INTERNATIONAL JOURNAL OF
COASTAL, OFFSHORE & ENVIRONMENTAL ENGINEERING IJCOE Vol.8/No. 4/Autumn 2023 (49-58)

Available online at: www. Ijcoe.org

Underwater Image Enhancement Using FPGA-Based Gaussian Filters with Approximation Techniques

Mehrnaz Monajati[1]

[1] *Assistant professor, Graduate University of Advanced Technology, Kerman; m.monajati@kgut.ac.ir*

ARTICLE INFO

Article History:
Received: 02 Jan 2022
Accepted: 05 Oct. 2022

Keywords:
Underwater image
Gaussian filter
Low power
High speed
FPGA

ABSTRACT

The major challenge in marine environment imaging lies in addressing the haziness induced by natural phenomena, such as absorption and scattering in underwater scenes. This haze significantly impacts the visual quality of underwater images, necessitating improvement. This paper presents a novel approach aimed at enhancing the efficiency of Gaussian filters for reducing Gaussian noise in underwater images. The method introduces a pipeline structure in the Gaussian filter implementation and evaluates the influence of employing approximate adders on overall performance. Simulation results reveal a notable speed enhancement exceeding 150%, coupled with a substantial reduction in power consumption exceeding 34%. However, these advantages are tempered by an increase in spatial requirements. The study recognizes the inherent tradeoff between output quality and power, highlighting the applicability of the proposed design in error-resilient applications, particularly in image and video processing domains. In essence, the presented approach offers a compelling solution where the benefits of accelerated speed and reduced power consumption outweigh spatial constraints, contributing to the advancement of underwater image enhancement techniques.

1. Introduction

Underwater imaging instruments are essential for remote sensing due to Earth's significant water coverage. Optical sensors capture acoustic signals, but resulting images suffer from issues like limited light range, lighting disruptions, low contrast, and color degradation. These disturbances necessitate pre-processing before crucial image processing tasks. Edge-preserving filters help denoise images without compromising quality or edges. Pre-processing involves correcting illumination, suppressing noise, enhancing contrast, and adjusting colors. Underwater images require pre-processing due to degradation caused by light transmission properties, environmental factors, and video capture challenges. Identifying suitable filters for effective pre-processing is a key focus in addressing these issues.

Exploring the underwater environment through video and images is a complex and fascinating endeavor, marked by challenges such as noise, limited visibility, light scattering, attenuation issues, non-uniform lighting, and other inherent factors of seawater [1, 2]. The use of artificial light sources exacerbates problems, introducing issues like image blur, haziness, and the presence of bluish or greenish hues in underwater images. This leads to absorption, scattering, color distortion, and noise [3]. These challenges pose a significant hurdle for researchers seeking to harness computer technology for in-depth studies, especially in the classification, recognition, and segmentation of coral reef components and various fish species.

Understanding underwater ecology is crucial for effective marine resource management and monitoring. Consequently, image analysis techniques rooted in computer vision and image processing technologies have been developed to facilitate proper monitoring of marine resources. Despite these advancements, there has been no comprehensive quantitative or qualitative assessment of all underwater marine resources, as noted in [4, 5].

Underwater images often exhibit poor visibility, a foggy appearance, and misty issues, with blurring occurring due to wave surfaces during object acquisition. Additionally, these images are prone to unwanted noise and increased scattering effects. Distortions vary across images acquired at different

times. Researchers globally are exploring a range of techniques, from simple approaches to multilayered deep techniques, to mitigate distortion effects on underwater images. The ideal technique should strike a balance between effectiveness and complexity.

The PCA fusion-based method integrates homomorphic filters, adaptive histogram equalization, and median filters for individual color channels. This technique is applied to fuse two images, enhancing color contrast and resulting in improved qualitative outcomes [6]. CLAHE proves effective in enhancing the visual quality of underwater images by addressing issues such as uneven illumination. It achieves adaptability through the application of adaptive histogram equalization and introduces a weighted map to enhance the visibility of distant objects [7]. The combination method involves integrating CLAHE and dark channel prior for image enhancement. This approach includes identifying the presence of artificial light, removing it if detected, and subsequently applying CLAHE to improve the overall quality of underwater images [8].

Histogram equalization image enhancement method was introduced to advance CLAHE, involving the conversion of RGB input images to the HSV color space. Individual components (Hue, Saturation, and Value) underwent histogram equalization, leading to improved visual quality in the output image compared to the input [9]. Additionally, a method named mixture contrast limited adaptive histogram equalization was developed, where the RGB input image was transformed into the HSV color space [10]. Both CLAHE-RGB and CLAHE-HSV images were generated and combined using Euclidean norm, contributing to enhanced underwater image quality.

In addressing the challenge of edge identification in acoustic underwater images, the study detailed in [11] enhanced these images by utilizing Wiener filtering to reduce speckle noise while preserving high-frequency components. Furthermore, a median filter was applied to eliminate small objects. Performance evaluation included the extraction of local minimum and maximum values through morphological operations. The resulting edge maps, when compared with Canny and Sobel algorithms, demonstrated superior performance over conventional methods, although some noise contamination persisted.

In [12], an approach to identify edges in underwater images was proposed, utilizing fractional order differentiation. The study introduced a texture enhancement filter based on the Grünwald-Letnikov (G-L) fractional differential operator. An analysis was conducted on diverse underwater images using both conventional and fractional differential operators, with results compared to the Riemann-Liouville fractional differential operator technique (R-L). The suggested method demonstrated superior performance compared to traditional and R-L-based approaches, particularly in recognizing edges in low-contrast underwater images. It offered heightened accuracy, improved brightness, and increased information extraction.

In [13], the enhancement of underwater image quality was pursued through the application of edge detection methods and the utilization of the Lab color model. The authors executed edge detection subsequent to color correction and contrast enhancement. Faced with challenges such as light illumination, water velocity, and suspended particles, the preference was given to color detection instead of direct edge recognition, aiming for improved results in underwater image processing. Although their approach successfully identified object shapes, some residual noise remained. To enhance future edge detection outcomes, the authors conveyed their intention to incorporate deep networks for automated operations.

Numerous underwater image enhancement algorithms have been proposed, leveraging texture and color features for feature database creation and improved image attribute description [14-16]. While histogram equalization is a common method for visual cue equalization, its simplicity often results in less appealing enhanced images. Model-based algorithms introduce some complexity, and recent deep neural network-based methods, although offering better quality, demand significant computational resources. The challenge lies in striking a balance between image quality and computational complexity. In this context, we propose an approximate pipeline Gaussian filter for enhanced underwater image improvement, aiming for an optimal trade-off.

2. Hardware Platforms for Image Processing

Multiple hardware platforms are accessible for deploying vision algorithms, including general-purpose computers, digital signal processors (DSPs), graphical processing units (GPUs), and reconfigurable devices like field-programmable gate arrays (FPGAs). In the field of image processing, FPGAs exhibit superior performance in complex computations, while GPUs excel in simpler tasks. Modern FPGAs empower system designers to create high-performance computing applications with significant parallelism [17].

Recent advances in computer vision, particularly in object detection, motion tracking, and semantic segmentation, have heightened interest in digital image processing (DIP) techniques. To address the need for rapid prototyping, low power consumption, and low latency, hardware accelerators, such as custom processors and co-processors, offer an alternative to software implementations. Meeting the demands of real-time image processing requires more processing power than conventional processors can provide. FPGA implementations outperform DSPs

and GPPs for algorithms leveraging substantial parallelism [18, 19]. Traditional DSP arrays, with fixed architectures and relatively short lifespans, can be expensive to program line by line with thousands of lines of code [20]. The memory bandwidth of contemporary FPGAs far exceeds that of GPPs or DSPs, running at two to ten times the speed of the FPGA [21]. Beyond their capability for highly parallel arithmetic architecture, FPGAs are well-suited for tasks such as digital filtering, Fast Fourier Transform, and image processing. FPGAs are a common choice for rapid prototyping, provide a favorable trade-off among design metrics, ensuring a swift time-to-market compared to other integrated circuit (IC) technologies [22]. FPGAs also find applications in control and communication [23]. Developing hardware accelerators in FPGA involves exploring various architectures, considering aspects such as the arithmetic used [24, 25], approximate computing techniques [26], and hardware/software communication architecture [27]. Features and techniques like these play a crucial role in implementing digital image processing algorithms on hardware. In this context, assessing potential implementation models and comparing their impact on system metrics is essential. Such evaluations assist hardware designers in making informed decisions about their designs. In [28], the mix module implementation of underwater image enhancement on FPGA relies on fusion by wavelet decomposition. This method significantly improves the visibility of underwater images, and qualitative results illustrate the enhanced quality of hazy underwater images.

In [29], the focus is on optimizing the preprocessing of optical images from autonomous underwater vehicles in challenging conditions. It highlights the efficiency of FPGAs for correcting image degradation through non-linear filtering, implementing two-dimensional FFT, its inverse, and logarithmic computation. Demonstrating FPGA effectiveness in parallel architectures, the study improves histograms in underwater image preprocessing via frequency-based filtering and introduces a method applicable in digital signal processing tasks.

In [30], a benchmarking analysis compared three Retinex model-based algorithms (SSR, MSR, MSRCR) on five embedded systems—Beagle Board, Odroid-XU4, Raspberry Pi 4, Jetson Nano, and Jetson TX2—for enhancing underwater images. Quality metrics (UIQM, UCIQUE, BRISQUE, Entropy) without a reference image were utilized. MSRCR performed best on Jetson TX2, with a 0.46s processing time difference compared to a high-performance PC. Implementing these algorithms on embedded systems proved cost-effective and held promise for artificial vision-equipped underwater vehicles.

[31] presented a low-cost, high-throughput design of the retinex video enhancement algorithm, renowned for restoring naturalness, especially in dark areas. Historically burdened by computational complexity, the hardware (HW) design was implemented on a field-programmable gate array (FPGA), achieving a throughput of 60 frames/s for a 1920×1080 image with minimal latency. The design optimized HW resources by utilizing a small line buffer, applying approximate computing for the Gaussian filter, and introducing a novel exponentiation operation. This approach significantly reduced HW resources (up to 79.22% of total resources) compared to existing systems while ensuring compatibility with commercial devices through standard HDMI/DVI video ports.

3. Approximate Computing

In error-resilient applications like multimedia, data mining, image processing, and machine learning, exact precision is not always essential [32]. Acceptable results with some accuracy degradation suffice for these applications. This flexibility enables trade-offs between accuracy and electrical performance, allowing for gains in power dissipation, occupied area, and delay by sacrificing a degree of accuracy [33].

Voltage over-scaling is a solution to reduce circuit power dissipation [34]. However, operating a circuit under normal voltage levels may lead to timing-induced failures, causing significant errors in the most significant bits. Approximate computing is an efficient paradigm to lower power consumption and enhance embedded system performance. Allowing errors at the outputs of a complex circuit simplifies logic expressions, reducing logic counts. This approach results in savings in areas, dynamic power dissipation, and shortened circuit delays [35].

In image and video processing, where error tolerance is permissible, adopting approximate computing techniques offers significant improvements in speed and power efficiency, albeit with a trade-off in output quality. The diverse accuracy requirements of various applications, coupled with the dynamic nature of accuracy needs within the same application at different processing stages or over time, underscore the flexibility and adaptability of approximate computing in meeting specific computational demands [36].

4. Gaussian Filter

In image processing, noise filtering is a crucial element designed to eliminate noise and its effects from the original image while striving to minimize distortion. However, these filtering operations typically involve intensive arithmetic operations, contributing to increased energy consumption in the

system. The Gaussian filter (GF) is a convolution operator employed for image blurring and noise removal [37]. The Gaussian filter, crucial for image smoothing, is a fundamental component in the initial stages of noise-sensitive edge detection algorithms like Canny, Sobel, and Laplace. Its primary role is to reduce distortions, ensuring the optimal performance of these algorithms.

Smoothing filters act as low-pass filters by suppressing the high-frequency components of an image. These components include characteristics like object contours, where the frequency increases with the abruptness of contour direction changes [37]. Consequently, smoothing filters work to alleviate these transitions, making them useful for tasks such as reconstructing incomplete contours caused by distortions from low resolution.

Eq.(1) represents a two-dimensional Gaussian function frequently employed in image and signal processing.

$$f(x.y) = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \quad (1)$$

The equation f(x,y) denotes the value of the Gaussian function at the coordinates (x,y). σ is the standard deviation parameter, influencing the width or spread of the Gaussian function. The standard deviation (σ) controls the width of the bell-shaped curve generated by the Gaussian function. Larger values of σ result in a broader curve. This Gaussian function is commonly applied in image processing for blurring and smoothing, emphasizing central values while suppressing those farther from the center

The Gaussian filter possesses the valuable property of separability. This means that filters with a size of n×n can be split into two masks with sizes n×1 and 1×n. This separation enables the convolution to be performed in two operations, promoting temporal parallelism by initiating the second operation before the completion of the first.

4. Proposed Architecture

To develop an efficient image processing algorithm for FPGA stream-based processing, a typical architecture combines row buffers (line buffers) capable of storing one image row and window buffers, 2D arrays storing a local image area required for computing the current output pixel. This setup enables sequential reading of input image/video pixels, optimizing the use of the FPGA's limited memory. Only the minimum pixels necessary for computing the current pixel's filtered value are stored at any given time. The required number of line buffers is estimated as H - 1, with H representing the vertical size of the kernel. The window buffer matches the kernel size (e.g., a 3×3 window buffer for a Gaussian filter with a 3×3 kernel). The overall architecture is illustrated in Figure 1.

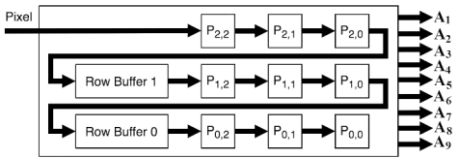


Figure 1. Delay line buffer structure

The delay line buffer optimizes memory access during convolution by storing recently accessed pixel values. In a typical digital image setup, pixels are stored sequentially in a frame buffer. The delay line buffer, using shift registers, retains the last values read, focusing on local storage for pixels still in use within upcoming convolution windows. This approach minimizes the need for continuous memory access. The structure involves nine 1-pixel registers for the current window and two row buffers storing additional pixels from the first two rows in the sliding window. The window buffer must match the size of the kernel.

To observe the filter's impact on the image, the mask must traverse all pixels, determining the brightness intensity of each new pixel based on Eq. (2) as illustrated in Figure 2.

$$h[i.j] = AP_1 + BP_2 + CP_3 + DP_4 + EP_5 + FP_{62} + GP_7 + HP_8 + IP_9 \quad (2)$$

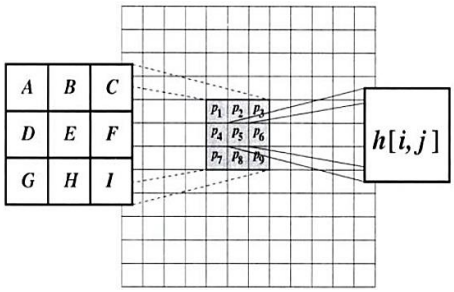


Figure 2. Convolution Operation on an image with a 3x3 kernel

4.1 Pipeline Gaussian Filter (PGF)

In this study, Figure 3 displays the Gaussian filter kernel, while Figure 4 presents the block diagram of the Gaussian filter implementation. The multiplication by 2 and 4 is achieved through left-shifting the pixel value by 1 and 2 bits, respectively. Similarly, division by 16 is performed by right-shifting the desired number by 4 bits. The Gaussian filter architecture consists of eight adders and six shifters for multiplication operations. However, the critical path from input to output includes fout adders and two shifters, leading to a slower calculation speed.

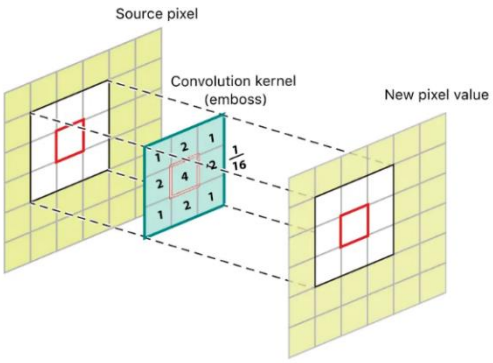


Figure 3. Application of the weighted kernel corresponding to the standard 3×3 Gaussian filter on the image

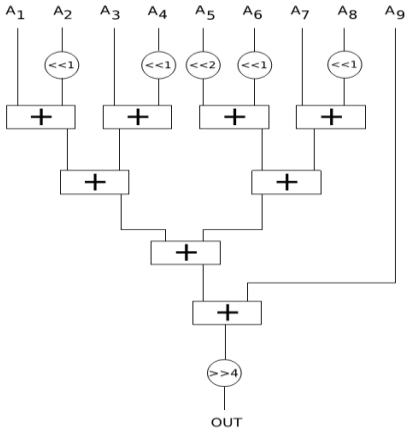


Figure 4. Block diagram of gaussian filter

A pipeline structure enhances processing speed by dividing the computation task into stages, allowing simultaneous execution of different stages. Each stage of the pipeline handles a specific aspect of the computation. As one stage completes its task, the results are seamlessly transferred to the next stage, enabling continuous processing. This concurrent operation minimizes idle time and optimizes resource utilization.

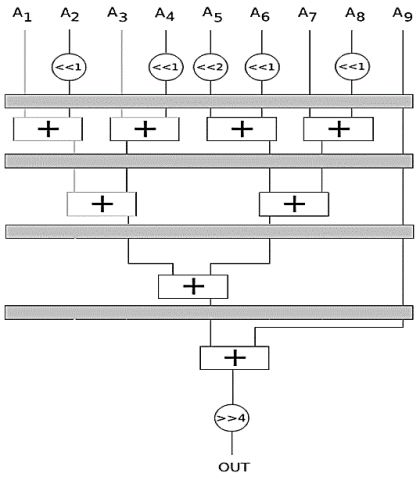


Figure 5. Pipeline Gaussian filter

To improve the speed of the Gaussian filter, a pipeline structure with four stages is implemented, as illustrated in Figure 5. This design approach ensures seamless computation, with each stage playing a role in enhancing the overall processing speed.

4.3 Approximate Adders

To implement various types of approximate Gaussian filters, we employed different approximate adders. Reducing the logic complexity of adders at the bit level offers enhanced power savings compared to conventional low-power design methods. The reduction in logic complexity for a conventional full adder (FA) cell was attained by minimizing the number of transistors. Several simplified versions of the FA were tested based on this logic complexity reduction.

The approximate full adders (APFAs) employed in the design of the Gaussian filter exhibit distinct characteristics. We employ some approximate adders in [38]. APFA1 approximates both Sum and Cout, with Sum being precise for 6 out of 8 cases and Cout for 7 out of 8 cases, excluding specific input combinations. In contrast, APFA2 focuses on approximating Sum alone, ensuring precision for 6 out of 8 cases and Cout for all cases. APFA3 involves approximation in both Sum and Cout, with Sum being precise for 5 out of 8 cases and Cout for 7 out of 8 cases. APFA4 approximates Sum and Cout, achieving precision in Sum for 5 out of 8 cases and Cout for 6 out of 8 cases at the logic level. APFA5 approximates both Sum and Cout, achieving precision in Sum for 4 out of 8 cases and Cout for 6 out of 8 cases, excluding specific input combinations. Meanwhile, APFA6 approximates both Sum and Cout, with Sum being correct for 5 out of 8 cases and Cout for 6 out of 8 cases. APFA7 demonstrates that Sum is correct for 6 out of 8 cases, and Cout is correct for all 8 cases, indicating a lower probability of error compared to APFA6. APFA8 introduces three errors in Sum, with Sum being correct for 5 out of 8 cases. APFA9 shows Sum is correct for 7 out of 8 cases. Lastly, APFA10 introduces three errors in Sum, with Sum being correct for 5 out of 8 cases, and Cout is correct for all 8 cases in APFA10. Table 1 displays the logic equations for different approximation methods. "None" indicates a precise full adder with no approximation applied.

Approximations are introduced to 16-bit adders through a carry ripple structure (Figure 1), specifically focusing on the least significant bits in each FA block. Evaluation reveals that acceptable output quality is sustained up to an 8-bit approximation, with a decline in quality beyond this threshold.

Table 1. Logical functions of approximate full adders

16-Bit FA	Logic function Sum	C_{out}
FA	$A'B'C_{in} + ABC_{in} + AB'C'_{in} + A'BC'_{in}$	$AB + BC_{in} + AC_{in}$
APFA1	$ABC_{in} + C'_{out}C_{in}$	$AC_{in} + B$
APFA2	C'_{out}	$AB + BC_{in} + AC_{in}$
APFA3	C'_{out}	$AC_{in} + B$
APFA4	$ABC_{in} + C'_{out}C_{in}$	A
APFA5	B	A
APFA6	$A' + BC_{in}$	A
APFA7	$A'(B + C_{in}) + BC_{in}$	$AB + BC_{in} + AC_{in}$
APFA8	$(A' + B)C_{in}$	$AB + BC_{in} + AC_{in}$
APFA9	$B'C'_{in} + ABC + A'C'_{in} + A'B'$	$AB + BC_{in} + AC_{in}$
APFA10	$A' + BC_{in}$	$AB + BC_{in} + AC_{in}$

The simulation results in Section 5 illustrate the impact of varying the number of least significant bits (LPL) on which the approximation is applied, shedding light on its influence on the output quality.

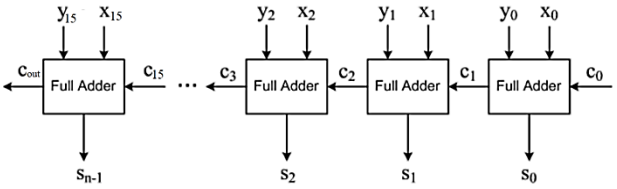


Figure 6. 16-bit carry ripple adder

5. Simulation Results

To evaluate the effect of utilizing an approximate adder on the output image quality, a 16-bit model of the Gaussian filter is designed in the MATLAB environment. Gaussian noise is then introduced to the original image, serving as input for both the exact Gaussian filter and the approximate Gaussian filters.

In the context of image quality assessment, several metrics are commonly employed to quantitatively evaluate the performance of image processing algorithms. The Peak Signal-to-Noise Ratio (PSNR) measures the ratio between the maximum possible power of a signal and the power of corrupting noise, providing insight into the fidelity of the reconstructed image. PSNR is calculated form Eq.(3). Mean Square Error (MSE) quantifies the average squared difference between corresponding pixel values in the original and reconstructed images, offering a measure of the overall image distortion. It is described in Eq.(4). Structural Similarity Index (SSIM) , which is computed using Eq.(5) , evaluates the similarity between two images, considering luminance, contrast, and structure, and providing a comprehensive measure of perceptual quality [39]. μ and σ denote the mean and variance of the image intensities respectively. C_1 and C_2 are constants. ω is Gaussian weighting function that is normalized to unit sum ($\sum_{i=1}^{N} \omega_i = 1$).

Error Distance (ED), Mean Error Distance (MED), and Normalized Error Distance (NED) gauge the spatial discrepancies between corresponding pixels in the original and reconstructed images, offering insights into the accuracy of the reconstructed image at both individual and average levels [40]. the Error Distance (ED) between two points ($x_{original}$, $y_{original}$) and ($x_{reconstructed}$, $y_{reconstructed}$) in an image can also be expressed using the Euclidean distance formula in the Eq.(6). NED is calculated by dividing the Euclidean distance by the diagonal length of the image. MED provides a measure of the average error distance between corresponding points in the original and reconstructed images. These metrics collectively contribute to a comprehensive evaluation of the performance of image processing algorithms.

$$PSNR(f.g) = 20\,log_{10}\left(\frac{2^B-1}{MSE(f.g)}\right) \quad (3)$$

$$MSE(f.g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(f_{ij} - g_{ij}\right)^2 \quad (4)$$

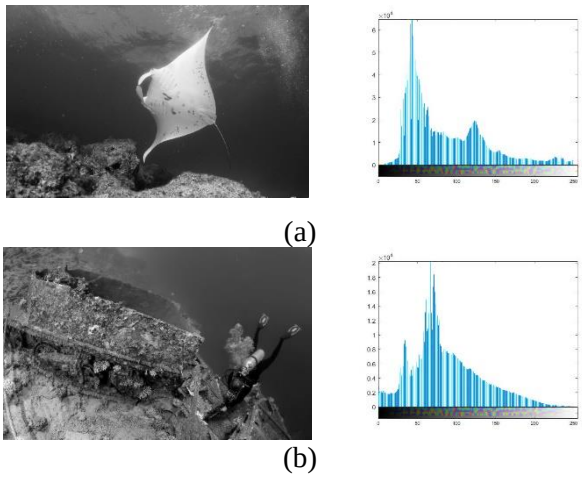
$$SSIM(f.g) = \frac{(2\mu_f\mu_g + C_1)(2\sigma_{fg} + C_2)}{\left(\mu_{f+}^2\mu_{f+}^2 + C_2\right)\left(\sigma_{f+}^2\sigma_{f+}^2 + C_2\right)} \quad (5)$$

$$ED = \sqrt{f^2 - g^2} \quad (6)$$

$$MED = \frac{\sum_{i=1}^{N} ED_i}{N} \quad (7)$$

Where, N represents the total number of pixels in the image, and i denotes the pixel index. The formulas provide quantitative measures for assessing the quality and accuracy of reconstructed images.

We selected two real-world underwater images [41] to simulate genuine underwater conditions. Gaussian noise was introduced to these images. The addition of noise was accomplished using the 'imnoise' function in MATLAB, incorporating a Gaussian distribution with a mean (m) of 0.0005 and a variance (v) of 0.005.



(a)

(b)

Figure 7. Two raw underwater images taken in diverse underwater scenes and their histograms. (a) flatfish, (b) diver [41]

Referring to Figure 8, it is evident that approximate adders up to 5 bits contribute to satisfactory image quality in the Gaussian filter output. However, an additional increase in the LPL causes a sudden decline in output quality, rendering the approximation ineffective. Significantly, the utilization of APFA9

and APFA2 results in the highest output quality, while employing APFA1 leads to the lowest. This observation underscores the critical role of both the approximation method and LPL in determining the balance between approximation and output quality in the Gaussian filter.

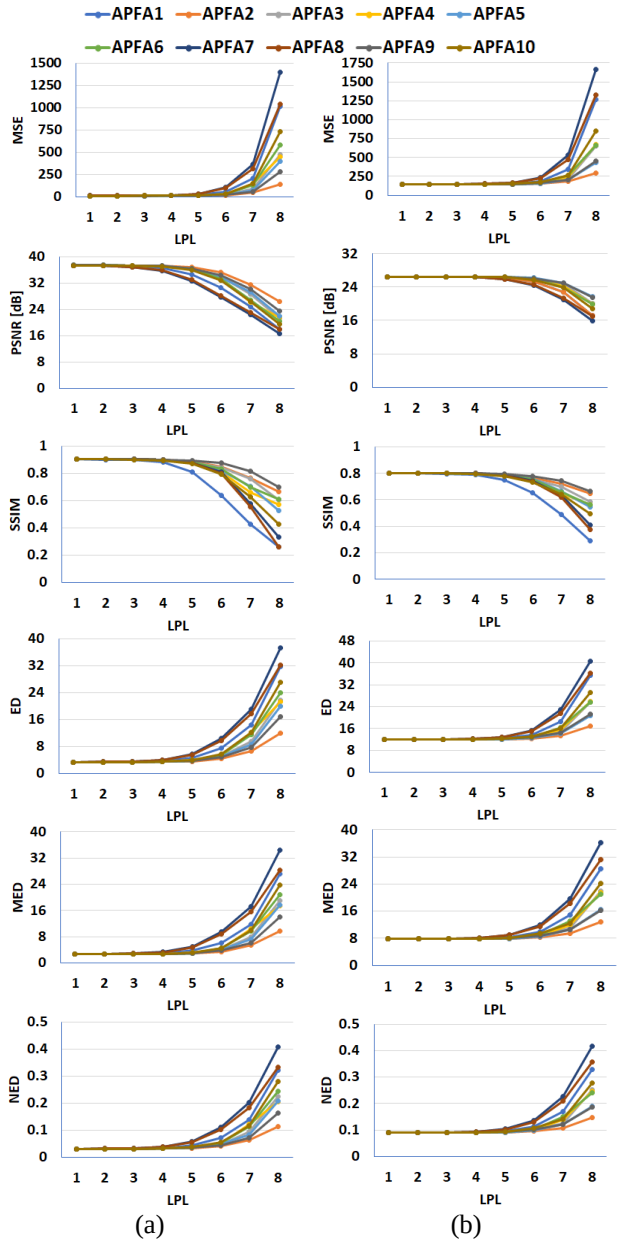


Figure 8. Evaluation Metrics for Image Enhancement of (a) flatfish and (b) diver

Approximate pipeline Gaussian filters (APGF) demonstrate effective performance up to a 5-bit approximation for both images, with the diver image showing satisfactory results up to 6 bits. The success is attributed to the Gaussian filter's role in smoothing and reducing image noise, especially in cases with a wide dynamic range of pixel intensities. When the histogram is concentrated within a narrow intensity range, the filter's impact may be more subtle due to fewer high-frequency components to smooth. Beyond smoothing, the Gaussian filter influences image contrast based on pixel intensity distribution in the histogram. In instances of low contrast, where pixels concentrate around a specific intensity, the filter enhances contrast by smoothing intensity transitions, contributing to overall image quality improvement.

6. Synthesis Results

To assess the physical characteristics of our approximate Gaussian filters, we developed a parameterizable and synthesizable Verilog HDL model for each architecture dedicated to the Gaussian filter. These models underwent validation through simulations using the ModelSim simulator and physical prototyping on the MAX10 device from Intel® FPGA using Quartus. Power consumption was estimated using the Power Analysis tool with a VCD file generated in post-synthesis simulation with 100,000 random inputs. Line buffers, essential for window-based spatial filters (see Figure 1) are excluded from synthesis results as their size depends on the input image size.

As per Table 2, the implementation of the pipeline method has proven to be instrumental in enhancing the efficiency of the Gaussian filter. This improvement is reflected in a notable 48% increase in speed and a commendable 12% reduction in power consumption. However, it comes at the cost of a 27% increase in the required area.

Table 2. Comparison on physical properties of gaussian filter (GF) and pipeline GF (PGF)

	Power (mW)	Delay (ns)	Logic
GF	119.45	9.82	414
PGF	104.96	5.09	529

Table 3 presents the hardware specifications of Gaussian filters implemented with approximate adders. The table highlights the power-efficient characteristics of certain approximate filters. The notation APFAiLj denotes the ith approximation of APFA, where j least significant bits are computed approximately. Filters utilizing approximate adders with the sum bit derived from the output carry bit exhibit slower performance. Notably, APFA5 stands out as the least power-consuming approximate filter, achieving a power reduction of over 20%. Additionally, APFA6, APFA7, and APFA8 significantly enhance the speed of the Gaussian filter by more than 70%.

The Power-Delay Product (PDP) is a metric used to evaluate the power efficiency of a digital circuit. It is calculated by multiplying the power consumption of the circuit by its propagation delay. The power delay product is particularly relevant in digital design because it provides insights into how efficiently a circuit performs in terms of both speed and power consumption. A lower PDP value indicates better

power efficiency because it signifies that the circuit achieves a balance between low power consumption and fast operation. Designers often aim to minimize the power-delay product to enhance the overall performance and energy efficiency of digital systems. Based on the data presented in Figure 9, it is evident that APFA 4, 5, and 6 exhibit the most favorable Power-Delay Product (PDP) values, indicating superior power efficiency and speed performance. On the other hand, APFA2 is identified as having the least favorable PDP among the analyzed approximate adders. This observation underscores the importance of considering the trade-off between power consumption and delay in the selection of approximate adders for specific applications.

Table 3. Comparative Analysis of Physical Properties between Gaussian Filter (GF) and Pipeline Gaussian Filter (PGF)

APGF	Power reduction%	Speed up %	Area reduction %
APFA1L8	4.09	31.19	6.62
APFA2L2	4.33	4.33	21.55
APFA2L3	5.81	5.81	20.04
APFA2L4	8.39	8.39	19.09
APFA2L8	4.92	4.92	13.80
APFA3L5	6.77	38.96	1.51
APFA3L6	7.85	47.05	1.70
APFA3L7	7.45	49.12	2.27
APFA3L8	5.81	62.55	2.27
APFA4L4	7.45	-7.64	-14.74
APFA4L6	5.81	-9.10	-14.93
APFA5L5	4.50	-28.64	-16.82
APFA5L6	10.49	-21.26	-21.36
APFA5L7	15.14	0.47	-25.90
APFA5L8	21.34	-12.81	-30.43
APFA6L4	4.10	-18.51	-10.78
APFA6L7	4.96	-30.26	-10.78
APFA6L8	4.95	-14.93	-10.78
APFA7L1	9.25	50.12	9.07
APFA7L3	5.31	68.02	18.34
APFA7L4	4.47	72.40	17.01
APFA7L5	7.45	70.45	15.88
APFA7L7	7.16	69.88	15.12
APFA8L1	4.06	50.59	26.47
APFA8L4	6.65	61.06	27.22
APFA9L1	4.06	50.59	26.47
APFA9L4	6.65	61.06	27.22
APFA10L6	9.91	59.92	29.30

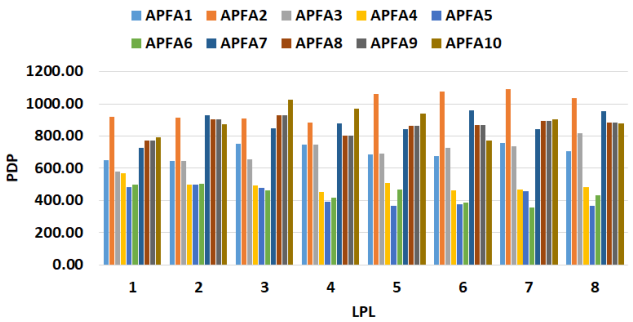


Figure 9. PDP values for different approximate gaussiaan filters

7. Conclusion

In this paper, we propose a novel architecture aimed at enhancing the efficiency of the Gaussian filter for the removal of Gaussian noise in underwater images. The key innovation lies in the utilization of a pipeline structure for the implementation of the Gaussian filter. Additionally, we conduct a comprehensive evaluation of the impact of employing ten approximate adders on the filter's performance.

The simulation results demonstrate that adopting the pipeline structure along with 2 to 8-bit approximation in adders leads to a significant improvement in the speed of the Gaussian filter, exceeding 150%. Moreover, this approach yields a noteworthy enhancement in power consumption, surpassing 34%. However, it is essential to note that these advantages come with an associated increase in the required area. While acknowledging the tradeoff between output quality and power, our design holds particular relevance for error-resilient applications, such as image and video processing. The proposed structure provides a valuable solution for scenarios where the benefits of heightened speed and reduced power consumption outweigh the increase in spatial requirements.

8. References

[1] Fatan, M., Daliri, M. R., & Shahri, A. M. (2016). Underwater cable detection in the images using edge classification based on texture information. Measurement, 91, 309-317.

[2] Saini, A., & Biswas, M. (2019). Object detection in underwater image by detecting edges using adaptive thresholding. 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI),

[3] Princess, P. J. B., Silas, S., & Rajsingh, E. B. (2019). Performance analysis of edge detection algorithms for object detection in accident images. 2019 Global Conference for Advancement in Technology (GCAT),

[4] Oliveira, A. J., Ferreira, B. M., & Cruz, N. A. (2021). A performance analysis of feature extraction algorithms for acoustic image-based underwater navigation. Journal of Marine Science and Engineering, 9(4), 361.

[5] Prasenan, P., & Suriyakala, C. (2022). A Study of Underwater Image Pre-processing and Techniques. In Computational Vision and Bio-Inspired Computing: Proceedings of ICCVBIC 2021 (pp. 313-333). Springer.

[6] Borker, S., & Bonde, S. (2017). Contrast Enhancement and Visibility Restoration of Underwater Optical Images Using Fusion. International Journal of Intelligent Engineering & Systems, 10(4).

[7] Mishra, A., Gupta, M., & Sharma, P. (2018). Enhancement of underwater images using improved CLAHE. 2018 International Conference on Advanced Computation and Telecommunication (ICACAT),

[8] Yang, S., Chen, Z., Feng, Z., & Ma, X. (2019). Underwater image enhancement using scene depth-based adaptive background light estimation and dark channel prior algorithms. IEEE Access, 7, 165318-165327.

[9] Bhairannawar, S. S. (2018). Efficient medical image enhancement technique using transform HSV space and adaptive histogram equalization. In Soft Computing Based Medical Image Analysis (pp. 51-60). Elsevier.

[10] Ulutas, G., & Ustubioglu, B. (2021). Underwater image enhancement using contrast limited adaptive histogram equalization and layered difference representation. Multimedia Tools and Applications, 80, 15067-15091.

[11] Priyadharsini, R., Sharmila, T. S., & Rajendran, V. (2016). An efficient edge detection technique using filtering and morphological operations for underwater acoustic images. Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies,

[12] Shourya, S., Kumar, S., & Jha, R. K. (2016). Adaptive fractional differential approach to enhance underwater images. 2016 Sixth International Symposium on Embedded Computing and System Design (ISED),

[13] Singh, N., & Bhat, A. (2023). A robust model for improving the quality of underwater images using enhancement techniques. Multimedia Tools and Applications, 1-22.

[14] Xiao, B., Wang, K., Bi, X., Li, W., & Han, J. (2018). 2D-LBP: an enhanced local binary feature for texture image classification. IEEE Transactions on Circuits and Systems for Video Technology, 29(9), 2796-2808.

[15] Ponomarev, A., Nalamwar, H. S., Babakov, I., Parkhi, C. S., & Buddhawar, G. (2016). Content-based image retrieval using color, texture and shape features. Key Engineering Materials, 685, 872-876.

[16] Humeau-Heurtier, A. (2019). Texture feature extraction methods: A survey. IEEE Access, 7, 8975-9000.

[17] Qasaimeh, M., Denolf, K., Lo, J., Vissers, K., Zambreno, J., & Jones, P. H. (2019). Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels. 2019 IEEE international conference on embedded software and systems (ICESS),

[18] Gupta, U., Babu, M., Ayoub, R., Kishinevsky, M., Paterna, F., Gumussoy, S., & Ogras, U. Y. (2018). An online learning methodology for performance modeling of graphics processors. IEEE Transactions on Computers, 67(12), 1677-1691.

[19] HajiRassouliha, A., Taberner, A. J., Nash, M. P., & Nielsen, P. M. (2018). Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms. Signal Processing: Image Communication, 68, 101-119.

[20] Diouri, O., Gaga, A., Ouanan, H., Senhaji, S., Faquir, S., & Jamil, M. O. (2022). Comparison study of hardware architectures performance between FPGA and DSP processors for implementing digital signal processing algorithms: Application of FIR digital filter. Results in Engineering, 16, 100639.

[21] De Haro, J. M., Bosch, J., Filgueras, A., Vidal, M., Jiménez-González, D., Alvarez, C., Martorell, X., Ayguadé, E., & Labarta, J. (2021). OmpSs@ FPGA framework for high performance FPGA computing. IEEE Transactions on Computers, 70(12), 2029-2042.

[22] Pirzada, S. J. H., Murtaza, A., Xu, T., & Jianwei, L. (2020). A reconfigurable model-based design for rapid prototyping on FPGA. International Journal of Computer Theory and Engineering, 12(3), 80-84.

[23] Chamola, V., Patra, S., Kumar, N., & Guizani, M. (2020). Fpga for 5g: Re-configurable hardware for next generation communication. IEEE Wireless Communications, 27(3), 140-147.

[24] Cabello, F., León, J., Iano, Y., & Arthur, R. (2015). Implementation of a fixed-point 2D Gaussian Filter for Image Processing based on FPGA. 2015 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA),

[25] Murray, K. E., Luu, J., Walker, M. J., McCullough, C., Wang, S., Huda, S., Yan, B., Chiasson, C., Kent, K. B., & Anderson, J. (2020). Optimizing FPGA logic block architectures for arithmetic. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28(6), 1378-1391.

[26] Nomani, T., Mohsin, M., Pervaiz, Z., & Shafique, M. (2020). xUAVs: Towards efficient approximate computing for UAVs—Low power approximate adders with single LUT delay for FPGA-based aerial imaging optimization. IEEE Access, 8, 102982-102996.

[27] Zhu, Z., Zhang, J., Zhao, J., Cao, J., Zhao, D., Jia, G., & Meng, Q. (2019). A hardware and software task-scheduling framework based on CPU+ FPGA heterogeneous architecture in edge computing. IEEE Access, 7, 148975-148988

[28] Guraksin, G. E., Deperlioglu, O., & Kose, U. (2019). A novel underwater image enhancement approach with wavelet transform supported by differential evolution algorithm. Nature Inspired Optimization Techniques for Image Processing Applications, 255-278.

[29] S. M. Alex Raj, M. H. S. (2015). FPGA Implementation of Underwater Image Enhancement using Nonlinear Filtering. Indian Journal of Science and Technology, 8(35),1-5. https://doi.org/10.17485/ijst/2015/v8i35/79110

[30] Aguirre-Castro, O., García-Guerrero, E., López-Bonilla, O., Tlelo-Cuautle, E., López-Mancilla, D., Cárdenas-Valdez, J., Olguín-Tiznado, J., & Inzunza-González, E. (2022). Evaluation of underwater image enhancement algorithms based on Retinex and its implementation on embedded systems. Neurocomputing, 494, 148-159.

[31] Park, J. W., Lee, H., Kim, B., Kang, D.-G., Jin, S. O., Kim, H., & Lee, H.-J. (2019). A low-cost and high-throughput FPGA implementation of the retinex algorithm for real-time video enhancement. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28(1), 101-114.

[32] Padhy, A. P., & Das, B. P. (2023). Lightweight Approximate Multiplier with Improved Accuracy in FPGA for Error Resilient Application. 2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID),

[33] Ullah, S., & Kumar, A. (2023). Approximate Arithmetic Circuit Architectures for FPGA-based Systems. Springer Nature.

[34] Amrouch, H., Ehsani, S. B., Gerstlauer, A., & Henkel, J. (2019). On the efficiency of voltage overscaling under temperature and aging effects. IEEE Transactions on Computers, 68(11), 1647-1662.

[35] Bahoo, A. A., Akbari, O., & Shafique, M. (2023). An Energy-Efficient Generic Accuracy Configurable

Multiplier Based on Block-Level Voltage Overscaling. IEEE Transactions on Emerging Topics in Computing.

[36] Jiang, H., Santiago, F. J. H., Mo, H., Liu, L., & Han, J. (2020). Approximate arithmetic circuits: A survey, characterization, and recent applications. Proceedings of the IEEE, 108(12), 2108-2135.

[37] Mafi, M., Martin, H., Cabrerizo, M., Andrian, J., Barreto, A., & Adjouadi, M. (2019). A comprehensive survey on impulse and Gaussian denoising filters for digital images. Signal Processing, 157, 236-260.

[38] Anusha, G., & Deepa, P. (2020). Design of approximate adders and multipliers for error tolerant image processing. Microprocessors and Microsystems, 72, 102940.

[39] Assessment, I. Q. (2004). From error visibility to structural similarity. IEEE transactions on image processing, 13(4), 93.

[40] Liang, J., Han, J., & Lombardi, F. (2012). New metrics for the reliability of approximate and probabilistic adders. IEEE Transactions on Computers, 62(9), 1760-1771.

[41] Li, C., Guo, C., Ren, W., Cong, R., Hou, J., Kwong, S., & Tao, D. (2019). An underwater image enhancement benchmark dataset and beyond. IEEE transactions on image processing, 29, 4376-4389.