Summary of Research Paper

Underwater Image Enhancement Using FPGA-Based Gaussian Filters with Approximation Techniques
Summarized on August 2, 2025

1. **Introduction / Abstract**
   The paper addresses the problem of poor visual quality in underwater images caused by natural phenomena like light absorption and scattering, which create haziness and noise. To improve image clarity, the study proposes an efficient implementation of Gaussian filters (a type of image smoothing filter) on **FPGA** (Field-Programmable Gate Array, a reconfigurable hardware device) platforms using pipeline structures and approximate adders (simplified arithmetic units that trade some accuracy for better speed and power efficiency). The main finding is that this approach significantly speeds up processing (by over 150%) and reduces power consumption (by more than 34%), though it requires more hardware area. This tradeoff is suitable for error-tolerant applications such as underwater image and video processing.

2. **Methodology**
   The researchers designed a pipeline architecture for the Gaussian filter on an FPGA, which breaks down the filtering process into stages that operate simultaneously to increase speed. They incorporated approximate adders that simplify addition operations by allowing minor errors, reducing power and delay. The design uses line buffers and window buffers to efficiently handle image data in a streaming fashion, minimizing memory access. They tested ten different approximate adder designs, applying approximations to the least significant bits of 16-bit adders, and evaluated their impact on image quality and hardware performance. Simulations were conducted in MATLAB with real underwater images corrupted by Gaussian noise, and hardware synthesis was performed on an Intel MAX10 FPGA.

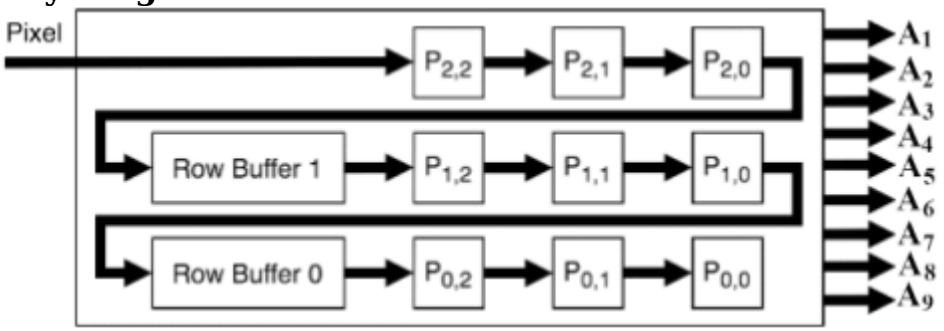
3. **Theory / Mathematics**
   The Gaussian filter is based on the two-dimensional Gaussian function:
   [f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}]
   where (f(x,y)) is the filter value at coordinates ((x,y)), and (\sigma) is the standard deviation controlling the spread of the filter. This function is used to blur images and reduce noise by weighting pixels near the center more heavily. The filter is separable, meaning a 2D convolution can be done as two 1D convolutions, improving efficiency.

Image quality was quantitatively assessed using:
- **Peak Signal-to-Noise Ratio (PSNR):**
[PSNR(f,g) = 20 \log_{10} \left(\frac{2^B - 1}{\sqrt{MSE(f,g)}}\right)]
where (B) is bit depth, and (MSE) is the mean squared error between original image (f) and filtered image (g).
- **Mean Squared Error (MSE):**

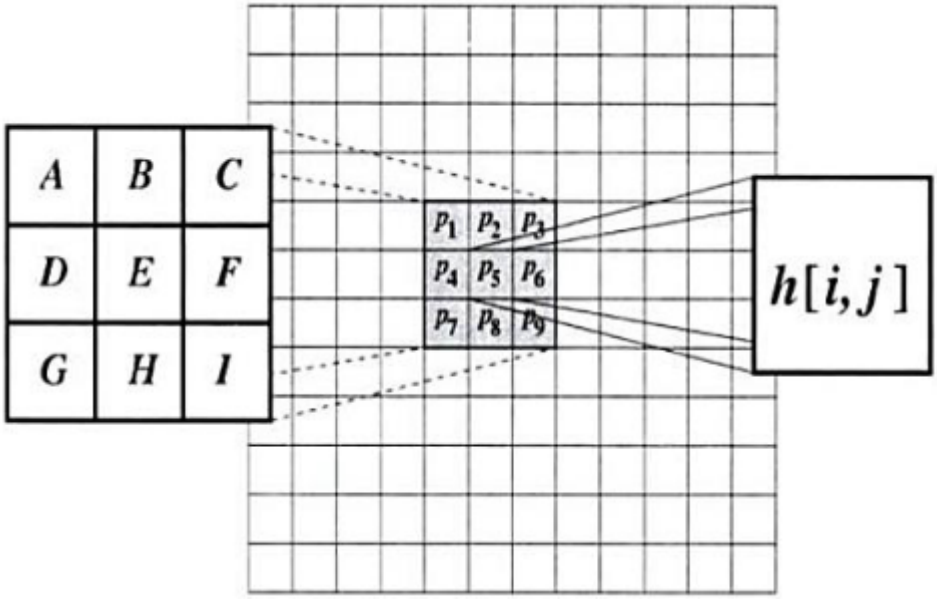
[MSE(f,g) = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (f_{ij} - g_{ij})^2]
- **Structural Similarity Index (SSIM):** A measure considering luminance, contrast, and structure to evaluate perceptual similarity.
- **Error Distance (ED), Mean Error Distance (MED), Normalized Error Distance (NED):** Metrics measuring spatial discrepancies between original and processed images.

These metrics help balance the tradeoff between approximation-induced errors and image quality.

1. **Key Diagrams or Visual Elements**

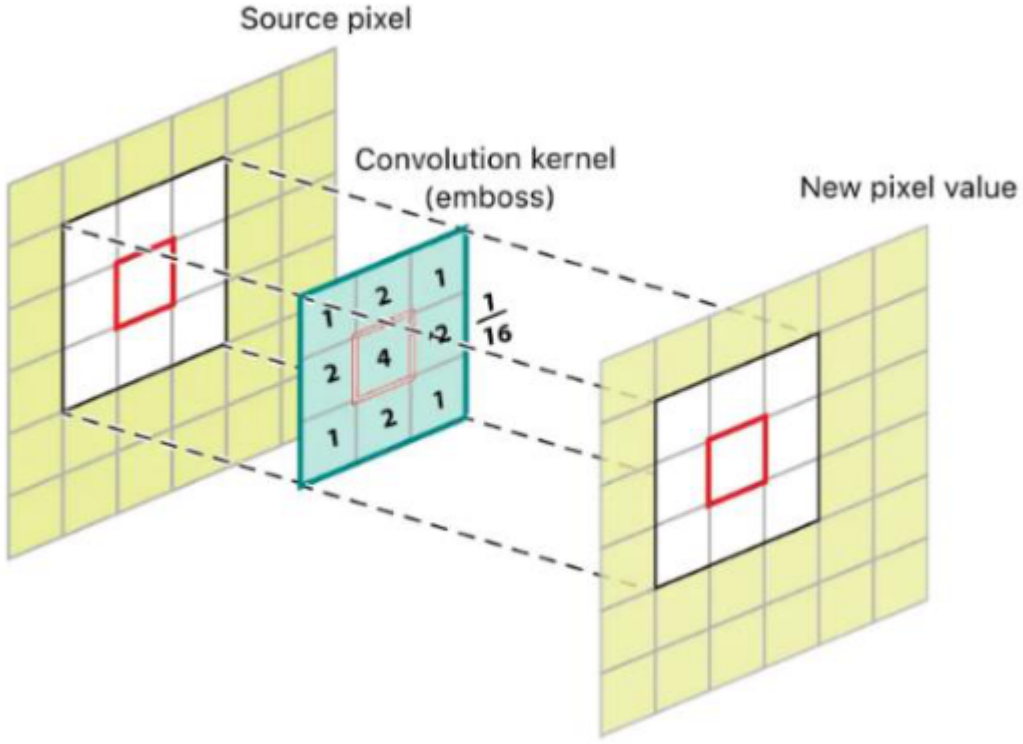


2. **Figure 1:** Shows the delay line buffer structure, which stores recent pixel values to optimize memory access during convolution. This reduces the need to repeatedly access large image memory.

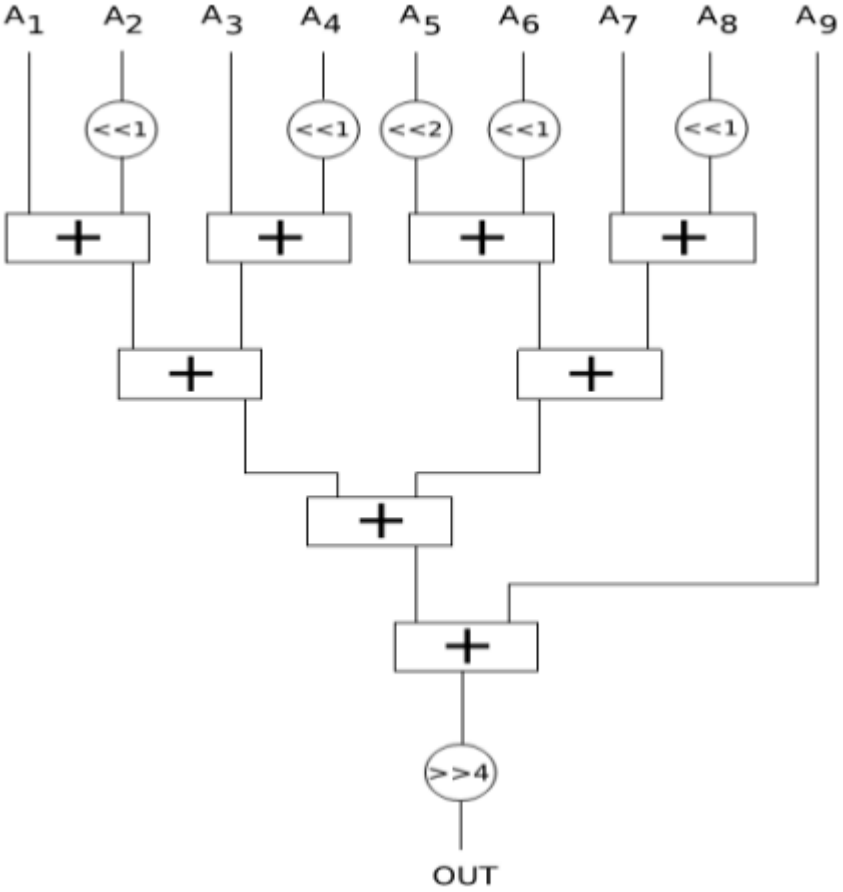


3. **Figure 2:** Illustrates the convolution operation of a 3×3 Gaussian kernel over an image, showing how pixel values are weighted and

summed.

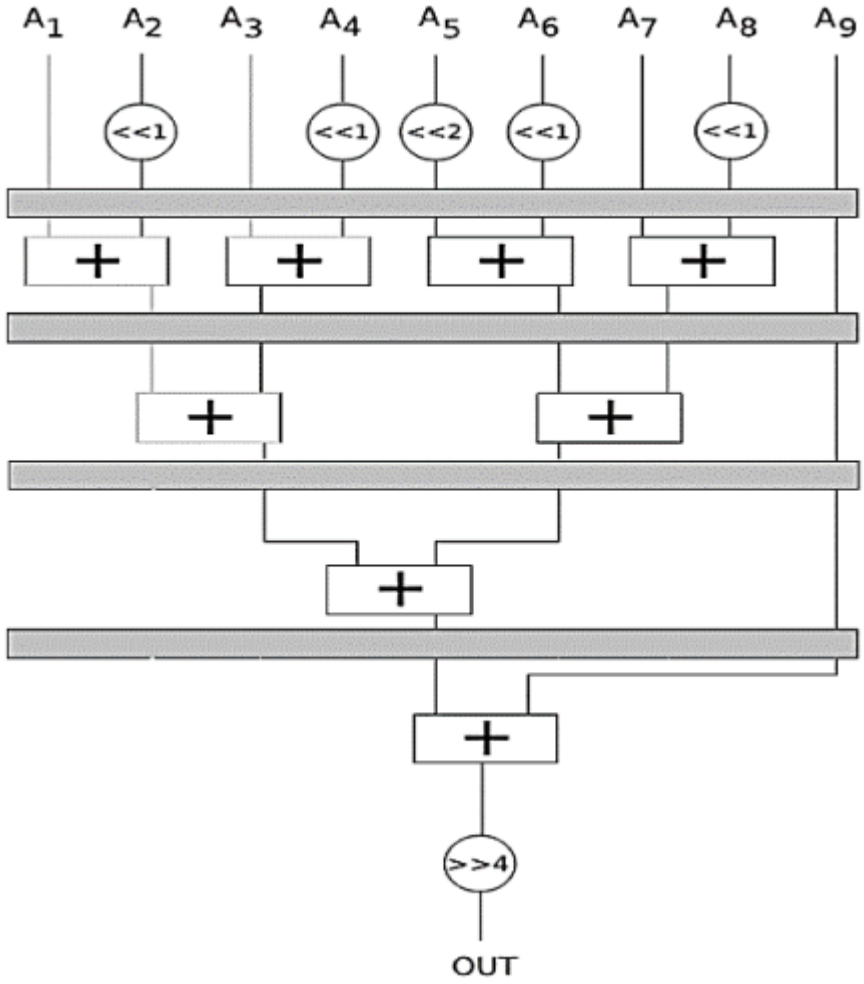


4. **Figure 3:** Displays the weighted 3×3 Gaussian kernel used for filtering.

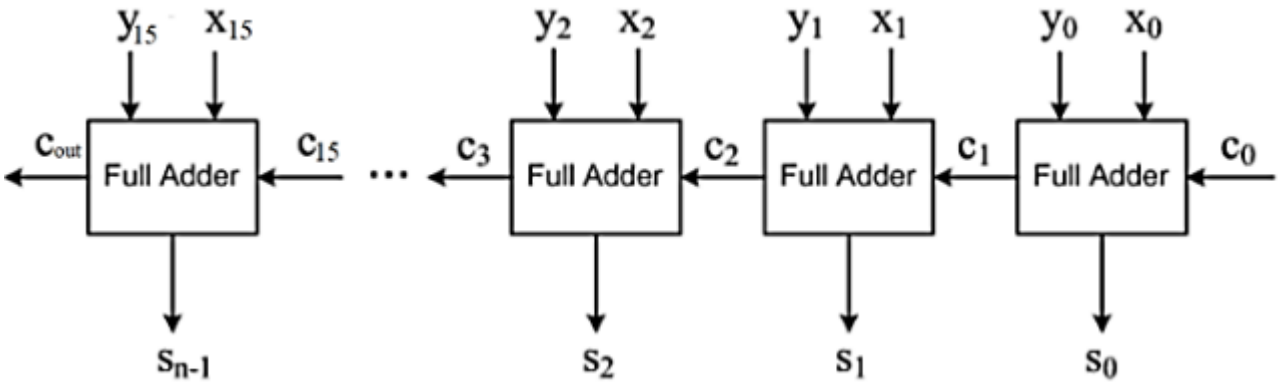


5. **Figure 4:** Block diagram of the Gaussian filter implementation, highlighting the use of adders and shifters for multiplication/division by

powers of two.

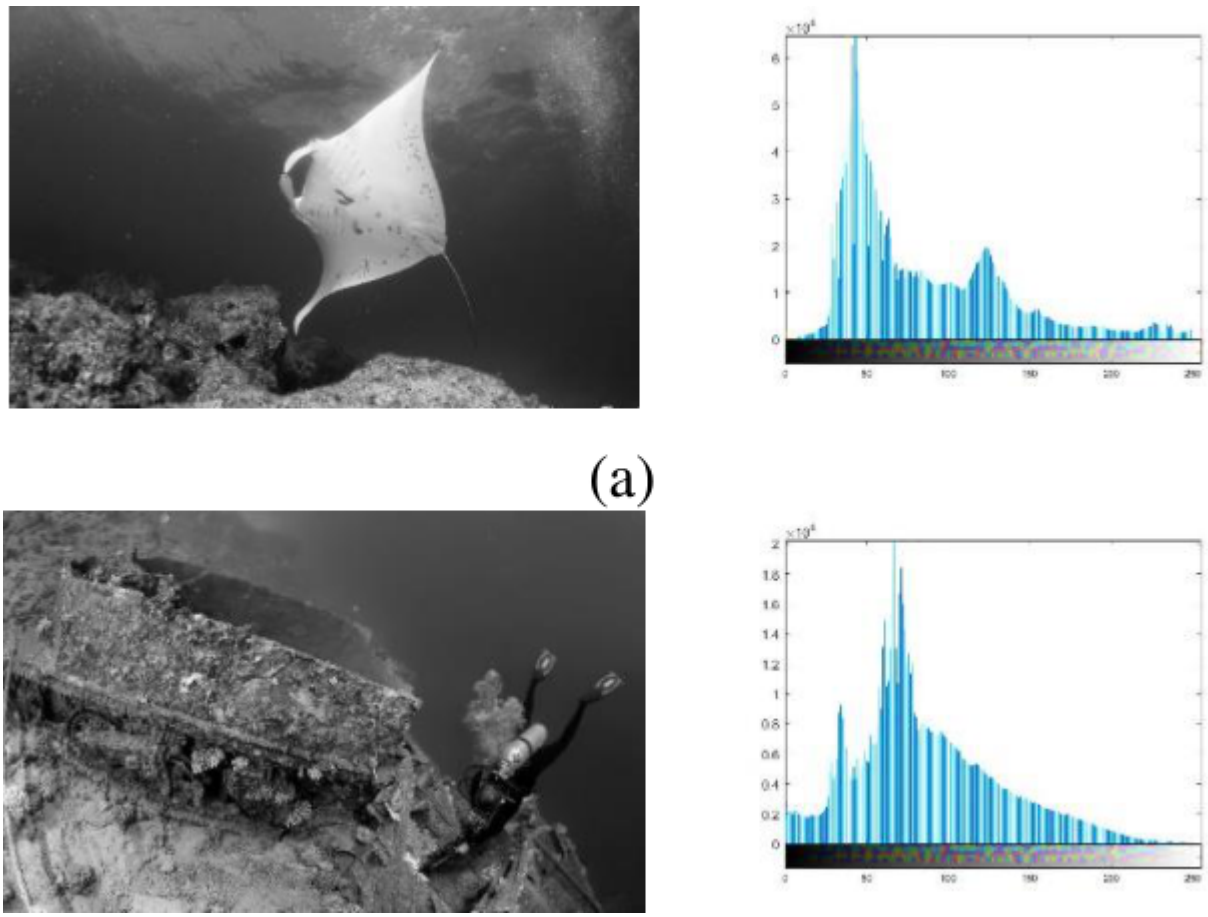


6. **Figure 5:** Depicts the pipeline structure of the Gaussian filter, dividing computation into four stages to increase throughput.



7. **Figure 6:** Shows the 16-bit carry ripple adder architecture used for addition in the filter, where approximate adders replace precise ones in

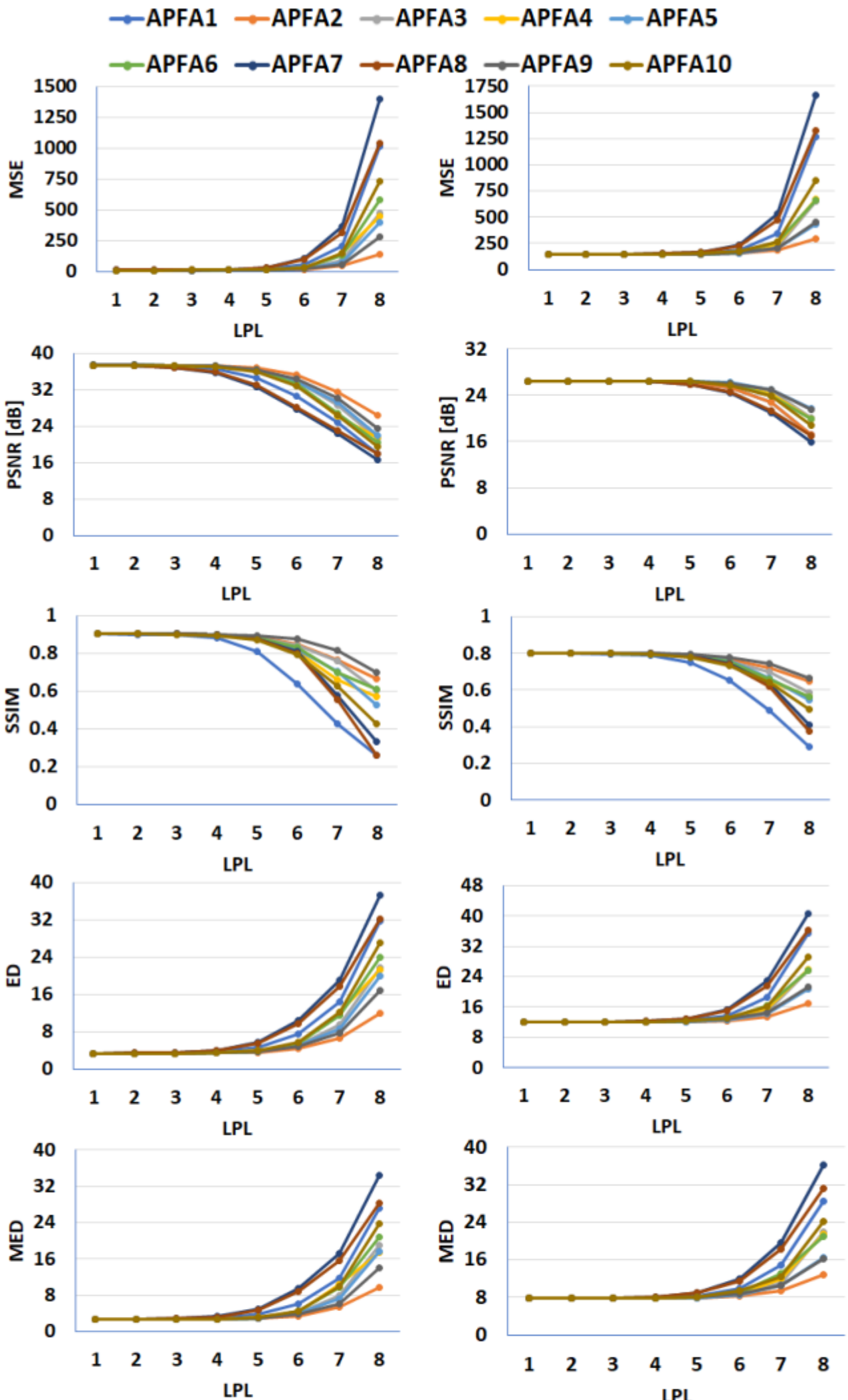
some bits.



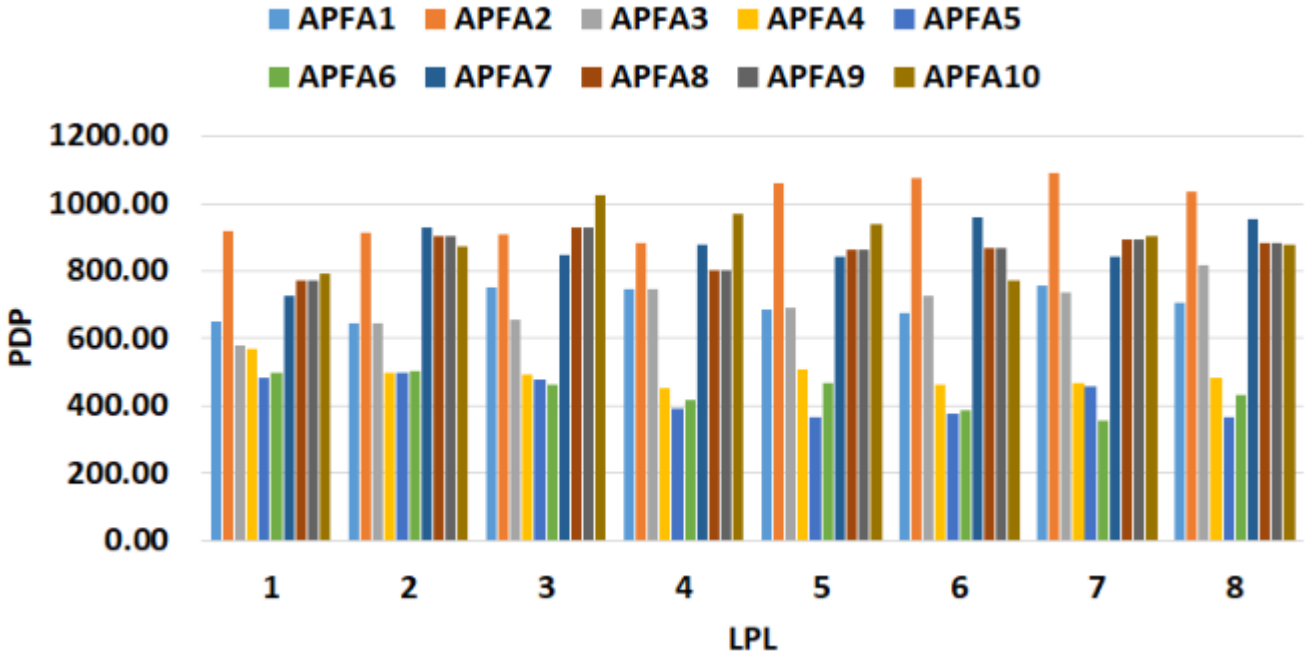
(a)

8. **Figure 7:** Presents two sample raw underwater images used for testing, along with their histograms showing pixel intensity

distributions.


APFA1
APFA2
APFA3
APFA4
APFA5
APFA6
APFA7
APFA8
APFA9
APFA10
MSE
PSNR [dB]
SSIM
ED
MED
LPL

9. **Figure 8:** Graphs evaluation metrics (PSNR, SSIM, etc.) for different approximate adders and bit approximations, demonstrating how image quality varies with approximation level.



10. **Figure 9:** Compares the Power-Delay Product (PDP) of different approximate adders, indicating their efficiency tradeoffs.

Tables summarize the logic functions of approximate adders and the synthesis results showing power, speed, and area tradeoffs.

1. **Conclusion**
   The study successfully demonstrates that implementing a pipeline Gaussian filter with approximate adders on FPGA significantly improves processing speed (over 150%) and reduces power consumption (over 34%) for underwater image enhancement. The tradeoff is an increased hardware area and some loss in output quality, which remains acceptable up to about 5-6 bits of approximation. This makes the approach well-suited for error-resilient applications like underwater video and image processing, where faster and more power-efficient hardware is critical. The research advances underwater imaging technology by providing a practical hardware solution that balances quality, speed, and energy use, enabling better real-time analysis of underwater scenes for marine ecology and resource management.