Summary of Research Paper

Underwater Image Enhancement Using FPGA-Based Gaussian Filters with Approximation Techniques
Summary Date: 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 authors propose an efficient method using a **Gaussian filter** (a mathematical tool for smoothing images and reducing noise) implemented on a **Field-Programmable Gate Array (FPGA)** (a type of reconfigurable hardware device). 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 platform. The pipeline divides the filtering process into stages that operate simultaneously, increasing speed. They used line buffers and window buffers to efficiently handle image data in real time. To further improve efficiency, they incorporated various types of approximate adders that simplify addition operations by allowing minor errors, reducing logic complexity, power, and delay. They tested ten different approximate adder designs, applying approximations mainly to the least significant bits of 16-bit adders. The system was simulated in MATLAB with real underwater images corrupted by Gaussian noise, and then synthesized on an Intel MAX10 FPGA for physical evaluation.

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 filter. This function creates a bell-shaped curve used to blur images and reduce noise by emphasizing central pixels and suppressing distant ones. The filter is separable, meaning a 2D filter can be applied as two 1D filters sequentially, improving computational efficiency.

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 a 3×3 window, and (A) to (I) are corresponding weights from the Gaussian kernel.

For image quality assessment, 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 image (f) and processed image (g).
- 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, minimizing repeated memory reads.
!

*Figure*2

(output_images/figure_2.png) - **Figure 2:** Illustrates the convolution operation on an image using a 3×3 Gaussian kernel, showing how pixel

values are weighted and summed.
!

*Figure*3

(output_images/figure_3.png) - **Figure 3:** Displays the weighted Gaussian kernel used for filtering.
!

*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.
!

*Figure*5

(output_images/figure_5.png) - **Figure 5:** Depicts the pipeline Gaussian filter architecture divided into four stages to improve processing speed.
!

*Figure*6

(output_images/figure_6.png) - **Figure 6:** Shows the 16-bit carry ripple adder structure used for addition in the filter, where approximate adders are applied to least significant bits.
!

*Figure*7

(output_images/figure_7.png) - **Figure 7:** Presents two sample raw underwater images (a flatfish and a diver) used for testing, along with their histograms showing pixel intensity distributions.
!

*Figure*8

(output_images/figure_8.png) - **Figure 8:** Graphs evaluation metrics (PSNR, SSIM, etc.) for image enhancement using different approximate adders, demonstrating that up to 5-bit approximation maintains good quality.
!

*Figure*9

(output_images/figure_9.png) - **Figure 9:** Compares Power-Delay Product (PDP) values for different approximate Gaussian filters, identifying the most power-efficient designs.

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. Although this comes with increased hardware area, the trade-off is justified in error-resilient applications like image and video processing where slight accuracy loss is acceptable. This work advances underwater imaging by providing a practical, efficient hardware solution to improve image quality in challenging marine environments, enabling better monitoring and analysis of underwater scenes. The approach balances computational efficiency and image quality, making it valuable for real-time underwater vision systems.