

Optimization of Filtering Algorithms and Video Signal Peak Search on Programmable Logic Integrated Circuits (FPGA) for Laser Cutting Tasks

A.A. Molotkov^{1,A}, Yu.A. Salkov^{2,B}, O.N. Tretiyakova^{3,A}, D.N. Tuzhilin^{4,B}

^A Moscow Aviation Institute (National Research University) MAI, Moscow, 125993, Russia

^B LLC "PROMIS LAB" of the group of companies "Lasers and Equipment TM",
Moscow, Russia

¹ ORCID: 0000-0002-9335-5219, karacerr@gmail.com

² ORCID: 0009-0002-4928-3171, salkov@laser-app.ru

³ ORCID: 0000-0003-0256-4558, tretiyakova_olga@mail.ru

⁴ ORCID: 0000-0002-8570-1732, tuzhilin@laserapr.ru

Abstract

In the process of laser cutting, it is crucial to accurately monitor the distance between the surface of the workpiece and the focusing lens of the laser beam. To achieve this, an optical gap-tracking system (OGTS) was developed. The operation of this optical setup requires high-frequency detection of the peak generated by the laser beam on camera frames.

For high-speed video processing, the Zynq-7000 system-on-chip (SoC) from Xilinx was employed. To determine the peak position within a frame, filtering algorithms and a center-of-mass peak detection method were developed. These algorithms were adapted for FPGA implementation.

This article extends previous research on the design and deployment of computer vision algorithms on FPGA. The paper presents implementation results as well as the performance of the algorithms under various parameter settings.

Keywords: Zynq, computer vision, video processing, FPGA, image filtering, moving average, HLS.

Introduction

This article continues a series of studies [1–2] focused on the development and implementation of computer vision algorithms on field-programmable gate arrays (FPGA). Earlier publications primarily addressed the general architecture of video signal processing systems and the principles of hardware design. The present study emphasizes a deeper mathematical analysis of the employed filters and algorithms, an evaluation of their efficiency in terms of hardware resource utilization, and a quantitative assessment of filtering quality.

Special attention is given to the implementation of a modified moving average filter and threshold filtering designed for video stream processing in the optical gap-tracking system (OGTS) used in laser cutting. The algorithms were developed using high-level synthesis (Vitis HLS) and adapted for execution on the Xilinx Zynq-7000 SoC platform.

To evaluate filtering quality, the study employs the PSNR metric, calculated under different window sizes. PSNR (Peak Signal-to-Noise Ratio) is widely applied in digital image and video processing due to its simplicity, interpretability, and direct comparability with a reference signal. Unlike more complex metrics such as SSIM (Structural Similarity), VMAF (Video Multi-Method Assessment Fusion), or VIF (Visual Information Fidelity), PSNR enables rapid estimation of noise suppression efficiency. The choice of this metric is motivated by the fact

that the goal of filtering is not high-fidelity visual reproduction, but rather robust detection of a distinct signal peak.

A number of studies on FPGA-based video processing [3-5, 12-16] demonstrate effective pipelines for real-time applications: from minimalist “camera–FPGA–display” designs using cost-effective CMOS interfaces to implementations of classical filters (e.g., Sobel) and smoothing filters optimized for chip resources (including coefficient approximations by powers of two to reduce DSP-block usage). These approaches confirm the relevance of FPGAs as platforms for high-frequency video analytics. However, most of them focus either on general visualization/edge detection frameworks or on architectural optimizations of filtering, without directly addressing the specific task of fast and stable peak detection in the narrow-band images of the laser processing zone.

Related Work

Studies [1–2] describe the specific features of implementing the considered computer vision algorithms on FPGAs. The present article continues the line of research on the development and deployment of computer vision algorithms for FPGA-based platforms.

In [3], a hardware-efficient real-time video processing system is presented, employing an FPGA with an OV7670 camera interface and VGA output. The system, implemented on Spartan-6 and Artix-7 FPGAs using Xilinx Vivado and Verilog HDL, is aimed at optimizing hardware resource utilization while maintaining real-time video processing capability. This work describes the acquisition and processing of video streams from a camera.

The study in [4] focuses on the design of a real-time video processing system on an FPGA using an edge-detection algorithm based on the Sobel filter. The system integrates a CMOS camera, a MicroBlaze processor, and a DVI display. Implemented on a Spartan-3A DSP FPGA, this project employs System Generator and the Embedded Development Kit (EDK) to optimize image processing performance in embedded environments. The article emphasizes video streaming for boundary detection, whose principles may be applied in our work.

In [5], the main focus is the development of a Gaussian smoothing filter optimized for FPGA implementation through the use of power-of-two coefficients. The study proposes an algorithm for approximating Gaussian filter coefficients that reduces hardware complexity while preserving noise suppression efficiency. Implemented on a Xilinx Artix-7 FPGA, the system enables effective real-time filtering, improving the signal-to-noise ratio with minimal computational overhead.

These works provide valuable insights into FPGA-based real-time video processing, addressing aspects such as Gaussian smoothing, efficient hardware utilization, and optimized filtering methods. However, in designing a filtering algorithm for the OGTS, it was necessary to take into account specific hardware constraints and stringent performance requirements. For this reason, many of the filter implementation approaches described in the aforementioned studies are not directly applicable to our development.

Hardware Configuration and Development Tools

The implementation of the OGTS was carried out using the AMD Zynq-7000 SoC ZC702 development board from Xilinx. This board represents a system-on-chip (SoC) platform that integrates both programmable logic and a processing subsystem.

During the development of the OGTS controller, the high-level synthesis (HLS) tool Vitis HLS was employed. High-level synthesis tools allow the creation of individual functional modules by writing code in high-level programming languages (such as C++ or Python) instead of hardware description languages (such as Verilog or VHDL). These tools enable developers of functional modules to abstract away from the register-transfer level (RTL) and focus on the level of data processing, thereby significantly reducing development time and complexity.

Controller Operation Algorithm

The controller input consists of a monochrome video stream in the CameraLink format, with each pixel encoded using 8 bits.

The output data, after video processing, represent the position and width of the detected peak within the frame, expressed in pixels.

The algorithm of the controller operation includes the following stages:

1. Video stream acquisition and conversion from the CameraLink format to the AXI-Stream format.
2. Preprocessing of the video stream using a modified moving average filter:

$$y_{ij} = \frac{1}{2N} \sum_{m=0}^1 \sum_{n=-N}^0 x_{i+m, j+n}$$

3. Filtering of the video stream using a threshold filter:

$$y_i = \begin{cases} 1, & \text{if } x_i \geq T \\ 0, & \text{if } x_i < T \end{cases}$$

4. Peak detection within the frame by computing the center of mass according to the defined formula:

$$y_{ij} = \frac{\sum_{i=k}^{k+n} (r_i * m_i)}{\sum_{i=k}^{k+n} (m_i)}$$

After determining the center of mass of the peak on the frame, the controller generates control signals for the servo driver and several auxiliary modules. The following sections of the paper describe in detail the implementation of the filtering and peak detection algorithms on the FPGA.

Development of the filtering algorithm

As mentioned above, the OGTS controller must rapidly receive and process the incoming video stream. Before performing peak detection on a frame, the incoming video must be filtered. For this purpose, a modified moving average algorithm in combination with threshold filtering was employed. These algorithms were chosen because the peak in the camera frame is distinctly pronounced, and the primary objective is to suppress high-amplitude noise to ensure accurate determination of the peak's center of mass.

The difference equation for the modified moving average filter can be expressed as follows:

$$y_{ij} = \frac{1}{2N} \sum_{m=0}^1 \sum_{n=-N}^0 x_{i+m, j+n}, \quad (1)$$

where N denotes the window size.

The difference equation for the threshold filter is defined as:

$$y_i = \begin{cases} 1, & \text{if } x_i \geq T \\ 0, & \text{if } x_i < T \end{cases} \quad (2)$$

where T represents the threshold value.

The mathematical models of these algorithms are relatively simple; however, when implementing them on an FPGA, several specific considerations must be addressed. In particular, it is essential to maximize parallelism in processing in order to achieve optimal performance and throughput within the hardware constraints of the system.

Algorithmic complexity and resource utilization

When implementing algorithms using the Vitis HLS tool, several parameters collectively define the temporal complexity of the synthesized design. Parameters such as latency, iteration latency, and interval determine the overall time performance of the algorithm. All these parameters are measured in clock cycles, since the same algorithm can operate at different clock frequencies depending on the target FPGA configuration.

FPGA resource utilization is estimated during the synthesis stage. Each FPGA device provides a finite number of hardware resources that can be allocated for algorithm implementation. Table 1 summarizes the key timing parameters and the estimated FPGA resource utilization obtained after synthesis.

Table 1 – Efficiency and Resource Utilization Assessment

Modules and loops	Latency (cycles)	Iteration latency	Interval	Pipe-lined	BRAM	DSP	FF	LUT
VideoProcessingSystem	4175	-	4176	no	2 (~0%)	6 (2%)	5846 (5%)	7399 (13%)
Pipeline	4143	-	4143	no	-	6 (2%)	4747 (4%)	4532 (8%)
Loop	4141	47	1	yes	-	-	-	-

The latency parameter indicates the overall temporal complexity of the algorithm in clock cycles, i.e., it shows how many cycles are required to process the input data. The iteration latency and interval parameters indicate the delay in cycles between algorithm iterations, which is necessary in some cases because, during implementation, Vitis HLS sequentially divides the written code into a certain number of finite-state machine states. Each state performs specific data operations. Thus, it can be simplified by saying that Vitis HLS uses a finite-state machine to translate high-level code into an RTL representation.

The main FPGA resource parameter is the number of LUTs (Look-Up Tables), which indicates how large a project can be implemented on a specific FPGA model. The FF parameter indicates how many flip-flops will be used. The DSP parameter indicates how many DSP blocks are used on the FPGA; these blocks are hardware-implemented functions such as multiplication or division of two floating-point numbers, etc. These blocks help reduce overall FPGA resource usage by providing ready-made implementations of some commonly used functions.

Based on the data presented in the table, it can be concluded that the algorithm implementation was successful. FPGA resources are used sparingly, and the temporal complexity of the algorithm is acceptable.

During implementation, the Vitis HLS tool provides detailed timing complexity reports, where the total temporal complexity of the algorithm can be observed after synthesis. When implementing the filtering algorithms, a delay of 4175 clock cycles was achieved. When running the algorithm at a frequency of 100 MHz, the processing delay was approximately 42 microseconds. It should be noted that the temporal complexity of the algorithm does not depend on the window size; this is due to the implementation characteristics described earlier.

Results of the filtering algorithms

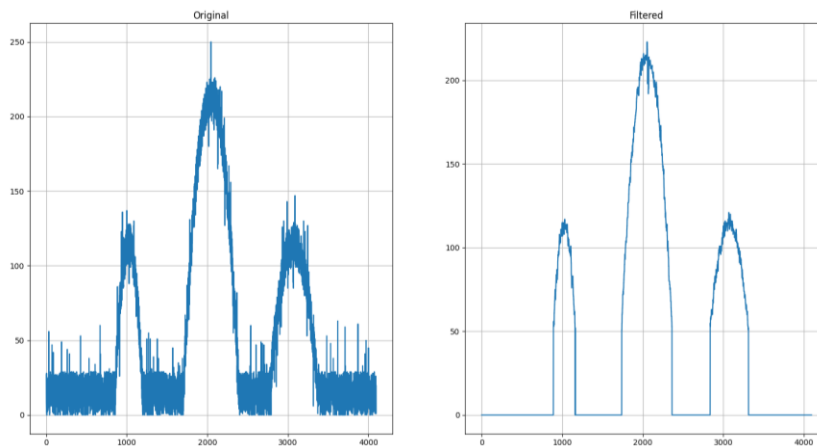


Figure 1. Result of filtering algorithms, window size 6

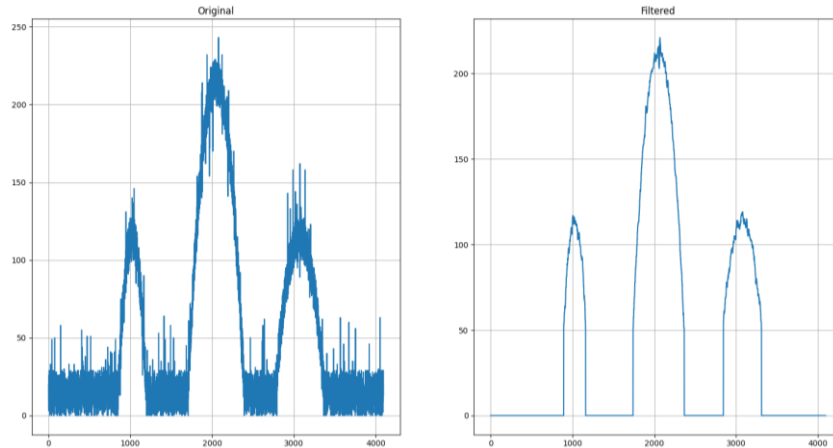


Figure 2. Result of filtering algorithms, window size 10

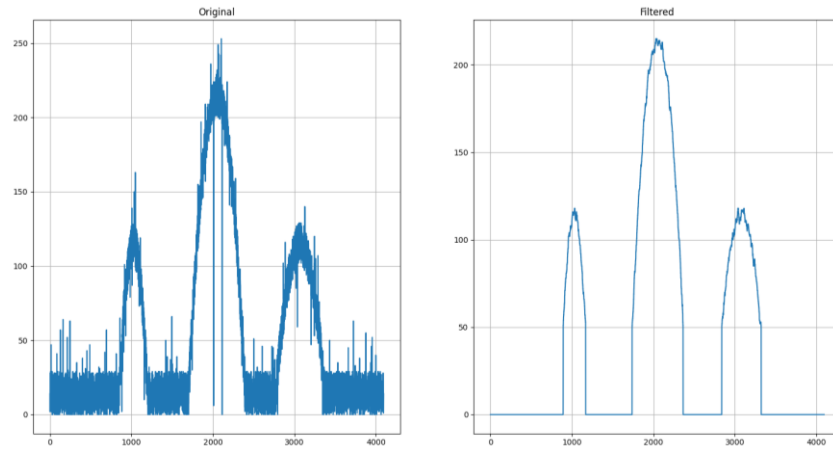


Figure 3. Result of filtering algorithms, window size 14

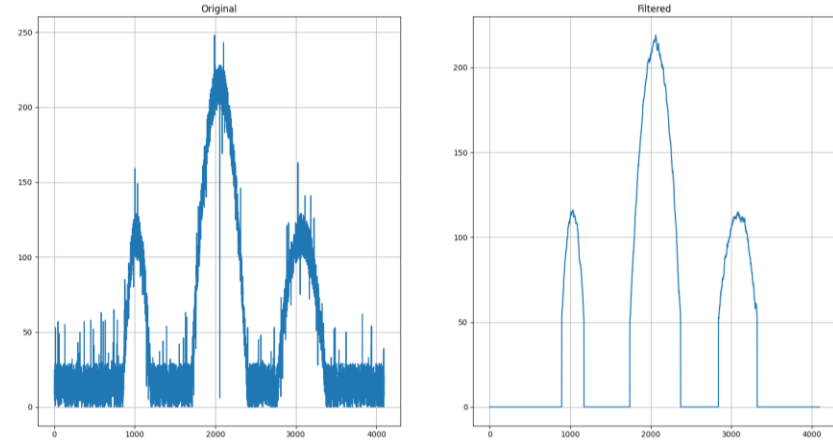


Figure 4. Result of filtering algorithms, window size 18

When generating input data, the standard C++ “random” library was used to create noise and outliers in the input image. During the generation of the peak, in addition to the main peak, two additional peaks were created with half the amplitude and width.

The Peak Signal-to-Noise Ratio (PSNR) is a widely used metric for measuring the quality of a reconstructed or filtered image/signal compared to the original. It is expressed in decibels (dB), and higher PSNR values indicate better quality (less distortion or noise).

$$PSNR = \frac{\max(X_f)}{MSE}$$

Here, X_f denotes the signal after filtering.

The PSNR results for different window sizes are shown in Figure 5.

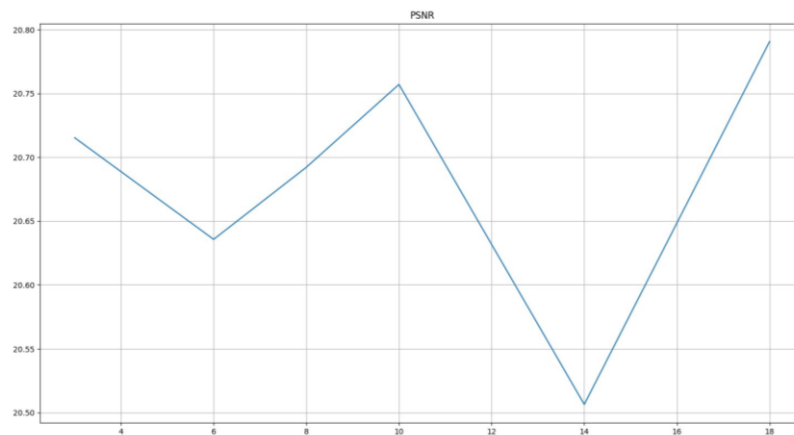


Figure 9. PSNR results for different window sizes

Table 2. PSNR Quality Interpretation

PSNR (dB)	Quality assessment
> 40 dB	Excellent (nearly lossless)
30 - 40 dB	Good (minor noise, unnoticeable distortion)
20 - 30 dB	Acceptable (visible noise)
10 - 20 dB	Poor (significant distortion, noticeable degradation)
<10 dB	Very poor (heavy noise, unusable signal)

A PSNR of 20 dB is at the lower boundary of “acceptable” quality. This means that some visible noise remains in the processed signal. For high-quality signal processing, values of 30 dB or higher are generally preferred. In noisy environments (such as medical imaging, low-light video, or sensor data processing), a value of 20 dB may still be acceptable. For the purposes of this project, a PSNR of 20 dB is considered satisfactory, as the peak in the image remains clearly visible.

The filtering algorithms successfully handled the task of filtering a noisy image with a small number of outliers. According to the graphs, we can observe that the optimal window width for performing this task is 14 pixels.

Current limitations and future improvements

At present, the controller algorithms are designed only for a frame resolution of 4096×2 . This imposes restrictions on the types of cameras that can be used. In the future, the algorithm should be refined to enable compatibility with various camera models.

The implementation of the algorithms using the Vitis HLS tool allows for relatively easy modification of the algorithm’s operation, making it possible in the future to incorporate median filters or Gaussian filters of various configurations.

Conclusion

As a result of this work, the FPGA resource utilization required for filter implementation was obtained, and the developed filters were tested on generated data to evaluate filtering quality.

The amount of FPGA resources used (10,000 LUTs) is considered acceptable for this type of algorithm implementation, as such an approach allows rapid modifications to the filter operation logic.

The PSNR metric for the moving average filter (approximately 20 dB) falls within the acceptable range of filtering quality. Since the peak in the camera image is distinctly pronounced, this level of filtering quality is satisfactory for the intended application.

Acknowledgments

The authors express their gratitude to the management of the Lasers and Equipment TM group of companies for their support in providing the material and technical resources necessary for experimental research and modeling of the process under consideration.

References

1. Salkov Yu.A., Tretiyakova O.N., Tuzhilin D.N. Development and application of algorithms of video stream filtering and processing on programmable logic integrated circuits FPGA // *Scientific Visualization*. 2024. Vol. 16, No. 4, pp. 102–108.
2. Сальков Ю.А., Сапрыкин Д.Л., Третьякова О.Н., Тужилин Д.Н. Разработка программного обеспечения контроллера системы соосного слежения за профилем образца при лазерной обработке в режиме реального времени [Software development of a controller for a coaxial sample profile tracking system during real-time laser processing] // *Приборы [Instruments]*. 2024. No. 5 (287), pp. 8–14.
3. Navaneethan, S., Varshini, M. A., Anil, A., & Thanusha, C. (2022). A Hardware Efficient Real-time Video Processing on ПЛИС [FPGA] with OV 7670 Camera Interface and VGA. *Proceedings of the Sixth International Conference on Electronics, Communication and Aerospace Technology (ICECA 2022)*. IEEE.
4. Said, Y., Saidani, T., Smach, F., Atri, M., & Snoussi, H. (2012). Embedded Real-Time Video Processing System on ПЛИС [FPGA]. In A. Elmoataz et al. (Eds.), *Lecture Notes in Computer Science (LNCS)*, vol. 7340, pp. 85–92. Springer-Verlag.
5. Ivashko, A., Zuev, A., Karaman, D., & Moškon, M. (2024). FPGA-Based Implementation of a Gaussian Smoothing Filter with Powers-of-Two Coefficients. *Advanced Information Systems*, Vol. 8, No. 2, pp. 39–46. DOI: 10.20998/2522-9052.2024.2.05
6. Molotkov A.A., Tretiyakova O.N. On possible approaches to visualizing the process of selective laser melting // *Scientific Visualization*, 2019, Vol. 11, No. 4, pp. 1–12. DOI: 10.26583/sv.11.4.01
7. Желтов С.Ю. и др. Обработка и анализ изображений в задачах машинного зрения [Image processing and analysis in machine vision tasks] // *Физматкнига [Fizmatkniga]*, 2010. 672 p.
8. Кондратенко В.С., Третьякова О.Н., Шевченко Г.Ю. Разработка средств управления лазерным технологическим оборудованием с различными кинематическими схемами [Development of control tools for laser technological equipment with various kinematic schemes] // *Вестник Московского авиационного института [Bulletin of Moscow Aviation Institute]*, 2015, Vol. 22, No. 2, pp. 121–131.
9. Молотков А.А., Сапрыкин Д.Л., Третьякова О.Н., Тужилин Д.Н. Программное обеспечение регистрации потоковых видеоданных FlexRecorder [FlexRecorder software for video stream registration]. State Registration Certificate of Computer Program No. 2021669002. Application No. 2021668322. Submission Date: 16 November 2021. Registration Date: 23 November 2021.
10. Третьякова О.Н., Молотков А.А., Семашко В.С. Программа FastDetector машинного зрения для создания детали в процессе селективного лазерного сплавления [FastDetector machine vision software for part creation in selective laser melting]. State Registration Certificate of Computer Program No. 2021669857. Application No. 2021669184. Submission Date: 26 November 2021. Registration Date: 3 December 2021.
11. Третьякова О.Н., Молотков А.А., Желябин И.А. Программа HarrisDetector машинного зрения для создания детали в процессе селективного лазерного сплавления [HarrisDetector machine vision software for part creation in selective laser melting]. State Registration Certificate of Computer Program No. 2021680165. Application No. 2021669247. Submission Date: December 2021. Registration Date: 3 December 2021.

12. Jia, Chuanmin, et al. "FPX-NIC: An FPGA-accelerated 4K ultra-high-definition neural video coding system." *IEEE Transactions on Circuits and Systems for Video Technology*, 32(9), 6385–6399, 2022.
13. Cong, Jason, et al. "FPGA HLS today: successes, challenges, and opportunities." *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 15(4), 1–42, 2022.
14. Bailey, Donald G. *Design for Embedded Image Processing on FPGAs*. John Wiley & Sons, 2023.
15. Cui, Beiyao, et al. "Design of Image Acquisition Circuit for Pixel Coupled Infrared Polarization Camera Based on FPGA." *2023 4th International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*. IEEE, 2023.
16. Viola P., Jones M. Rapid object detection using a boosted cascade of simple features // *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE, 2001. pp. I–511.