An FPGA-based design for real-time Super-Resolution Reconstruction

Yoan Marin, Johel Mitéran, Julien Dubois, Barthélémy Heyrman, Dominique Ginhac

Noname manuscript No.
(will be inserted by the editor)

An FPGA-based design for real-time Super-Resolution Reconstruction

Yoan MARIN · Johel MITERAN · Julien DUBOIS · Barthélémy HEYRMAN · Dominique GINHAC

Received: date / Accepted: date

Abstract For several decades, the camera spatial resolution is gradually increasing with the CMOS technology evolution. The image sensors provide more and more pixels, generating new constraints for suitable optics. As an alternative, promising solutions propose Super Resolution (SR) techniques to reconstruct high-resolution images or video without modifying the sensor architecture. However, most of the SR implementations are far from reaching real-time performance on a low-budget hardware platform. Moreover, convincing state-of-the-art studies reveal that artifacts can be observed in highly textured areas of the image. In this paper, we propose a Local Adaptive Spatial Super Resolution (LASSR) method to fix this limitation. LASSR is a two-step SR method including a machine learning-based texture analysis and a fast interpolation method that performs a pixel-by-pixel SR. Multiple evaluations of our method are also provided using standard image metrics for quantitative evaluation and also a psycho-visual assessment for a perceptual evaluation. A first FPGA-based implementation of the proposed method is then presented. It enables high-quality 2k to 4k super-resolution videos to be performed at 16 fps, using only 13% of the FPGA capacity, opening the way to reach more than 60 fps by executing several parallel instances of the LASSR code on the FPGA.

Keywords FPGA · Super-Resolution · real-time texture analysis · spatial interpolation · real-time implementation

All authors are with Université Bourgogne Franche-Comté (UBFC), 9 Avenue Alain Savary, 21000 Dijon, FRANCE
E-mail: firstname.lastname@u-bourgogne.fr

1 Introduction

Continuous advancements in imaging technology have boosted the emergence of cameras featuring smaller pixels and higher resolution, allowing to capture finer details. In most applications, high-resolution contents (i.e. images or video) are desired for better human perception and often required for efficient image analysis and automatic interpretation. So, the development of such megapixel sensors has been naturally pulled by the pressing technological needs of several key markets such as medical, security or broadcasting. And today, Ultra High Definition (UHD) cameras, able to capture up to 8k video (8192 x 4320 pixels), are commercially available. Unfortunately, constructing such cameras with high precision optics is a very costly process. It is one of the most important issues in many real applications, limiting their use only to specific niche markets. Moreover, increasing sensor resolution has also additional limitations on camera performance. It tends to slow the acquisition framerate, the storage, and the data processing because of the higher number of pixels. Important hardware resources (i.e high-speed interfaces, powerful processor) are required to cope with this unprecedented bandwidth, making them not compliant with real-time embedded systems constraints.

So, to satisfy the high demand for UHD content, an alternative and affordable solution is Super Resolution (SR). It is a set of algorithmic techniques reconstructing high-resolution images or videos from one or multiple low-resolution observations of the same scene, thereby increasing the high-frequency components and removing the degradation caused by the imaging process of the low-resolution camera[56]. It relies on off-the-shelf optical components and lower resolution cameras (LR, typically < 2k) combined with image processing for up-

scaling video into higher resolutions (HR, up to 8k). Since the pioneering works by Tsai and Huang[50], SR has been an area of very active research[3,39,38], resulting in many research papers, each describing a new SR technique for a specific purpose (see [36]for a taxonomy covering the different types of SR algorithms). With the rapid shift of traditional computer vision to deep learning methods in recent years, deep learning-based SR models have been actively explored[53]. SR has also received a significant boost in performance using deep learning-based methods and often achieves the state-of-the-art performance[23,28,22,10,49]. Several reasons can explain that SR has been a spotlighted area for the research community over the last two decades. First, SR is a challenging problem because it is a computationally complex and numerically ill-posed problem, leading to a multiplicity of solutions for a given set of observation images[12]. Secondly, SR produces video sequences with more level of details, improving the performance of computer vision applications, typically in medical imaging, in satellite imaging or surveillance[45,60,55,58]. Finally, algorithmic SR may be more cost-effective than a hardware approach because any existing LR imaging system with low-cost optical components can be re-used[4,40,43].

Historically, the seminal SR algorithms were based on image processing techniques in the frequency domain. For example, Tsai and Huang[50] proposed the first multiple-image SR algorithm using a discrete Fourier transform. HR images were built from the relationship between consecutive images acquired by Landsat satellite, exploiting the relative motion between the LR images. Other techniques such as discrete cosine transform[44] and wavelet transform methods[6] have also been proposed. However, despite high computational efficiency, frequency-domain algorithms have also serious limitations, limiting their applications. So, nowadays, the main trend in SR is the spatial domain using two different approaches: multi-image (MISR) or single image (SISR). Multiple image-based algorithms named "Reconstruction-based SR" assume that LR instances of the same scene are available. MISR methods overcome the inherent resolution limitation by exploiting the explicit redundancy of information available in subpixel shifts of LR images. They usually perform a geometric registration and blur estimation at first to compensate for the misalignment between the LR images, and finally, recover details by a fusion step. Iterative back projection (IBP) methods[41] were among the first methods developed for spatial-based MISR. Typical other approaches include maximum likelihood (ML)[27], Maximum A Posteriori (MAP)[18] or Projection Onto Convex Sets (POCS)[48]. These methods significantly counterbalance the disadvantages of frequency-domain methods but require more computationally intensive operations (i.e. registration and fusion), the accuracy of which directly impacts the quality of the final result[37]. While MISR relies on the complementary information available in LR images to build HR images, SISR uses only one single image for this task. SISR algorithms named "Example-based SR"[14] or "Image Hallucination"[2] often use machine learning or deep learning techniques, trying to hallucinate the missing information of the LR images[36,58]. The principle is to learn correspondences between LR and HR images patches from a database of LR and HR images pairs. During the HR computation, an LR patch is analyzed, and the prediction step of the machine learning algorithm recovers its more likely HR version.

Even if state-of-the-art SISR and MISR techniques appear as efficient alternatives to achieve great success in SR reconstruction accuracy, they both suffer from heavy computational cost. Most of the above-mentioned works on SR mainly focus on the mathematical algorithms behind SR and the ability to overcome issues such as noise, non-uniform blur, and motion estimation errors to reach high accuracy[4]. Executing these algorithms at the framerate of the imaging system remains a challenging task for high-performance desktop computers, and may be unreachable for low-power embedded processors[43]. Some of the literature addresses the hardware implementation with a parallel acceleration of SR techniques[16]. Focus is made on the design of highly parallelizable algorithms and dedicated implementations to try to achieve real-time SR reconstruction using CPU[42], GPU[21,57] or FPGA[47,30,25]. Nevertheless, most of these SR implementations are far from reaching ideal performance as well as scalability or energy efficiency. So, in this paper, inspired by the above-mentioned state-of-the-art, we propose a hardware-friendly single image super-resolution method and its dedicated hardware implementation on an FPGA. Overall, the contributions of this paper can be summarized as follows:

- We propose LASSR, a "Local Adaptive Spatial Super-Resolution" method that is compliant with real-time embedded constraints. LASSR is a two-step SR method including a machine learning-based texture analysis and a fast interpolation method that performs a pixel-by-pixel SR. According to the texture analysis, an edge enhancement algorithm can be applied to reveal finer details and better contrast.
- We provide a thorough evaluation of the LASSR method. The quantitative evaluation relies on widely used metrics in SR problems (i.e. PSI and SSIM) to compare the output of SR to the ground truth. Since

it is known that such metrics may correlate poorly with human perception, we also provide a Psycho-Visual Assessment (PVA) in which forty-five participants are asked to give scores to assess the image quality based on their visual perception.
– We design a real-time implementation of the LASSR method that demonstrates the generation of 4K high-quality content from 2K sources on an embedded FPGA board while limiting run-time and hardware budget.

The remainder of the paper is organized as follows: Section 2 proposes an analysis of some real-time single image super-resolution demonstrating their limitations in the low-frequency textured image regions. Section 3 introduces our "Local Adaptive Spatial Super-Resolution" (LASSR) method which includes a real-time classification-based texture analysis in the resolution enhancement process. Performance is evaluated using a psycho-visual metric. Finally, Section 4 describes the dedicated hardware FPGA-based implementation of this method which enables a high image quality to be maintained while achieving real-time processing on high spatial resolution images (2K, 4K).

2 Analysis of limitations of some real-time interpolation-based SR methods

2.1 Interpolation-based SR methods

Sophisticated SR methods are usually not compliant with hardware limitations. Simpler interpolation-based methods are then considered for increasing resolution by a magnification factor of 2, typically in TV applications with Full-to-Ultra-High-Definition conversion or 4K-to-8K conversion[24]. However, a basic interpolation of a single image is unable to recover the high-frequency components of the image and can not be considered as an SR technique[39]. So interpolation-based SR techniques frequently adopt a two-step process, starting with an interpolation step to match the target resolution, followed by a quality enhancement step to add high-frequency information in the HR image. Such methods, sometimes called "Super interpolation", intelligently incorporate both the simplicity of interpolation and the quality enhancement capability of SR. In [8], the proposed method utilizes edge-orientation-based pre-learned kernels. It requires an off-line training phase in which LR patches are clustered based on their edge orientations. During the on-line up-scaling phase, HR images patches are generated by applying a linear mapping function based on the edge orientation of the corresponding LR input patches. In [24], this method is improved to fit with the hardware requirements of an FPGA. The resulting implementation reconstructs 4K UHD HR images in real-time. A similar approach, described in [20] combines a neural network and an interpolation-based method. It crops each frame into blocks, measures their total variation values, and dispatches them accordingly to the neural network or to the interpolation module for upscaling. The hardware FPGA implementation achieves real-time for a Full-HD-to-UHD conversion.

All the above-mentioned super-interpolation techniques are example-based SR working on image patches or blocks to generate their corresponding SR version. The missing high-resolution information is assumed to be learned during the training phase from the LR/HR pairs in the database. Not only it is not guaranteed that the reconstructed HR patch provides true details, but also the training phase is a long process requiring a large set of images.

On the contrary, Gohshi[17] proposes an SR method without the use of any neural network technique. He describes a real-time method for increasing image spatial resolution, suitable for FPGA implementation. The principle of this method is to add to the interpolated image a proportion of high frequencies computed using a Laplacian operator, as depicted in Fig. 1, where the output named GFI (i.e Gohshi Filtered Image) is a two-times (2X) up-scaled image in both horizontal and vertical directions.

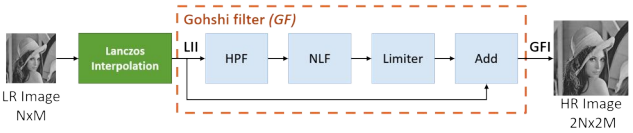

Fig. 1 Gohshi's method

The first block is a standard interpolation based on the Lanczos method[29] allowing to upscale the original image by factor 2. The interpolation step is followed by a high pass filter (HPF) based on a 5x5 Laplacian, then the result is amplified using a nonlinear function (NLF) (cube value of the Laplacian). A limiter ensures that the output value of the NLF is in the range $[-255, 255]$. Then, the final merging between the initial interpolated image and the computed high frequencies is performed using a weighted addition.

The method is powerful and provides high-quality SR images. Nevertheless, significant artifacts appear in some "smooth" regions, i.e. the low spatial frequencies regions, as depicted in Fig. 2, in which the edges are well enhanced in the area defined by rectangle 1 whereas noise is highly visible in rectangle 2.

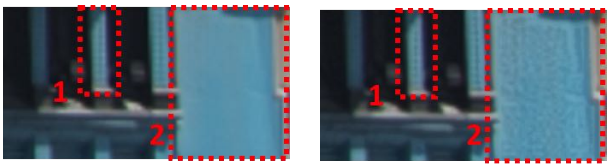

Fig. 2 Differences between a Lanczos Interpolation (left) and Gohshi's method (right) on both areas 1,2

Another limitation is illustrated in Fig. 3. Gohshi's process increases and may saturate the luminance in the "smooth" areas, losing details and original information of the scene. Therefore, the human visual perception of the image is significantly impacted, particularly in the case of high-resolution video broadcasting and screening. In smooth areas, a simple Lanczos interpolation does not saturate as well as not generate information loss.

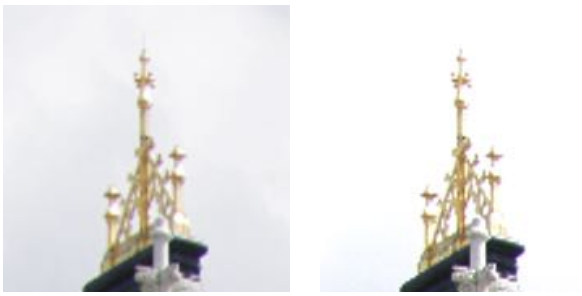

Fig. 3 Differences between an Lanczos Interpolation (left) and Gohshi's method (right). The resulting image is saturated, details are lost in high levels of luminance regions

2.2 Evaluation of the quality of interpolation-based SR methods

To quantify the impact of the information loss and the artifacts in smooth regions, we compared the Gohshi's performance with a standard Lanczos interpolation, according to the level of the image spatial frequencies. We built a data set S_1 of 350 patches P manually extracted from 35 pictures representing various scenes, including several landscapes and studio portraits (8k pictures captured with a camera Nikon D800). Each patch has a 128x128-pixel size, and has been annotated by an human expert by using a label $y_i = \{LF, MF, HF\}$ where LF, MF, and HF are respectively the label for texture of low, medium and high frequencies. Thus, the set S_1 is defined as follow: $S_1 = \{P_i, y_i\}_{i=1}^{350}$. Some samples are presented in Fig. 4.

For each patch, three criteria have been evaluated, quantifying the intrinsic quality of the image: the Structural SIMilarity[52] (SSIM), the Perceptual Sharpness Index[13] (PSI), and a Psycho-Visual Assessment[31] (PVA). SSIM is a well-known metric correlated with the quality perception of the human visual system, used to measure the similarity between two images. SSIM models the image distortion as a combination of loss of three factors: the luminance, the contrast, and the correlation. SSIM is a full reference metric that compares the processed SR image to the ground truth SR image.

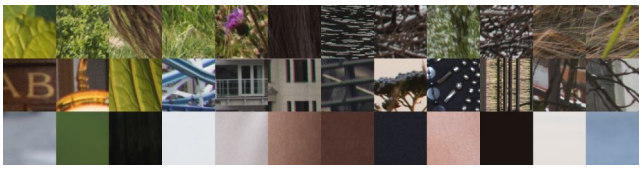

Fig. 4 Example from the data set S sorted by texture , up to bottom : 1^{st} row : high frequencies 2^{nd} row: medium frequencies, 3^{rd} row : low frequencies

PSI is a no-reference metric, based on a statistical analysis of local edge gradients. This method takes into account human visual system properties: it correlates highly with human perception and exhibits low computational complexity. Both values of SSIM and PSI are close to 1 when the image enhancement is of good quality, and close to 0 in the opposite case.

Psycho-Visual Assessment (PVA) consists of the use of human observers who score image quality during experiments. Contrary to objective measures such as SSIM or PSI, PVA provides a subjective measure of the perceptual quality of images. In our studies, the participants have been selected with good skills in computer vision because we assumed that experts in computer vision are the best panels for evaluating the quality of SR images. Participants were informed about the main goal of the study (i.e. evaluate the quality of reconstructed high-resolution images) but they were unaware of the different algorithms used to build the SR images. Following the expertise criterion, the recruitment process has been limited only to academic researchers and PhD students from our laboratory and also to students from the local engineering school in electrical engineering and computer science. For each of the two studies, flyers were posted, and messages have been sent to mailing lists of the laboratory and the engineering school to recruit the different participants. For this first evaluation, twenty participants were recruited. They had to evaluate 350 pairs of images produced with the Lanczos Interpolated Image (LII) and the Gohshi Filtered Image (GFI) algorithms. The duration of the experiment was about 30 to 40 minutes for each participant. For each trial, the LII and GFI images were randomly displayed side by side on the screen. Moreover, the LF, MF and HF frequencies were also randomly dis-

tributed to avoid any bias. For each pair of SR images, participants were asked to select the image that corresponds to the highest perceived quality. The response time and the ranking were recorded. The response time has been systematically used to verify that the participants spent enough time before ranking the images. For this first study, no outliers were identified and eliminated. The analysis of ranking was then performed on all the twenty participants' data by computing the percentage of LII and GFI images classified respectively in the LF, MF and HF categories, as depicted in Table 1.

Table 1 Comparison of Gohshi Filtered Image (GFI) and Lanczos Interpolated Image (LII)

		HF	MF	LF
GFI	PSI	0.25	0.23	0.48
	SSIM	0.89	0.88	0.93
	PVA(%)	**76.48**	**63.05**	40.54
LII	PSI	0.27	0.20	0.47
	SSIM	0.91	0.91	0.94
	PVA(%)	23.52	36.95	**59.46**

As shown in Table1, the difference between LII and GFI is not significant with the SSIM and PSI assessments. However, quality metrics are originally designed to mainly account for image signal and noise rather than human visual perception. Several authors[46,35, 61] highlight that full-reference metrics such as SSIM fail to match the visual perception of human subjects well for SR performance evaluation.

Regarding PVA, the GFI results outperform the LII for medium to high frequencies (MF and HF). However, the difference between MF and HF is not significant. Therefore, it is possible to merge these two classes into only one class named MHF for the following experiments. For low frequencies (LF), the best enhancement is obtained using the Lanczos interpolation method. Therefore, this evaluation demonstrates that the image texture impacts the quality of the Gohshi's results. As expected, this approach is efficient for the highly textured part of the image. Nevertheless, it adds noise in the low textured part where a standard Lanczos interpolation is preferable. Moreover, Gohshi's method increases the global luminance of the image.

Based on this observation, we propose a hybrid approach combining Gohshi and Lanczos methods. This new approach is named LASSR for Local Adaptive Spatial Super-Resolution. LASSR uses a local texture analysis and classification to automatically select the appropriate algorithm. More precisely, each pixel is processed using one of the two methods according to a texture image analysis. LASSR is compliant with real-time constraints and can be implemented on an embedded vision system.

3 LASSR Method

The LASSR method, depicted in Fig. 5, can be viewed as an upgrade of the Gohshi's filter, adding two processing steps: a Texture Classification (TC) and a Merging step, designed respectively to overcome the limitations previously highlighted: noise in low-frequency areas and saturation of the luminance.

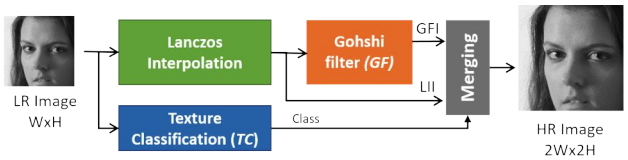

Fig. 5 LASSR method

3.1 Texture Classification

The Texture Classification step is based on a standard machine learning workflow, including the definition of a training dataset, features extraction, training, and final pixel-wise classification (as seen in Fig. 6). The decision phase enables the texture analysis to be performed on-line based on the model generated during the off-line learning step. The output of the classifier (pixel class) is a binary information (LF or MHF) and can be used to select either GFI or LII in the merging operation.

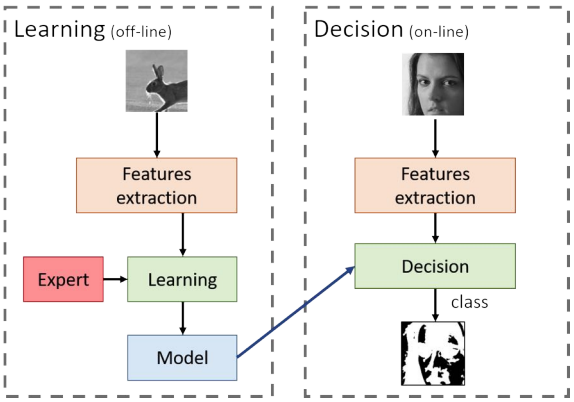

Fig. 6 Machine learning based texture analysis: off-line learning, on-line classification

3.1.1 Training DataSet

A learning phase is required and is performed off-line, based on a set of patches annotated by a human ex-

pert, according to their texture. Example of patches from the two different classes are shown in Fig. 7 where high frequencies patches are represented on the top row, and low frequencies patches on the bottom row. Thirty five 8k photographies have been used to extract a total number of 2000 patches with a distribution of 1000 patches per class. Thus, the labelled dataset is defined by $S_2 = \{P_i, y_i\}_{i=1}^{2000}$ with $y_i = \{LF, MHF\}$.

Fig. 7 Sample of textured patches: High frequencies(MHF) patches (1^{st} row), low frequencies(LF) patches (2^{nd})

3.1.2 Classification methods

Numerous methods are suitable for embedded classification, such as Neural Networks[59], Support Vector Machine[51], Boosting[15], Random Forest[5], k-NN[9], etc. Here, a very fast decision step is required, since the decision frequency is the pixel frequency. We developed in previous studies some specific FPGA-based hardware accelerators for SVM and Boosting[33]. During this study, we developed a tool allowing the user to find automatically the appropriate trade-off between classification performance and hardware implementation cost for SVM and Boosting. This tool is based on the LIBSVM library[7] and an adaptive boosting method for the estimation of the SVM and boosting parameters. For SVM, this tool uses a Radial Basis Function (RBF) kernel (K) of parameter σ with the main objective to automatically optimize the classification rate. For this purpose, the SVM decision function providing the class C of a feature vector F of dimension d is presented in Eq. 1.

$$C(F) = sgn\left(\sum_{i=1}^{N_v} y_i \epsilon_i K(S_i, F) + b\right) \quad (1)$$

where ϵ_i are the Lagrange coefficients, and b a constant value obtained during the optimization process. The separating plane is constructed from those N_v input vectors, for which $\epsilon_i \neq 0$. These vectors $S_i, i = 1, ..., N_v$, of label $y_i \in \{-1, +1\}$, are *support vectors* and reside on the boundary margin.

For Boosting, the basic idea of this method is to build a "strong" classifier from "weak" ones, focusing at each iteration on misclassified samples. The resulting Boosting decision function providing the class C of a feature vector F is presented in Eq. 2.

$$C(F) = sgn\left(\sum_{t=1}^{T} \lambda_t h_t(F)\right) \quad (2)$$

Where both λ_t and $h_t(F)$ are to be acquired by the training boosting procedure. h_t is a threshold operator applied to k component of the F vector. The threshold value and the k values are also determined during the training process. They are considered as constant values during the real-time decision step, therefore they can be integrated into the architecture. In this case, the output $C_i \in \{0, 1\}$ corresponds respectively to the classes MHF and LF.

Both for SVM and Boosting, our tool automatically estimates the classification global error, the FPGA resources and generates the SVM and Boosting classification model in various formats (C++, VHDL, ROM configuration file). Specifically, for SVM, an approximation of the decision boundary is described automatically in VHDL using a combination of hyper-rectangles. In case of a 2D feature vector, the ROM file containing the decision function, compatible with Xilinx software, is automatically generated.

For Boosting, the VHDL decision function is automatically generated, as well as the C++ header containing the definition of all constant values of the decision function (thresholds values of h_t, polarity associated with these thresholds, and λ_t coefficients). The C++ header can then easily be included in the High-Level Synthesis description of the decision function and implemented using, for instance, Vivado HLS.

Despite the powerful of SVM training and the low-complexity of the decision algorithm, the hardware implementation of the decision function is still hardware resource consuming, due to the RBF kernel K and the high number of support vectors. If the feature vector dimension d is lower than 4, it is possible to discretize the feature space and store the whole decision in an embedded FPGA memory configured as a ROM. For example, if $d = 2$, the ROM contains the decision function in a 256x256 feature space. If $d > 4$, it can be more efficient to implement the Boosting decision function.

3.1.3 Feature extraction for texture analysis

A large number of features are available in the literature to quantify the local frequencies around the pixel $p(i, j)$ to be classified. Classical approaches are generally based on the Fourier Transform[32], the Wavelet Transform[1], the Haralick features[19], or standard statistics.

Due to the hardware implementation constraints, we defined a set of features based on standard statistics

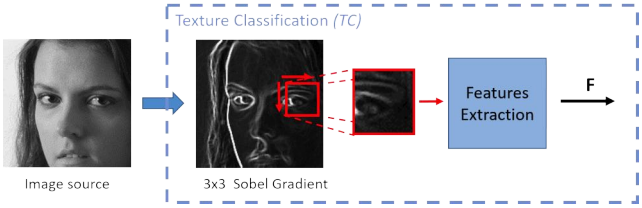

Fig. 8 Feature extraction based on Sobel Gradient

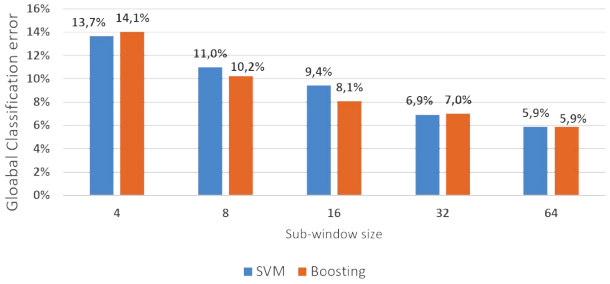

Fig. 9 Global classification error and FPGA resources evolution for SVM and Boosting method with different sub-window size.

computed from the Sobel gradient image SGI, which is a high-pass filter. A neighborhood or sub-window N_1 of size $n \times n$ centered around each pixel $p(i,j)$ to be classified is defined, where (i,j) are the coordinates of the considered pixel. In the following experiments, the impact of N_1 size has been considered as larger neighborhoods enable larger texture information to be captured. The mean $\overline{X}$ of SGI and the variance σ^2 of SGI are computed in N_1. A second sub-window N_2 of size 3×3 is defined in order to capture local texture information. The nine gradient values in N_2 are used as sub feature vector component named $\mathbf{G_2}$. Finally, The mean $\overline{X_2}$ of SGI is computed in N_2. The resulting feature vector is $\mathbf{F}(i,j) = \{\overline{X}, \sigma^2, \mathbf{G_2}, \overline{X_2}\}$.

3.1.4 Classification performance and feature selection

To choose the best classification method, we had to find the best trade-off between global classification error and hardware resources (complexity, FPGA resources utilization). For this purpose, we studied the impact of the parameter n on SVM and Boosting performance with $n = 4, 8, 16, 32, 64$. The dimension of the initial set of features $\mathbf{F}(i,j)$ is $d = 12$ and then is reduced using the boosting ability so select discriminative features. Results are presented in Fig. 9. Differences between SVM and Boosting are not significant: Boosting obtains slightly better scores for $n = 8$ and $n = 16$ while SVM lightly outperforms for the other values of n . The best trade-off between classification performance and complexity is then the implementation of the boosting method. Moreover, a powerful intrinsic property of boosting is to perform feature selection automatically during the training step. Finally, boosting algorithm selection removed all feature components except σ^2. The final feature vector is $\mathbf{F}(i,j) = \{\sigma^2\}$. This allows to implement the decision function with a single iteration of Eq. 2 or using a simple 256x1 bit ROM. The hardware resource utilization depends mainly on the architecture of the filter needed for feature extraction (see Section 4.3 for a detailed study).

We provide in Fig. 10-a an example of texture classification for a portrait. High Frequencies pixels are depicted in white and low frequencies in black. As expected, hairs, eyes, mouth or nose contours, will be sharpened, whereas background or "smooth" skin will remain unchanged. The same statement is observed with low frequencies areas of Fig. 10-b (London) where smooth areas (sky and cloud) are considered as no-textured areas, and all the details in regions with textures like tower's stones will be enhanced. In Fig. 10-c (Vienna), near all the wall details will be enhanced.

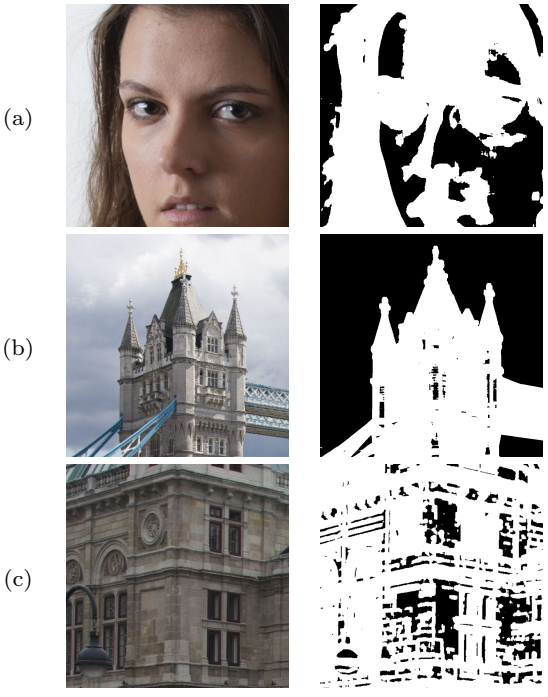

Fig. 10 (a) Original image "Portrait" *(left)* and Texture Classification *(right)*, (b) Original image "London" *(left)* and Texture Classification *(right)*, (c) Original image "Vienna" *(left)* and Texture Classification *(right)*

3.2 Merging

The role of this step is to merge the Lanczos Interpolated Image LII with the Gohshi filtered image GFI according to the texture class y. Since luminance of GFI and LII are slightly different, it is necessary to perform a local equalization, taking into account the local mean of luminance of both images. The merged image is computed as follows (Eq. 4):

$$LASSR(i,j) = \begin{cases} LII(i,j) \times (1 - y(i,j)) + \\ GFI_{eq}(i,j) \times y(i,j) \end{cases} \quad (3)$$

where (i,j) are the coordinates of the considered pixel, $LII(i,j)$ is the luminance value of interpolated image, and the $GFI_{eq}(i,j)$ is the equalized luminance of the Gohshi's image, computed using the mean of their respective luminance in 5×5 neighborhoods:

$$GFI_{eq}(i,j) = GFI(i,j) + (\overline{X_{LII}(i,j)} - \overline{X_{GFI}(i,j)}) \quad (4)$$

where

$$\overline{X_{LII}(i,j)} = \frac{1}{25} \sum_{k=i-2}^{2+i} \sum_{l=j-2}^{2+j} LII(k,l) \quad (5)$$

and

$$\overline{X_{GFI}(i,j)} = \frac{1}{25} \sum_{k=i-2}^{2+i} \sum_{l=j-2}^{2+j} GFI(k,l) \quad (6)$$

Examples of merging with and without local equalization are presented in Fig. 11. Since the luminance of GFI (see Fig. 11-a) is higher and sometimes saturated compared to the LII (see Fig. 11-b), an halo appears around areas classified as textured (see Fig. 11-c). These artifacts are eliminated by the local luminance equalization (see Fig. 11-d).

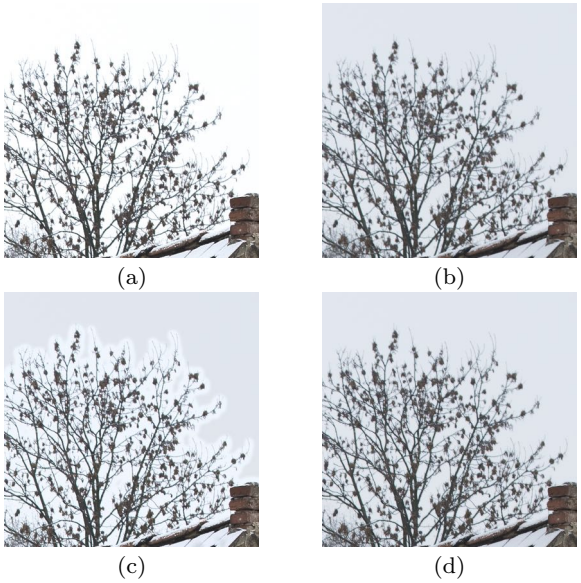

Fig. 11 Merging with and without local equalization: (a) GFI with high or saturated values on the sky, (b) LII with lower values, (c) apparition of halo around the trees after merging GFI and LII, (d) no halo on merged image after local luminance equalization

3.3 LASSR evaluation

3.3.1 Experimental setup and dataset

To validate the LASSR method, we also conducted experiments using a Psycho-Visual Assessment (PVA) as in [46]. For this second PVA evaluation, twenty-nine new participants were recruited according to the inclusion criterion. None of them had participated in the first study described into 2.2. After analysis of the response times, the data of four participants have been eliminated and the PVA ranking results have been evaluated using the twenty-five remaining participants. The main goal of this second study was to evaluate the quality of the proposed LASSR algorithm compared with GFI and LII methods. The participants were informed about this goal but they were unaware of the different algorithms used to build the SR images and they were also unaware of the detailed purpose of the experiment (i.e. compare the quality of the proposed LASSR algorithm versus other algorithms). For each trial, they were shown the ground truth image on the upper left of the screen along with the SR images (obtained with LASSR, GFI and LII algorithms) randomly displayed on the three other quadrants of the screen. For this purpose, we used a desktop computer with its calibrated 4K screen that displayed simultaneously the 4 images at a 1024x1024 spatial resolution. The participants were asked to rank the three SR images from higher to lower perceived quality, by clicking first on the image featuring the higher quality, and next on the second-ranked image. After the selection of the two first images, the last one was automatically ranked third. For ranking the images, participants have received the following instructions:

- A high-quality SR image must provide a high level of precise details;
- The color and luminance of a high-quality SR image must match the ground truth image as precisely as possible.

We built a dataset S_2 of 150 different images with a 1024x1024-pixel resolution extracted from 8k-resolution

images. The dataset includes various scenes (urban and scenic landscapes, portraits, animals, etc.) with different levels of details. Image samples from S_2 are depicted in Fig. 12. The participants had to evaluate all the 150 images of the dataset during a test session. The average evaluation time was around 22 minutes to rank the full dataset.

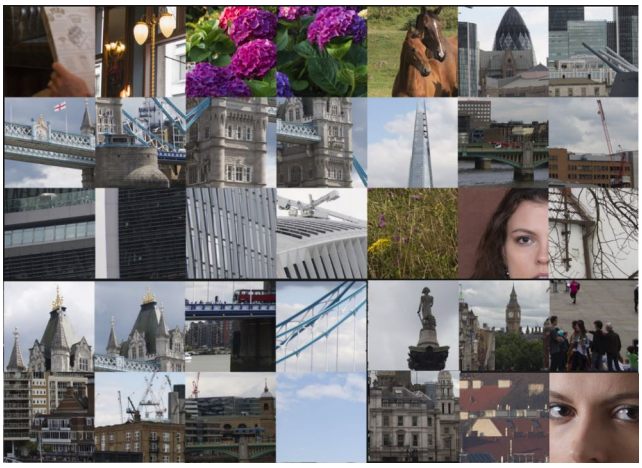

Fig. 12 Example of 1024 × 1024 sub-images from S_2 used for PVA

3.3.2 Objective and subjective evaluation of the quality of LASSR images

The LASSR method has been implemented and evaluated in Matlab. Three example images from the dataset S_2 are presented in Fig. 13: a portrait, an urban landscape, and a forest. The first row is the ground truth while the subsequent rows display the SR images obtained respectively with LII, GFI and LASSR methods. For each image, we also provide a detailed zoom of a specific region to highlight the details generated by the different algorithms. The global luminance, as well as the colors of the LASSR resulting images, is quite similar to the ground truth images. Furthermore, there are no significant local luminance artifacts between textured and no-textured areas (for example in hair and skin details of the portrait image) validating the local luminance equalization of the GFI processed before the merging operation.

Table 2 SSIM and PSI metrics for GFI, LII and LASSR

	GFI	LII	LASSR
SSIM	0.79	**0.92**	0.87
PSI	**0.31**	0.25	0.30

Concerning the objective metrics, SSIM and PSI have been evaluated for each image of the dataset and their mean values are shown in Table 2. The values of SSIM and PSI are not significantly different for GFI, LII, and LASSR. For SSIM, LASSR gets a score of 0.87 close to the higher value (0.92) obtained with LII. For PSI, LASSR obtains performance (0.30) similar to GFI (0.31). One common feature of the objective image quality metrics is that they emphasize the importance of distortions on the image structure. Based on the consensus that the structure is dominant in the human visual perception, quality metrics have been mainly designed to efficiently detect the presence of severe image distortions but they are unable to evaluate the subtle differences present in the SR images[35] Today, most of the SR algorithms succeed to accurately reproduce the image structure. The main difference between the SR algorithms is their capacity to precisely generate image details. Since metrics such as SSIM or PSI are unable to evaluate image details with enough precision, they can not reflect image quality well and are not best suited for SR[54].

Table 3 presents the PVA results obtained with a panel of 25 participants. Contrary to quality metrics, the PVA results highlight a significant difference between the three algorithms. Table 3 indicates that LASSR is ranked as the best method to improve image resolution while preserving a good level of details. With a score of 40%, LASSR outperforms GFI (32%) and LII (28%). By combining the ranks 1 and 2, LASSR obtains a huge score of 84%, compared to 61% for GFI and 55% for LII. Finally, LASSR is only considered as the worst method in 16% of cases, a very low score compared to 39% for GFI and 45% for LII.

To summarize, the quality metrics SSIM and PSI are unable to significantly exhibit the best SR method but the PVA rankings experimentally validate the visual quality of the proposed method LASSR for generating SR images with a high level of details.

Table 3 PVA ranking results for GFI, LII and LASSR

Rank	GFI	LII	LASSR
1^{st}	32%	28%	**40%**
2^{nd}	29%	27%	**44%**
3^{rd}	39%	**45%**	16%

4 Hardware implementation

The LASSR's method is efficient in terms of visual perception as demonstrated in Section 3. Therefore, to

Fig. 13 Visual comparison for three ground truth patches with a 1024x1024-pixel resolution cropped from 8k images (a) with processed SR images obtained with Lanczos Interpolation *LII* (b), Goshi's method *GFI* (c) and proposed method *LASSR* (d).

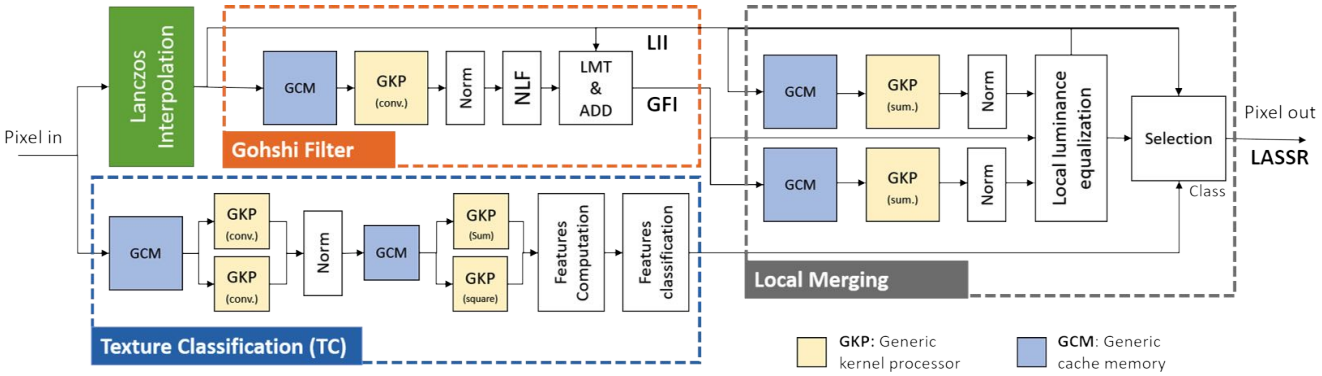

Fig. 14 Block diagram of the LASSR method architecture highlighting the multiple use of 2-D filter, gradient, mean $\bar{X}$ and variance σ^2 computation, local luminance equalization

achieve our goal, we propose a specific hardware description to reach real-time processing on an FPGA target. As a proof of concept, a demonstrator has been developed on a Xilinx VC707 development board including a Virtex7 FPGA.

The hardware description of LASSR has been manually developed in VHDL and Verilog to have better control on the architecture design and optimizes the performance. Nevertheless, to limit the demonstrator development time, a rapid prototyping strategy based on Matlab/Simulink has been considered. This prototyping method requires two steps for a demonstrator validation:

1. *A HDL Co-Simulation* that interfaces the Matlab workspace with the HDL simulator (ModelSim) with the objective to make a Hardware/Software validation. It consists of simulating the entire HDL core with ModelSim. For this purpose, Matlab/Simulink enables a high level of abstraction description of the data transfer to be achieved: input images are sent by Simulink to ModelSim and simulation outputs are saved in the Matlab workspace using an internal socket that performs block by block communication.
2. *A Co-Design method*, named "FPGA-in-the-loop" (FIL) in Simulink, to interface the Matlab workspace with the Xilinx VC707 board. Simulink uses Vivado from Xilinx to generate the FPGA configuration file and to program the FPGA board. This method gives access to the FPGA resource utilization and the maximum clock frequency. The FPGA-in-the-loop methodology also automatically provides the communication channel (i.e. an Ethernet connection in our case) for sending and receiving data between Simulink and the FPGA board.

Consequently, using FPGA-in-the-loop simulation, most of the development time has been focused on the design of the LASSR IP core architecture. The validation of the LASSR demonstrator (in simulation and on-board) has been simplified and significantly accelerated by the Simulink development environment that automatically generates the data interfaces between the validation environment and the FPGA board.

The proposed architecture for a hardware implementation of the LASSR method is depicted in Fig. 14. This architecture details the three main steps previously described in Fig. 5: the Gohshi's filter, the texture classification, and the final merging. These three steps, based on a pipeline architecture, are processed simultaneously increasing the task's parallelism. Moreover, based on the fact that multiple 2-D filtering is required in these three main blocks, we developed a generic HDL description which enables the kernel size and coefficients to be customized, before the HW generation, according to the user requirements. The generic model description enables high performance, in s of processing and resulting pixel's throughput, to be guaranteed. Indeed, for any kernel size, the proposed design enables the pixel's neighborhood to be processed at each clock cycle.

The generic filter model including the Generic Cache Memory (GCM) and the Generic Kernel Processor (GKP) are detailed in Section 4.1. All blocks are detailed in the following sections.

4.1 Generic filter model

As different 2-D filters with different kernel parameters (size, coefficients) are requested in the hardware description of the LASSR method, we designed a customizable 2-D filter. This generic filter regroups two generic components: a Generic Cache Memory (GCM) and a Generic Kernel Processor (GKP). GCMs are represented with *blue blocks* in the global implementation (Fig. 14) and GKPs with *yellow blocks*. For each block, the kernel size is specified in brackets.

The ***Generic Cache Memory (GCM)*** guarantees parallel access to all the pixels of a neighborhood. The architecture overview of this component is depicted in Fig. 15.

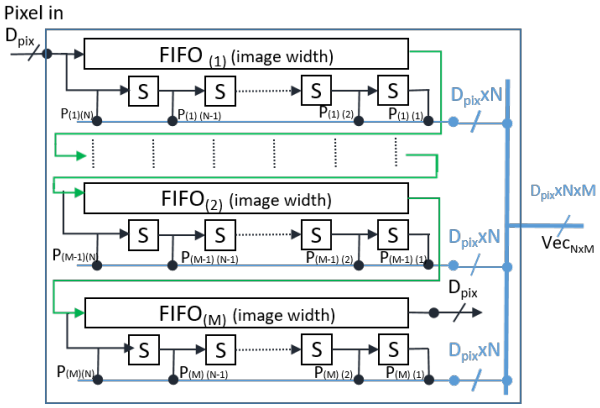

Fig. 15 Description of Generic Cache Memory implementation

According to the user's configuration, the cache memory enables a $W \times H$ pixel neighborhood (where W and H are respectively its width and its height) to be accessible in only a clock cycle. The neighborhood size, as well as the input pixel dynamic (D_{pix}), are fixed according to the user's constraints before the core synthesis. According to the user's customization (i.e. the kernel size and the image width), the number of delay lines (FIFO) and the Shift registers (S) are automatically fixed and synthesized. Each delay line is generated using the FPGA's embedded memory blocks and the memory's depth is fixed according to the image width. The core concatenates the pixels of the neighborhood in one vector ($Vec_{(W \times H)}$) and enables all kernel pixels to be accessible in one clock cycle. The $Vec_{(W \times H)}$ dynamic is automatically fixed at $W \times H \times D_{pix}$ bits.

The ***Generic Kernel Processor (GKP)*** has been described to be specifically bounded to the GCM (see Fig. 16). The generic description of GKP enables three processing to be considered. Indeed, the user can select either the mean $\bar{X}$ or the mean of the squared $\bar{X^2}$ or a 2-D convolution using the configurable kernel. Each processing is performed on the considered pixel neighborhood.

According to the user's configuration, the appropriated number of multipliers is fixed for the first pipeline stage of the generated architecture. For a $W \times H$ neighborhood, the GKP generates $2^{\max(W,H)}$ signed multipliers. Each coefficient is bound to a multiplier and a pixel neighborhood. The coefficient precision (number of bits) is a parameter, meanwhile, all internal signal

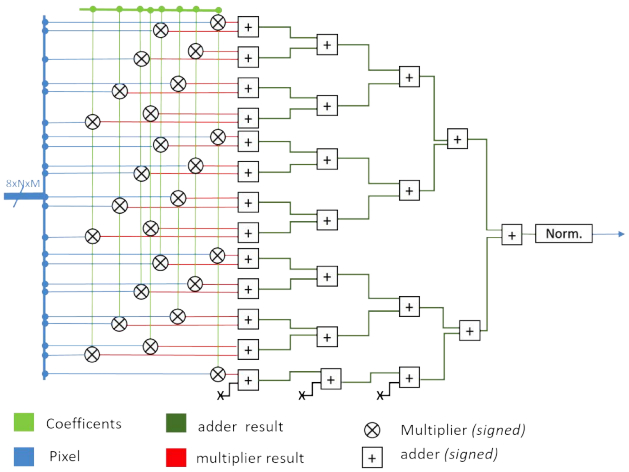

Fig. 16 Description of Generic Kernel Processor implementation processing a 5×5 2-D convolution

precision is automatically fixed according to the selected user's configuration. The core generates $ceil(log2(W \times H))$ stages of the signed adders. An optional normalization is included to control the GKP's output precision.

4.2 Gohshi Filter (GF)

Gohshi's method requires a classical interpolation, a high pass filter, a nonlinear function, and a final adder. The proposed architecture, depicted in Fig. 17, enables the implementation of this method and each block's structure is detailed below.

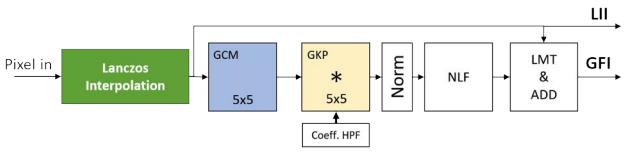

Fig. 17 Gohshi Filter implementation

Interpolation: We implemented the Lanczos's bi-linear technique as the interpolation filter as a trade-off between complexity and quality. This interpolation core[26] allows up and down scaling.

High Pass Filter: The 5×5 Laplacian edge detector is performed using configured GCM and GKP. The 5×5 GKP performs convolution function using 25 signed coefficients which have been fixed for optimizing the hardware implementation cost. The values are defined in Eq. 7.

$$\mathbf{X} = \frac{1}{256} \times \begin{bmatrix} 0 & -3 & -5 & -3 & 0 \\ -3 & -15 & -25 & -15 & -3 \\ -5 & -25 & +214 & -25 & -5 \\ -3 & -15 & -25 & -15 & -3 \\ 0 & -3 & -5 & -3 & 0 \end{bmatrix} \quad (7)$$

Non Linear Function (NLF) The non linear function applied is a simple cubic function, implemented by cascading two multipliers, the output $Pixel_{\mathrm{NLF}}$ is signed.

Limiter & Adder (LMT & ADD): The limiter bounds $Pixel_{\mathrm{NLF}}$ to the output $Pixel_{\mathrm{LMT}}$ range from [-255,255]. The role of the *Add* block is to merge $Pixel_{LMT}$ (value in $[-255, 255]$) with the corresponding interpolated pixel $Pixel_{Int}$ (value in $[0, 255]$). The cubic function is monotonic for positive values and increases fast. The resulting pixel value is often saturated.

4.3 Texture Classification

As mentioned in Section 3.1.4, the Texture Classification block aims to classify the local texture. The process is pipelined in several steps: a Sobel gradient is performed, and then the features $F(i, j)$ (variance in N_1 neighborhood) are extracted from the Gradient. The final step corresponds to the decision phase (single boosting iteration).

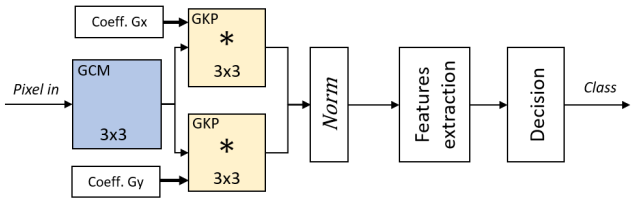

Fig. 18 TC architecture block

The 3×3 Sobel gradient is performed using configured GCM and two GKPs. The variance is performed on the gradient resulting image relying on Eq. 8.

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i^2) - (\sum_{i=1}^{N} x_i)^2/N)}{N}, N = n \times n. \quad (8)$$

Based on an architecture that has been originally proposed in previous studies[34,11], an implementation is proposed using our generic filter model. As discussed in 3.1.4, the size n of the kernel used to compute variance does impact the global classification error. Therefore, the genericity of the proposed architecture provides flexibility and enables the classification performance to be adjusted according to the hardware resources available onto the FPGA target. Therefore, a $n \times n$ GCM bounded to two GKPs is generated. One GKP is configured to process the sum of GCM pixels, and the other one is configured to process the sum of the squared pixel (Fig. 19).

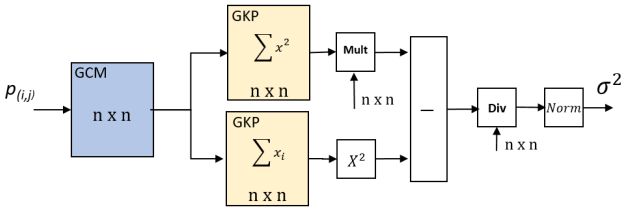

Fig. 19 Computing the variance using GKP and GCM.

This filter is not only easy to use and to configure but also very fast due to the high level of parallelism. However, it may be hardware resource-consuming when n increases. So, the impact of n on classification performance and hardware resources is presented in Fig. 20 using a Xilinx Virtex-7 as the target. The classification error is acceptable (lower than 10%) from $n = 8$ and decreases when n increases. However, the proportion of used hardware resources is acceptable (according to our application constraints) for $n = 8$ and $n = 16$ which require respectively 2.8% and 10.5% of the selected FPGA resources. We choose $n = 16$ as the best trade-off between global classification error and hardware resources.

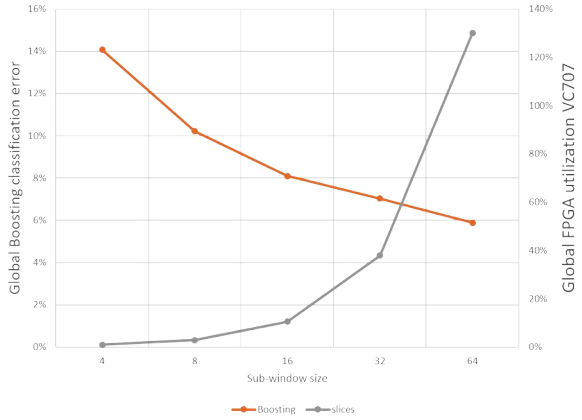

Fig. 20 Global Boosting classification error and FPGA resources evolution depending on sub-window size.

4.4 Local Merging

This final step of the LASSR method merges the GFI and the LII according to their local texture as depicted in Fig. 21. Since the Gohshi's method increases the luminance, a 5×5 local luminance equalization is performed (following Eq. 3 previously described in Section 3.2). The luminance equalization requires to compute two local means by connecting a 5×5 GCM to a GKP configured to sum pixels of the neighborhood. The final merging is performed using a simple 2-input multiplexer (with class y as selection input).

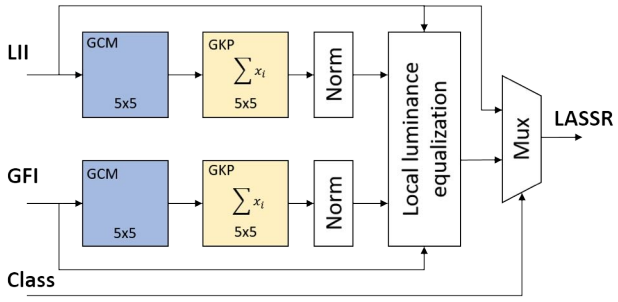

Fig. 21 Local Merging principle block

option is to increase the output framerate by duplicating the LASSR core on the current FPGA device, and executing each instance in parallel on independent blocks of the input source. Since the current HW prototype consumes few hardware resources, we decide to implement four parallel instances of LASSR (named LASSR4), each of them processing a quarter of the input image. As expected, the resulting LASSR4 implementation is 4 times faster and reaches 4K SR at 68 fps. It consumes 60285 Slices LUT, 88 BRAMs, and 1260 DSP blocks, representing 19.86% of total Slices LUT, 8.54% of total BRAMs, and 45% of total DSP in the targeted Virtex-7 device.

Table 4 FPGA utilization summary of Super Resolution, Texture Classification & Local Merging

Hardware Resources	Slices LUT	BRAMs	DSP blocks
Super Resolution	1546	6	38
Texture Classification	11450	10.5	275
Local Merging	2249	11.5	2
Global Utilization	15245 (5.02%)	28 (2.72%)	315 (11.25%)

4.5 Implementation results

The proposed LASSR method has been implemented through the use of Vivado 2019.2. It is mapped to a Xilinx Virtex-7 xc7vx485tffg1761-2. As indicated in Table 4, LASSR HW uses 15245 Slices LUT, 28 BRAMs, and 315 DSP blocks, representing only 5.02% of total Slices LUT, 2.72% of total BRAMs, and 11.25% of total DSP of the targeted Virtex-7 device.

Among the 3 modules, the Texture Classification requires the largest number of Slices LUT (11450/15245, i.e 75%) that corresponds to about 4% of the FPGA LUT resources and the largest number of DSP blocks (275/315, i.e 87%) that corresponds to about 10% of the FPGA DSP resources. An upcoming study plans to limit the complexity of the Texture Classification by reducing the size of the sliding sub-window required for the estimation of the SVM parameters while preserving a low global error.

With an operating frequency of 142.85MHz, the resulting implementation supports high quality 4K SR videos from 2K at 16 fps. These results lead to two potential scenarios. First, a lower budget FPGA could efficiently replace the Xilinx Virtex-7 xc7vx485tffg1761-2 and still reach the same performance. The second

Table 5 compares our two LASSR implementations to state-of-the-art HW real-time implementations of Super-Resolution on FGPA. Two of them [30, 25] are CNN-based methods while the third one implements an iterative algorithm [47]. Seyid et al. [47] presented a real-time HW implementation of SR using an iterative back-projection method, that can reconstruct 512x512 images from a set of lower resolutions (up to 20 images) at 25 fps. The FPGA utilization scales with the number of input images and with the number of iterations of the algorithm. Manabe et al. [30] proposed a real-time super-resolution system for moving images using a convolutional neural network. They applied horizontal and vertical flips to network input images instead of commonly used pre-enlargement techniques. This method performed super-resolution from 960x540 pixels to 1920x1080 pixels at 48fps. Yongwoo et al. [25] proposed an FPGA implementation able to generate 4K images at 60fps, with the better visual quality compared to other CNN-based SR methods. To cope with the FPGA constraints, their method used a limited number of convolutional filter parameters by incorporating depth-wise separable convolutions with a residual connection.

Compared to the Iterative Back Projection, the LASSR method can process larger input data (1920x1080 vs 256x256) for a comparable framerate (16 fps vs 25 fps)

with a straightforward implementation of LASSR. With a more sophisticated and more efficient implementation based on four parallel instances, LASSR4 reaches the same level of performance (4K at 67 fps) of the best state-of-the-art CNN-based SR techniques. Moreover, LASSR4 consumes fewer logic resources because CNNs are known to require many arithmetic operations. For example, about 50% of LUT and 100% DSP are used in [30, 25] while LASSR4 requires less than half the resources of the CNN methods (i.e. 20% of LUT and 45% of DSP).

5 Conclusion and Future work

In this paper, we have proposed a new method to perform real-time image super-resolution even for high-resolution input images. Moreover, as an improvement of real-time state-of-art methods, the proposed solution includes a classification-based texture analysis. This step enables to process differently the low-frequency texture regions, limiting the noise increasing. According to the texture analysis, an edge enhancement algorithm is applied to reveal finer details and better contrast. The process has been validated using a PVA metric and a panel of forty-five participants. We have designed a real-time implementation of our SR method that demonstrates the generation of 4K high-quality content from 2K sources on an embedded FPGA board while limiting run-time and hardware resources. This system allows producing high-quality 4k SR videos from 2k videos at 16 fps. Since less than 20% of the FPGA hardware resources were used for a single-core implementation, we have also proposed and implemented an extended and optimized version reaching 67 fps. This high-performance version, named LASSR4, is based on the parallel execution of four independent LASSR cores on the FPGA.

All these results open interesting avenues for future exploration both on software and hardware issues. First, we plan to reduce the complexity of the Texture Classification block that consumes more than 75% of the slices LUT and 87% of the DSP blocks used for the LASSR core. The TC complexity is directly linked to the classification error and the size n of the sliding window used to estimate the SVM parameters. For the TC implementation, we have pragmatically chosen $n = 16$ with a classification error rate lower than 10%. A deeper analysis must be conducted to explore the challenging trade-off between image quality, classification accuracy, and resource utilization. The main idea is to maintain a high quality of SR data while releasing constraints on the classification error and then reducing the hardware complexity.

Secondly, this paper has validated the hardware design of the LASSR core using a rapid prototyping strategy based on Matlab, Simulink, and ModelSim. In the FPGA-in-the-loop approach, the design of LASSR has been deployed to the Xilinx VC707 board and runs in real-time on the Xilinx Virtex 7. This approach helped us to validate the different blocks of the LASSR core. However, all the other surrounding components are simulated in the software environment. Typically, the image sensor is simulated and input images are passed from Matlab/Simulink to the HDL design on the FPGA device. After processing, the SR output images are then passed back from the FPGA to Matlab for further analysis. So, the next step will be the full implementation on an FPGA-based platform including an image sensor to experimentally validate an SR smart camera able to produce a 4K SR live video from a captured 2K flow.

Acknowledgements This work was funded by EU Horizon 2020 ECSEL JU research and innovation programme under grant agreement 662222 (EXIST).

References

1. S Arivazhagan and L Ganesan. Texture classification using wavelet transform. *Pattern recognition letters*, 24(9-10):1513–1521, 2003.
2. S. Baker and T. Kanade. Hallucinating faces. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 83–88, 2000.
3. S. Borman and R.L. Stevenson. Super-resolution from image sequences-a review. In *1998 Midwest Symposium on Circuits and Systems (Cat. No. 98CB36268)*, pages 374–378, 1998.
4. Oliver Bowen and Christos-Savvas Bouganis. Real-time image super resolution using an FPGA. In *2008 International Conference on Field Programmable Logic and Applications*, pages 89–94, 2008.
5. Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.
6. Raymond H Chan, Tony F Chan, Lixin Shen, and Zuowei Shen. Wavelet Algorithms for High-Resolution Image Reconstruction. *SIAM Journal on Scientific Computing*, 24:1408–1432, 2003.
7. Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.
8. Jae-Seok Choi and Munchurl Kim. Super-Interpolation With Edge-Orientation-Based Mapping Kernels for Low Complex 2 x Upscaling. *IEEE Transactions on Image Processing*, 25:469–483, 2016.
9. Danny Coomans and Désiré Luc Massart. Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-nearest neighbour classification by using alternative voting rules. *Analytica Chimica Acta*, 136:15–27, 1982.
10. Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *Computer Vision – ECCV 2014,*

Table 5 Characteristics of State-of-the-art real-time FPGA hardware implementations

Publication	[47]	[30]	[25]	Our work	
Method	IBP	CNN	CNN	LASSR *(1 core)*	LASSR *(4 cores)*
FPGA Device	Xilinx Virtex-7 XC7VX485T	Virtex UltraScale XCVU095	Xilinx Kintex UltraScale XCKU040	Xilinx Virtex-7 XC7VX485T	
FPGA Resources	N/A	266k LUT (49%) 131 BRAM (7%) 768 DSP (100%)	110k LUT (45%) N/A 1920 DSP (100%)	31k LUT (10%) 28 BRAM (3%) 315 DSP (11.25%)	60k LUT (20%) 88 BRAM (9%) 1260 DSP (45%)
Max. Frequency	265 MHz	133 MHz	150 MHz	142 MHz	
Input Resolution	256x256	960x540	1920x1080	1920x1080	
Output Resolution	512x512	1920x1080	3940x2160	3940x2160	
Framerate	25 fps	48 fps	60 fps	16 fps	67 fps

13th European Conference, Zurich, Switzerland, pages 184–199, 2014.

11. Julien Dubois and Marco Mattavelli. Embedded coprocessor architecture for cmos based image acquisition. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 2, pages II–591. IEEE, 2003.
12. Sina Farsiu, Dirk Robinson, Michael Elad, and Peyman Milanfar. Advances and challenges in super-resolution. *International Journal of Imaging Systems and Technology*, 14:47–57, 2004.
13. C. Feichtenhofer, H. Fassold, and P. Schallauer. A perceptual image sharpness metric based on local edge gradient analysis. *IEEE Signal Processing Letters*, 20(4):379–382, 2013.
14. W.T. Freeman, T.R. Jones, and E.C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22:56–65, 2002.
15. Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.
16. Georgios Georgis, George Lentaris, and Dionysios Reisis. Acceleration techniques and evaluation on multi-core CPU, GPU and FPGA for image processing and super-resolution. *Journal of Real-Time Image Processing*, pages 1–28, 2016.
17. Seiichi Gohshi. *A New Signal Processing Method for Video Image-Reproduce the Frequency Spectrum Exceeding the Nyquist Frequency Using a Single Frame of the Video Image*, pages 593–604. Springer New York, New York, NY, 2013.
18. Jinchen Guan, Jianyu Yang, Yulin Huang, and Wenchao Li. Maximum a posteriori–based angular superresolution for scanning radar imaging. *IEEE Transactions on Aerospace and Electronic Systems*, 50:2389–2398, 2014.
19. R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, May 1979.
20. Zhuolun He, Hanxian Huang, Ming Jiang, Yuanchao Bai, and Guojie Luo. FPGA-Based Real-Time Super-Resolution System for Ultra High Definition Videos. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 181–188, Boulder, CO, 2018.
21. Cheolkon Jung, Peng Ke, Zengzeng Sun, and Aiguo Gu. A fast deconvolution-based approach for single-image super-resolution with GPU acceleration. *Journal of Real-Time Image Processing*, 14:501–512, 2018.
22. Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016.
23. Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-Recursive Convolutional Network for Image Super-Resolution. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1645, 2016.
24. Yongwoo Kim, Jae-Seok Choi, and Munchurl Kim. 2X Super-Resolution Hardware Using Edge-Orientation-Based Linear Mapping for Real-Time 4K UHD 60 fps Video Applications. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65:1274–1278, 2018.
25. Yongwoo Kim, Jae-Seok Choi, and Munchurl Kim. A Real-Time Convolutional Neural Network for Super-Resolution on FPGA with Applications to 4K UHD 60 fps Video Services. *IEEE Transactions on Circuits and Systems for Video Technology*, PP:1–1, 2018.
26. D Kronsteiin. Video stream scaler : www.opencores.org, 2011.
27. Haoyang Li, Yujia Huang, Cuifang Kuang, and Xu Liu. Method of super-resolution based on array detection and maximum-likelihood estimation. *Applied Optics*, 55:9925, 2016.
28. Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1132–1140, 2017.
29. B. N. Madhukar and R. Narendra. Lanczos resampling for the digital processing of remotely sensed images. In Veena S. Chakravarthi, Yasha Jyothi M. Shirur, and Rekha Prasad, editors, *Proceedings of International Conference on VLSI, Communication, Advanced Devices, Signals & Systems and Networking (VCASAN-2013)*, pages 403–411, India, 2013. Springer India.
30. Taito Manabe, Yuichiro Shibata, and Kiyoshi Oguri. FPGA implementation of a real-time super-resolution system using a convolutional neural network. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 249–252, Xi'an, 2016.
31. Čadík Martin. *Perceptually Based Image Quality Assessment and Image Transformations*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, January 2008.
32. Takashi Matsuyama, Shu-Ichi Miura, and Makoto Nagao. Structural analysis of natural textures by fourier transformation. *Computer vision, graphics, and image processing*, 24(3):347–362, 1983.

33. Johel Mitéran, Jiri Matas, E Bourennane, Michel Paindavoine, and Julien Dubois. Automatic hardware implementation tool for a discrete adaboost-based decision algorithm. *EURASIP Journal on Applied Signal Processing*, 2005:1035–1046, 2005.
34. Romuald Mosqueron, Julien Dubois, Marco Mattavelli, and David Mauvilet. Smart camera based on embedded hw/sw coprocessor. *EURASIP Journal on Embedded Systems*, 2008(1):597872, Oct 2008.
35. Vinay P Namboodiri, Vincent De Smet, and Luc Van Gool. Systematic evaluation of super-resolution using classification. In *2011 Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2011.
36. Kamal Nasrollahi and Thomas B. Moeslund. Super-resolution: a comprehensive survey. *Machine Vision and Applications*, 25:1423–1468, 2014.
37. Kien Nguyen, Clinton Fookes, Sridha Sridharan, Massimo Tistarelli, and Mark Nixon. Super-resolution for biometrics: A comprehensive survey. *Pattern Recognition*, 78, 2018.
38. J.D. van Ouwerkerk. Image super-resolution survey. *Image and Vision Computing*, 24:1039–1052, 2006.
39. Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *IEEE Signal Processing Magazine*, 20:21–36, 2003.
40. Joel Pérez, Eduardo Magdaleno, Fernando Pérez, Manuel Rodríguez, David Hernández, and Jaime Corrales. Super-Resolution in Plenoptic Cameras Using FPGAs. *Sensors*, 14:8669–8685, 2014.
41. Feng-qing Qin, Xiao-hai He, Wei-long Chen, Xiao-min Yang, and Wei Wu. Video superresolution reconstruction based on subpixel registration and iterative back projection. *Journal of Electronic Imaging*, 18:013007–013007–11, 2009.
42. E. Quevedo, L. Sánchez, G. M. Callicó, F. Tobajas, J. de la Cruz, V. de Armas, and R. Sarmiento. Super-resolution with selective filter based on adaptive window and variable macro-block size. *Journal of Real-Time Image Processing*, 15:389–406, 2018.
43. Rodolfo Redlich, Luis Araneda, Antonio Saavedra, and Miguel Figueroa. An Embedded Hardware Architecture for Real-Time Super-Resolution in Infrared Cameras. In *2016 Euromicro Conference on Digital System Design (DSD)*, pages 184–191, 2016.
44. Seunghyeon Rhee and Moon Gi Kang. Discrete cosine transform based regularized high-resolution image reconstruction algorithm. *Optical Engineering*, 38:1348–1356, 1999.
45. M. Dirk Robinson, Stephanie J. Chiu, Cynthia A. Toth, Joseph A. Izatt, Joseph Y. Lo, and Sina Farsiu. New Applications of Super-resolution in Medical Imaging. In Peyman Milanfar, editor, *Super-resolution imaging*, pages 383–412. CRC Press, 2010.
46. Mehdi SM Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4491–4500, 2017.
47. Kerem Seyid, Sebastien Blanc, and Yusuf Leblebici. Hardware implementation of real-time multiple frame super-resolution. In *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 219–224, Daejeon, South Korea, 2015.
48. Wanqiang Shen, Lincong Fang, Xiang Chen, and Honglin Xu. Projection onto Convex Sets Method in Space-frequency Domain for Super Resolution. *Journal of Computers*, 9, 2014.
49. Assaf Shocher, Nadav Cohen, and Michal Irani. Zero-Shot Super-Resolution Using Deep Internal Learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3118–3126, 2018.
50. R. Y. Tsai and T. S. Huang. Multiframe image restoration and registration. In R. Y. Tsai and T. S. Huang, editors, *Advances in Computer Vision and Image Processing*, pages 317–339. JAI Press Inc., 1984.
51. Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.
52. Z. Wang, H. R. Bovik, A. C.and Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
53. Zhihao Wang, Jian Chen, and Steven C H Hoi. Deep Learning for Image Super-resolution: A Survey. *arXiv*, 2019.
54. Chih-Yuan Yang, Chao Ma, and Ming-Hsuan Yang. Single-image super-resolution: A benchmark. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 372–386, Cham, 2014. Springer International Publishing.
55. Daiqin Yang, Zimeng Li, Yatong Xia, and Zhenzhong Chen. Remote Sensing Image Super-Resolution: Challenges and Approaches. In *2015 IEEE International Conference on Digital Signal Processing (DSP)*, pages 196–200, 2015.
56. Jianchao Yang and Thomas Huang. Image Super-Resolution: Historical Overview and Future Challenges. In Peyman Milanfar, editor, *Super-resolution imaging*, pages 1–33. CRC Press, 2010.
57. Yuan Yuan, Xiaomin Yang, Wei Wu, Hu Li, Yiguang Liu, and Kai Liu. A fast single-image super-resolution method implemented with CUDA. *Journal of Real-Time Image Processing*, 16:81–97, 2019.
58. Linwei Yue, Huanfeng Shen, Jie Li, Qiangqiang Yuan, Hongyan Zhang, and Liangpei Zhang. Image super-resolution: The techniques, applications, and future. *Signal Processing*, 128:389–408, 2016.
59. Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.
60. Liangpei Zhang, Hongyan Zhang, Huanfeng Shen, and Pingxiang Li. A super-resolution reconstruction algorithm for surveillance images. *Signal Processing*, 90:848–859, 2010.
61. F. Zhou, R. Yao, B. Liu, and G. Qiu. Visual quality assessment for super-resolved images: Database and method. *IEEE Transactions on Image Processing*, pages 1–1, 2019.