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 effects like light absorption and scattering, which create haziness and noise. To improve these images, the study proposes an efficient method using a **Gaussian filter** (a mathematical tool for smoothing images and reducing noise) implemented on an **FPGA** (Field-Programmable Gate Array, a type of reconfigurable hardware that allows fast and parallel processing). The key innovation is a pipeline architecture combined with approximate adders (simplified arithmetic units that trade some accuracy for better speed and lower power use). The main findings show that this approach speeds up processing by over 150% and reduces power consumption by more than 34%, though it requires more hardware area. This trade-off is suitable for error-tolerant applications like image and video processing.

2. **Methodology**
   The researchers designed a pipeline Gaussian filter architecture on an FPGA, which processes image pixels in stages to increase speed. They used line buffers to store image rows and window buffers to hold local pixel areas for filtering, minimizing memory use. The Gaussian filter applies a weighted convolution mask (a 3×3 kernel) to smooth images and reduce noise. To further improve efficiency, they replaced exact adders with various types of approximate adders that simplify calculations by allowing small errors. They tested ten different approximate adder designs, focusing on approximating the least significant bits to balance quality and performance. The system was simulated in MATLAB with real underwater images corrupted by Gaussian noise, and then synthesized on an Intel MAX10 FPGA to measure power, speed, and area.

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) (standard deviation) controls the spread of the smoothing effect. This function creates a bell-shaped curve that emphasizes central pixels and reduces the influence of distant pixels, effectively blurring noise while preserving image structure.

The convolution operation for filtering is expressed as:
\

$h[i,j$

= A P_1 + B P_2 + C P_3 + D P_4 + E P_5 + F P_6 + G P_7 + H P_8 + I P_9]
where (P_1) to (P_9) are pixel values in the 3×3 window, and (A) to (I) are the corresponding Gaussian weights.

For image quality evaluation, the paper uses:
- 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 Mean Square Error between original (f) and processed (g) images.
- Mean Square 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), which measures perceptual similarity considering luminance, contrast, and structure.

- Error Distance (ED) and Normalized Error Distance (NED) quantify spatial differences between images.

- **Key Diagrams or Visual Elements**
  !

*Figure*1

(output_images/figure_1.png) - **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.
!

*Figure*2

(output_images/figure_2.png) - **Figure 2:** Illustrates the convolution operation with a 3×3 Gaussian kernel sliding over the image pixels, showing how each output pixel is computed as a weighted sum of neighboring pixels.
!

*Figure*3

(output_images/figure_3.png) - **Figure 3:** Displays the Gaussian filter kernel weights used in the 3×3 mask.
!

*Figure*4

(output_images/figure_4.png) - **Figure 4:** Block diagram of the Gaussian filter implementation, highlighting the use of adders and shifters for multiplication and division by powers of two (implemented as bit shifts for efficiency).
!

*Figure*5

(output_images/figure_5.png) - **Figure 5:** Depicts the pipeline structure of the Gaussian filter, dividing the computation into four stages to allow simultaneous processing and increase throughput.
- **Table 1:** Lists the logical functions of various approximate full adders (APFAs), showing how sum and carry outputs are simplified to reduce hardware complexity.
!

*Figure*6

(output_images/figure_6.png) - **Figure 6:** Shows the 16-bit carry ripple adder structure used to implement approximate addition by focusing approximation on least significant bits.
!

*Figure*7

(output_images/figure_7.png) - **Figure 7:** Presents two sample raw underwater images used for testing, illustrating typical underwater conditions with haziness and color distortion.
!

*Figure*8

(output_images/figure_8.png) - **Figure 8:** Graphs evaluation metrics (PSNR, SSIM, etc.) for different approximate adders and bit approximations,

showing that up to 5-bit approximation maintains good image quality.
- **Table 2:** Compares physical properties (power, delay, logic area) of the standard Gaussian filter and the pipeline Gaussian filter, showing speed and power improvements with some area increase.
- **Table 3:** Summarizes power reduction, speed-up, and area reduction for Gaussian filters using different approximate adders, highlighting trade-offs.
!

*Figure*9

(output_images/figure_9.png) - **Figure 9:** Displays Power-Delay Product (PDP) values for approximate filters, indicating which designs offer the best balance of power efficiency and speed.

1. **Conclusion**
   The study successfully demonstrates that implementing a pipeline Gaussian filter on FPGA with approximate adders significantly improves processing speed (over 150%) and reduces power consumption (over 34%) for underwater image enhancement. Although this comes with increased hardware area, the trade-off is acceptable for error-resilient applications like image and video processing where some loss in precision is tolerable. This approach offers a practical solution for real-time underwater imaging systems, enabling better visual quality with efficient hardware use.

Why It Matters:
Improving underwater image quality is vital for marine research, resource management, and underwater vehicle navigation. This research provides a hardware-efficient method to enhance images in real time, supporting better monitoring and exploration of underwater environments while saving energy and increasing processing speed.