comparison of Optical Flow Algorithms on SIMD and GPU Architectures,» chez *2018 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 2018.

[20] T. Kroeger, R. Timofte, D. Dai et L. V. Gool, «Fast Optical Flow using Dense Inverse Search,» chez *(ECCV)*, 2016.

[21] A. Plyer, G. L. Besnerais et F. Champagnat, «Massively parallel Lucas Kanade optical flow for real-time video processing applications,» *Journal of Real-Time Image Processing,* Vols.

%1 sur %211,4, pp. 713-730, 2016.

[22] M. Kunz, A. Ostrowski et P. Zipf, «An FPGA-optimized architecture of horn and schunck optical flow algorithm for real-time applications,» chez *International Conference on Field Programmable Logic and Applications (FPL)*, 2014.

[23] L. Bako, S. Hajdu, S.-T. Brassai, F. Morgan et C. Enachescu, «Embedded Implementation of a Real-Time Motion Estimation Method in Video Sequences,» *Procedia Technology,* vol. 22, pp. 897-904, 2016.

[24] Middlebury, *Optical Flow Database http://vision.middlebury.edu/flow/.*

[25] C. Zach, T. Pock et H. Bischof, «A duality based approach for realtime TV-L 1 optical flow,» chez *Joint Pattern Recognition Symposium*, 2007.

[26] B. D. Lucas, T. Kanade et others, «An iterative image registration technique with an application to stereo vision,» 1981.

[27] C. Tomasi et R. Manduchi, «Bilateral filtering for gray and color images,» chez *null*, 1998.

[28] A. Davy, T. Ehret, G. Facciolo, J.-M. Morel et P. Arias, «Non-Local Video Denoising by CNN,» *arXiv preprint arXiv:1811.12758,* 2018.

[29] Derf, *Derf's Test Media Collection https://media.xiph.org/video/derf/.*

[30] L. Lacassagne, D. Etiemble, A. Hassan-Zahraee, A. Dominguez et P. Vezolle, «High Level Transforms for SIMD and low-level computer vision algorithms,» chez *ACM Workshop on Programming Models for SIMD/Vector Processing (PPoPP)*, 2014.

[31] H. Ye, L. Lacassagne, D. Etiemble, L. Cabaret, J. Falcou et O. Florent, «Impact of High Level Transforms on High Level Synthesis for motion detection algorithm,» chez *IEEE International Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 2012.

[32] H. Ye, L. Lacassagne, J. Falcou, D. Etiemble, L. Cabaret et O. Florent, «High Level Transforms to reduce energy consumption of signal and image processing operators,» chez *IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2013.

[33] F. C. Crow, «Summed-area tables for texture mapping,» chez *ACM SIGGRAPH computer graphics*, 1984.

[34] N. Rambaux, D. Galayko, G. Guignan, J. Vaubaillon, L. Lacassagne, P. Keckhut, A. C. Levasseur-Regourd, A. Hauchecorne, M.Birlan, G. Augarde, S. Barnier, S. B. Kemmoum, A.Bigot, P. Boisse, M. Capderou, A. Chu, F. Colas, F. Deshours, Y. Fargeix, A. Hennequin, T. Koehler, M. Lumbroso, J.-F. Mariscal, D. Portela-Moreira, J. Raffard, J.-L. Rault, T. Romera, C. Tob et B. Zanda, «METEORIX: a cubesat mission dedicated to the detection of meteors,» chez *42nd Assembly of Committee on Space Research (COSPAR)*, 2018.

[35] N. Rambaux, J. Vaubaillon, L. Lacassagne, D. Galayko, G. Guignan, M. Birlan, M. Capderou, F. Colas, F. Deleflie, F. Deshours, A. Hauchecorne, P. Keckhut, A. C. Levasseurd-Regourd, J. L. Rault et B. Zanda, «Meteorix: a cubesat mission dedicated to the detection of meteors and space debris,» chez *ESA Space Safety Programme Office, NEO and Debris Detection Conference (ESA NDDC)*, 2019.

[36] T. Q. Pham et L. J. Van Vliet, «Separable bilateral filtering for fast video preprocessing,» chez *2005 IEEE International Conference on Multimedia and Expo*, 2005.

[37] N. N. Schraudolph, «A fast, compact approximation of the exponential function,» *Neural Computation,* vol. 11, pp. 853-862, 1999.

[38] B. K. P. Horn et B. G. Schunk, «Determining optical flow,» *ACM Computing Surveys (CSUR),* vol. 17, pp. 185-203, 1981.

[39] S. Piskorski, L. Lacassagne, S. Bouaziz et D. Etiemble, «Customizing CPU instructions for embedded vision systems,» chez *Computer Architecture, Machine Perception and Sensors (CAMPS)*, 2006.

[40] L. Lacassagne, D. Etiemble et S. Kablia, «16-bit Floating Point Instructions for embedded Multimedia Applications,» chez *CAMP: Computer Architecture and Machine Perception*, 2005.

[41] D. Etiemble, L. Lacassagne et S. Bouaziz, «Customizing 16-bit floating point instruction on a NIOS II processor for FPGA image and media processing,» chez *Estimedia - Embedded Systems for Real-Time Multimedia*, 2005.

[42] D. Etiemble et L. Lacassagne, «Introducing image processing and SIMD computations with FPGA soft-cores and customized instructions,» chez *Workshop on Reconfigurable Computing Education (WRCE)*, 2006.

[43] K. Aneja, F. Laguzet, L. Lacassagne et A. Merigot, «Video rate image segmentation by means of region splitting and merging,» chez *IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, 2009.

[44] L. Lacassagne et D. Etiemble, «16-bit floating point operations for low-end and high-end embedded processors,» chez *ODES: Optimizations for DSP and Embedded Systems*, 2005.