Underwater Image Enhancement Using FPGA-Based Gaussian Filters with Approximation Techniques
Summary Date: August 5, 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 these images, the study proposes an efficient method using **Gaussian filters** (a type of image smoothing filter) implemented on **FPGA** (Field-Programmable Gate Array, a reconfigurable hardware device) with approximation techniques to speed up processing and reduce power consumption. The main finding is that using a pipeline structure combined with approximate adders in the Gaussian filter design significantly increases processing speed by over 150% and reduces power use by more than 34%, though it requires more hardware area. This tradeoff is suitable for error-tolerant applications like 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 introduced approximate adders—simplified arithmetic units that trade some accuracy for lower power and faster operation—into the filter design. Various types of approximate adders were tested, each differing in how much they simplify the addition operation and how this affects output quality. The filter was tested on real underwater images with added Gaussian noise, and performance was evaluated using image quality metrics. The design was synthesized and implemented on an Intel MAX10 FPGA to measure power, speed, and area.

3. **Theory / Mathematics**
   The Gaussian filter uses a two-dimensional Gaussian function to smooth images and reduce noise. The function is given by:

$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

   where (x, y) are coordinates in the image and σ controls the spread of the smoothing effect. Larger σ means more blurring. The filter is applied via convolution, which combines the Gaussian kernel with the image pixels to produce a smoothed output. The convolution for a 3×3 kernel is:

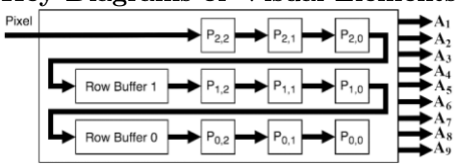
$$h[i,j] = AP_1 + BP_2 + CP_3 + DP_4 + EP_5 + FP_6 + GP_7 + HP_8 + IP_9$$

   where P_1 to P_9 are pixel values in the window and A to I are kernel weights. The pipeline structure divides this computation into stages to

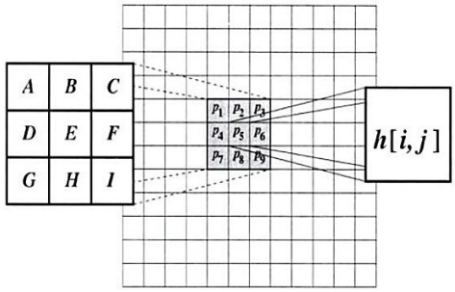
improve throughput. Approximate adders simplify the addition logic by allowing some errors in less significant bits, reducing power and delay.

Image quality was assessed using:
- **PSNR (Peak Signal-to-Noise Ratio):** Measures how close the filtered image is to the original, higher is better.
- **MSE (Mean Square Error):** Average squared difference between original and filtered images, lower is better.
- **SSIM (Structural Similarity Index):** Measures perceptual similarity considering luminance and contrast, higher is better.
- **Error Distance (ED), Mean Error Distance (MED), Normalized Error Distance (NED):** Quantify spatial differences between images.

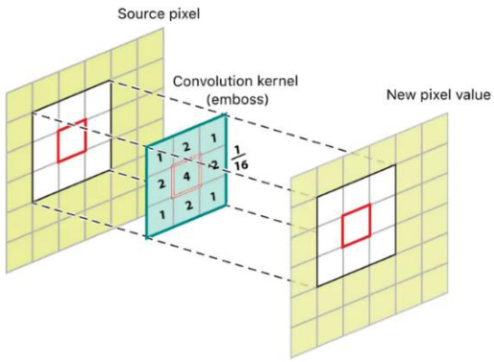
4. **Key Diagrams or Visual Elements**



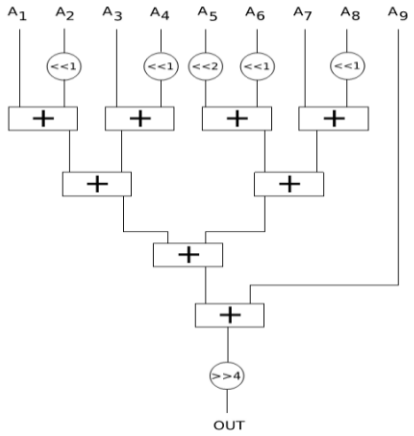
- **Figure 1:** Shows the delay line buffer structure used to store image rows and local pixel windows, optimizing memory access during convolution.



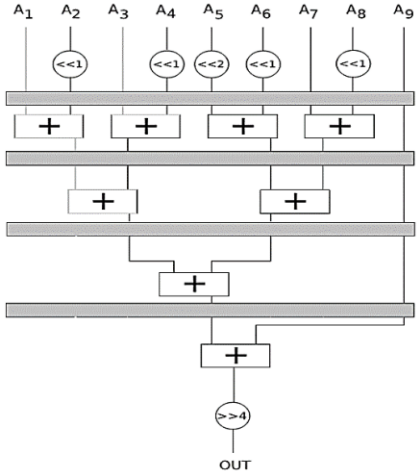
- **Figure 2:** Illustrates the convolution operation of a 3×3 Gaussian kernel sliding over an image, explaining how each output pixel is computed.



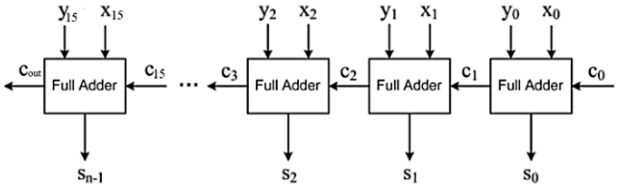
- **Figure 3:** Displays the weighted Gaussian kernel used in the filter.



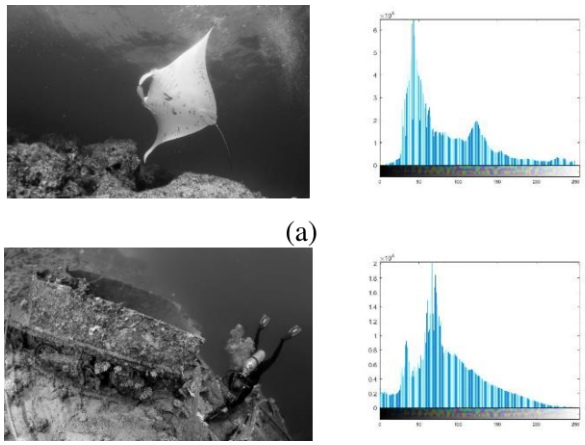
- **Figure 4:** Block diagram of the Gaussian filter implementation, showing adders and shifters used for multiplication and division by powers of two.



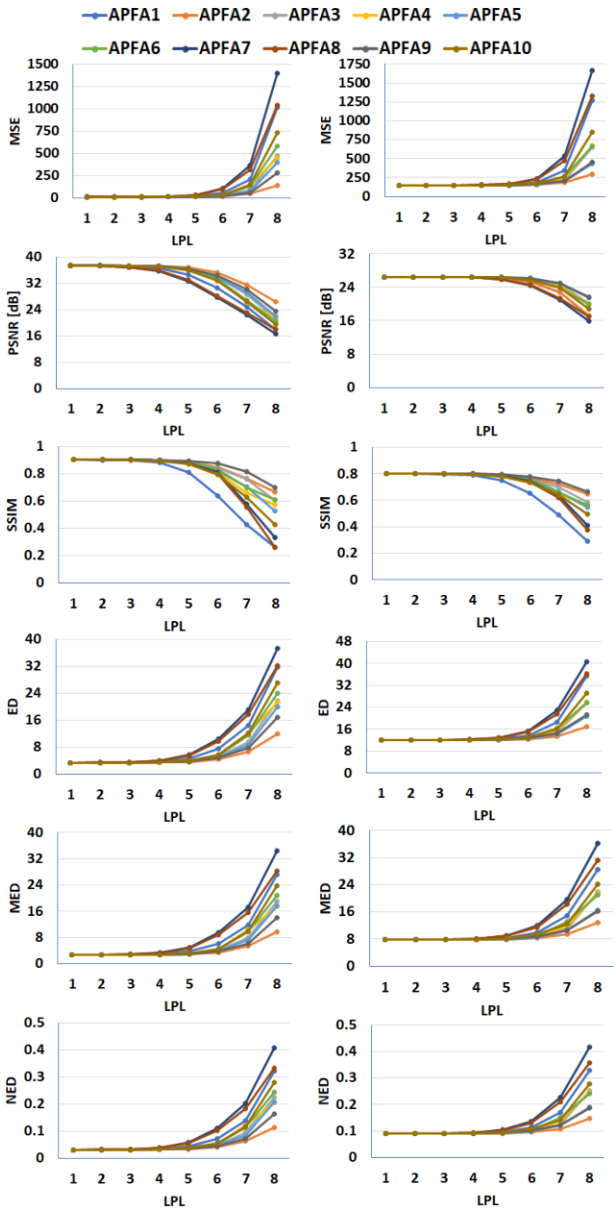
- **Figure 5:** Pipeline Gaussian filter architecture with four stages, enabling simultaneous processing steps to increase speed.



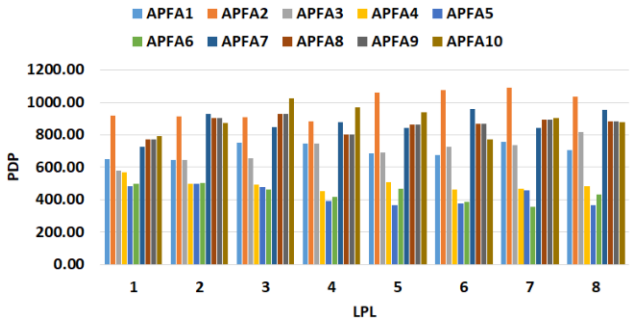
- **Figure 6:** Diagram of a 16-bit carry ripple adder used in the design, highlighting where approximation is applied.



- **Figure 7:** Two sample underwater images (a flatfish and a diver) used for testing, along with their histograms showing pixel intensity distributions.



- **Figure 8:** Graphs showing image quality metrics (PSNR, SSIM, etc.) for different approximate adders and levels of approximation, indicating that up to 5-bit approximation maintains good quality.



- **Figure 9:** Power-Delay Product (PDP) comparison for different approximate adders, identifying which designs offer the best tradeoff between speed and power.

5. **Conclusion**

The study successfully demonstrates that implementing a pipeline Gaussian filter with approximate adders on FPGA can greatly enhance processing speed (over 150% faster) and reduce power consumption (by more than 34%) for underwater image enhancement. The tradeoff is an increased hardware area requirement and some loss in output quality, which remains acceptable for error-resilient applications like image and video processing. This approach offers a practical solution for real-time underwater imaging systems where speed and power efficiency are critical, such as in marine exploration and monitoring. The results, supported by simulation and FPGA synthesis data, highlight the potential of approximate computing combined with hardware acceleration to advance underwater image enhancement technology.