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

Summarized on: October 26, 2023

1. Introduction / Abstract

What is the core problem the paper addresses? Underwater photos and videos are often blurry, hazy, and have a blue or green tint. This is caused by how light scatters and gets absorbed in water, which makes it difficult to see details clearly. This poor image quality is a major problem for scientists studying marine life, companies inspecting underwater equipment, and autonomous underwater vehicles trying to navigate.

What is its proposed solution and main finding? The researchers developed a new, highly efficient method to clean up these underwater images. Their solution uses a special type of customizable computer chip called an **FPGA** (**Field-Programmable Gate Array**; a flexible chip that can be rewired for a specific task, making it very fast for repetitive jobs like image processing). They implemented a common image-smoothing technique called a **Gaussian filter** but made two key changes to it: they structured the calculation like an assembly line (a "pipeline") and used "approximate" math circuits that trade a tiny, often unnoticeable, amount of accuracy for huge gains in speed and power efficiency.

The main finding is that their new method is remarkably effective, achieving a speed increase of over 150% and a power consumption reduction of over 34% compared to standard approaches.

2. Methodology

How did the researchers conduct their work?

The researchers designed and tested their new image enhancement method through a combination of software simulation and hardware modeling. Here are the key steps:

1. **Hardware Platform Selection:** They chose to build their design for an **FPGA**. Unlike a standard computer processor (CPU), an FPGA can be programmed at the hardware level to perform a specific task, like filtering an image, with extreme efficiency and parallelism (doing many things at once).
2. **Core Filtering Technique:** They used a **Gaussian filter**, a standard algorithm that reduces image noise by blurring it slightly. This filter works by performing a **convolution** (**a mathematical operation where a small grid of numbers, called a kernel, is slid across an image**

to calculate new pixel values based on a weighted average of neighboring pixels).

3. **Architectural Innovation (Pipelining):** Instead of having one large circuit perform the entire filter calculation for a pixel, they broke the process down into four smaller, sequential stages. This "pipeline" works like a factory assembly line: as one stage finishes its task, it passes the result to the next, allowing the filter to work on multiple pixels simultaneously. This dramatically increases the overall processing speed.
4. **Efficiency Innovation (Approximate Computing):** To further boost speed and reduce power, they replaced the standard, perfectly accurate addition circuits with "approximate adders." These are simplified circuits that are much faster and use less energy but can sometimes make tiny errors. For image processing, where the human eye won't notice a slight change in a pixel's color value, this is an excellent trade-off. They tested 10 different types of these approximate adders to see which performed best.
5. **Evaluation:** They tested their designs by feeding them noisy underwater images and measuring the results using both image quality metrics (to see how good the output looked) and hardware performance metrics (to measure speed, power use, and chip area).

3. Theory / Mathematics

The research is based on the **Gaussian filter**, which uses the following mathematical concepts.

First, the filter's weights are determined by the two-dimensional Gaussian function:

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

What it means: This formula creates a 2D "bell curve." When used to create a filter kernel, it ensures that the pixel at the very center has the most influence on the final result, while pixels further away have progressively less influence. This creates a smooth, natural-looking blur that is very effective at removing random noise. The symbol (sigma) controls the "width" of the bell curve, which translates to the amount of blurring.

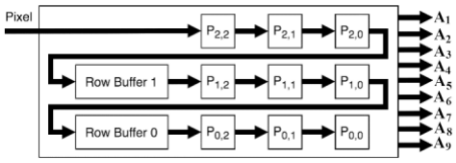
Next, the filter is applied to the image using the convolution operation, which can be simplified for a 3x3 kernel as:

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

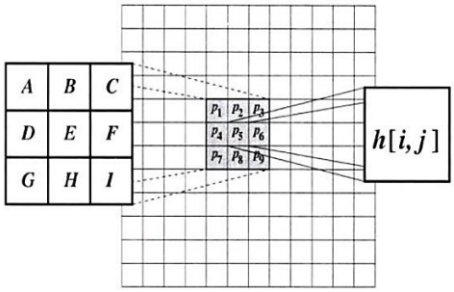
What it means: This equation shows how the new brightness value for a single output pixel (`h[i,j]`) is calculated. The values `P1` through `P9` represent the brightness of the original pixel and its eight immediate neighbors. The values `A` through `I` are the weights from the Gaussian kernel (calculated using the formula

above). Each pixel in the 3x3 window is multiplied by its corresponding weight, and all the results are added together to get the new, smoothed pixel value. This process is repeated for every pixel in the image.

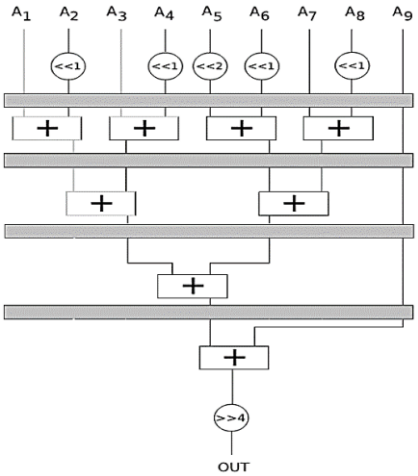
4. Key Diagrams or Visual Elements



- **Figure 1:** This diagram shows the "delay line buffer" structure. This is an efficient memory layout that stores just the few rows of image pixels needed for the filter to work. It prevents the system from having to constantly access the main memory, which saves time and power.

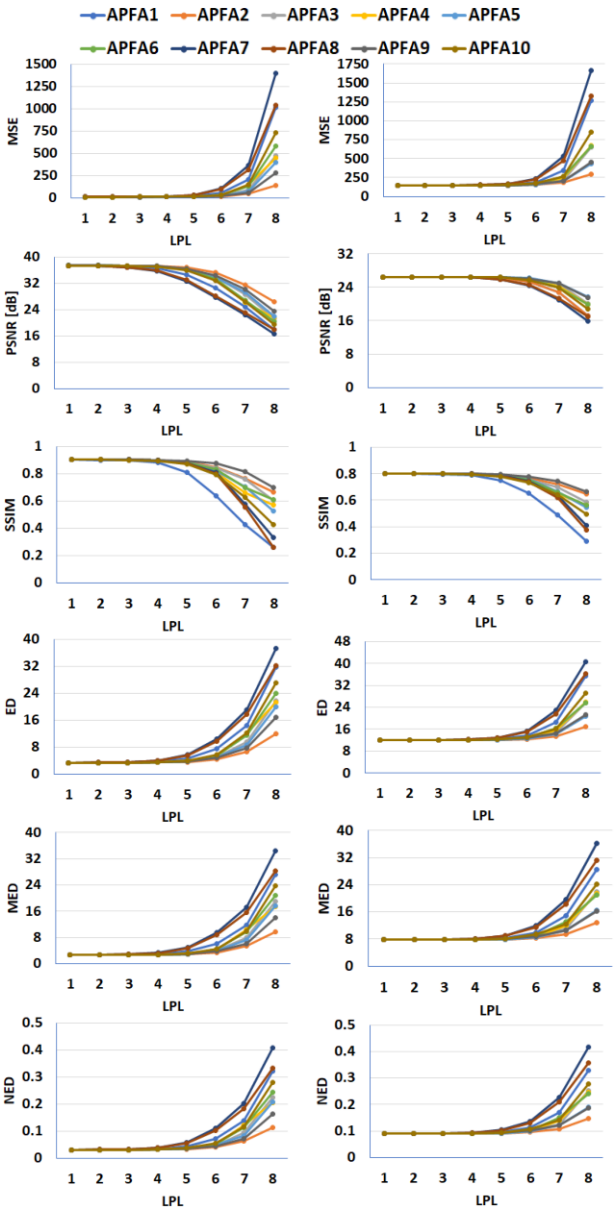


- **Figure 2:** This visual perfectly illustrates the **convolution** operation. It shows a 3x3 "kernel" (a grid of weights) sliding over the image. To calculate the new value for the center pixel, the values in the kernel are multiplied by the image pixel values directly underneath them, and all nine products are summed up.



- **Figure 5:** This diagram shows the researchers' "Pipeline Gaussian Filter." It visualizes their key architectural improvement, breaking the complex calculation into four simpler stages that run

one after another. This assembly-line approach is what allows their design to be so much faster than a traditional, non-pipelined filter (shown in Figure 4).



- **Figure 8:** These graphs display the critical trade-off between performance and quality. They plot image quality metrics (like PSNR and SSIM) against the level of approximation used in the adders. The graphs clearly show that image quality remains very high up to a certain point (using 5-6 "approximate" bits) and then suddenly drops. This helps designers find the "sweet spot" that gives the best performance boost

with minimal loss in visual quality.

5. Conclusion

The key result of this study is a novel hardware design for a **Gaussian filter** that is exceptionally fast and power-efficient. By combining a pipelined architecture with approximate computing techniques on an **FPGA**, the researchers created a system that enhances underwater images with over a 150% increase in speed and a 34% reduction in power consumption.

The main takeaway is that for "error-resilient" tasks like image processing, intentionally sacrificing a tiny amount of mathematical precision can lead to massive improvements in real-world performance. The results, shown in hardware performance tables (Tables 2 and 3) and image quality graphs (Figure 8), prove that this trade-off is not only possible but highly beneficial.

Why It Matters: This research provides a practical blueprint for building low-cost, high-speed, and energy-efficient video processing systems. This is especially critical for battery-powered autonomous underwater drones, which need to process images in real-time to navigate and explore murky environments where every bit of power is precious.