Proceedings of CONF-CDS 2025 Symposium: Application of Machine Learning in Engineering
DOI: 10.54254/2755-2721/2025.AST26339

Systematic Research on Hardware Design of Bilateral Filter

Zhiming Wang

Wuhan Britain-China School, Wuhan, China
1372304135@qq.com

Abstract. In daily life, during the acquisition of images and videos, the presence of noise is often unavoidable due to factors such as sensor performance and environmental interference. Such noise degrades image quality and interferes with subsequent processing tasks like image analysis and target recognition. Therefore, the development of efficient image denoising techniques to improve image quality has become an important research direction in the field of digital image processing. Among numerous image filtering methods, bilateral filtering has attracted extensive attention due to its excellent performance in effectively suppressing noise while well preserving image edges and details. This paper conducts a comprehensive study on bilateral filtering methods, including research on typical algorithms and hardware design. The research in this paper is conducive to a systematic and thorough understanding of the progress in bilateral filtering hardware design methods, and provides a reference for future research on better bilateral filtering hardware as well as the design of other image processing hardware.

Keywords: Image processing, Bilateral filter, Hardware and software design, Algorithm optimization

1. Introduction

Digital image processing is fundamental across diverse fields. A core challenge in this domain is effectively enhancing image quality by reducing noise while preserving critical structural details, such as edges and textures. Image filtering encompasses techniques designed to decrease noise. Generally, signals in images can be broadly categorized as spatial details, spectral components, and noise artifacts. The key characteristics for image filtering include locality (operations often use local neighborhoods), sensitivity to noise type, and the critical trade-off between noise suppression and detail preservation. Therefore, diverse image filtering methods are given.

Classical filtering methods include three typical kinds as follows. Firstly, the Gaussian Filtering is mentioned in [1]. This is about using an algorithm for wavelet transform to denoise the image, which can improve the signal-to-noise ratio of the image and retain as much edge information as possible. Secondly, the typical Median Filtering (MF) methods are analyzed in [2] for their variants for removing impulse noise from grayscale images. It compares Extended Median Filtering (EMF), Modified Boundary Discriminative Noise Detection (MBDND) and Improved Adaptive Median Filtering (IAMF) methods, respectively. In terms of functionality, time complexity and performance (using metrics like Peak Signal to Noise Ratio (PSNR)), it finds that the EMF and MBDND perform

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.AST26339

well, while the IAMF has optimal time complexity. While effective for noise smoothing, they inherently blur edges and fine textures. Hence, the famous Bilateral Filtering (BF) is proposed in [3] finally. It is becoming a cornerstone. The BF combines a standard spatial Gaussian kernel with a range (intensity) Gaussian kernel, weighting pixels based on both their geometric closeness and their photometric similarity. This dual-domain approach allows it to smooth homogeneous regions effectively while preserving strong edges.

Despite its effectiveness, the standard BF is computationally intensive, especially for large kernels or high-resolution images. Significant research focuses on effectiveness optimization can be achieved by approximation techniques. DHBF(Dual-Histogram BF) [4] is a method aimed at improving the working efficiency and increasing the effect of reducing the noise level. DHBF achieves this via a two-stage framework. It utilizes local histograms and box filters to approximate the range kernel effect, achieving significant speedups (>10x). However, the filtering accuracy, especially near edges and for high-variance range kernels, can be compromised. Therefore, it is required to note the limitations in handling strong edges gracefully. Another example is given in [5]. This research proposes a fast bilateral filtering algorithm that achieves significant acceleration while maintaining high accuracy by elevating the image to a higher-dimensional space for convolution and downsampling.

In addition to the software acceleration methods above, hardware acceleration is crucial for real-time BF applications. Typical hardware platform based BF architectures are designed include of FPGA-based implementation [6], ASIC implementations [7] and many-core GPU Implementation [8]. In [6], a novel method presents an FPGA-based scalable bilateral filtering architecture with O(1) complexity via Taylor approximation and box filters, outperforming conventional $O(\omega^2)$ methods. It offers stable resources, reconfigurability, minor errors compensable by higher orders. Choi et al. proposes a method focusing on the ASIC implementation of the bilateral filtering [7]. It exploits the Similar Weight Grouping (SWG) and Zero Value Suppression (ZVS) methods, reducing the size of the Look-Up Table (LUT) by approximately 95% while maintaining almost unchanged performance. Implemented using a 65 nm process, it utilizes 93.1 K gates and 7.68 KB memory with a power consumption of 5.46 mW, suitable for high-performance image processing. A method in [8].It utilizes GPU multi-threading to optimize memory access, achieving a speedup of over 30 times compared to the CPU version without losing computational accuracy.

It can be seen that, different optimal bilateral filtering methods can achieve different efficiency including its implementation complexity. Especially for the hardware design of bilateral filtering, efficient designs and implementation methods will be help a lot for application. In order to gain a more systematic and comprehensive understanding and research of bilateral filtering hardware design, this paper aims to study typical bilateral filtering algorithms, hardware design architectures, and the process of hardware implementation method.

The remainder of this paper is organized in the following sections. Section II reviews the algorithm of BF methods. Section III describes the typical BF hardware design. Section IV provides the implementation of hardware BF architecture. Finally, Section V draws the conclusions of this paper.

2. Bilateral filtering algorithms

In this section, the bilateral filtering algorithm will be introduced, including the introduction of the typical bilateral filtering algorithm and one fast hardware-oriented bilateral filtering algorithm.

DOI: 10.54254/2755-2721/2025.AST26339

2.1. Bilateral filtering algorithm

First of all, the basic definition and formulas of bilateral filtering is for an input image I, the output Js of bilateral filtering at pixel s is defined in [8] as,

$$J_s = \frac{1}{k(s)} \sum_{p \in \Omega} f(p-s) \cdot g(I_p - I_s) \cdot I_p \tag{1}$$

where the normalization term $k(s)$ is computed by,

$$k(s) = \sum_{p \in \Omega} f(p-s) \cdot g(I_p - I_s) \tag{2}$$

In Eq. (2), the term $f(p-s)$ denotes the spatial gaussian weight, depending on the spatial distance between pixels p and s by,

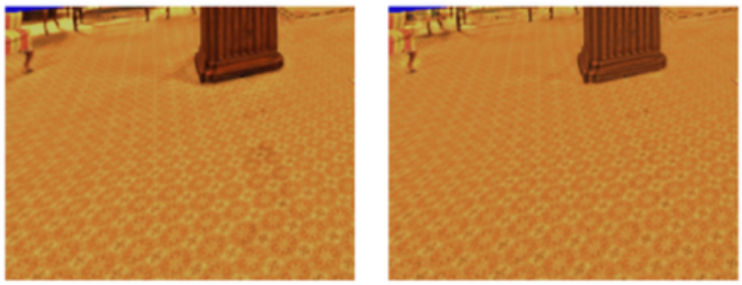
$$f(x) = e^{-\frac{x^2}{2\sigma_s^2}} \tag{3}$$

where σ_s is the spatial standard deviation. The term $g(I_p - I_s)$ is the range weight, depending on the intensity difference between pixels p and s by,

$$g(x) = e^{-\frac{x^2}{2\sigma_r^2}} \tag{4}$$

where σ_r is the range standard deviation. Additionally, the term Ω is neighborhood of pixel s.

Bilateral filtering achieves edge-preserving smoothing by combining spatial distance and intensity similarity weights, avoiding edge blurring caused by Gaussian filtering. One experimental result is shown in Figure 1. Comparing the detail preservation effects of Gaussian filtering and Bilateral Filtering, it can be seen that the BF can achieve more details.



(a) Gaussian filtering (b)Bilateral Filtering

Figure 1. Filtering results comparison for Gaussian filtering and Bilateral Filtering [8]

2.2. Fast bilateral filtering algorithm

One way to increase the speed of the bilateral filtering algorithm is defined in [8], named the Piecewise Linear Approximation method. The core idea is to replace computationally expensive nonlinear components with a set of simpler linear approximations. To approach it, the method divides the intensity range into multiple intervals, and each interval is approximated using a linear function. Convolution is then applied separately to each interval, and the results are combined. The

DOI: 10.54254/2755-2721/2025.AST26339

advantage is that it can transform the original nonlinear filtering operation into a series of efficient linear operations, significantly improving computational efficiency.

The acceleration strategy is thus as follows. The authors discretize the set of possible signal intensities into segment values i^j, and compute a linear filter for each such value,

$$\begin{aligned} J_s^j &= \frac{1}{k^j(s)} \sum_{p\in\Omega} f\Big(p-s\Big) g\left(I_p - i^j\right) I_p \\ &= \frac{1}{k^j(s)} \sum_{p\in\Omega} f\Big(p-s\Big) H_p^j \end{aligned} \tag{5}$$

and

$$\begin{aligned} k^j\Big(s\Big) &= \sum_{p\in\Omega} f\Big(p-s\Big) g\left(I_p - i^j\right) \\ &= \sum_{p\in\Omega} f\Big(p-s\Big) G^j\Big(p\Big) \end{aligned} \tag{6}$$

The final output of the filter for a pixel s is then a linear interpolation between the output J_s^j of the two closest values I^j of I_s. This corresponds to a piecewise-linear approximation of the original bilateral filter.

The comparison with traditional method in speed is shown in Figure 2. We can safely conclude that the Piecewise Linear method can significantly decrease the time.

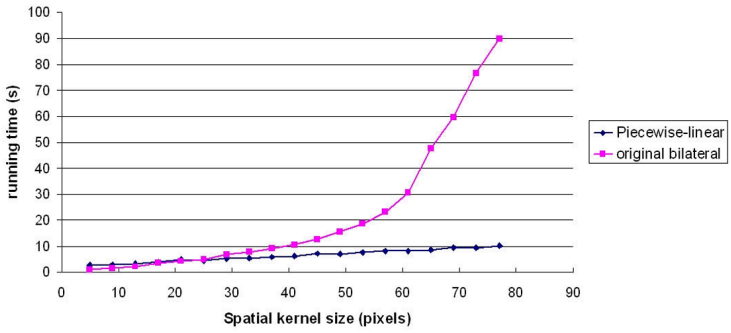


Figure 2. Speed-up of the piecewise-linear acceleration for 17 segments and a 576x768 image [8]

3. Bilateral filtering hardware structures

In this section, we will elaborate on the hardware architecture of bilateral filtering, encompassing both the hardware-oriented fast bilateral filtering algorithm and its corresponding hardware structure.

3.1. Bilateral filtering algorithm for hardware optimization

In [9], Xie A et al. developed an FPGA-based hardware architecture of gaussian-adaptive bilateral filter for real-time image denoising. Based on the basic equations of bilateral filtering as (1) and (2), this algorithm optimize the basic process in two ways.

Firstly, the implementation of the bilateral filter requires complex exponential operations, which consume a lot of logical resources. According to the properties of exponents, $0 < f(p-q) \leq 1$ and $0 < g(p-q) \leq 1$, fixed-point numbers with only fractional parts can be used instead of

DOI: 10.54254/2755-2721/2025.AST26339

floating-point numbers. This method is named look-up table-based fast exponential calculation. For example, in FPGA implementation, the pre-calculated results of the spatial domain filter kernel as Eq. (3) are stored in ROM (Read Only Memory). When using 17-bit fixed-point floating-point numbers, 17 bit memory resources are consumed. In this way, exponential operations can be directly implemented with integer multipliers and shift operations, saving logical resources.

Secondly, in FPGA hardware implementation, its pipeline and parallel characteristics are fully utilized to improve operation speed. When processing images, the input image enters the $N \times N$ window register after $N-1$ line buffers, the product of the filter kernel coefficient and the neighborhood pixel value is generated through the row calculation unit, and then the relevant terms are summed respectively and divided to obtain the filtered pixel value. Taking the processing of a 5×5 window bilateral filtering as an example, compared with the input clock, it can be processed with more than five times the clock (processing five pixels at a time). The data input module caches low-speed image data and reads data at a high-speed clock for processing.

The above optimization methods enable the bilateral filtering algorithm to run on hardware platforms with limited resources.

3.2. Introduction to bilateral filter hardware structure

In this section, three modules of the critical bilateral filter hardware structures defined in [10,11]will be introduced. They are Distance-Oriented Grouping Module, Summation and Normalization Module and Pipelining and Parallel Processing.

The First module is the Distance-Oriented Grouping Module. The function is to Groups pixels in the 5×5 window by their Euclidean distance from the center pixel as shown in Figure.3, reducing redundant computations. It can be seen that pixels with the same distance (e.g., distance=1, 2, 4, 5) are grouped into 6 parallel processing channels, while for distance=5 (8 pixels), two sub-groups are created to balance parallelism.

DOI: 10.54254/2755-2721/2025.AST26339

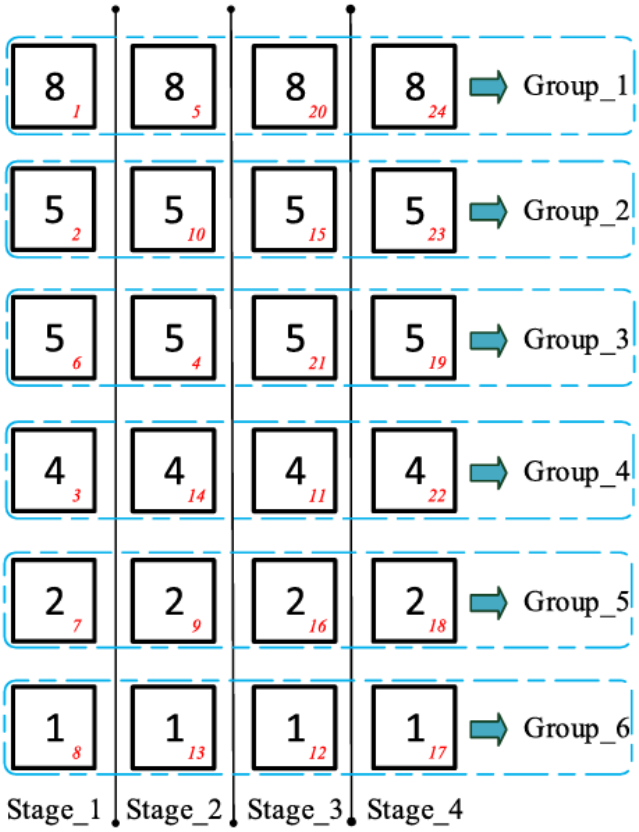


Figure 3. Proposed distance-oriented grouping of window size 5x5 [10]

The second module is Summation and Normalization Module. Multiplies each pixel I_p by its combined weight $f(p-s)\cdot g(Ip-Is)$ and sums the products via an adder tree as shown in Figure 4, generating the numerator of Eq. (1) for kernel result calculation. Then, to achieve the Normalization Term Calculation, sums the combined weights $f(p-s)\cdot g(Ip-Is)$ via another adder tree in Figure 5 by generating the denominator of Eq. (1). Finally, we can perform integer division of the kernel result by the normalization term to produce the final filtered pixel J_s in the division unit.

DOI: 10.54254/2755-2721/2025.AST26339

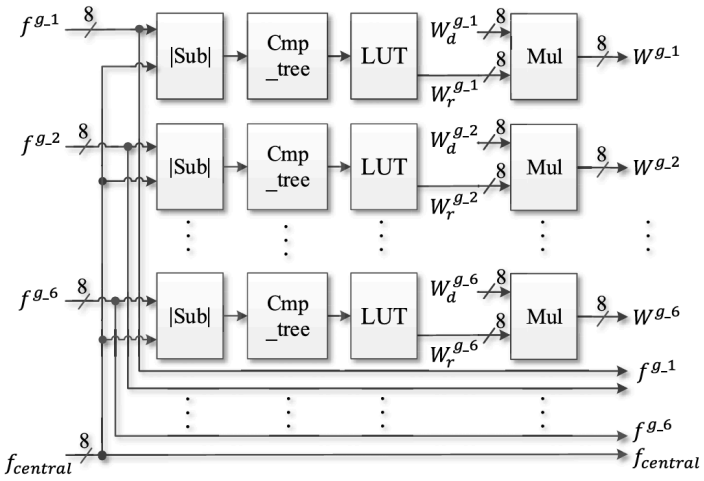


Figure 4. Hardware architecture of kernel result calculation [10]

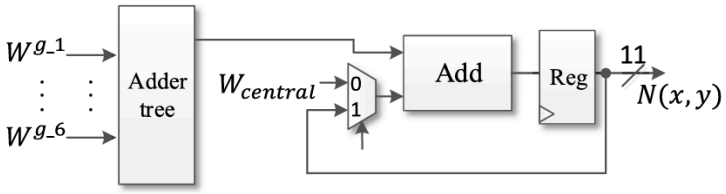


Figure 5. Hardware architecture of normalization term calculation [10]

Table 1. A comparison between the proposed design and a previous design

Parameter	[10]	[6]
Max Clock Frequency	236.7 MHz	220 MHz
Frame Rate (1024×1024)	56.43 fps	52.45 fps
Multipliers Used	12	23
LUT Entries per Channel	12	256
Slice Usage (FPGA)	2304	3180
DSP48Es Used	36	115

The third module is Pipelining and Parallel Processing. It utilizes a 33-stage pipeline that is optimized to overlap computation across the 5×5 window, ensuring that each pixel is processed in 4 clock cycles. Parallel channels for each distance group enable simultaneous processing of 24 neighborhood pixels, matching the input pixel stream rate with 6 groups as shown in Figure 3. By using these combined methods, it can achieve excellent performance results while maintaining image quality; the hardware cost is significantly reduced, as listed in Table 1. It can be seen that the design referenced in [10] appears to be more efficient in terms of hardware resource utilization while also offering better performance in terms of clock frequency and frame rate compared to the design in [6]. It uses fewer multipliers, LUT entries, FPGA slices, and DSP48E slices, which can lead to lower costs, reduced power consumption, and potentially easier integration into larger systems.

DOI: 10.54254/2755-2721/2025.AST26339

4. Achieve flow of bilateral filtering hardware

This section introduces two typical hardware implementation flows of bilateral filtering, including the traditional RTL-based approach and the high-level EDA software synthesis-based approach.

4.1. RTL implementation of bilateral filtering

The RTL implementation of bilateral filtering can be simply introduced by the following several steps.

Firstly, we can use window extraction to utilize line buffers and shift registers to extract a local window (e.g., 5×5) centered on the target pixel, processing the image stream synchronously. Secondly, we can compute spatial weights based on the fixed positional coordinates of pixels within the window. Employs parallel look-up operations (LUT) from a precomputed ROM to obtain coefficients reflecting the principle “closer proximity yields higher weight”. Thirdly, the intensity difference between each window pixel and the central pixel is calculated by maps the difference to a “greater similarity yields higher weight” coefficient via a dedicated range kernel LUT stored in ROM, then performs real-time multiplication of each pixel’s spatial weight and range weight to generate the composite weight. Simultaneously computes the weighted pixel value for each location. At last, we use employs adder trees for parallel accumulation of all weighted pixel values and the sum of all composite weights. A divider module outputs the normalized result. To improve the efficiency, we can use a multi-stage pipeline for precise timing control, processing one pixel per clock cycle. Synchronizes with horizontal synchronization signals to ensure real-time throughput. It is a traditional way to implement hardware bilateral filtering. For example, in [11], Gabiger-Rose A et al. use RTL method to achieve it.

4.2. Vivado HLS implementation of bilateral filtering

Another solution to bilateral filtering is given in [12]. It is achieved based on the EDA named Vivado HLS. The process can be simply divided into several steps.

Firstly, the core task is to develop a behavioral-level bilateral filter function in C++ (no RTL details), and use Vivado HLS to synthesize C++ code into RTL. Secondly, we need to guide hardware generation by adding compilation instructions. The aim of optimization is parallelization, including array partitioning to concurrent pixel processing and pipelining to ensure continuous data flow. The resource control includes Direct ROM storage to LUTRAM (conserves BRAM). Thirdly, the automated synthesis and verification integration of the system are conducted. The typical workflow is shown in Figure 6.

DOI: 10.54254/2755-2721/2025.AST26339

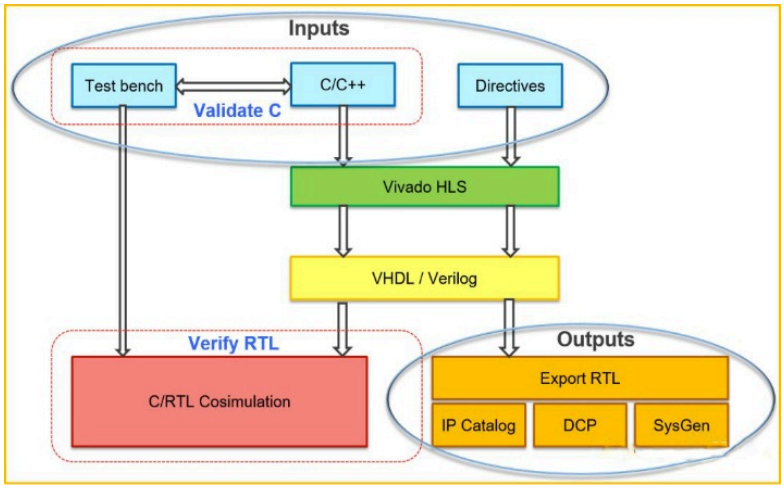


Figure 6. The full steps of RTL implementation

4.3. The hardware achievement method comparsion

For these two methods, there are three differences between them. The first one is methodological divergence. The RTL implementation employs manual register-transfer level design, requiring precise construction of finite state machines, data paths, and memory architectures (e.g., chained line buffers and custom adder trees) to achieve cycle-accurate timing control and resource reuse. In contrast, Vivado HLS synthesizes hardware from behavioral C++/C algorithm specifications (spatial/range weighting and normalization) through pragma directives, abstracting low-level timing details. Secondly, optimization capability is also different. RTL delivers ultimate optimization with power, performance, and area, which can customize a pipeline design for high throughput. While Vivado HLS prioritizes development efficiency with C/RTL co-simulation for functional verification, it incurs ≈10-20% resource overhead in auto-generated hardware, with timing closure dependent on tool scheduling. The last main difference is the application context. RTL suits ASIC/ultra-low-power designs with strict constraints, while Vivado HLS excels in rapid algorithm iteration, trading marginal degradation for agility.

5. Conclusion

This paper conducts a detailed investigation into research methods for hardware-based bilateral filtering. Firstly, it introduces typical bilateral filtering algorithms, followed by optimized fast bilateral filtering algorithms. Secondly, it presents the hardware structures of bilateral filtering, and finally, it describes the implementation of hardware-based bilateral filtering. The research in this paper is conducive to a systematic and comprehensive understanding of the progress in bilateral filtering hardware design methods, and provides a reference for future research on better bilateral filtering hardware as well as the design of other image processing hardware.

References

[1] You N, Han L, Zhu D, et al. Research on image denoising in edge detection based on wavelet transform [J]. Applied Sciences, 2023, 13(3): 1837.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Shah A, Bangash J I, Khan A W, et al. Comparative analysis of median filtering and its variants for removal of impulse noise from gray scale images [J]. Journal of King Saud University-Computer and Information Sciences,

DOI: 10.54254/2755-2721/2025.AST26339

2022, 34(3): 505-519.K. Elissa, “Title of paper if known, ” unpublished.
[3] Tomasi C, Manduchi R. Bilateral filtering for gray and color images [C]//Sixth international conference on computer vision (IEEE Cat. No. 98CH36271). IEEE, 1998: 839-846.Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface, ” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[4] Cheng S W, Lin Y T, Peng Y T. A Fast two-stage bilateral filtering using constant time O (1) histogram generation [J]. Sensors, 2022, 22(3): 926.
[5] Paris S, Durand F. A fast approximation of the bilateral filtering using a signal processing approach [C]//European conference on computer vision. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006: 568-580. Mittal S. A survey of FPGA-based accelerators for convolutional neural networks [J]. Neural computing and applications, 2020, 32(4): 1109-1139.
[6] Dabhade S D, Rathna G N, Chaudhury K N. A reconfigurable and scalable FPGA architecture for bilateral filtering [J]. IEEE Transactions on Industrial Electronics, 2017, 65(2): 1459-1469.
[7] Choi J M, Jang S J, Lee S S, et al. Memory optimization of bilateral filtering and its hardware implementation [C]//The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014). IEEE, 2014: 1-2.
[8] Paris S. A gentle introduction to bilateral filtering and its applications [M]//ACM SIGGRAPH 2007 courses. 2007: 3-es.
[9] Xie A, Zhang A, Mei G. An FPGA-based hardware architecture of gaussian-adaptive bilateral filter for real-time image denoising [J]. IEEE Access, 2024.
[10] Lien C Y, Tang C H, Chen P Y, et al. A low-cost VLSI architecture of the bilateral filter for real-time image denoising [J]. IEEE Access, 2020, 8: 64278-64283.
[11] Gabiger-Rose A, Kube M, Weigel R, et al. An FPGA-based fully synchronized design of a bilateral filter for real-time image denoising [J]. IEEE Transactions on Industrial Electronics, 2013, 61(8): 4093-4104.
[12] Cortes A, Velez I, Irizar A. High level synthesis using Vivado HLS for Zynq SoC: Image processing case studies [C]//2016 Conference on design of circuits and integrated systems (DCIS). IEEE, 2016: 1-6.