symmetry

MDPI

Article

A Low-Latency Noise-Aware Tone Mapping Operator for Hardware Implementation with a Locally Weighted Guided Filter

Qianwang Liang, Tianyu Yan, Nan Wang, Zhiying Zhu and Jiongyao Ye *

School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
* Correspondence: yejy@ecust.edu.cn

Abstract: A tone mapping operator (TMO) is a module in the image signal processing pipeline that is used to convert high dynamic range images to low dynamic range images for display. Currently, state-of-the-art TMOs typically take complex algorithms and are implemented on graphics processing units, making it difficult to run with low latency on edge devices, and TMOs implemented in hardware circuits often lack additional noise suppression because of latency and hardware resource constraints. To address these issues, we proposed a low-latency noise-aware TMO for hardware implementation. Firstly, a locally weighted guided filter is proposed to decompose the luminance image into a base layer and a detail layer, with the weight function symmetric concerning the central pixel value of a window. Secondly, the mean and standard deviation of the basic layer and the detail layer are used to estimate the noise visibility according to the human visual characteristics. Finally, the gain for the detail layer is calculated to achieve adaptive noise suppression. In this process, luminance is first processed by the log2 function before being filtered and then symmetrically converted back to the linear domain by the exp2 function after compression. Meanwhile, the algorithms within the proposed TMO were optimized for hardware implementation to minimize latency and cache, achieving a low latency of 60.32 µs under video specification of 1080 P at 60 frames per second and objective metric smoothness in dark flat regions could be improved by more than 10% compared to similar methods.

Keywords: video tone mapping; guided filter; low-latency; noise suppression

Citation: Liang, Q.; Yan, T.; Wang, N.; Zhu, Z.; Ye, J. A Low-Latency Noise-Aware Tone Mapping Operator for Hardware Implementation with a Locally Weighted Guided Filter. *Symmetry* **2024**, *16*, 356. https://doi.org/10.3390/sym16030356

Academic Editor: Theodore E. Simos

Received: 8 February 2024
Revised: 13 March 2024
Accepted: 13 March 2024
Published: 15 March 2024

1. Introduction

With the development of image sensors and imaging technology, high dynamic range (HDR) imaging has found widespread applications in medical imaging [1], autonomous driving [2], gaming [3], and various other fields. Current imaging devices, which often produce images with a high bit depth, coupled with the maturity of HDR synthesis techniques, lead to the expectation of a growing number of HDR videos in the future. However, most current display devices have a limited range of 8 bits [4], necessitating the use of tone mapping operators (TMOs) to map images into the display range. The TMOs help both highlights and shadows to display more details, providing better subjective quality.

Because of the different purposes of applying TMOs on images, contingent upon the target application [4], diverse TMOs and their hardware implementations have continued to be proposed in recent years. For instance, Refs. [5–7] introduced TMOs based on neural networks, while [8–10] presented the hardware implementations optimized for different TMOs. Therefore, research on TMOs and their hardware implementation remains an active and important research field.

Traditional TMOs can be categorized into two types. Global TMOs, such as the adaptive logarithmic transformation [11] and histogram-based method [12], apply a single tone curve to the entire image, and it is straightforward and effective for most scenes.

However, global TMOs may result in insufficient local contrast, leading to the loss of some local details. Local TMOs, such as TMOs based on Retinex theory [13], the image color appearance model [14], and the TMO proposed in [15], adjust the tone curve based on local information around a pixel. These TMOs have been extensively developed to meet the requirements of many applications.

In recent years, with the development of neural networks, there has been an increasing trend in utilizing deep neural networks to perform tone mapping (TM) tasks. Numerous TMOs based on deep neural networks have emerged. For example, Cao et al. [5] proposed a unified framework for unsupervised image and video tone mapping based on neural networks. Notably, to facilitate the unsupervised training process for video tone mapping, Ref. [5] constructed a large-scale unpaired HDR-LDR video dataset. Zhang et al. [7] proposed a TMO that is based on the generative adversarial network architecture and introduced a new high-quality 4K HDR-SDR dataset of image pairs, covering a wide range of brightness levels and colors. Wang et al. [16] proposed a learning-based self-supervised tone mapping operator that is trained at test time specifically for each HDR image and does not need any data labeling. Despite offering superior visual subjective quality, these algorithms often incur substantial latency, which can be hundreds of milliseconds, even when running on high-performance devices, such as graphics processing units (GPUs) [17]. Therefore, it is challenging to apply such algorithms in applications with strict latency requirements, such as autonomous driving [18] and real-time medical image processing [19].

Moreover, TMOs tend to amplify the existing noise in the image, particularly in the darker regions [20], and this may potentially impact subsequent processing algorithms, such as altering the results of the following feature extraction [21]. In pursuit of higher image quality, Gödrich et al. [22] employed the U-Net architecture in a deep learning approach to approximate an optimized multiscale Retinex method, effectively reducing noise in tone-mapped images [22]. However, this network structure exhibits a runtime of several tens of milliseconds on a GPU [17]. With power and resource constraints in embedded vision applications, a straightforward porting of software algorithms to hardware platforms may result in poor performance or even system failures, failing to meet the increasing demands of numerous embedded vision applications (such as advanced automotive systems, medical imaging, and unmanned aerial vehicles) [4]. The primary components of such vision-based embedded systems include image sensors, image processing algorithms, and display monitors. For these embedded applications with stringent time constraints, hardware acceleration is indispensable, and the ability to build optimized custom hardware makes field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) better than GPUs [4]. The hardware implementation of image processing algorithms must be optimized for hardware portability, as this redesigning effort can leverage hardware platforms to achieve optimal performance. Consequently, this has led to the development of numerous novel hardware TM algorithms and architectures, such as [8–10,23]. However, the majority of these hardware TMOs are primarily focused on optimizing hardware implementation or architecture, aiming to utilize fewer hardware resources, lower latency, or achieve higher throughput. They either lack dedicated consideration for noise suppression [8,10,23] or do not comprehensively address the issue of noise [9]. The issue of noise can potentially alter the outcomes of subsequent algorithms in the system, such as feature extraction [21] and object detection [24], thereby impacting the result of the overall system.

In summary, for certain embedded vision applications, the currently available hardware-implemented TMOs do not fully address the issue of noise. To further alleviate the impacts of noise, this paper proposes a low-latency noise-aware TMO for hardware implementation. While replacing the filter to enhance image quality, the TMO's capability of noise suppression (NS) is enhanced by introducing a new method for noise suppression. However, these enhancements increase the system latency and line buffer requirements, compromising hardware performance. Therefore, we optimize the algorithms for hardware implementation, reducing latency and line buffer requirements. Additionally, several operations, such as cut and data

prediction, are introduced to refine the entire system. Finally, the feasibility of the proposed TMO is validated on an FPGA.

Our main contributions are summarized as follows:

1. A local weighting method with a symmetric weight function is proposed in the guided filter (GF) to enhance the edge-preserving capability of a GF under a large regularization parameter ϵ. This enables the GF, when applied to TM or image enhancement, to effectively mitigate halo artifacts, thereby improving the visual subjective quality of the results.
2. A hardware-oriented noise visibility assessment method based on human visual characteristics is proposed, facilitating the implementation of introduced NS in hardware. It leverages the intrinsic information of the image without the need for calibration. This endows the hardware TMO with noise suppression capabilities, surpassing the current counterpart in a similar TMO.
3. Optimizations are applied to the GF and the algorithm for finding the K-th value to reduce latency and minimize buffer requirements in hardware implementation. This results in approximately a $\frac{1}{3}$ reduction in overall system latency and a $\frac{2}{5}$ reduction in line buffer demands. These enhancements enable the TMO to handle continuous video stream inputs in hardware, facilitating implementation on hardware.

Building upon the three aforementioned key contributions, a comprehensive hardware-based TMO system with noise suppression has been designed and implemented, offering a novel solution for noise-sensitive embedded visual applications.

In the following sections, the development of hardware- and software-implemented TMOs is first introduced in Section 2. Subsequently, the three major contributions mentioned above are detailed in Section 3, followed by the hardware implementation and optimization discussed in Section 4. Finally, in Section 5, to demonstrate the usefulness of the proposed TMO, we use a series of input HDR video sequences to evaluate the proposed TMO and similar approaches, showing the advantages of proposed TMO in NS and system latency of hardware implementation.

2. Related Work

Currently, hardware-implemented TM algorithms predominantly rely on traditional TM techniques, and most of them are the improvements of algorithms proposed in [25,26] and the algorithms based on Retinex theory [8] and retinal information processing mechanisms [23]. For instance, Upadhyay et al. [8] proposed a Retinex-based algorithm that employs a low-cost edge-preserving filter for illumination estimation and implemented it on an FPGA, but their algorithm does not take into account the problem of noise. Nosko et al. [26] designed a comprehensive HDR real-time processing system based on the Durand TMO, which filters the maximum and minimum values by averaging from previous frames to ensure smooth adaptation and facilitates implementation on an FPGA, enabling the system to process the video under specification of 1080 P at 96 frames per second (1080 P @ 96 FPS). However, with the use of a Gaussian filter, there may be some artifacts in certain situations, and the issue of noise was not taken into consideration. Park et al. [27] proposed a hardware implementation of a low-cost, high-throughput video enhancement system based on the Retinex algorithm that minimizes hardware resources while maintaining quality and performance by applying the concept of approximate computing to Gaussian filters and by designing a new nontrivial exponentiation operation. Xiang et al. [23] implemented a biological retina-inspired tone mapping processor for high-speed, low-power image enhancement that introduced various hardware design techniques, such as adjacent frame feature sharing, multi-layer convolution pipelining, etc. Nevertheless, both [23,27] focused solely on optimizing the hardware implementation of the algorithm without improving the algorithm itself. Ambalathankandy et al. [28] presented an adaptive global and local tone mapping algorithm and its FPGA implementation, which is based on local histogram equalization, but it demands a relatively large number of memory resources as it requires the utilization of a downscaled frame. Lang et al. [9] noticed the issue of noise but did not

comprehensively take into account the characteristics of the human visual system (HVS). They proposed a method for expanding the dynamic range of a low-light night vision system based on guided filter and adaptive detail enhancement. In the process of adaptive detail enhancement, only the impacts of edges on noise visibility were considered, the impacts of background brightness were ignored.

In software-implemented algorithms, some authors have taken the issue of noise into account. Eilertsen et al. [29] addressed noise-related problems by utilizing a noise model, the contrast sensitivity function of the human eye, and visual system threshold curves to calculate the noise level and the gain of detail. However, this approach requires a lot of calibration data and runs on a GPU. Hu et al. [30] designed a neural network structure that combines TM with denoising to accomplish both tasks and explored optimal structural arrangements, but the number of multiply–accumulate operations and learnable parameters of this model significantly increased compared to a simple TM task. Gödrich et al. [22] employed the U-Net architecture in a deep learning approach to approximate an optimized multiscale Retinex method and used a self-supervised deep learning method to implicitly reduce the noise in the image after tone mapping. However, these algorithms are primarily designed for high-performance GPU platforms, which have high complexity and computational demands. Therefore, they are not suitable for applications in scenarios with strict requirements for low latency and low noise, especially when deployment on edge devices is necessary.

Therefore, considering the requirements for low latency and low noise in real-time visual applications on embedded devices, such as autonomous driving [18] and real-time medical image processing [19], we propose a hardware-oriented, low-latency TMO with NS and validate the algorithm's effectiveness on the FPGA platform.

3. The Proposed Low-Latency Noise-Aware TMO

Inspired by the TMO in [26], our proposed low-latency noise-aware TMO is shown in Figure 1. The HDR color image input is initially transformed into a luminance image through the luma module. And, the image is then converted into the log_2 domain by the log module following the cut operation. Subsequently, the luminance image is decomposed into a base layer and a detail layer by the HLWGF module. Then, compression and NS are applied to the base layer and the detail layer, respectively. The compressed image is then symmetrically converted back into the linear domain, followed by normalization and color reproduction.

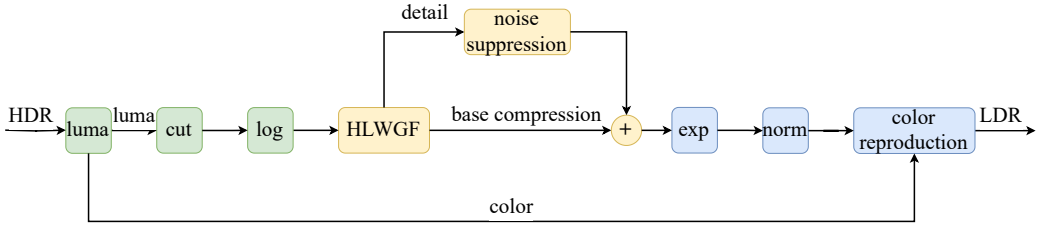


Figure 1. Diagram of proposed low-latency noise-aware TMO.

The functions of each module are summarized as follows:

1. The luma module converts the color image to a luminance image and color coefficients.
2. The cut module clamps the upper and lower bounds of the pixel value of the whole image.
3. The log module converts luminance from the linear domain to the log_2 domain.
4. The HLWGF module decomposes the image into a base layer and detail layer using a hardware-oriented locally weighted guided filter (HLWGF).
5. The noise suppression module calculates noise visibility based on human visual characteristics and uses a threshold to calculate gains to suppress noise in the detail layer.
6. The exp module converts the luminance from the log_2 domain to the linear domain symmetrically.

7. The norm module normalizes the compressed luminance image.
8. The color reproduction module reproduces the color of the normalized image.

This paper enhances the TMO for hardware implementation through three key improvements. Firstly, optimization is applied to the method of finding the K-th largest/smallest value, enabling the TMO to efficiently process continuous video streams on hardware. Secondly, hardware-oriented optimization is performed on the GF, along with the introduction of a local weighting mechanism, successfully enhancing its edge-preserving capability while concurrently reducing artifacts generated under large regularization parameter ϵ. Thirdly, to meet the high image quality requirements of the target application, an NS mechanism based on the characteristics of HVS is introduced, which is easy to implement in hardware. The details of these three parts are described in the following subsections.

3.1. Fast Estimation of the K-th Largest/Smallest Value for Hardware Implementation

For the cut operation, it is necessary to know the maximum clamp value and the minimum clamp value. This problem can be described as finding the K-th largest and smallest values in an array. There are two kinds of algorithms to solve this problem: sorting and histogram.

For hardware implementation, the K-th largest/smallest value of the current frame requires a latency of one entire frame. To reduce system latency and the extensive frame buffer, we utilize an exponential filter to smooth the K-th largest/smallest value of all previously input frames. These smoothed values are then utilized as predictions for the K-th largest and smallest values of the current frame, as discussed in Section 4.5.

Whether it is complete sorting or partial sorting, these algorithms require significant latency and hardware resources (such as logic resources and buffers). The latency tends to increase linearly or logarithmically with the data size N, and some algorithms do not support real-time continuous data stream inputs [31]. In our case, the data size is the number of all pixels in an image. For a 1080 P image, the number of pixels can reach 2,073,600, and the input is in the form of a stream. Therefore, we adopt a histogram-based algorithm to address this issue. However, for 16-bit depth pixels, 65,536 bins are needed to store the count of each pixel value. The worst-case latency to obtain the final result is 65,536 cycles, which is not suitable for continuous frame inputs with intervals of only a few thousand cycles, as it may lead to system instability when the latency exceeds the inter-frame interval cycles. To ensure the stable processing of continuous video streams, the use of a compressed histogram allows for lower latency and reduced buffer overhead. The specific steps of the method are outlined below.

To locate the value at x% in an image (with dimensions $W \times H$), it is necessary to find the value at the $W \times H \times x$% position in the sorted image data. Taking the example of finding the top 0.1% and bottom 0.1% values in a 1920×1080 image, we need to find the value at the $1920 \times 1080 \times 0.001 = 2073.6 \approx 2074$th position. Since the output image data will be compressed to 10 bits or 8 bits, the number of histogram bins is compressed to 1024 or 256, with 13 bits each. After the data stream input of a frame is completed, the values of 1024 or 256 registers are traversed from the first and last bin to the other end, and the value range of the top/bottom 0.1% is found when the accumulator is detected to be greater than 2074 for the first time. Here, to reduce the impacts of the cut operation on normal pixel values, the upper bound of the top 0.1% interval is taken as the desired value. Similarly, the lower bound of the bottom 0.1% interval is taken as the desired value (other methods can be selected according to the situation, such as selecting the middle value of the interval). The pseudo-code flow is shown in Algorithm 1, where img is the input image; min_num and max_num are the bottom and top positions, respectively, calculated according to the cut ratio; $hist\_bin, min\_cnt, max\_cnt, i, j$ are temporary variables; and Kth_min and Kth_max are the estimated K-th smallest and largest values, respectively. Finally, the algorithm's latency and buffer requirements are reduced to 1/64 or 1/256 of the original, corresponding to compression to 10 bits and 8 bits, respectively.

Algorithm 1 Fast estimation of the K-th value for hardware implementation.

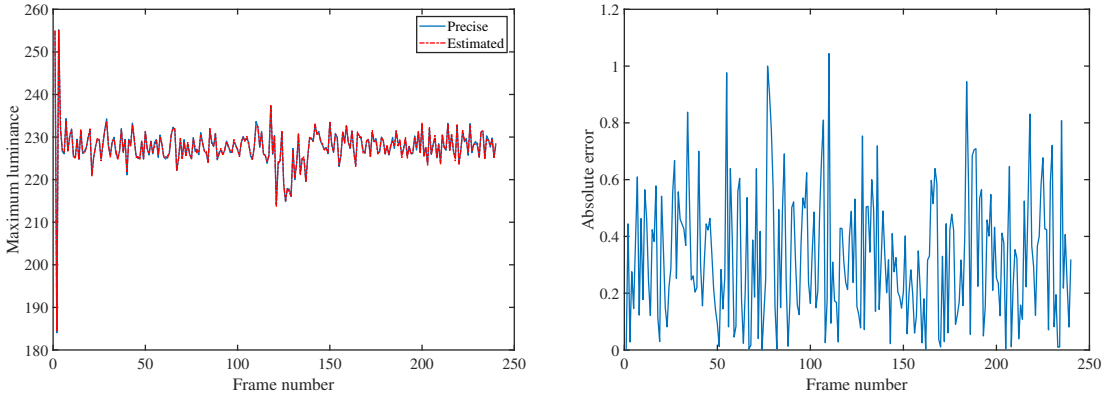
$hist\_bin[256] \leftarrow 0$
$min\_cnt \leftarrow 0$
$max\_cnt \leftarrow 0$
for $i \leftarrow 0$ **to** $N-1$ **do**
  $hist\_bin[img[i]/256] \leftarrow hist\_bin[img[i]/256] + 1$
end for
for $i \leftarrow 0$ **to** 255 **do**
  $min\_cnt \leftarrow min\_cnt + hist\_bin[i]$
  if $min\_cnt \geq min\_num$ **then**
    break
  end if
end for
for $j \leftarrow 255$ **to** 0 **do**
  $max\_cnt \leftarrow max\_cnt + hist\_bin[j]$
  if $max\_cnt \geq max\_num$ **then**
    break
  end if
end for
$Kth\_min = i \times 256$
$Kth\_max = (j+1) \times 256$

In summary, the fast estimation of the K-th value can be summarized in the following four steps:

1. Initialize the compressed histogram bins with 0.
2. Traverse and accumulate counts from both ends of the histogram toward the other end.
3. When the cumulative sum, while traversing either from left to right (right to left), exceeds the ranking of the bottom K-th *min_num* (or top K-th *max_num*) position, exit the traversal and record the corresponding bin index i (or j).
4. Utilize the bin index i (or $j+1$) obtained in step 3, multiplied by the compression ratio 256, to estimate the value of the K-th largest/smallest.

To assess the impacts of using the compressed histogram to obtain the K-th largest and smallest values on the results, with other parameters held constant, we conducted the following experiment: within the entire TM process, using the exponential filtering method, we calculated the K-th largest and smallest values using both precise and estimated methods, respectively, and then observed the normalized maximum luminance values of video frames after the TMO. The test was conducted using the "Fireplace" teaser clipof the HdM-HDR-2014 dataset [32]. For better result observation, the maximum luminance values of the first frame for both methods were set to 255 when plotting with Matlab.

Figure 2 shows the impacts of two different methods for obtaining the K-th largest and smallest values, and the results of the two methods in Figure 2a are almost the same. Figure 2b presents the absolute errors in the maximum luminance between the results obtained by the two methods, with the maximum absolute error and average absolute error being 1.0441 and 0.3295, respectively. Consequently, the improved estimation method minimally affects the results while significantly reducing latency and buffer requirements.



(**a**) The maximum luminance of images after TM. (**b**) The absolute errors between the two methods.

Figure 2. The maximum luminance of images after TM and the absolute errors between the two methods.

3.2. Hardware-Oriented Locally Weighted Guided Filter

It is well known that a bilateral filter tends to cause gradient reversal when used for detail enhancement or TM. In contrast, GF can alleviate this phenomenon [9,33]. Therefore, we adopted the GF [33] to replace the bilateral filter in our TMO. For the convenience of comparison with subsequent improvements, we recapitulate the GF algorithm here. The GF assumes that the output image q is a linear transform of guidance image I in a square window w_k (with radius r) centered at the pixel k:

$$q_i = a_k I_i + b_k, \forall i \in w_k, \tag{1}$$

where the subscript i represents a certain pixel in the image and (a_k, b_k) are the linear coefficients obtained by least square method in w_k.

The GF minimizes the difference between q and the filter input image p, and the cost function in the window w_k is defined as follows:

$$E(a_k, b_k) = \sum_{i \in w_k} ((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2), \tag{2}$$

where ϵ is a regularization parameter preventing a_k from being too large. The solution to Equation (2) can be given by linear regression:

$$a_k = \frac{\frac{1}{|w|}\sum_{i \in w_k} p_i I_i - \overline{p}_k \mu_k}{\frac{1}{|w|}\sum_{i \in w_k} (I_i - \mu_k)^2 + \epsilon} \tag{3}$$

$$b_k = \overline{p}_k - \mu_k a_k, \tag{4}$$

where μ_k is the mean of I in w_k, $|w|$ is the number of pixels in w_k, and $\overline{p}_k$ is the mean of p in w_k. However, a pixel i is involved in all the windows w_k that contain i so that GF averages all the possible values of q_i. The final filter output is computed by

$$q_i = \frac{1}{|w|} \sum_{k: i \in w_k} (a_k I_i + b_k) = \overline{a}_i I_i + \overline{b}_i, \tag{5}$$

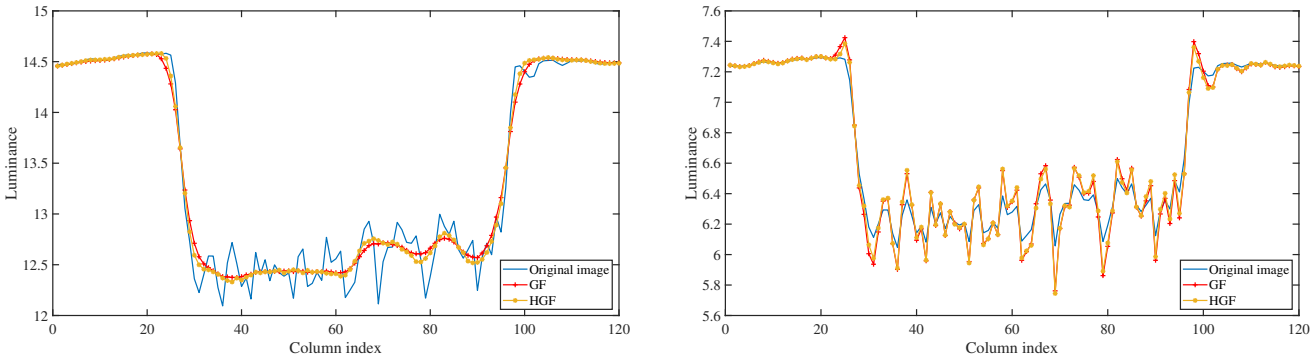
where $\overline{a}_i = \frac{1}{|w|}\sum_{k \in w_i} a_k$ and $\overline{b}_i = \frac{1}{|w|}\sum_{k \in w_i} b_k$.

The GF averages the a_k and b_k before obtaining the result q (as expressed in Equation (5)). This operation is equivalent to expanding the "receptive field" of the current point, enlarging the range that can influence the result. While this enhances the smoothness of the filtering result, it diminishes the edge-preserving capability in large edge regions. Furthermore, it introduces increased latency and line buffer requirements for hardware implementation. To address these challenges, optimizations are implemented for the GF.

To reduce the latency and line buffer cost of the GF in hardware implementation, a hardware-oriented guided filter (HGF) is introduced. This is achieved by eliminating the step of averaging a_k and b_k and directly computing the result q_i using a_k and b_k: $q_i = a_k I + b_k$. This hardware-oriented approach results in an almost $\frac{1}{2}$ reduction in latency and a $\frac{2}{3}$ reduction in line buffer.

According to the principles of the proposed TMO, which involves decomposing the image through a GF and compressing the base layer, the algorithm aims to achieve smoother results in relatively flat regions while preserving gradients as much as possible in large edge regions. This facilitates retaining more detail after the TMO. To achieve a smoother result in relatively flat regions, the parameter ϵ can be appropriately increased, typically chosen as 2.56 (calculated as $(0.1 \times 16)^2$). Further improvements in edge-preserving capability will be addressed in subsequent work.

Figure 3 illustrates the results of image processing using a GF and an HGF with parameters $r = 2, \epsilon = 2.56$. These lines represent the 1D profiles of the result images at coordinates (800, 425:544) in Matlab. Both input images for the GF and HGF are from the first frame of FireplaceTeaser, transformed into the log_2 domain. It can be observed from Figure 3a that, in cases with a relatively large ϵ, the difference between the results of the GF and HGF is negligible in flat regions, producing good smoothing outcomes. In large edge regions, the HGF exhibits a slightly better capability of preserving large edges than the GF. Figure 3b demonstrates the results after restoring details, showcasing halo artifacts near $Column\ index = 25$ and $Column\ index = 98$. The disparity in results between the GF and HGF is not significant, as the HGF does not fundamentally enhance edge preservation near large edges.



(**a**) The 1D profiles of filtered images. (**b**) The 1D profiles of images after restoring detail.

Figure 3. The results of the GF and HGF.

To enhance the edge-preserving capability of HGF, it is essential to fundamentally analyze the principles of the HGF. The scenario depicted in Figure 3 arises because the least squares method assigns equal weights to every point within the window, resulting in a significant impact from points on the other side of the edge with a large difference from the central point I_k. To address this issue, the weights of points with a larger difference from the center point I_k are reduced. The modified cost function of the HGF is defined as follows:

$$E(a_k, b_k) = \sum_{i \in w_k} (\omega_i (a_k I_i + b_k - p_i)^2 + \epsilon a_k^2), \tag{6}$$

where the ω_i is

$$\omega_i = \begin{cases} 1, & dw_i < ths_{low} \\ 1 - (\frac{dw_i - ths_{low}}{\lambda}), & ths_{low} < dw_i < \lambda + ths_{low} \\ 0, & \lambda + ths_{low} < dw_i \end{cases}, \tag{7}$$

where the dw_i is

$$dw_i = \frac{abs(I_i - I_k)}{I_{max} - I_{min}}, \tag{8}$$

where I_{max} and I_{min} are the maximum and minimum values of I, respectively. The function dw_i is symmetric with respect to I_k, thereby ensuring that the weights w_i also exhibit symmetry with respect to I_k, which guarantees an equal consideration for both higher and lower edges relative to the central pixel.

The solution to Equation (6) can be given by linear regression:

$$a_k = \frac{\frac{1}{W}\sum_{i \in w_k} \omega_i p_i I_i - \overline{p}_k \mu_k}{\frac{1}{W}\sum_{i \in w_k} \omega_i (I_i - \mu_k)^2 + \epsilon} \tag{9}$$

$$b_k = \overline{p}_k - \mu_k a_k, \tag{10}$$

where W represents the sum of weights ω_i in the local window w_k and μ_k and $\overline{p}_k$ represent the weighted means of I and p in w_k, respectively:

$$\mu_k = \frac{1}{W}\sum_{i \in w_k} \omega_i I_i \tag{11}$$

$$\overline{p}_k = \frac{1}{W}\sum_{i \in w_k} \omega_i p_i \tag{12}$$

Finally, the HLWGF is shown as Algorithm 2, where "$f_{w\_mean}(I, r)$" means to calculate the weighted average of image I within a window with radius r, as shown in Equation (11). "$.*$" and "$./$" represent the corresponding operations in Matlab, that is, two matrices of the same size multiply or divide the data at the corresponding position.

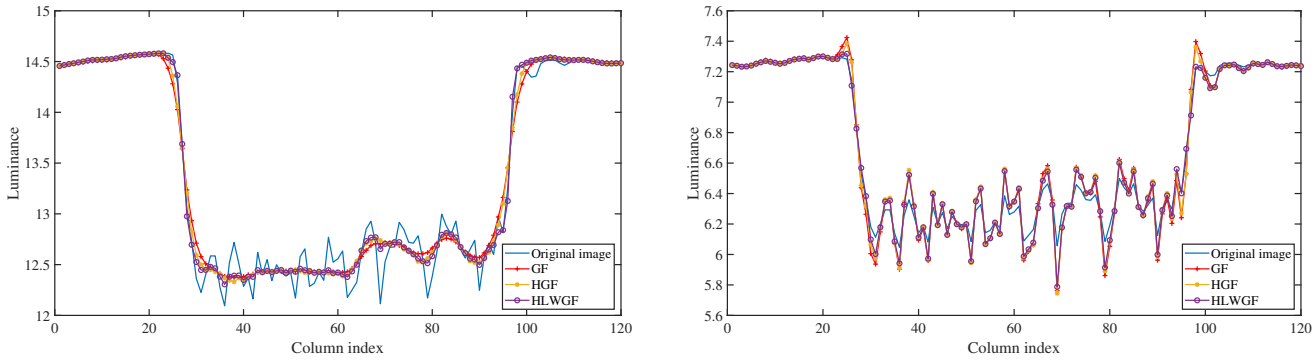
Algorithm 2 Hardware-oriented locally weighted guided filter.

$w\_mean_I = f_{w\_mean}(I, r)$
$w\_mean_p = f_{w\_mean}(p, r)$
$w\_corr_I = f_{w\_mean}(I. * I, r)$
$w\_corr_{Ip} = f_{w\_mean}(I. * p, r)$
$w\_var_I = w\_corr_I - w\_mean_I. * w\_mean_I$
$w\_cov_{Ip} = w\_corr_{Ip} - w\_mean_I. * w\_mean_p$
$a = w\_cov_{Ip}./(w\_var_I + \epsilon)$
$b = w\_mean_p - a. * w\_mean_I$
$q = a. * I + b$

In summary, the HLWGF can be outlined in the following four steps:

1. Compute the weighted averages of I and p in a window with radius r.
2. Calculate the weighted averages of $I. * I$ and $I. * p$ in a window with radius r.
3. Compute the weighted variance of $I. * I$ and the weighted covariance of $I. * p$ in a window with radius r.
4. Determine the parameters a and b using Equations (9) and (10), respectively, to obtain the result $q = a. * I + b$.

The first frame of FireplaceTeaser was converted into the log_2 domain, and it was used as the input and guide images for the above three GFs. Other parameters were kept consistent ($r = 2, \epsilon = 2.56, ths_{low} = 0.02, \lambda = 0.11, I_{max} = max(I), I_{min} = min(I)$). Figure 4 shows the results of the GF, HGF, and HLWGF. It can be seen that the introduction of the local weighting mechanism significantly alleviates the halo artifacts in large edge regions. Additionally, the capability of preserving details in flat regions is similar to other methods. Therefore, the HLWGF effectively meets our target requirements.

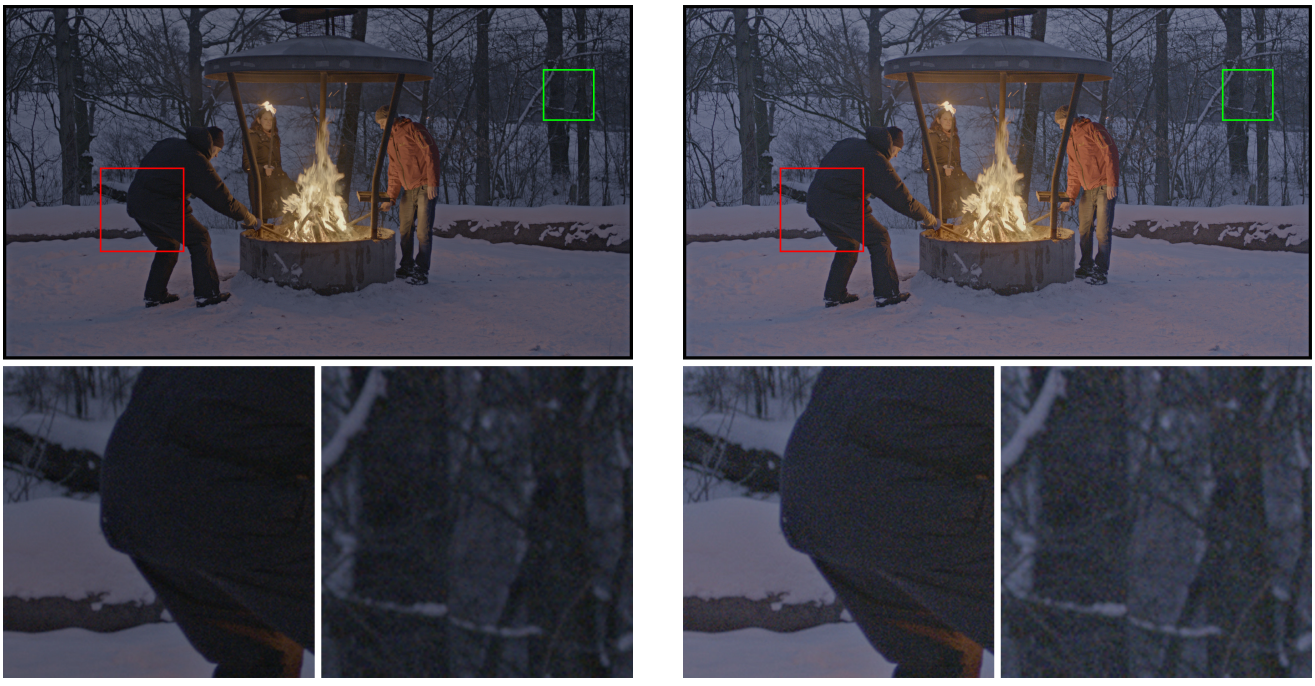


(**a**) The 1D profiles of filtered images. (**b**) The 1D profiles of images after restoring detail.

Figure 4. The results of the GF, HGF, and HLWGF.

3.3. Noise Suppression

In the process of image acquisition, noise is inevitably introduced, especially in the case of low-light conditions. To capture well-exposed images, the gain value of camera sensors is usually increased, leading to increased noise in the dark regions. After TM, which enhances luminance and contrast in the dark regions, the noise in these regions is further amplified [20], resulting in an undesirable visual subjective quality. Figure 5 shows the results before and after the proposed TMO without NS. It can be observed that the luminance and contrast in most regions of the original image are increased, and the noise in the dark regions becomes more pronounced (such as the zoomed-in regions). It is difficult for traditional denoising algorithms to remove all noise without introducing new artifacts and significantly increasing computational complexity [20]. Therefore, it is crucial to incorporate NS during the TM process, leveraging the characteristics of noise and the HVS to achieve improved image quality.



(**a**) Original image. (**b**) Image after TMO.

Figure 5. The images before and after the TMO.

In the process of TM, if the detail layer remains unchanged, the TM preserves local contrast. However, if the values of the detail layer are amplified, image details are enhanced, but there is also a risk of amplifying noise [4], which is the detail layer that contains a lot

of noise. Therefore, we perform NS on the detail layer here. The suppression involves multiplying each pixel in the detail layer by a gain value "g", which depends on the local information around each pixel. We borrow the concept of noise visibility [20] to calculate gain "g", whose equation is defined as follows:

$$g = e \cdot min\left(1, \frac{ths_{nv}}{nv}\right), \tag{13}$$

where e and ths_{nv} are parameters that can be specified manually. e is the detail enhancement factor, which determines the maximum multiple of enhancement. ths_{nv} is the threshold of nv (noise visibility), which together with the noise visibility nv determines the gain "g" of detail. We design the equation of noise visibility based on the following two characteristics of HVS:

1. The HVS perceptual threshold for changes of luminance under a dark background is significantly lower than that under a bright background [34]. In other words, the human eye is more sensitive to the luminance change in the dark background.
2. The contrast sensitivity of the HVS decreases with the sharpness of the transition in the image [9]. In other words, the human eye is more sensitive to the luminance change in flat regions than that near the sharp edges.

Firstly, we use the standard deviation std_d within a window of the detail layer as an estimate of the noise level. Then, for the first point mentioned above, we take the average value $mean_b$ within a window of the base layer as an estimate of the background luminance of each pixel, defining the noise visibility factor 1 as $\frac{1}{mean_b}$. Secondly, we use the standard deviation std_b of the base layer as a measure of image edge sharpness. Therefore, we define noise visibility factor 2 using the formula $\frac{1}{std_b+1}$. Additionally, considering the level of noise in the detail layer, if the noise variation is greater than the variation of edges, we consider that the noise visibility should increase. Hence, noise visibility factor 2 is modified to $\frac{1}{(\theta \cdot std_b/std_d)+1}$, where θ is a manually configured parameter and the default is 3. Finally, we define the noise visibility as follows:

$$nv = std_d \cdot \frac{1}{mean_b} \cdot \frac{1}{\frac{\theta \cdot std_b}{std_d} + 1} = \frac{std_d^2}{mean_b \cdot (\theta \cdot std_b + std_d)} \tag{14}$$

Considering the significant latency and hardware resources required for hardware implementation of calculating standard deviation, we replace the standard deviation with the "SAD", which represents the sum of the absolute difference between each pixel of a window and the mean of a window. Consequently, the noise visibility function is changed as follows:

$$nv = \frac{SAD_d^2}{mean_b \cdot (\theta \cdot SAD_b + SAD_d)}, \tag{15}$$

where SAD_d and SAD_b are the SAD values of the detail layer and base layer, respectively.

In summary, the introduced NS method can be divided into the following three steps:

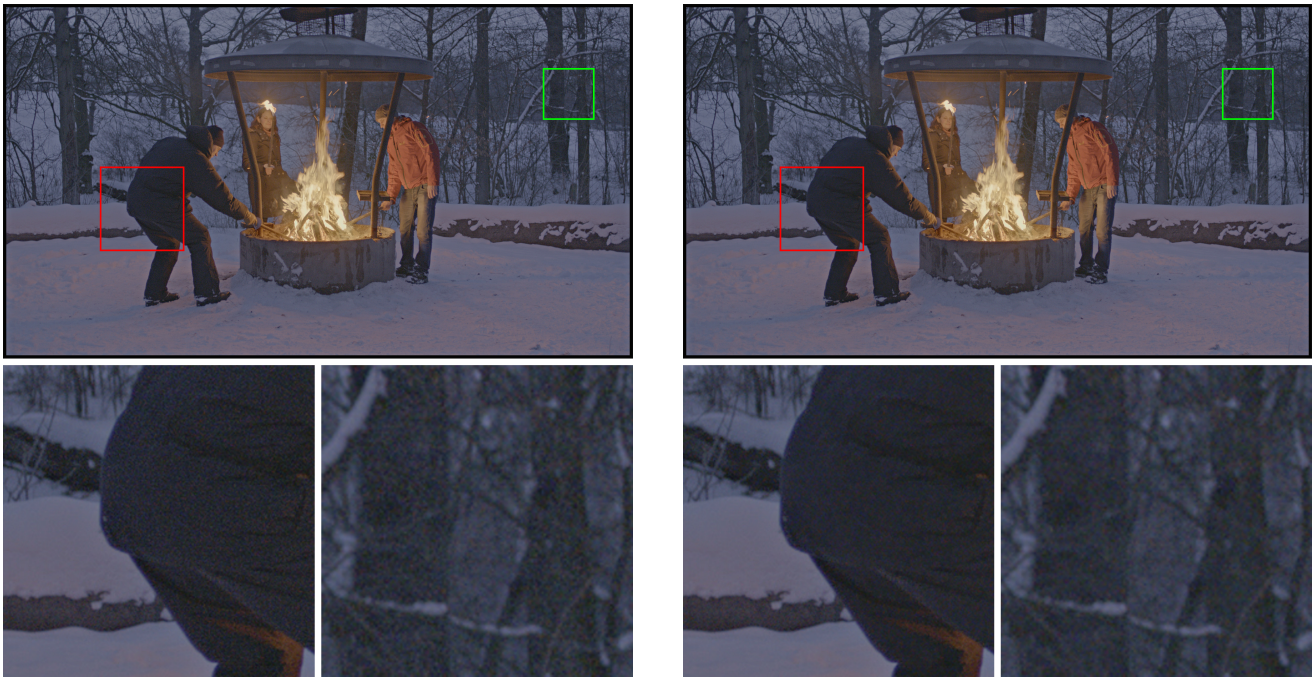
1. Compute the noise visibility, nv, in the image using Equation (15).
2. Set parameters e and ths_{nv} and calculate the gain g for each point in the detail layer using Equation (13).
3. Multiply the gain g by the corresponding points in the detail layer and add it to the compressed base layer to restore details.

An example visualization results of noise visibility and the gain of the detail layer for the 10th frame of FireplaceTeaser are presented in Figure 6. In Figure 6a, brighter regions indicate higher noise visibility, while darker regions represent lower noise visibility. In Figure 6b, brighter pixels indicate larger gain values, while darker pixels represent smaller gain values. It is evident from the figure that noise visibility is the largest in dark, flat regions, and the degree of NS is also the largest.

(**a**) Noise visibility of detail layer. (**b**) Gain map of detail layer.

Figure 6. Visualization of noise visibility and gain map of detail layer.

Figure 7 shows the result of the 10th frame in FireplaceTeaser being processed by the proposed TMO without and with NS, and the zoomed-in images of the red and green areas are displayed on the lower left and right sides, respectively. It can be observed that, without NS (Figure 7a), the noise of the trees in the background and the back of the person on the left is significantly amplified compared to the original image in Figure 5a. Upon the introduction of NS (Figure 7b), the noise level in the two areas was significantly reduced, and there was no significant effect on detail. This illustrates the adaptive capability of the introduced NS method, effectively suppressing dark noise while preserving details. Further detailed tests and data are presented in Section 5.1.



(**a**) The image after TMO without NS. (**b**) The image after TMO with NS.

Figure 7. The results of the proposed TMO without and with NS.

4. Hardware Implementation of Proposed TMO

To validate the feasibility and performance of the proposed TMO in hardware, we implemented the entire real-time TMO system on an FPGA using a pipelined design. The target video specification for this implementation was 1080 P @ 60 FPS, operating at a frequency of 148.5 MHz. The hardware implementation diagram is shown in Figure 8. The module design was basically the same as Figure 1, but the function of data prediction was designed as a hardware module, which made it easier for hardware implementation.

The color coefficients separated from *lum* module were delayed by first in first out (FIFO) cache for a certain period and used to reproduce color in *color* module. And, the red bold arrow in the figure shows the flow of the pixel data stream of the video, which was finally output by the *color* module for display. In the following subsections, we introduce the functionality and implementation details of the modules not mentioned in Section 3.

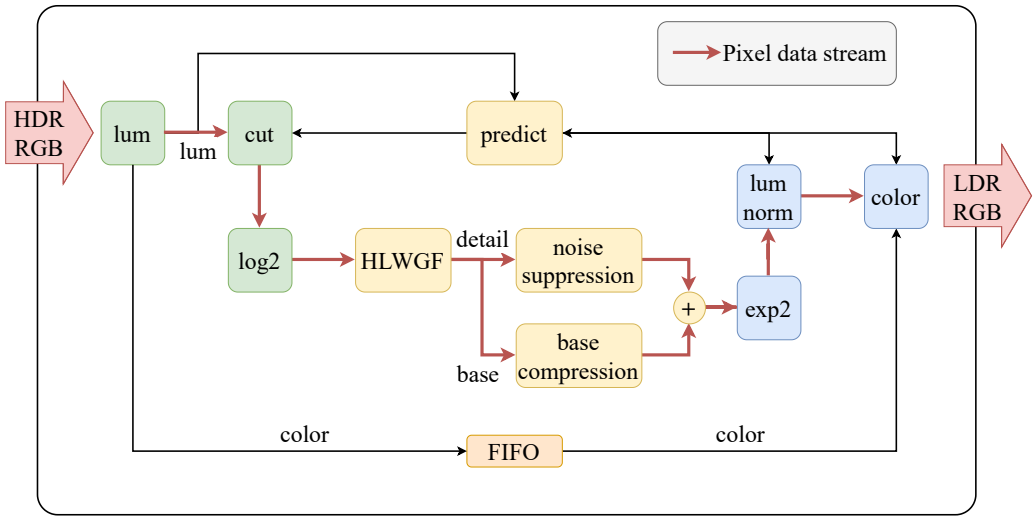


Figure 8. Hardware implementation block diagram of the whole system.

4.1. Cut

There are often some "outliers" that significantly exceed the distribution range of the main pixel values in the image. During the following normalization and mapping ("lum norm" and "color" modules), these outliers compress the contrast of the main pixels in the image, thereby reducing the subjective visual quality of the TM result [35]. To mitigate the impacts of these outliers, the cut operation (also known as clamping) is introduced to remove them, which means cutting off a certain ratio of high and low pixel values. The cut ratio needs to be determined based on the experiments under different scenarios, with common ratios such as 1% or 0.1%. Section 5 of this paper gives the chosen cut ratio.

The algorithm that obtains the K-th largest/smallest value in the cut operation is the method described in Section 3.1.

4.2. Log2 and Exp2

In hardware circuits, implementing the $log2$ and $exp2$ functions with floating-point numbers requires a significant amount of hardware and incurs substantial latency. Therefore, we employed the lookup table method to implement them.

For the $log2$ function, the following transformation is applied:

$$log_2(x) = log_2\left(2^n 2^{-n} x\right) = n + log_2(2^{-n} x), \tag{16}$$

where x is a 16-bit integer input and n is the highest bit position in the binary representation of x. For example, if $x = 27$, which is "11011" in binary, then $n = 4$ and $(2^{-n}x) \in [1,2)$. Therefore, we use an 11-bit fixed-point representation to quantize $(2^{-n}x)$, generating 2048 values of $log_2(2^{-n}x)$, use a 16-bit fixed-point fraction to quantize $log_2(2^{-n}x)$, and then store these 2048 values in ROM for the lookup table.

For the $exp2$ function, the following transformation is applied:

$$2^x = 2^n \cdot 2^f = 2^f \ll n, \tag{17}$$

where x is a 20-bit fixed-point number, with the lower 16 bits representing the fractional part. n is the integer part of the x, and f is the fractional part. To reduce ROM cost, we

utilize the first 12 bits of f as input indices for the lookup table of 2^f and use a 17-bit fixed-point number to quantize 2^f, with the lower 16 bits representing the fractional part.

4.3. Base Layer Compression

In the hardware implementation of the proposed TMO, to make full use of existing information and reduce unnecessary waste of resources, the compression factor is changed to Equation (13), and the lookup table of the $log2$ function can be shared with those in Section 4.2.

$$F = \frac{T}{log_2(max(L)) - log_2(min(L))}, \tag{18}$$

where T is the contrast adjustment parameter. It is considered that the adjustment parameter L is the original luminance of the whole image. Since T is an adjustable parameter, the effect of Equation (13) is the same as that of the original TMO.

Finally, the compressed base layer is obtained by multiplying the compression factor F by each pixel of the base layer.

4.4. Normalization and Color Reproduction

After the compressed luminance image passes through the exp2 module, it needs normalization and color reproduction, and finally, it is mapped to the color image within the range of 0–255. The formula of luminance normalization is defined as follows:

$$L_{out} = \frac{L_{cp}}{max(L_{cp})}, \tag{19}$$

where L_{cp} is the luminance image after compressing the base layer and suppressing the noise of the detail layer and L_{out} is the luminance image after normalization.

The color reproduction method is similar to the Durand TMO [15], and the formula is defined as follows:

$$C_{out} = \left(\frac{C_{in}}{L_{in}}\right)^s L_{out}, \tag{20}$$

where C_{in} is the values of the input color channels, which are red, green, and blue (RGB); L_{in} is the luminance obtained by the input RGB conversion; s is the parameter to adjust the color saturation, generally taken as 1; and C_{out} is the value of the output color channel.

To maintain the luminance difference between frames, the maximum and minimum ratio of the final mapped luminance is kept consistent with the ratio of the original video image, and the final mapped to the range of 0–255 and quantized:

$$C_d = uint8(round(C_{out} \cdot (max(L_{255}) - min(L_{255})) + min(L_{255}))), \tag{21}$$

where C_d is the value of the different color channels of the final output and L_{255} is the luminance of the original HDR image after linear compression to the 0-255 range.

If the application scenario does not require maintaining the luminance contrast between frames, the maximum and minimum values of L_{255} can be directly set to 255 and 0, respectively, so that each frame of the result video sequence can make full use of the luminance range of the display.

4.5. Prediction

Some of the operations described previously require the use of statistics about the current frame, such as the maximum and minimum value, the K-th largest/smallest value, and the maximum luminance after compression and NS. In order to meet the requirements of low latency, the statistics of N-1 frames input before the current N-th frame are filtered and used as the predicted statistics for the current N-th frame. At the same time, this filter can also reduce the flicker effect caused by the image luminance changing too fast.

The simple and effective exponential filter is used to obtain the prediction of statistics for the current frame. The formula for the exponential filter is

$$PD_N = \alpha \cdot PD_{N-1} + (1 - \alpha) \cdot D_{N-1}, \tag{22}$$

where D_{N-1} is the accurate statistic of the (N-1)-th frame, PD_{N-1} is the predicted statistic of the (N-1)-th frame, PD_N is the predicted statistic of the N-th frame, and α is the adjustable parameter.

4.6. Fixed Point Implementation

The software implementation of the proposed TMO utilizes double-precision floating-point numbers. However, for FPGA or ASIC platforms, the complexity and resource costs associated with floating-point implementations led to the final decision to adopt a fixed-point approach. The input of the system is 16-bit unsigned RGB channels, and the output is 8-bit unsigned RGB channels. The fixed-point precision (Q-format) for different modules is detailed in Table 1.

Table 1. Fixed point precision for each module's main processing flow.

Module	Integral Bit	Fractional Bit	Q-Format
luma	16	0	Q16.0
cut	16	0	Q16.0
log2	4	16	Q4.16
predict	16	0	Q16.0
HLWGF	4	16	Q4.16
base compression	4	16	Q4.16
noise suppression	4	16	Q4.16
exp2	16	0	Q16.0
norm	4	12	Q4.12
color	8	12	Q8.12

5. Results

5.1. Noise Suppression

To assess the impacts of the NS introduced in this paper on image noise, we conducted tests using the HdM-HDR-2014 dataset [32], specifically selecting three test teasers: FireplaceTeaser, Showgirl2Teaser, and BeerfestTeaser. The specific evaluation method is outlined as follows: Each test teaser underwent processing using the algorithms proposed by Durand et al. [15], Lang et al. [9], and this paper. The algorithm in [9] was proposed to process luminance images and does not affect the comparison of experimental results, as the comparison of smoothness was conducted in the luminance domain. Examine the results of the 10th frame and calculate smoothness metrics for different local regions after transforming them into luminance images for comparison. Select two regions for evaluation: one in the bright to assess the impact of noise suppression on low-noise areas and one in the dark to evaluate the noise suppression capability. Additionally, 1D profiles extracted from these two regions are plotted to provide a more intuitive representation of noise and details. To maintain consistency with a single variable, the compression ratio for the first two methods was kept at 0.75, and the cut ratio of the largest and smallest values were both 0.1%. Other parameters of the algorithm in this paper are $r = 2$, $\epsilon = 2.56$, $\alpha = 0.1$, $ths_{low} = 0.02$, $\lambda = 0.18$, $e = 1$, $ths_{nv} = 0.06$, $\theta = 3$. The parameters of Lang et al. [9] are set as follows: $r = 2$, $\epsilon = 100$, $\beta = 0.75$, $\theta = 1$, $g_{min} = 1$,$g_{max} = 2.5$, $\eta = 1.5$, $\tau = 1,000,000$ (the choice of τ aims to ensure that the average luminance value of results align with other algorithms, as significant luminance differences can affect comparison of smoothness). Smoothness is defined as follows [36]:

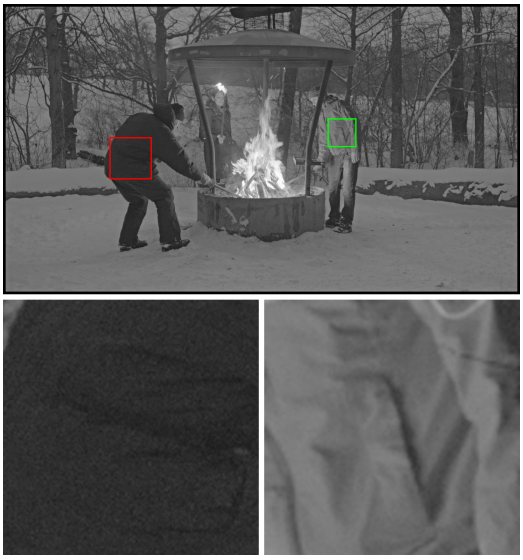
$$Smoothness = \frac{mean(I)}{std(I)}, \tag{23}$$

where $mean(I)$ and $std(I)$ represent the mean and standard deviation of the luminance I of selected regions.

Figure 9 shows the results of the 10th frame of the FireplaceTeaser being processed by three algorithms (Durand et al. [15], Lang et al. [9], and this work). The coordinates of the dark region and the bright region are (500:649, 400:549) and (430:529, 1210:1309), respectively, in Matlab. The results in Figure 9 indicate that, after being processed by different algorithms, the noise in dark regions exhibits varying degrees of amplification, while the noise in bright regions remains almost unchanged. Compared to the works of Durand et al. [15] (Figure 9b) and Lang et al. [9] (Figure 9c), the proposed TMO (Figure 9d) exhibits lower noise level in dark regions with preserved details, showcasing superior subjective perceptual effects. In bright regions, all three algorithms exhibit a comparable level of detail preservation, with specific smoothness values provided in the subsequent tables. It is evident that the NS method proposed in this paper effectively suppresses noise in dark regions while preserving details in both bright and dark regions.

(**a**) Original image.

(**b**) Durand et al.

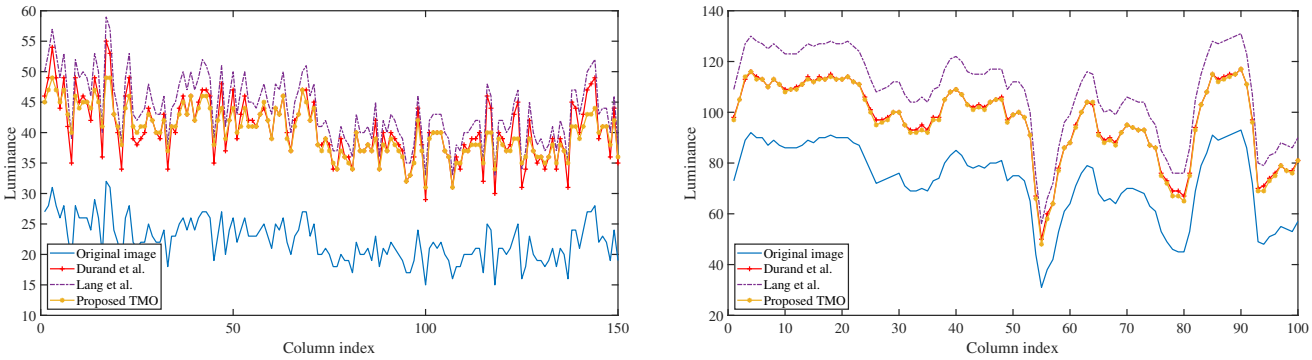


(**c**) Lang et al.

(**d**) This work.

Figure 9. Comparison of images processed by different algorithms: (**a**) Original image. (**b**) Durand et al. [15] (**c**) Lang et al. [9] (**d**) This work.

Figure 10 shows the 1D profiles of dark and bright regions in Figure 9, with the coordinates of the profiles being the middle row of the dark and bright regions. It provides a more intuitive observation. In the dark flat region, variations between signals are predominantly influenced by noise, and the proposed TMO exhibits the minimum pixel value changes, reflecting minimal noise and aligning with subjective observation and objective smoothness results. In the bright region, variations between signals are dominated by detail changes. The results from Lang et al. [9] demonstrate the highest contrast between pixel values, showcasing the most prominent details, followed by the proposed TMO and, lastly, Durand et al.'s [15].



(**a**) The 1D profiles of dark region. (**b**) The 1D profiles of bright region.

Figure 10. The 1D profiles of the zoomed-in dark and bright regions in Figure 9.

The smoothness values of the dark and bright regions of the 10th frame of FireplaceTeaser are shown in Table 2, and the test results of the other two teasers are shown in Tables 3 and 4. The coordinates of the dark region and the bright region of Showgirl2Teaser are (100:249, 250:399) and (400:549, 500:649), respectively, and those of BeerfestTeaser are (700:849, 990:1139) and (30:79, 470:519), respectively.

Table 2. The smoothness values of the 10th frame of FireplaceTeaser.

Smoothness	Original HDR Image	Durand et al. [15]	Lang et al. [9]	This Work
Dark region	5.0407	6.9238	6.9168	7.7688
Bright region	5.2403	6.7723	6.7143	6.6359

Table 3. The smoothness values of the 10th frame of Showgirl2Teaser.

Smoothness	Original HDR Image	Durand et al. [15]	Lang et al. [9]	This Work
Dark region	5.3340	6.7083	6.8288	8.7255
Bright region	7.5807	9.6034	9.5276	9.5410

Table 4. The smoothness values of the 10th frame of BeerfestTeaser.

Smoothness	Original HDR Image	Durand et al. [15]	Lang et al. [9]	This Work
Dark region	3.6156	5.0322	4.5193	5.2724
Bright region	2.8762	3.7553	3.7312	3.7144

The smoothness results are consistent with the qualitative analysis above, and the proposed TMO exhibits the lowest noise level in the dark region, resulting in maximum smoothness, and it preserves more details in the bright region with minimal smoothness. As can be seen from the table above, the improved algorithm proposed in this paper can reduce the noise level in the dark regions (improve the smoothness) and has better detail preservation in the bright regions, which is attributed to the HLWGF, which separates

the base layer from the detail layer to a greater extent. Additionally, the NS mechanism introduced can effectively suppress noise according to the characteristics of HVS.

5.2. Hardware Implementation Accuracy

We utilized two objective metrics, peak signal-to-noise ratio (PSNR) and structural similarity image index (SSIM), to evaluate the computational accuracy of the hardware implementation. The evaluation method was as follows: The algorithm parameters were defined, quantified as fixed-point numbers, and applied to the hardware system, and subsequently, Matlab's SSIM and PSNR functions were employed to compare the output results of Matlab and hardware implementation. All the frames of three different teasers (240 frames of Fireplace, 339 frames of Showgirl2, and 329 frames of Beerfest) were used as input test HDR video stream. Table 5 shows the average PSNR and SSIM values for the three teaser results, and it is evident from the data that the error between the hardware and the software implementation is very small, with average SSIM values of 0.9993, 0.9995, and 0.9997, respectively, and average PSNR values of 50.0642, 58.6609, and 55.9096, respectively.

Table 5. The SSIM and PSNR between the software- and hardware-implemented results.

Evaluation Index	Fireplace (240 Frames)	Showgirl2 (339 Frames)	BeerFest (329 Frames)
SSIM	0.9993	0.9995	0.9997
PSNR	50.0642	58.6609	55.9096

5.3. Throughput And Latency

The hardware system implemented in this paper is designed for real-time processing at 1080 P @ 60 FPS. Each clock cycle processes one pixel, and based on the resolution of 1920 × 1080 and a frame rate of 60, a minimum frequency of 124.4 MHz is required. However, with the non-continuous nature of data in standard video timing, a higher frequency is necessary. Generally, Xilinx-supported video processing modules use a clock frequency of 148.5 MHz to process videos 1080 P @ 60 FPS [27]. Therefore, the proposed system in this paper is also designed to operate at a clock frequency of 148.5 MHz. While a higher clock frequency improves the frame rate, it also leads to increased power consumption. Hence, it is crucial to carefully set an appropriate clock frequency to achieve the desired frame rate while avoiding unnecessary power consumption.

Given that one of the objectives of this system design is to minimize latency, line buffers are employed, with efforts made to reduce their quantity. In the *HLWGF*, *base compression*, and *noise suppression* modules, the window size is set to 5 × 5 for each, and each module requires 4 line buffers, leading to a total requirement of 12 line buffers. Table 6 shows the latency of different sub-modules in the system. The latency of the entire hardware system is 60.32 µs.

Table 6. Processing latency of different modules.

Module Name	Latency Cycle	Specific Latency at 148.5 MHz
Luma	2	13.47 ns
Cut	1	6.73 ns
HLWGF	4484	30.20 µs
Base compress and noise suppression	4448	29.95 µs
Lum norm	21	141.41 ns
Color recover	2	6.73 ns
TOTAL	8958	60.32 µs

As the system latency is related to the algorithms in the system, Tables 7 and 8 are provided to compare the latency and the smoothness metric before and after the improvements, considering a total of four scenarios that involve different combinations of

the HLWGF and NS proposed in this work. The data in Table 8 present the test results of the 10th frame in the FireplaceTeaser.

Table 7. The latency of different solution combinations.

Solution Combinations	Latency (Cycle)	Specific Latency Time (μs)
Proposed TMO with GF and NS	13,282	89.44
Proposed TMO with HLWGF and NS	8958	60.32
Proposed TMO with GF and without NS	8834	59.49
Proposed TMO with HLWGF and without NS	4510	30.37

Table 8. The smoothness of different solution combinations.

Solution Combinations	Smoothness (Dark)	Smoothness (Bright)
Proposed TMO with GF and NS	8.1518	6.6971
Proposed TMO with HLWGF and NS	7.8686	6.6403
Proposed TMO with GF and without NS	6.4899	6.4908
Proposed TMO with HLWGF and without NS	6.6107	6.6246

Analysis of Table 7 reveals a notable reduction in system latency with the implementation of the proposed HLWGF. Additionally, Table 8 demonstrates enhanced smoothness in flat dark regions and superior preservation of details in bright regions. The selection of a specific solution should be guided by practical considerations.

Compared to other hardware implementations of TMO in recent years, the latency of the system implemented in this paper is still best. Table 9 shows the latency of the TMO proposed in this paper compared to other hardware implementations.

Table 9. Latency comparison of FPGA-implementations of proposed TMO against other TMOs.

TMOs	Image Size	Clock (Mhz)	Latency (ms)
Park et al. [27]	1920 × 1080	148.5	0.24100
Muneer et al. [37]	1920 × 1080	200	6.96000
This work	1920 × 1080	148.5	0.06032

5.4. Hardware Resource Cost

We used the Xilinx ZCU102 evaluation board for development, utilizing Vivado software version 2021.2. Table 10 illustrates the synthesized resource cost of the system. The utilization of LUT and Register resources is primarily attributed to the dividers used in the system, amounting to 21649 and 34918, respectively. DSP resources are predominantly contributed by the HLWGF module, totaling 188. With the improvement of the algorithm, the memory required for this study is reduced, only necessitating 1026 Kb/48 Block RAMs.

Table 10. Resource cost report after synthesis.

Device	LUT	Register	Memory (Kb)/ Block RAM	DSP
xc7vx485tffg1157-1	21649 (7.13%)	34918 (5.75%)	1026/48 (4.66%)	188 (6.71%)

6. Discussion

In the methods and experimental results above, we compared our TMO with prior studies. Firstly, the proposed HLWGF effectively reduces halo artifacts when the GF is utilized for image enhancement or TM tasks with large regularization parameter ϵ, as compared to the original GF. Secondly, we employed a smoothness metric to assess the noise suppression capability of our TMO. Compared to similar hardware-implemented TMOs, our NS method achieves superior noise suppression in dark flat regions, enhancing

the smoothness metric by at least 10% and up to 20% in test scenarios. Additionally, it can be finely optimized through parameter adjustments. Concurrently, our TMO effectively preserves edge details in both dark and bright regions. Thirdly, we also compare the latency before and after the hardware optimizations, as well as with other hardware-implemented TMOs in Section 5.3. Results show that, despite adding NS, system latency remains nearly unchanged, attributed to tailored hardware-specific algorithm optimizations. Our system also exhibits lower latency compared to other hardware TMOs.

We proposed the novel HLWGF, which effectively preserves large edges in images compared to the original GF, offering new possibilities for filtering tasks and image enhancement. Additionally, we proposed a noise visibility assessment method based on characteristics of the human visual system, adaptable to various image contents and convenient for hardware implementation, providing a practical option for both software- and hardware-based image denoising tasks. For practice, our study combines hardware optimizations, offering an effective hardware TMO solution for noise-sensitive embedded visual applications.

In more detail, the local weighting mechanism in the HLWGF adjusts pixel weights based on their differences in value within the window. The larger the differences between pixel values and the center pixel, the more likely they belong to different sides of an edge. Consequently, the weights of the corresponding pixels are reduced to enhance edge preservation. Additionally, the proposed novel noise visibility assessment method comprehensively considers the influence of brightness and edges on human visual perception. Specifically, human eyes are more sensitive to brightness changes in dark backgrounds and flat regions. Based on this effect, the corresponding computational formula is designed to effectively evaluate the visibility of noise in images. However, these two methods increase the hardware resources needed for hardware implementation, requiring more multiplication and division operations. This can limit the usage of large integrated systems with a TMO on smaller FPGAs.

7. Conclusions

In this paper, we proposed a low-latency noise-aware TMO tailored for hardware implementation to meet the demands of noise-sensitive embedded vision applications. Firstly, we enhanced the edge-preserving capability of the guided filter under large regularization parameter ϵ by introducing a weighted mechanism, addressing the issue of halo artifacts commonly observed in image enhancement or TMO applications. Secondly, a noise suppression method was introduced in the TMO, accompanied by a novel noise visibility assessment method based on two human visual characteristics. Compared to similar TMOs, our approach achieved an improvement of over 20% in noise suppression. Thirdly, the algorithm for finding the K-th largest/smallest value and the HLWGF were optimized for hardware implementation, enabling the TMO to handle continuous video stream inputs in hardware and reducing the latency and line buffer by $\frac{1}{3}$ and $\frac{2}{5}$ respectively. The final hardware system was implemented on an FPGA, demonstrating a latency of 60.32 µs at 1080 P @ 60 FPS, validating the effectiveness of the algorithm.

The TMO presented in this paper does not enhance other aspects of visual quality in the tone-mapped images, such as contrast. Improvement can be achieved by exploring alternative types of TMOs, such as those based on histogram adjustments. This research, however, is subject to several limitations. For instance, the introduced HLWGF and NS require additional multiplication and division operations, increasing the utilization of certain hardware resources (such as DSP) in the system. This may restrict the deployment of large integrated systems containing TMO on smaller FPGAs. Future research may focus on innovative hardware design methodologies to minimize hardware resource costs.

Author Contributions: Conceptualization, Q.L.; data curation, Q.L.; formal analysis, Q.L.; investigation, Q.L.; methodology, Q.L.; project administration, J.Y.; resources, Q.L.; software, Q.L. and T.Y.; supervision, J.Y.; validation, Q.L. and T.Y.; visualization, Q.L.; writing—original draft, Q.L.;

writing—review & editing, N.W., Z.Z. and J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HDR	High dynamic range
TMO	Tone mapping operator
GPU	Graphics processing unit
HVS	Human visual system
NS	Noise suppression
FPGA	Field-programmable gate array
ASIC	Application-specific integrated circuit
GF	Guided filter
TM	Tone mapping
FPS	Frames per second
HGF	Hardware-oriented guided filter
HLWGF	Hardware-oriented locally weighted guided filter
RGB	Red, green, blue
FIFO	First in first out

References

1. Tohidypour, H.R.; Wang, Y.; Pourazad, M.T.; Nasiopoulos, P.; Zhao, D.; Xie, M.; Kamat, D. Investigating Suitability of Inverse Tone Mapping for Medical Images. In Proceedings of the 2023 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 6–8 January 2023; pp. 1–2. [CrossRef]
2. Shopovska, I.; Aelterman, J.; Van Hamme, D.; Philips, W. Low-Complexity Deep HDR Fusion And Tone Mapping for Urban Traffic Scenes. In Proceedings of the 2023 IEEE Intelligent Vehicles Symposium (IV), Anchorage, AK, USA, 4–7 June 2023; pp. 1–6. [CrossRef]
3. Barman, N.; Martini, M.G. User Generated HDR Gaming Video Streaming: Dataset, Codec Comparison, and Challenges. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 1236–1249. [CrossRef]
4. Ou, Y.; Ambalathankandy, P.; Takamaeda, S.; Motomura, M.; Asai, T.; Ikebe, M. Real-Time Tone Mapping: A Survey and Cross-Implementation Hardware Benchmark. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 2666–2686. [CrossRef]
5. Cao, C.; Yue, H.; Liu, X.; Yang, J. Unsupervised HDR Image and Video Tone Mapping via Contrastive Learning. *IEEE Trans. Circuits Syst. Video Technol.* **2024**, *34*, 786–798. [CrossRef]
6. Shopovska, I.; Stojkovic, A.; Aelterman, J.; Van Hamme, D.; Philips, W. High-Dynamic-Range Tone Mapping in Intelligent Automotive Systems. *Sensors* **2023**, *23*, 5767. [CrossRef] [PubMed]
7. Zhang, J.; Wang, Y.; Tohidypour, H.; Pourazad, M.T.; Nasiopoulos, P. A Generative Adversarial Network Based Tone Mapping Operator for 4K HDR Images. In Proceedings of the 2023 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 20–22 February 2023; pp. 473–477. [CrossRef]
8. Upadhyay, B.B.; Sarawadekar, K. A Low Cost FPGA Implementation of Retinex Based Low-Light Image Enhancement Algorithm. *IEEE Trans. Circuits Syst. II Express Briefs* 2024, *early access*. [CrossRef]
9. Lang, Y.Z.; Qian, Y.S.; Wang, H.G.; Kong, X.Y.; Wu, S. A real-time high dynamic range intensified complementary metal oxide semiconductor camera based on FPGA. *Opt. Quantum Electron.* **2022**, *54*, 304. [CrossRef]
10. Kashyap, S.; Giri, P.; Bhandari, A.K. Logarithmically Optimized Real-Time HDR Tone Mapping With Hardware Implementation. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *71*, 1426–1430. [CrossRef]
11. Fang, X.; Feng, X. Domain-Aware Adaptive Logarithmic Transformation. *Electronics* **2023**, *12*, 1318. [CrossRef]
12. Zhang, F.; Dai, Y.; Peng, X.; Wu, C.; Zhu, X.; Zhou, R.; Wu, Y. Brightness segmentation-based plateau histogram equalization algorithm for displaying high dynamic range infrared images. *Infrared Phys. Technol.* **2023**, *134*, 104894. [CrossRef]
13. Zhao, L.; Li, G.; Wang, J. Tone Mapping Method Based on the Least Squares Method. *Electronics* **2023**, *12*, 31. [CrossRef]
14. Li, Y.; Liao, N.; Wu, W.; Deng, C.; Li, Y.; Fan, Q.; Liu, C. Tone Mapping Operator for High Dynamic Range Images Based on Modified iCAM06. *Sensors* **2023**, *23*, 2516. [CrossRef]
15. Durand, F.; Dorsey, J. Fast bilateral filtering for the display of high-dynamic-range images. In Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, San Antonio, TX, USA, 23–26 July 2002; pp. 257–266.

16. Wang, C.; Chen, B.; Seidel, H.; Myszkowski, K.; Serrano, A. Learning a self-supervised tone mapping operator via feature contrast masking loss. *Comput. Graph. Forum* **2022**, *41*, 71–84. [CrossRef]
17. Zhang, Z.; Jiang, Y.; Jiang, J.; Wang, X.; Luo, P.; Gu, J. STAR: A Structure-Aware Lightweight Transformer for Real-Time Image Enhancement. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 4106–4115.
18. Liu, S.; Liu, L.; Tang, J.; Yu, B.; Wang, Y.; Shi, W. Edge Computing for Autonomous Driving: Opportunities and Challenges. *Proc. IEEE* **2019**, *107*, 1697–1716. [CrossRef]
19. Jha, D.; Tomar, N.K.; Ali, S.; Riegler, M.A.; Johansen, H.D.; Johansen, D.; de Lange, T.; Halvorsen, P. NanoNet: Real-Time Polyp Segmentation in Video Capsule Endoscopy and Colonoscopy. In Proceedings of the 2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS), Aveiro, Portugal, 7–9 June 2021; pp. 37–43. [CrossRef]
20. Eilertsen, G.; Mantiuk, R.K.; Unger, J. A comparative review of tone-mapping algorithms for high dynamic range video. *Comput. Graph. Forum* **2017**, *36*, 565–592. [CrossRef]
21. Wang, Y.; Jing, Z.; Ji, Z.; Wang, L.; Zhou, G.; Gao, Q.; Zhao, W.; Dai, S. Lane Detection Based on Two-Stage Noise Features Filtering and Clustering. *IEEE Sens. J.* **2022**, *22*, 15526–15536. [CrossRef]
22. Gödrich, A.; König, D.; Eilertsen, G.; Teutsch, M. Joint tone mapping and denoising of thermal infrared images via multi-scale Retinex and multi-task learning. In Proceedings of the Infrared Technology and Applications XLIX SPIE, Orlando, FL, USA, 30 April–4 May 2023; Volume 12534, pp. 275–291.
23. Xiang, X.; Liu, L.; Que, L.; Jia, C.; Yan, B.; Li, Y.; Guo, J.; Zhou, J. A biological retina inspired tone mapping processor for high-speed and energy-efficient image enhancement. *Sensors* **2020**, *20*, 5600. [CrossRef]
24. Kim, Y.; Hwang, H.; Shin, J. Robust object detection under harsh autonomous-driving environments. *IET Image Process.* **2022**, *16*, 958–971. [CrossRef]
25. Reinhard, E.; Stark, M.; Shirley, P.; Ferwerda, J. Photographic tone reproduction for digital images. In *Seminal Graphics Papers: Pushing the Boundaries*; ACM: New York, NY, USA, 2023; Volume 2, pp. 661–670.
26. Nosko, S.; Musil, M.; Zemcik, P.; Juranek, R. Color HDR video processing architecture for smart camera. *J. Real-Time Image Process.* **2020**, *17*, 555–566. [CrossRef]
27. Park, J.W.; Lee, H.; Kim, B.; Kang, D.G.; Jin, S.O.; Kim, H.; Lee, H.J. A low-cost and high-throughput FPGA implementation of the retinex algorithm for real-time video enhancement. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *28*, 101–114. [CrossRef]
28. Ambalathankandy, P.; Ikebe, M.; Yoshida, T.; Shimada, T.; Takamaeda, S.; Motomura, M.; Asai, T. An Adaptive Global and Local Tone Mapping Algorithm Implemented on FPGA. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 3015–3028. [CrossRef]
29. Eilertsen, G.; Mantiuk, R.K.; Unger, J. Real-time noise-aware tone mapping. *ACM Trans. Graph. (TOG)* **2015**, *34*, 1–15. [CrossRef]
30. Hu, L.; Chen, H.; Allebach, J.P. Joint multi-scale tone mapping and denoising for HDR image enhancement. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 4–8 January 2022; pp. 729–738.
31. Ray, S.S.; Ghosh, S. k-Degree Parallel Comparison-Free Hardware Sorter for Complete Sorting. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *42*, 1438–1449.
32. Froehlich, J.; Grandinetti, S.; Eberhardt, B.; Walter, S.; Schilling, A.; Brendel, H. Creating cinematic wide gamut HDR-video for the evaluation of tone mapping operators and HDR-displays. In Proceedings of the Digital Photography X, San Francisco, CA, USA, 2–6 February 2014; Sampat, N., Tezaur, R., Battiato, S., Fowler, B.A., Eds.; International Society for Optics and Photonics SPIE: Bellingham, WA, USA, 2014; Volume 9023, p. 90230X. [CrossRef]
33. He, K.; Sun, J.; Tang, X. Guided Image Filtering. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1397–1409. [CrossRef]
34. Pu, X.; Yang, K.; Li, Y. A Retinal Adaptation Model for HDR Image Compression. In Proceedings of the Computer Vision, Tianjin, China, 11–14 October 2017; Yang, J., Hu, Q., Cheng, M.M., Wang, L., Liu, Q., Bai, X., Meng, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; pp. 37–47.
35. Gu, B.; Li, W.; Zhu, M.; Wang, M. Local edge-preserving multiscale decomposition for high dynamic range image tone mapping. *IEEE Trans. Image Process.* **2012**, *22*, 70–79.
36. Lee, J.W.; Park, R.H.; Chang, S. Noise reduction and adaptive contrast enhancement for local tone mapping. *IEEE Trans. Consum. Electron.* **2012**, *58*, 578–586. [CrossRef]
37. Muneer, M.H.; Pasha, M.A.; Khan, I.R. Hardware-friendly tone-mapping operator design and implementation for real-time embedded vision applications. *Comput. Electr. Eng.* **2023**, *110*, 108892. [CrossRef]