HARDWARE-EFFICIENT DEBANDING AND VISUAL ENHANCEMENT FILTER FOR INVERSE TONE MAPPED HIGH DYNAMIC RANGE IMAGES AND VIDEOS

Qing Song*† *Guan-Ming Su*† *Pamela C. Cosman**

* UC San Diego, Dept. of Electrical and Comp. Engr., 9500 Gilman Dr, La Jolla, CA 92093-0407
† Dolby Laboratories, Inc., 432 Lakeside Drive, Sunnyvale, CA 94085

ABSTRACT

If a low dynamic range (LDR) image or video is inverse tone mapped to a higher dynamic range, there can be banding artifacts in the output high dynamic range (HDR) image or video. We design a selective sparse filter to remove the banding artifacts and at the same time preserve edges and details. The filter is able to reduce other artifacts, such as blocky artifacts which are due to the compression of the LDR image/video. The filter is computationally efficient.

Index Terms— Sparse filter, high dynamic range, false contour removal, debanding, visual enhancement

1. INTRODUCTION

In recent years there is a growing interest in HDR images and videos as well as HDR displays. HDR, represented in 12+ bit-depth (i.e., 12 or more bits per color component), provides a wider range of brightness and a larger color gamut than the traditional 8-bit LDR, and is thus closer to the human visual system. However, the majority of existing displays only support 8-bit image and video content. In order to make the HDR content compatible with typical LDR displays, the HDR content is usually converted to LDR by some tone mapping operator (TMO). Conversely, even today there is limited content captured in HDR, since HDR devices are not widely spread yet. So it is necessary to apply an inverse tone mapping operator (iTMO) to the LDR content for presenting on HDR displays. This area has been investigated in [1, 2, 3, 4].

It is observed that the HDR generated by iTMO usually suffers from false contours which are also referred to as banding artifacts and ringing artifacts. It happens especially when iTMO is a one-to-one mapping function. Banding artifacts usually occur in smooth gradient regions. The cause of the artifacts is that 8-bit LDR has at most 256 codewords (there are only 220 codewords in Rec. 601 [5] and Rec. 709 [6]), and after mapping the HDR also has 256 codewords. But the 16 bit-depth HDR has a total of 65,536 codewords. Lack of codewords in the inverse tone mapped HDR results in the banding artifacts which are visually annoying.

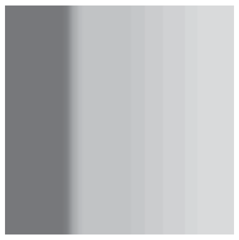
To remove the banding artifacts, i.e., debanding or decontouring, some suggested dithering [7, 8, 9], but the output is often not visually pleasant either. Those works aim at producing the output at the same bit-depth as the original image, so there is no new codeword available. But in our case, only a few codewords have been utilized in the HDR image generated by iTMO, so there is extra room to generate new codewords to smooth the banding artifacts. In [10], this is achieved by linear interpolation in the banding area after the banding width is identified by median filtering. The banding area can be detected by the algorithm in [11]. Some people apply low pass filters to increase codewords. Daly and Feng [12] proposed to predict and reduce false contours by low pass filtering and quantization. In [13], false contours are reduced by 1D directional smoothing filters whose directions are orthogonal to the false contours. This method requires high computational complexity to detect false contours.

In this paper, we propose a selective sparse filter which combines smooth region detection and banding reduction. It was originally designed to remove the banding artifacts, but later we found it can also reduce some coding artifacts, such as blocky artifacts. The properties of the inverse tone mapped HDR are exploited in the filter design. The selection of some parameters of our filter is based on a piecewise iTMO which is proposed in [4] and [14]. But our filter can be extended to other iTMO algorithms with a few changes. We aim at implementing the filter in hardware.

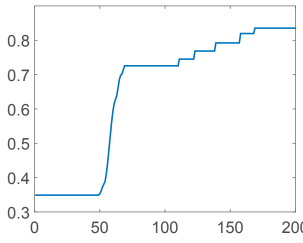
The rest of the paper is organized as follows: In Sec. 2 we describe our proposed filter and discuss the parameter selection. In Sec. 3, we show the experimental results. Sec. 4 concludes the paper.

2. PROPOSED EDGE-AWARE SPARSE FILTER

Banding artifacts usually occur in the regions of small gradient, and the artifacts appear stepwise. Fig. 1a shows an example of vertical banding in the right part of the image. We take one row of pixels and plot the values in Fig. 1b. The values are normalized to $[0, 1]$. The steps are where the banding artifacts lie. To remove the artifacts, we want to smooth the area by adding more codewords between each step. One simple method is to apply a low pass filter. A traditional dense 2D FIR filter can be represented as: $y[m, n] =$



(a)



(b)

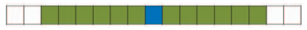
Fig. 1: Example of banding artifacts.

$\frac{1}{(2u+1)(2v+1)} \sum_{i=-u}^{u} \sum_{j=-v}^{v} x[m+i, n+j]$, where $x[m,n]$ is the input signal at row m and column n, and $y[m,n]$ is the corresponding output signal. The filter averages a total of $(2u+1)(2v+1)$ input pixels centered at $x[m,n]$. The 2D filtering is equivalent to applying a 1D $(2v+1)$-tap horizontal filter and a 1D $(2u+1)$-tap vertical filter sequentially which is much more efficient. For simplicity, we use the 1D filters with the same number of taps and define only the horizontal 1D filter in the following.

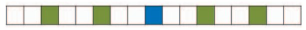
In order to remove banding, we have to apply a filter whose span is wide enough. Fig. 2c shows a signal with many steps. Fig. 2d-2f show the outputs of the dense filter with different number of taps. The 9-tap filter is not able to remove the false contours as there are still many steps left. The 29-tap filter works better. The banding is almost gone when the filter has 49 taps. When the span of the filter is not wide enough, many consecutive pixels of the output will have the same codeword, because the pixels taken for averaging are all on the same step. If the step size is uniform, the false contours can be completely smoothed out when the span of the filter is $2w-1$, where w is the width of each step. That means, when the step is wide, we would have to increase the span of the filter, i.e., increase the number of taps of the filter. To implement the filtering in hardware, we need to put each row of pixels into one line buffer for vertical filtering. That is, the filtering module would need one line buffer for each tap of the filter. The cost of the dense filter is too high.

2.1. Sparse filter

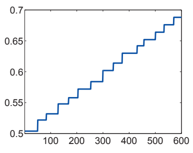
From the observations above, we learn that to remove banding, the key is to get samples from different steps. A sparse filter would be more efficient to achieve that. A simple 1D sparse FIR filter is defined as: $y[n] = \frac{1}{2u+1} \sum_{i=-u}^{u} x[n+s_i]$, where s_i is the distance from the original pixel to the sampled input signal. The number of taps is $2u+1$. Fig. 2b shows a 5-tap sparse filter with the same span as the 13-tap dense filter in Fig. 2a. The origin is marked by blue. Fig. 2g shows the filtering results by a 5-tap sparse filter whose span is 29. Though the output of a sparse filter has some ripples thus would not be as smooth as the output of a dense filter with enough taps, the interpolated codewords between each



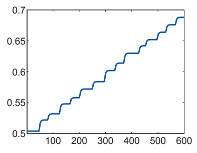
(a) 13-tap dense filter



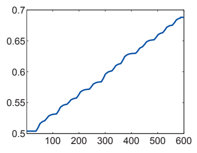
(b) 5-tap sparse filter, span = 13



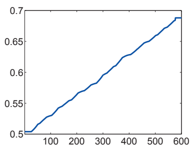
(c) original signal



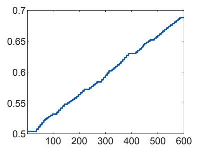
(d) 9-tap dense filter



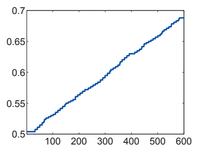
(e) 29-tap dense filter



(f) 49-tap dense filter



(g) 5-tap sparse filter, span = 29



(h) 5-tap sparse filter, span = 37

Fig. 2: Performance of dense filter and sparse filter.

two steps can be sufficient to make the banding unnoticeable to human eyes. Though the memory traffic is increased, the required memory size of the sparse filter is much smaller than the dense filter, because the 5-tap sparse filter only requires 5 line buffers, while the dense filter requires more than 29. We also need fewer adders and multipliers for the sparse filtering.

From our preliminary tests, we found that 5 taps are usually enough for the sparse filter to remove the banding artifacts. Note that increasing the span of the sparse filter may make the signal smoother, but would not increase the cost.

2.2. Edge-aware selective filter

It is clear that sparse FIR filters can help remove the false contours, but they can also blur true edges and remove details. Edge-preserving filters, such as the bilateral filter [15, 16, 17] and guided filter [18], can preserve edges, but they are dense filters essentially which require high memory cost. To address this issue, we propose to apply the sparse filter selectively: we want to apply the filter to the smooth areas only. On one hand, banding is only observed at smooth areas. On the other hand, smoothing a smooth area would not cause a loss of many details even if there is no banding in the area.

Fig. 3 shows the flowchart of our proposed filter. The original image means the HDR image mapped from the LDR image by iTMO. For each pixel $x[n]$ in the original image, we sample three pixels on its left and right side respectively. The positions of the sampled pixels are denoted as $n+s_i$ where $i \in \{-3,-2,-1,1,2,3\}$. For simplicity, $x[n]$ is also denoted as $x[n+s_0]$ where $s_0 = 0$. We compute the difference between the central pixel $x[n]$ and each of the sampled pixels $x[n+s_i]$ where $i \neq 0$. If the absolute value of the difference is below a threshold Δ, we determine that the sampled pixel has a similar value to the central pixel. If the central pixel and all the sampled pixels have similar values, we determine the

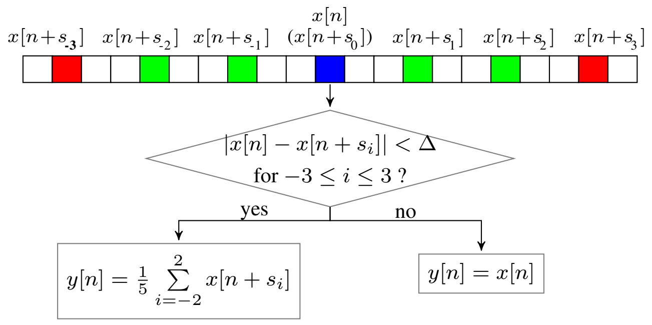


Fig. 3: Flowchart of proposed edge-aware sparse filter.

area as a smooth area, and we apply the sparse filter to the central pixel. The sparse filter takes only five inputs: $x[n+s_{-2}], x[n+s_{-1}], x[n], x[n+s_1], x[n+s_2]$. The output $y[n]$ is the average of them. However, if the difference between the central pixel and any of the sampled pixels is greater than the threshold, there may be edges or texture in the area. Then the sparse filter is not applied, and the original pixel value remains unchanged: $y[n] = x[n]$.

This selective filter combines edge detection and sparse filtering. It only requires one line buffer for the horizontal filtering and seven line buffers for the vertical filtering. The selection of the threshold Δ for the selective sparse filter is critical to the debanding performance. Note that the filtering process only takes five pixels, and the extra two pixels are for edge detection. As mentioned above, banding artifacts are in the smooth regions, and would not occur near object boundaries. Therefore, we do not want to involve pixels near boundaries. In the following, we first describe how to select the threshold, then explain why the extra two pixels are necessary for the edge and details preservation.

2.2.1. *Adaptive threshold*

The threshold Δ indicates how much difference we want to tolerate in the decision process. If Δ is too small, we will only average pixels with small differences, and thus small areas would be filtered. In other words, the banding artifacts may not be removed. If Δ is too large, the filter would be applied to areas with sharp edges and details, which leads to blurred edges and loss of details.

One important observation of the banding areas is that the codewords of the corresponding pixels in the LDR image are very similar to each other. The difference of the LDR codewords between the neighboring pixels is 1 or 2 most of the time. After the inverse tone mapping, the difference between these neighboring pixels shown on a HDR display becomes larger and that results in the banding artifacts. Since the input of our filter is the inverse tone mapped HDR, Δ can be related to the mapping function.

Assume that the codeword of a pixel in LDR is b where $0 \leq b \leq 255$ for an 8-bit LDR. The corresponding inverse tone mapped HDR codeword is $T(b)$. The difference between two neighboring HDR codewords is denoted as $dT(b) = |T(b+1) - T(b)|$. In the HDR image, if an area is relatively smooth, we expect the difference among the nearby pixels to be roughly within the range of a small number times $dT(b)$. If a pixel is in a textured area, the difference among the nearby pixels could be much larger than that.

Since our selection of Δ depends on the iTMO function, we would like to select one iTMO algorithm from the existing algorithms for discussion. The piecewise polynomial iTMO in [4] is selected. Assume that there are K segments in the iTMO curve in total. The pivot points, i.e., the points at the boundary of segments, are denoted as s_k where $1 \leq k \leq K$. The differential function $dT(b)$ is partitioned into $K+1$ segments. The mapping slope of each segment can be very different, so selecting different thresholds is necessary for different segments. An empirical result shows that the following gives a good trade-off between the removal of banding and the preservation of texture and edges. When the codeword of the central pixel is $T(b)$, the threshold

$$\Delta = \alpha \cdot \max_{s_k \leq T(b) < s_{k+1}} \{dT(b)\}, \quad (1)$$

where α is positive. In other words, the threshold depends on which segment the pixel lies in. The threshold is set to the maximum differential of the segment multiplied by a factor α. In our test, it usually works well when α is 2 or 3.

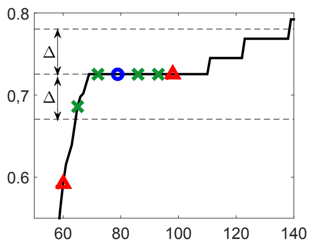
Though the discussion above is based on the piecewise iTMO, this method can be extended to other iTMO algorithms. The differential function $dT(\cdot)$ of any one-to-one mapping can be built. The point is that we only allow averaging a few codewords. So the threshold can be set to $\alpha \cdot dT(b)$, or $\alpha \cdot \max_{0 \leq b < 255}\{dT(b)\}$.

2.2.2. *Extra samples for edge preservation*

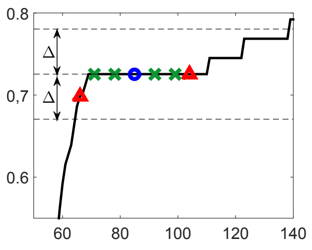
We include seven pixels for edge detection but only take five pixels for filtering. The reason why we add two more samples for detection than for filtering is that we observed false ringing introduced to the output image when only five pixels are used for both detection and filtering. The false ringing usually occurs near edges.

We again use the patch in Fig. 1a to explain why it happens. It is clear there is an edge between the dark and bright regions, and there are banding artifacts in the bright region. Our goal is to preserve the dark region and the edge, and smooth the banding on the right part. We set the filter parameters $s_2 = -s_{-2} = 14$, $s_1 = -s_{-1} = 7$, and set the threshold Δ to $2H$ where H is the maximum difference between each two steps. Fig. 4a shows the mid region of Fig. 1b. We want to determine whether we want to apply the sparse filter to the pixel marked by blue circle. The range of $[x[n]-\Delta, x[n]+\Delta]$ is marked by dash lines.

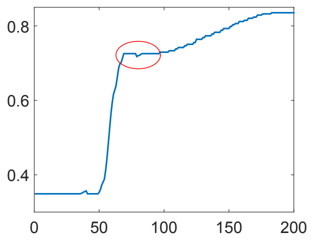
Now assume that we only sample the four pixels marked by green crosses, and we determine to apply the filtering when the four samples have similar values to the central pixel. In



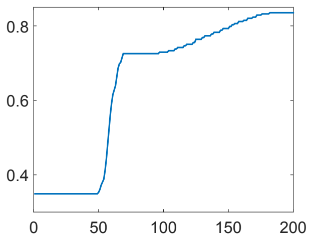
(a) original signal



(b) original signal



(c) filtered signal: 5-pixel edge detection



(d) filtered signal: 7-pixel edge detection

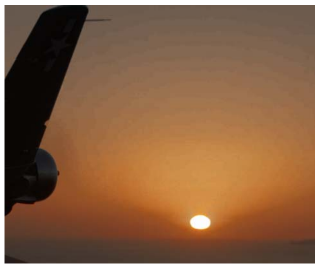
Fig. 4: Comparison between 5-sample and 7-sample edge detection.

this case the filtering condition would be satisfied. However, the leftmost green cross sample is actually at a transition area. That sample is an outlier, whose value is slightly different from the others though the difference is still within the threshold. The average of the five pixels would be slightly lower than the original value which brings an undershoot (marked by the red circle in Fig. 4c). In the image, it will appear as a faint false ringing. It may not be very noticeable on a LDR display, but it can be annoying on a HDR display.

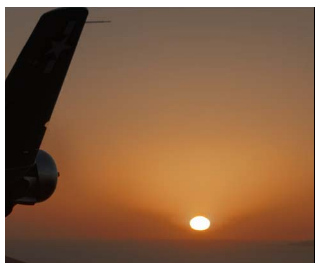
Now we consider two more samples which are marked by red triangles in Fig. 4a, and we apply the filter only when all the six samples have similar values as the central pixel. In this case, the difference between the central pixel and the leftmost sample will exceed the threshold, so the filtering will not be applied. For the pixel marked by a blue circle in Fig. 4b, all the samples marked by green crosses and red triangles are within the threshold. We apply the sparse filtering by averaging only the five central pixels, not all seven pixels. So we exclude the leftmost sample which is an outlier from the averaging. The filtering result is shown in Fig. 4d. The undershoot is not introduced, the banding artifacts are smoothed, and the edge is well preserved. Therefore, we prevent introducing new artifacts by sampling two more pixels to probe if there is an edge nearby.

3. EXPERIMENTAL RESULTS

The proposed filter is verified by visual evaluation of ten sample video clips frame by frame on the Dolby Pulsar HDR monitor. One example of the performance is shown in Fig. 5. There are severe banding artifacts in the sky above the sun in the original image Fig. 5a. We apply the proposed filter with $s_1 = -s_{-1} = 12$, $s_2 = -s_{-2} = 24$, $s_3 = -s_{-3} = 27$, and $\alpha = 3$. The filtered image is shown in Fig. 5b. The artifacts are smoothed out, and the edges and details are well preserved. We apply the filter in the YCbCr gamma color space. Note that this filter can be applied to a single color component (e.g., luma), or more components (e.g., chroma).



(a) original HDR



(b) debanded HDR

Fig. 5: Results. Note that banding artifacts in (a) are more noticeable on a screen than on paper.

In our tests with many HD videos, we found it best to use a symmetric filter ($s_1 = -s_{-1}, s_2 = -s_{-2}, s_3 = -s_{-3}$), and to set equal distance between neighboring samples in the central five pixels ($s_2 = 2s_1$). The span of the filter depends on the width of banding. When the banding is very wide, we need to increase the span of the filter. The farthest samples on both ends are several pixels away from the next samples. We obtained good results when $s_3 = s_2 + 5$ for nine out of ten video clips. For the other clip, we found it better to set $s_3 = s_2 + 3$ so that more pixels could be filtered.

An interesting observation is that if there are coding artifacts (like blocky artifacts) in the video due to low bit rate compression, our algorithm is able to remove, or at least reduce them. That is because pixels in the regions with blocky artifacts usually have similar values, and our filter can smooth them out. Note that pixels in those regions can have different gradient directions, so the algorithms using directional features to detect and reduce artifacts may not work well. Our algorithm does not depend on the direction or gradient, so it is also effective for the blocky artifacts.

4. CONCLUSION

We proposed an edge-aware selective sparse filter to remove banding artifacts and reduce some coding artifacts in inverse tone mapped HDR videos and images. Meanwhile, the filter is able to preserve edges and details. It combines edge detection and filtering. No banding map or filtering map is required to store in memory. We described how to select the threshold Δ using the inverse tone mapping function. The filter can be implemented in hardware efficiently. We provided empirical settings of the filter parameters. How to select the parameters automatically would be interesting to explore in the future.

5. REFERENCES

[1] A. G. Rempel, M. Trentacoste, H. Seetzen, H. David Young, W. Heidrich, L. Whitehead, and G. Ward, “Ldr2Hdr: on-the-fly reverse tone mapping of legacy video and photographs,” in *ACM SIGGRAPH*, 2007.

[2] F. Banterle, P. Ledda, K. Debattista, and A. Chalmers, “Inverse tone mapping,” in *Proc. the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, New York, NY, USA, 2006, pp. 349–356.

[3] P.-H. Kuo, C.-S. Tang, and S.-Y. Chien, “Content-adaptive inverse tone mapping,” in *Proc. IEEE International Conference on Visual Communications and Image Processing (VCIP)*, Nov 2012, pp. 1–6.

[4] Q. Chen, G.-M. Su, and P. Yin, “Near constant-time optimal piecewise LDR to HDR inverse tone mapping,” in *Proc. SPIE*, 2015, vol. 9404, pp. 94040O–11.

[5] Recommendation ITU-R BT.601-7, “Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios,” March 2011.

[6] Recommendation ITU-R BT.709-6, “Parameter values for the HDTV standards for production and international programme exchange,” June 2015.

[7] W. Ahn and J.-S. Kim, “Flat-region detection and false contour removal in the digital TV display,” in *Proc. IEEE International Conference on Multimedia and Expo*, July 2005, pp. 1338–1341.

[8] S. Bhagavathy, J. Llach, and J. Zhai, “Multi-scale probabilistic dithering for suppressing banding artifacts in digital images,” in *Proc. IEEE International Conference on Image Processing*, Sept 2007, vol. 4, pp. IV–397 – IV–400.

[9] K. Yoo, H. Song, and K. Sohn, “In-loop selective processing for contour artefact reduction in video coding,” *Electronics Letters*, vol. 45, no. 20, pp. 1020–1022, September 2009.

[10] G.-M. Su, T. Chen, P. Yin, and S. Qu, “Guided post-prediction filtering in layered VDR coding,” Nov. 25 2014, US Patent 8,897,581 B2.

[11] G.-M. Su, S. Qu, and S. Daly, “Adaptive false contouring prevention in layered coding of images with extended dynamic range,” Oct. 28 2014, US Patent 8,873,877 B2.

[12] S. J. Daly and X. Feng, “Decontouring: prevention and removal of false contour artifacts,” in *Proc. SPIE*, 2004, vol. 5292, pp. 130–149.

[13] J. W. Lee, B. R. Lim, R.-H. Park, J.-S. Kim, and W. Ahn, “Two-stage false contour detection using directional contrast and its application to adaptive false contour reduction,” *IEEE Transactions on Consumer Electronics*, vol. 52, no. 1, pp. 179–188, Feb 2006.

[14] Z. Mai, H. Mansour, R. Mantiuk, P. Nasiopoulos, R. Ward, and W. Heidrich, “Optimizing a tone curve for backward-compatible high dynamic range image and video compression,” *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1558–1571, June 2011.

[15] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *Sixth International Conference on Computer Vision*, Jan 1998, pp. 839–846.

[16] F. Porikli, “Constant time O(1) bilateral filtering,” in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2008, pp. 1–8.

[17] T. Q. Pham and L. J. van Vliet, “Separable bilateral filtering for fast video preprocessing,” in *IEEE International Conference on Multimedia and Expo*, July 2005, pp. 4–7.

[18] K. He, J. Sun, and X. Tang, “Guided image filtering,” in *11th European Conference on Computer Vision*, September 2010, pp. 1–14.