INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2026, VOL. 72, NO. 3, PP. 1-10
Manuscript received May 04, 2026; revised June, 2026. doi: 10.24425/ijet.2026.1723

Meta-filtration: Adaptive selection of multiple filter cascades for images with quality analysis

Dominika Kanty, and Jędrzej Sikora

***Abstract*—This paper proposes a meta-filtration framework for denoising images corrupted by mixed noise (Gaussian, salt&pepper, speckle). Instead of fixed pipelines or AI-based methods, it uses a manually defined filter set and automatically selects combinations that improve image quality metrics (PSNR or MS-SSIM). The approach was tested on images of different sizes and mixed noise levels. Results show PSNR improvements of 7-11dB, with MS-SSIM confirming preservation of fine details. An embedded implementation on a Zynq-7000 SoC achieved similar quality to MATLAB (within 0.1-0.8dB) and significantly reduced runtime, demonstrating practical efficiency on resource-constrained hardware.**

***Keywords*—computer vision; automatic optimalization; adaptive filter cascades; mixed-noise image denoising; embedded vision systems**

I. Introduction

ONE of the human abilities is the capacity to make decisions and draw conclusions from experience, as well as to receive and analyze stimuli through sensory organs. By using appropriate tools and optimizations, such as vision systems, these activities contribute to the development of further fields, including artificial intelligence. [1] Modern digital image processing systems are increasingly taking over the functions of human vision, offering much higher speed, repeatability, and efficiency wherever precise image analysis and recognition are crucial.

Vision systems are now used in an ever-growing number of domains; with suitable image processing operations, it is possible to extract more information than the human eye is able to perceive. [2] To use an image as a source of information, it must first be converted into a digital form in order to carry out an analysis process that includes segmentation and object localization. The purpose of operations on pixel values and their modification is to remove undesired components, enhance key features such as edges, textures, and details, and selectively amplify or suppress specific bands. These actions improve the readability and simplicity of image analysis, i.e., image processing, which in turn provides an excellent foundation for automatic object recognition and machine learning. [3]

In contemporary image processing systems, there is an increasing emphasis on full automation: the algorithm combines multiple features and selects the configuration that yields the highest quality, while simultaneously reducing the operator's time involvement. An example of this is the work of researchers from Shanghai, in which the authors proposed the use of median and averaging filters in different orders for denoising CCD camera images during automatic laser beam alignment. [4] Another study applied filtering to improve the quality of static thermal images from a mine refuge chamber; the results showed that an appropriate combination of filters makes it possible to better preserve image details. [5] The described approaches are based on combining different filtering methods in an appropriate sequence and with properly selected parameters, in such a way that their mutual interactions do not lead to degradation of image quality. Selection of filter order and sensitivity is a key element of the denoising process, as individual operations can be significantly influence subsequent stages of processing.

Despite the extensive body of knowledge on the behavior of individual filtering methods, relatively few publications discuss and analyze the mutual interactions between different filters. The first mentions date back to 2014, when it was shown that the order of applying smoothing (Gaussian) and median filters has a significant impact on the final result. The choice of filter order followed from the fact that the images were dominated by a specific type of noise, and therefore a preliminary image analysis was required before further processing. [4] A year later, a paper was published that focused on the removal of mixed noise through the use of a classical (median) filter and a transform-based approach (wavelet-domain denoising, DWT). [6] However, the studies discussed did not address images degraded by unknown mixed noise, in which individual noise components occur with varying proportions and intensities.

The aim of this study was to analyze the effectiveness of automatically selected filters for images corrupted by unknown mixed noise of varying intensity. In particular, attention was focused on assessing the relevance of applying individual filters and their mutual interactions within a cascade configuration, with the goal of making the processed image resemble the reference image as closely as possible.

The topic addressed in this paper is also related to the improvement of automatic vision systems, with particular emphasis on one of the key aspects, namely minimizing the load on the GPU computing unit.

II. Image Denoising Method

In this study, a set of selected contextual filters and geometric operations was used, corresponding to different types of possible image degradations:

Authors are with Wrocław University of Science and Technology, Poland (e-mail: dominika.kanty@pwr.edu.pl).

- **Gaussian filtering** – the classical 2-D Gaussian function acts as a low-pass filter based on the normal distribution. It enables image smoothing with minimal distortion of geometric structures. [7] The value of each pixel is computed as weighted average of the neighbouring pixel values, where the weights follow a Gaussian distribution that decreases with the distance from the centre of the kernel. This filter enables noise reduction while preserving image edges. The two-dimensional Gaussian kernel with standard deviation σ is definied as follows:

$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \quad (1)$$

x, y- denote spatial coordinates relative to the cemtre of the filter,
σ- controls the amount of smoothing.

- **Median filtering** – an optimal filter for suppressing impulse noise (so-called salt-and-pepper noise) while preserving edges and fine details that are important for image objects. For each pixel, it replaces its value with the median of the intensity levels within a predefined neighborhood around that pixel. [8] Formally, it can be expressed as follows:

$$\hat{f}(x,y) = median\{g(r,c)\} \quad (2)$$

$$(r,c) \in S_{xy} \quad (3)$$

$\hat{f}(x,y)$- is the filtered image,
$g(r,c)$- is the input image,
S_{xy}- denotes a neighbourhood (window) centred at (x,y).

- **Standard deviation filter** – emphasizes local intensity variations and detects regions with high variability, which is essential for texture analysis. It can be definied as

$$\sigma(x,y) = \sqrt{\frac{1}{|S_{x,y}|}\sum_{(i,j)\in S_{x,y}}\left(f(i,j)-\mu(x,y)\right)^2} \quad (4)$$

Where $|S_{x,y}|$ denotes the number of pixels in the neighbourhood and

$$\mu(x,y) = \frac{1}{|S_{x,y}|}\sum_{(i,j)\in S_{x,y}} f(i,j) \quad (5)$$

Is the local mean intensity in $S_{x,y}$. Finally, the resulting image serves as a measure of local contrast or texture.

- **Wiener filter** – enables estimation of unknown random noise by filtering the corrupted signal. Wiener filtering allows local estimation of the noise level based on the statistical properties of the neighborhood of the analyzed pixel. [9] For a discrete grayscale image f(x,y), the local Wiener filter estimates the noise-free signal by adapting the amount of smoothing to the local statistics of the image.

$$g(x,y) = \mu(x,y) + \frac{\sigma^2(x,y)-\varepsilon_n^2}{\sigma^2(x,y)}\left(f(x,y)-\mu(x,y)\right) \quad (6)$$

Where:
$g(x,y)$- the value of the pixel in the image after Wiener filtering,
$\mu(x,y)$- the local mean in the neighbourhood,
$\sigma^2(x,y)$- the local variance in the same neighbourhood,
ε_n^2- the noise variance (typically estimated globally)

- **Sobel filter** – one of the most common filters for edge detection in digital images. The behavior with respect to noise depends on the kernel size. Small kernels (3×3) are more sensitive to noise but produce sharper edges, whereas larger kernels (9×9) reduce false detections and smooth the image at the cost of increased blurring. [10] In „Digital Image Processing", an example is presented in which the Sobel operator is applied to a simple test image with sharp intensity transitions. This results in strong responses along the edges while maintaining values close to zero in homogeneous regions. [8] This confirms that the operator approximates the local intesity gradient and is therefore a suitable tool for edge detection.
- **Laplacian filter** – detects intensity changes in the image based on the second derivative, i.e., object edges, which allows objects to be localized, shapes to be recognized, and characteristic features to be detected. It effectively detects edges in different directions and enhances image details. [11] The Laplacian operator is a second-order derivative filter used to highlight regions of rapid intensity change in an image. For a continuous image f(x,y), the 2D Laplacian is definied as

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \quad (7)$$

For a discrete grayscale image f(x,y), the Laplacian-filtered image g(x,y) is obtained as

$$g(x,y) = (L * f)(x,y) = \sum_{i=-1}^{1}\sum_{j=-1}^{1} L(i,j) f(x-i, y-j) \quad (8)$$

Where L denotes kernels (masks).

CLAHE (Contrast Limited Adaptive Histogram Equalization) – improves local contrast while limiting noise amplification. [12] This filter is an extension of adaptive histogram equalization designed to enhance local cotrast. The image is devided into small blocks, and for each block local brightness (intensity) histogram is computed, which is then „clipped" to a specified contrast limit (clip limit). It is widely used in medical, satellite, and machine-vision imaging. [13] [14].

The order of the applied filters, illustrated in Fig. 1, has a significant impact on the result. In the proposed model, all possible combinations of six filters are tested, with the option of repeating individual filters or skipping a given step (no filter). The final filtering outcome depends not only on the choice of specific filters, but also on the sequence in which they are applied.

A key element of the proposed approach is the filter selection rule: a given filter is retained only if it improves the image quality relative to its previous version; otherwise, it is rejected and replaced with another filter or omitted altogether.

To determine whether the filter results improve the quality of the image reconstruction, we focus on PSNR, the peak signal-to-noise ratio. It makes it possible to quantify the degree of distortion of the original image with respect to the noisy/processed one and is expressed in dB. In the context of the experiment, it can be observed that a higher PSNR value indicates better image quality.

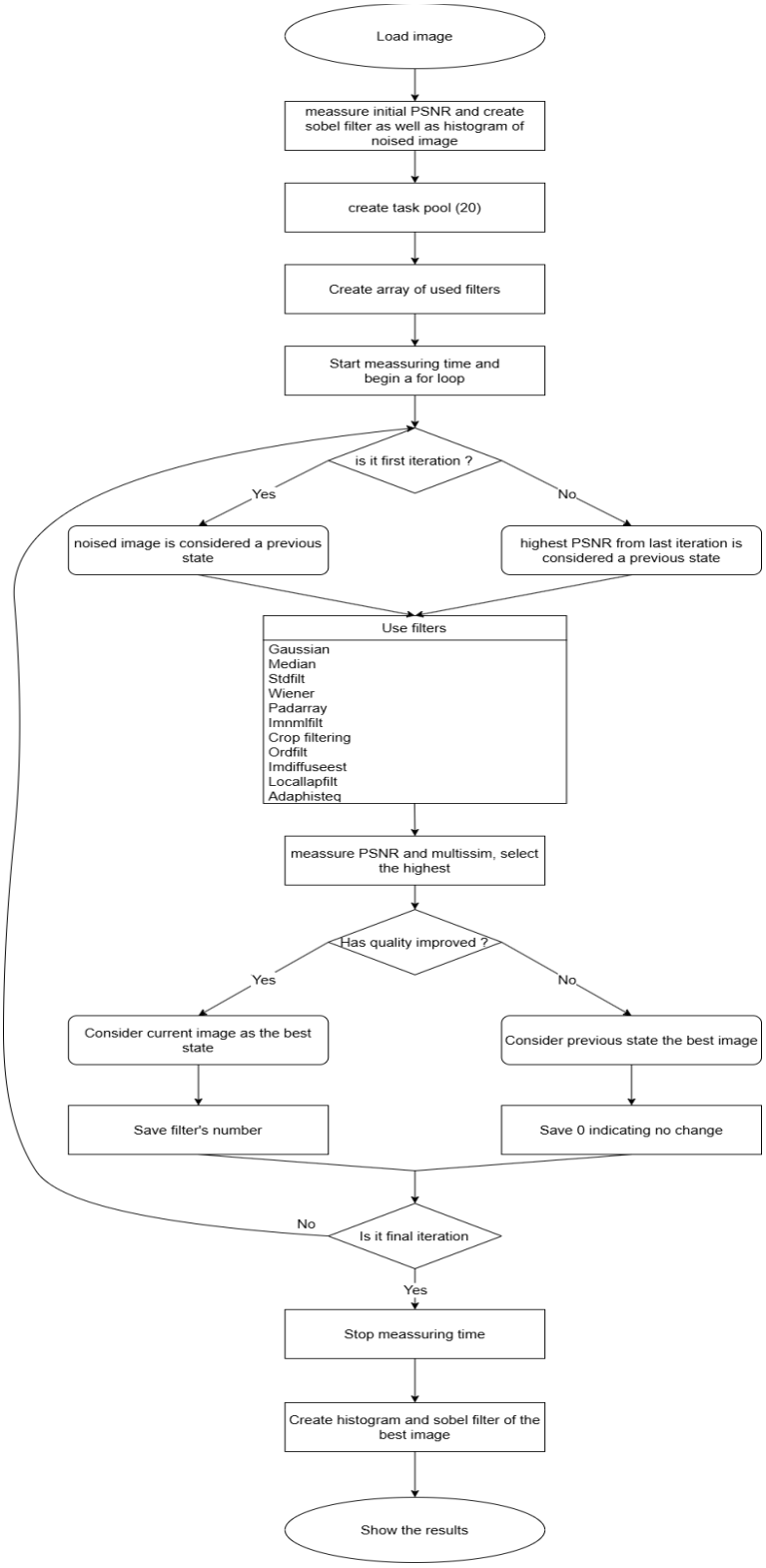

Fig. 1. Block diagram of the automatic image filtering algorithm incorporating a quality-driven decision-making mechanism.

The PSNR can be calculated using the following formula:

$$PSNR = 10 * \log_{10}\left(\frac{N*max_I^2}{\sum_{i=1}^{N}\left(I_{org}(i)-I_{comp}(i)\right)^2}\right) \ [dB] \qquad (9)$$

Where:

max_I^2 – the maximum possible pixel value,
N – the numer of pixels,
$I_{org}(i)$ – the value of th i-th pixel in the original image,
$I_{comp}(i)$ – the value of th i-th pixel in the processed image.

In this study, the structural similarity index SSIM (Structural Similarity Index Measure) was additionally used as a metric for comparing the structural similarity between two images. In contrast to the previous method, the one described here consists in computing the SSIM index and then quantifying the

differences between the original and the distorted image. By using attributes such as brightness, contrast, and structure of the two images the level of similarity between them can be determined.

$$SSIM(x,y) = a(x,y)^{\alpha} b(x,y)^{\beta} c(x,y)^{\gamma} \quad [\%] \qquad (10)$$

Where:

$a(x,y)$- represents brightness,
$b(x,y)$- represents the contrast ratio,
$c(x,y)$- represents the structure,
$x-$ the original picture,
$y-$ denoised picture

It was necessary to approach this problem in such a way as a way as exclude operations that don't improve the quality of the filters. It should be remembered that in the case of an inappropriate choice of filter, the opposite of the intended effect occurs, i.e. the filtering results become worse. An example of this can be found in the article "Image Denoising Using Hybrid Filtering Techniques" [15] in which certain filters were selected- Gaussian, Wiener and Median and also a mixed filter- and the order of these filters is of crucial importance. In the photograph below, a table from the article is shown, where it can be observed that the optimization and selection of filters in an appropriate order result either in an improvement or a deterioration of the situation.

In the literature, there are numerous examples of combining filters in fixed sequences to reduce different types of noise. [16] In the article "A Review of Denoising Filters in Image Restoration", the filters are discussed in the following order: averaging (linear) filter, median (nonlinear) filter, and adaptive filter (which adjusts its parameters to the image statistics). The authors of the article indicate that the sequence they propose is the best approach to the problem under discussion; however, no comparative experiments were presented. [17]

In 2021, researchers conducted a study on noise reduction in images using a hybrid approach that combines several filters, including Gaussian, Wiener, and median filters. The results of the analysis were presented, among others, in terms of the PSNR index: the best obtained values are around 35.84 dB, which indicates a noticeable improvement in image quality after filtering. It should be noted, however, that in the vast majority of cases the PSNR values remain below 30 dB, which points to the limited effectiveness of the individual filtering methods across the full range of analyzed distortions. A major limitation of the discussed work is the lack of a precise description of the parameters of the applied noise. A detailed characterization of the type and intensity of noise, as well as its nature, is a key element in assessing the effectiveness of filtering, and its omission significantly hinders both the comparability and interpretation of the results. [18]

After analysing the available publications in this area, it becomes apparent that there is no information on how noise affects images of different sizes. Many studies that attempted to address this issue were based mainly on theoretical considerations. [19] The available literature also does not indicate which parameters for image comparison are the most appropriate. For this reason, the authors of this paper have attempted to find a golden mean, which were MS-SSIM (for matlab) and PSNR.

In contrast to these approaches, the solution proposed in this work automatically tests all possible combinations and orders of filters, using a quality metric as the selection criterion. Operations that have a beneficial impact on the quality of the processed image are retained in the final filter sequence, which results in an adaptive and optimization-oriented approach to the image filtering problem.

III. Image Resolution Selection and Quality Evaluation Measure

For images with high spatial resolution (dense sampling grid) and a wide range of gray levels (dense gray-level representation), the amount of fine detail can become problematic. For this reason, the experiments used images of size 256×256, 512×512, and 1024×1024 pixels. Most of these are relatively small by the standards of contemporary applications, yet sufficiently large to evaluate filter performance and to clearly observe their effects.

To compare the distorted images with the original ones, the Multi-Scale Structural Similarity (SSIM) index was used instead of the Peak Signal-to-Noise Ratio (PSNR). Although PSNR is commonly employed in the available literature, SSIM was chosen in order to assess image quality in a way that is more consistent with human visual perception, while maintaining sensitivity to object details. [20]

IV. Development environment

As part of the project, two different development environments were used to implement and verify the proposed image filtering algorithms. The first environment was MATLAB, language designed for matrix operations as well as focused on data processing and simulations. Basic environment was extended with packages such as:

- **Image Processing Toolbox** – Package that provides wide plethora of functions regarding overall image processing ranging from loading and showing images to processing functions such as filtering.
- **Parallel Computing Toolbox** – Toolbox giving its user to perform parallel operations on GPU rather than CPU. This package enabled creating images based on GPU with use of GPUarray() and split code's execution onto parallel jobs with parpool() . Implementation of this package lead to decrease of run time and made use of RTX 4060 laptop GPU.

The second approach involved implementing the filtering algorithms in the C programming language using the Vitis environment, which enabled execution of the code directly on the ZedBoard hardware platform equipped with an AMD Zynq-7000 SoC. For this purpose, it was necessary to prepare a complete hardware–software setup consisting of:

- **Vivado (VHDL)** – used to generate the hardware design of the system, including the configuration of the ARM Cortex-A9 processor, the AXI interconnect, and the required peripherals. The resulting project was exported as an *.xsa* file, which then served as the basis for creating the application in Vitis.
- **Vitis (C)** – the development environment used to implement and compile the C code, generate the executable *.elf* file, and run it on the Zynq processing system. Communication and monitoring of the results

were carried out via a UART interface, which enabled the display of logs and diagnostic information directly from the ZedBoard.

- **PetaLinux** – installed and configured to provide a file system and support more advanced image-processing operations. It enabled loading image files, performing the processing, and saving the results in the form of output images directly within the system running on the ZedBoard.

In this project, the image filtering algorithms were implemented manually in the C language and no pre-defined, hardware-accelerated functions from the AMD Vitis Vision library were used. The Vitis Vision library, which is part of the open-source Vitis Libraries (Apache 2.0 license), provides a set of optimized image processing functions and could be employed in future work as an alternative approach to further accelerate computations and offload selected operations to the programmable FPGA fabric.

Thanks to the prepared environment, it was possible not only to compare the efficiency of the algorithms at the software level, but also to practically verify their operation under hardware conditions, which constitutes an essential part of the overall project validation.

The presented set of filters was selected to enable effective analysis and denoising of images with unknown and potentially mixed types of degradation, while preserving the key structural features of the image.

To protect image borders from artifacts resulting from filtering in the vicinity of edges, the padarray operation was applied. In addition, subsequent cropping (crop) was used to restore the image to its original size and to remove artifacts introduced by the earlier padding.

A. GPU

The most time-consuming stage, and the one that requires the greatest computational resources, is the image filtering process; however, the most complex and difficult operation to implement is the image analysis itself.

MATLAB is a high-level programming environment that enables users to perform tasks requiring substantial computational power [19]. One of the key measures of code performance is its execution time; therefore, the execution time of the code on the GPU was measured using the gputimeit function.

V. RESULTS

Table I summarizes the detailed compositions of the noise mixtures (including their proportions) and the results obtained in MATLAB and in the C implementation. All experiments in this section were carried out on images of size 512×512 pixels. The considered noise configurations comprise different proportions of Gaussian, salt-and-pepper, and speckle noise; some cases contain only a single noise type, while most represent mixed-noise scenarios. In this study we focus on images affected by information loss during acquisition, transmission, or file storage, which in practice leads to the superposition of several degradation mechanisms (e.g. Gaussian, impulse salt-and-pepper, and speckle noise). In such conditions, identifying a single dominant noise component is often ambiguous, and the use of a single filter rarely yields application-level acceptable quality; moreover, an inadequate choice of method may further deteriorate the reconstruction. For this reason, the proper selection and sequencing of denoising techniques is regarded as a key issue. A review of the existing literature shows that the emphasis is typically placed on evaluating individual filters in isolation. In contrast, the present work proposes and investigates a solution based on a set of complementary filters applied in the presence of mixed noise, aimed at achieving the highest possible reconstruction quality while maintaining computational efficiency.

Within the examined dataset, both processing variants clearly improve the quality with respect to the noisy input images. At the same time, the implementation running on the Zynq platform (C/ARM) exhibits a noticeable advantage both in terms of reconstruction quality and computation time. Following an initial analysis of the results, the set of evaluation metrics for the MATLAB-based pipeline was extended to include MS-SSIM in order to capture perceptual aspects of image quality. In the interpretation of the reported metrics, PSNR (before) denotes the quality of the noisy image with respect to the original, whereas PSNR (after) refers to the quality of the denoised image relative to the original.

For each noise configuration, Table I also reports execution times for the MATLAB implementation. In the baseline variant, the code is executed on the host CPU using functions from the Image Processing Toolbox, without explicit parallelization. The corresponding runtimes range from approximately 56s to 388s. per 512x512 image, with an average of about 158s. which reflects both the complexity of the adaptive cascade search and the overhead of repeatedly applying multiple filters. To reduce this computational cost, the implementation was extended with Parallel Computing Toolbox. This GPU-accelerated variant reduces the runtime to several tens of seconds per image (on average around 30s), corresponding to a multi speed-up compared to the CPU-only Matlab version. These results confirm that the proposed algorithm exhibits a high degree of data parallelism that can be efficiently exploited by modern GPU hardware. Nevertheless, even the optimized Matlab/GPU implementation remains significantly slower than the embedded C implementation on the Zynq-7000 platform, which processes the same test images.

The execution time of the proposed algorithm on the Zynq-7000 platform (up to approximately 10 s for the considered test resolutions) is consistent with values reported in the literature for complex image filtering and denoising tasks. For comparison, Mándi reported processing times on the order of 40–50 ms per frame for a pipeline comprising thresholding, Gaussian filtering, and Sobel filtering on a PYNQ-Z2 board (Zynq-7000), which corresponds to about 20–24 fps for a single image. [21] This demonstrates that even relatively complex filter sequences can be executed on this class of platforms within a few tens of milliseconds per frame. Similarly, Tsiktsiris reports approximately 13 ms per frame for denoising large images with a resolution of 4000×3000 pixels using an image-stacking technique on a PYNQ-Z1 platform. [22] This is an example where, for significantly higher resolutions, computation times on the order of a few to several milliseconds can be achieved, provided that sufficient parallelism is exploited. [23]

Other studies employing the Vitis Vision library indicate speedups of about 10–30× over CPU implementations for filtering and edge detection on Zynq-7000 platforms, both for classical linear filters and more advanced operators. [24] Although many of these solutions offload computations directly to the programmable logic (PL), whereas in the present work the algorithm was implemented in software on the ARM Cortex-A9 processor, the observed speedup of approximately 15× relative to the reference MATLAB implementation falls within the typical range of performance gains obtained when moving from an interpreted environment to optimized C/C++ code or hardware-based implementations. This indicates that the proposed implementation is not only functionally correct, but also computationally efficient in the context of results reported in the literature, and that further work on migrating selected operations to the FPGA fabric (e.g., using Vitis Vision) may yield additional, significant reductions in processing time.

TABLE I
IMPACT OF NOISE CONFIGURATION ON QUALITY (PSNR, MS-SSIM) AND PROCESSING TIME- MATLAB VS VITIS/C

file	noise			Matlab						Vitis		
	Gaussian	salt&pepper	speckle	before	after	before	after	before	after	before	after	t
	(var)	(amount)	(var)	MS-SSIM	MS-SSIM	PSNR [dB]	PSNR [dB]	t [s]	t [s]	PSNR [dB]	PSNR [dB]	[s]
1	0,01	0,03	0,10	0,52	0,86	13,93	26,10	253	-	13,93	26,00	2,06
2	0,10	0,50	0,20	0,20	0,52	7,54	15,73	137	-	7,54	15,05	5,35
3	0,00	0,00	0,40	0,49	0,78	10,72	22,57	89	26,87	10,72	23,09	2,42
4	0,00	0,20	0,00	0,44	0,95	12,06	28,65	210	30,90	12,06	32,37	1,39
5	0,00	0,40	0,00	0,27	0,52	8,19	18,11	201	32,40	8,19	17,68	4,87
6	0,20	0,00	0,00	0,33	0,61	9,60	20,14	215	32,14	9,60	19,94	2,63
7	0,40	0,00	0,00	0,26	0,53	8,18	18,06	92	32,61	8,18	17,64	4,13
8	0,40	0,40	0,40	0,15	0,39	6,80	14,14	168	31,77	6,80	13,57	4,23
9	0,20	0,05	0,04	0,31	0,59	9,38	19,14	197	30,94	9,38	18,81	2,65
10	0,60	0,60	0,60	0,11	0,33	6,20	13,03	105	32,47	6,20	12,41	6,43
11	0,02	0,01	0,05	0,55	0,88	14,94	26,87	155	31,45	14,94	26,56	1,78
12	0,30	0,01	0,25	0,26	0,63	8,35	17,67	388	31,26	8,35	16,61	2,48
13	0,02	0,20	0,25	0,34	0,70	9,75	20,62	167	31,42	9,75	19,84	1,10
14	0,10	0,20	0,05	0,31	0,64	9,46	20,52	233	31,57	9,46	18,85	1,21
15	0,00	0,00	0,10	0,64	0,86	15,74	27,74	120	32,44	15,74	27,89	0,76
16	0,00	0,10	0,00	0,56	0,96	15,08	29,94	174	40,64	15,08	35,13	0,56
17	0,10	0,00	0,00	0,41	0,77	11,52	22,69	73	30,43	11,52	22,53	0,85
18	0,10	0,10	0,10	0,34	0,72	10,04	20,23	111	23,67	10,04	19,76	1,25
19	0,20	0,20	0,20	0,25	0,55	8,25	17,17	224	27,36	8,25	16,43	1,62
20	0,00	0,00	0,20	0,57	0,84	13,11	25,61	87	24,12	13,11	25,64	0,84
21	0,00	0,00	0,30	0,52	0,81	10,33	23,77	56	25,25	11,66	24,21	2,15
22	0,00	0,30	0,00	0,36	0,93	8,72	27,36	206	29,97	10,33	30,4	0,80
23	0,30	0,00	0,00	0,30	0,57	7,39	18,95	156	25,97	8,72	18,57	1,55
24	0,80	0,80	0,80	0,08	0,40	5,85	12,35	122	32,12	5,85	12,31	2,33
25	0,70	0,70	0,70	0,09	0,28	6,01	12,55	69	32,11	6,01	12,36	2,31
26	1,00	1,00	1,00	0,05	0,45	5,62	12,21	95	32,10	5,62	12,2	8,74

The Matlab results are shown below.

Fig. 2. Illustration of the proposed denoising method: original image, (before) mixed noise image and (after) reconstructed image after filtering in Matlab.

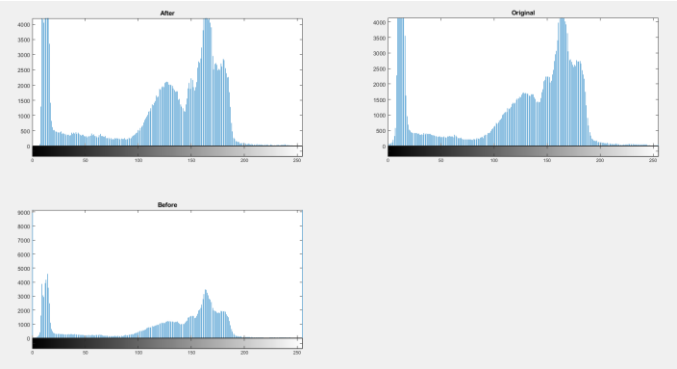

Fig. 3. Histograms were created based on the images presented above in Matlab.

The hardware design environment results are shown below.

Fig. 4. Input image mixed noise (a) and reconstructed image (b) after filtering in Vitis.

For better visualization of the quality differences, a plot of the PSNR gain (ΔPSNR) was generated, where PSNR (before) denotes the ratio of the noisy image to the original, and PSNR (after) represents the quality of the filtered image with respect to the same reference.

In most of the evaluated cases, the Zynq-based implementation yields higher PSNR values than the Matlab pipeline, as illustrated by both the numerical results and Fig 5. For both variants, the obtained gains are substantial, typically ranging from about 6 dB to over 20 dB, which confirms that the proposed filtering strategy consistently improves the quality of the noisy input images. The two curves follow a very similar trend across all noise configurations, indicating that the overall behavior of the algorithm is preserved between implementations.

At the same time, the Zynq variant tends to achieve slightly higher ΔPSNR values than Matlab for the majority of files, with the largest advantages observed for selected configuration-around cases 4, 19, and 25, where the PSNR gain approaches 20 dB. Only a few cases exhibit nearly identical performance of both implementations. In combination with the clearly shorter execution times on the Zynq-7000 platform, these results show that the embedded C implementation can deliver at least comparable, and often marginally better, reconstruction quality than Matlab baseline, while being significantly more efficient computationally.

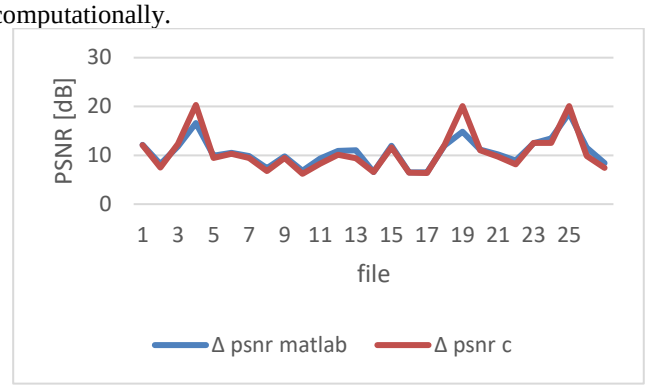

Fig. 5. Comparison of image quality improvement (ΔPSNR) for the given noise configuration- Matlab vs Vitis.

To assess the sensitivity of the algorithm to increasing levels of mixed noise, we considered cases in which all three components (Gausssian, salt & pepper and speckle) shared identical parameters, gradually increased in the range 0,1-1,0. Figure 6 shows the PSNR (after) values for both implementations for images of size 512x512 pixels. In both cases a monotonic decrease of PSNR can be observed as the noise level increases, with the quality dropping from about 20 dB at the lowest noise level to 12-13 dB for the strongest distortion. The two curves follow a very similar trend: the Matlab implementation yields slightly higher PSNR for most noise levels, but the difference remains within roughly 0.5-1 dB and becomes negligible for high noise intensities. Overall, both implementations respond to mixed noise in a similar way, while the proposed filtering scheme preserves a noticeable quality advantage even for heavily corrupted images, though with smaller gains at the highest noise level.

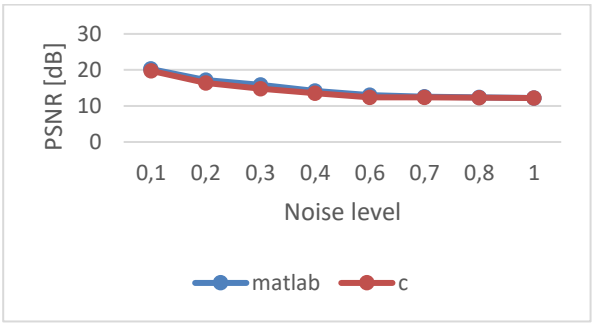

Fig. 6. Degradation of image quality (PSNR) with increasing mixed noise level (Gaussian = Salt & Pepper = Speckle) – Matlab vs Vitis.

To assess the sensitivity of the algorithm to changes in input scale, a series of experiments was carried out using identical mixed noise parameters and three input resolutions: 256x256, 512x512 and 1024x1024 pixels (Table II). Reconstruction quality was evaluated using the PSNR and MS-SSIM metrics, while the computational load was quantified in terms of execution time on the Matlab and Vitis platforms. For each configuration, the PSNR before filtering is identical in MATLAB and in athe Vitis (C/ARM) implementation, since both pipelines operate on the same noisy inputs. At the lower noise level (0.1, 0.1, 0.1) the pre-filtering PSNR is approximately 10 dB, whereas for the stronger noise (0.2, 0.2, 0.2) it decreases to about 8.3 dB, which clearly reflects the impact of increasing noise intensity on the signal-to-noise ratio.

After applying the proposed filter cascade, PSNR increases by roughly 7–11 dB, depending on the image resolution and noise level. For the 0.1 case, the post-filtering PSNR in MATLAB rises from about 18.8 dB (256×256) to 21.1 dB (1024×1024), while the Zynq implementation achieves values from 18.5 dB to 20.6 dB over the same range. For the 0.2 case, the corresponding PSNR values are lower, ranging from approximately 15.9 dB to 17.3 dB in MATLAB and from 15.8 dB to 16.8 dB on Zynq. Overall, increasing the resolution from 256 to 1024 pixels yields a moderate PSNR improvement of about 1–2.5 dB in both environments, whereas higher noise levels reduce the attainable PSNR after filtering by roughly 2–3 dB compared to the milder noise scenario.

The direct comparison between MATLAB and Zynq shows that the PSNR differences after filtering remain small, typically within 0.1–0.8 dB and usually in favour of MATLAB. From a practical standpoint, this indicates that the reconstruction quality obtained on the Zynq platform is very close to the MATLAB reference, and the observed deviations are largely negligible from a perceptual point of view. Combined with the substantially shorter execution times on Zynq, these results suggest that the C/ARM implementation preserves almost the same PSNR level as MATLAB while operating at a significantly lower computational cost, making it a favorable choice for time-constrained applications.

TABLE II
IMPACT OF IMAGE RESOLUTION 256-1024PX ON QUALITY AND PROCESSING TIME- MATLAB VS VITIS UNDER MIXED-NOISE CONDITIONS

file	noise			Resize	Matlab						Vitis		
	Gaussian	salt& pepper	speckle		before	after	before	after	before	after	before	after	t
	(var)	(amount)	(var)		MS-SSIM	MS-SSIM	PSNR [dB]	PSNR [dB]	t [s]	t [s]	PSNR [dB]	PSNR [dB]	[s]
1	0,1	0,1	0,1	256	0,41	0,68	10,03	18,77	123	31,41	10,03	18,54	0,21
2	0,1	0,1	0,1	512	0,34	0,72	10,04	20,23	111	23,67	10,04	19,76	1,25
3	0,1	0,1	0,1	1024	0,28	0,63	10,05	21,06	656	23,94	10,05	20,57	9,45
4	0,2	0,2	0,2	256	0,30	0,48	8,25	15,93	91	24,25	8,25	15,79	0,39
5	0,2	0,2	0,2	512	0,25	0,55	8,25	17,17	224	27,36	8,25	16,43	1,62
6	0,2	0,2	0,2	1024	0,20	0,63	8,26	17,31	422	23,96	8,26	16,81	7,10

VI. CONCLUSIONS

In this work, a meta-filtration concept was proposed, based on adaptive selection and cascading of multiple filters for images by unknown mixed noise. The algorithm automatically tests different combinations and orders of filters, allowing repetitions and omissions, and a given stage is accepted only if it improves the selected quality metric. As a result, an individually tailored sequence of filtering operations is obtained, without the need for prior knowledge of the dominant type of disturbance or manual tuning of the filtering chain.

Operation that yield a measurable improvement in a quality with respect to the reference image are retained, where as stages that don't provide any benefit are discarded. This mechanism can be interpreted as a form of heuristic optimization, related to concepts well known from the literature, like:

1. *Greedy algorithm*- making decisions based on the immediate, local gain in quality.
2. *Evolutionary or generic algorithms*- where each filter is treated as a gene in the processing sequence and image quality acts as the fitness function.
3. *Reinforcement learning*- where the system adaptively selects actions (filters) that maximize a chosen quality metric.

The proposed solution was implemented in two environments: as a high-level prototype in matlab (with GPU support using the Image Processing Toolbox and Parallel Computing Toolbox) and as a C implementation on the ZedBoard platform (Zynq-7000 SoC), developed in the Vitis environment with the use of Vivado and PetaLinux. For a wide range of mixed-noise configurations (Gaussian, salt&pepper, speckle), both processing paths achieved a clear improvement in quality- the PSNR gain typically fell within approximately 7-11dB and in favorable cases, approached 20dB. Complementing the evaluation with the MS-SSIM metric confirmed that essential structures and details of the image preserved ale from the perceptual point of view.

In contrast to classical approaches, in which filter sequences are predetermined based on expert knowledge or predefined heuristics, the proposed method relies on a fully automatic exploration of the space of possible filter configurations and their orderings. The core of the algorithm is local quality

assessment performed at each processing stage using a selected image similarity metric.

The proposed solution can therefore be interpreted as a form of intelligent, adaptive selection of image filters, providing an alternative to traditional approaches based on rigid, predefined processing pipelines.

The authors recognize the potential for further extension and validation of the proposed software. An interesting direction for future work would be to evaluate its performance in the presence of one of the most challenging types of degradation, namely motion blur. It would also be justified to carry out an analysis of the code execution time. [25] [26]

Although the available literature describes numerous algorithms designed for this type of task, it usually lacks detailed quantitative data comparing the original image with its denoised or reconstructed counterpart.

The authors also highlight the potential benefits of using the Vitis Vision library to increase the efficiency of algorithm implementation. [27] An alternative direction for further development would be to migrate the design to a more modern hardware platform, such as Zynq UltraScale+ or Artix-7 devices, which could potentially yield shorter processing times and improved overall system performance.

VII. REFERENCES

[1] A. A. Khan, A. A. Laghari i S. A. Awan, „EAI Endorsed Transactions Scalable Information Systems," *Machine Learning in Computer Vision: A Review,* pp. 1-11, August 2021. doi: 10.4108/eai.21-4-2021.169418

[2] R. Tadeusiewicz i P. Korohoda, „Komputerowa analiza i przetwarzanie obrazów," Kraków, Wydawnictwo Fundacji Postępu Telekomunikacji, 1997.

[3] V. Wiley i T. Lucas, „Computer Vision and Image Processing: A Paper Review," *International Journal Of Artificial Intelligence Research,* pp. 28-36, June 2018. doi: 10.29099/ijair.v2i1.42

[4] P. Zeng, B. Zhu, D. Liu i J. Zhu, „Application of Image Processing Based on Multiple Filters in an Alignment System," *High Power Laser Science and Engineering,* pp. 1-6, 1 July 2014. doi: 10.1017/hpl.2014.23

[5] W. Zhang, „Image Denoising Algorithm of Refuge Chamber by Combining Wavelet Transform and Bilateral Filtering," *International Journal of Mining Science and Technology* , pp. 221-225, 13 May 2013. doi: 10.1109/ICCT.2018.8599921

[6] W. Yu, F. Liu, L. Zheng i D. Wang, „De-noising Method Based on Median Filter and Wavelet Transform," w *Proceedings of the 5th International Conference on Computer Engineering and Networks (CENet 2015)*, Shanghai, 2015. doi: 10.1109/SOPO.2012.6270998

[7] M. R. Sunitha i D. Asha, „Comparision of Gaussian and Median Filters to Remove Noise in Dental Images," *Institute of Scholars,* pp. 1430-1436, 8 August 2020. doi: 10.59934/jaiea.v4i3.1033

[8] R. C. Gonzales i R. E. Woods, w *Digital Image Processing. Fourth Edition*, Edinburgh, Pearson, 2018, pp. 1-1022.

[9] S. k. Jadwa, „Wiener Filter based Medical Image De-noising," *International Journal od Science and Engieneering Applications,* pp. 318-323, September 2018. doi: 10.7753/IJSEA0709.1014

[10] K. S. Chethan, K. R. Nataraj, G. S. Sinchana i A. L. Choodarathnakara, „Analysis of Image Quality using Sobel Filter," w *International Conference on Inventive Systems and Control (ICISC 2019)*, Mysore, 2019. doi: 10.1109/ICISC44355.2019.9036369

[11] K. Krawczyk, I. Winnicki, J. Jasiński, K. Kroszczyński i S. Pietrek, „Maski wybranych krawędziowych filtrów Laplace'a w przetwarzaniu danych cyfrowych," *Biuletyn WAT,* pp. 145-170, 2012.

[12] S. Venkatesh, P. Subhashini i K. Somasundaram, „Image Enhancement and Implementation of CLAHE Algorithm and Bilinear Interpolation," *Cybernetics and Systems,* 17 November 2022. doi: 10.1080/01969722.2022.2147128

[13] M. J. Alwazzan, M. A. Ismael i A. N. Ahmed, „A Hybrid Algorithm to Enhance Colour Retinal Fundus Images Using a Wiener Filter and CLAHE," *Journal of Digital Imaging,* pp. 750-759, 22 April 2021. doi: 10.1007/s10278-021-00447-0

[14] M. Braik, M. Azmi Al-Betar, M. A. Mahdi, M. Al-Shalabi, S. Ahamad i S. A. Saad, „Enhancement of satellite images based on CLAHE and augmented elk herd optimizer," *Artificial Intelligence Review,* pp. 1-74, December 2024. doi: 10.1007/s10462-024-11022-8

[15] A. Aishvarrya, G. N. Diksha i H. S. Prashantha, „Image Denoising Using Hybrid Filtering Techniques," *International Journal of Creative Research Thoughts (IJCRT),* pp. 488-496, 6 June 2021. DOI:10.1729/JOURNAL.27404

[16] K. Dabov, A. Foi, V. Katkovnik i K. Egiazarian, „Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," *IEEE Transactions on Image Processing,* pp. 2080-2095, August 2007. doi: 10.1109/TIP.2007.901238

[17] M. Aggarwal, R. Kaur i B. Kaur, „A Review of Denoising Filters in Image Restoration," *International Jouranl of Current Research and Academic Review,* pp. 83-89, 2014.

[18] A. Aishvarrya, G. N. Diksha i H. S. Prashantha, „Image Denoising Using Hybrid Filtering Techniques," *International Journal of Creative Research Thoughts,* pp. 488-496, 6 June 2021. dOI:10.1729/JOURNAL.27404

[19] A. Horé i D. Ziou, „Image quality metrics: PSNR vs. SSIM," w *2010 International Conference on Pattern Recognition*, Istanbul, 2010. doi: 10.1109/ICPR.2010.579

[20] M. N. Moindop i B. O. Yenke, „Hybrid Image Filtering Method Based on Wavelet Transform," *European Journal of Electrical Engineering and Computer Science,* pp. 38-45, 8 August 2024.

[21] Á. Mándi, J. Máté, D. Rózsa i S. Oniga, „Hardware accelerated image processing on FPGA based PYNQ-Z2 board," *Carpathian Journal od Electronic and Computer Engineering,* pp. 20-23, 2021. doi: 10.2478/cjece-2021-0004

[22] D. Tsiktsiris, D. Ziouzios i M. Dasygenis, „HLS Accelerated Noise Reduction approach using Image Stacking on Xilinx PYNQ," w *8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, Thessaloniki, 2019. doi: 10.1109/MOCAST.2019.8741574

[23] P. Palazzari, M. Faltelli i F. Iannone, „FIPLib: An Image Processing Library for FPGAs Using High-Level Synthesis," *International Journal od Parallel Programming,* pp. 1-23, 2025. doi: 10.1007/s10766-025-00784-5

[24] K. Sehairi i T. Bouwmans , „Accelerating image processing functions on ZYNQ- FPGA using the Vitis Vision Library," w *IEEE International Conference on Advanced Electrical Engineering*, 2024. doi: 10.1109/ICAEE61760.2024.10783257

[25] Y. Chen i J. Hua, „Image Restoration Algorithm Research on Local," *Information Engineering and Electronic Business,* pp. 23-29, March 2011. doi: 10.5815/ijieeb.2011.03.04

[26] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis i W. T. Freeman, „Removing Camera Shake from a Single Photograph," *ACM Transactions on Graphics (TOG),* pp. 784-794. doi: 10.1145/1141911.1141956

[27] K. Sehairi i T. Bouwmans , „Accelerating image processing functions on ZYNQ- FPGA using the Vitis Vision Library," w *IEEE International Conference on Advanced Electrical Engineering*, 2024. doi: 10.1109/ICAEE61760.2024.10783257