A Real-Time High Dynamic Range HD Video Camera

Rajesh Narasimha
Samsung Research America
1301 East Lookout Drive | Richardson, TX 75082
rajesh.n1@samsung.com

Umit Batur
Perception and Analytics Lab
Texas Instruments Inc.
batur@ti.com

Abstract

Standard cameras suffer from low video quality in high dynamic range scenes. In such scenes, parts of the video are either too dark or too bright. This is because lighting is very unevenly distributed in a high dynamic range scene. A standard camera, which typically has a 10 or 12 bit sensor, cannot capture the information both in dark and bright regions with one exposure. High dynamic range (HDR) imaging addresses this problem by fusing information from multiple exposures and rendering a dynamic range compressed result, which is also called dynamic range enhancement (DRE) or tone mapping. DRE is a challenging problem because human perception of brightness/contrast is quite complex and is highly dependent on image content. We present an embedded real-time HDR video camera system that provides 30fps, 720p/1080p high dynamic range video.

1. Introduction

High dynamic range is critical for various imaging applications such as security cameras, mobile cameras and automobile cameras. Non-HDR cameras take photographs with a limited exposure range, resulting in loss of detail in bright or dark areas. HDR imaging devices may compensate for this loss of detail by capturing two or more images at different exposure levels and combining them to produce an image with a broader tonal range. Merging multiple exposures preserves both the saturated and the shadow regions and thus provides a higher dynamic range than a single exposure. There are several known techniques for generating an HDR image (also referred to as a wide dynamic range (WDR) image) from two or more exposures. In some approaches, the sensor merges multiple exposures using spatial interleaving and provides a native HDR Bayer image with a pixel depth ranging from 12 to 20 bits. In other approaches, the imaging system captures multiple temporally spaced exposures from the sensor and these exposures are merged to form an HDR image in the imaging device. Whether with the native HDR Bayer data or with the imaging device generating the HDR image using temporal merging, tone mapping may need to be performed on the HDR image to permit processing of the HDR image in an imaging pipeline with a lesser pixel bit depth, *e.g.*, 10 -12 bits. Tone mapping essentially compresses the incoming image from 16-20 to 10/12 bits while preserving the content in both the bright and dark regions.

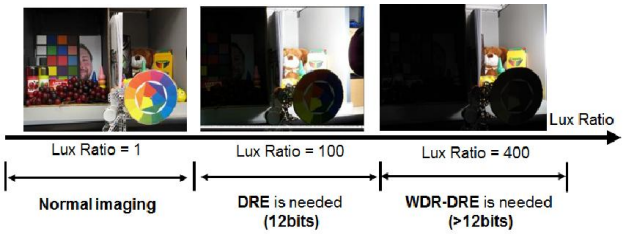


Figure 1. Dynamic range of normal and WDR scenes as a function of light intensity. Lux ratio indicates the ratio of the lighting level in the bright and dark regions of the scene.

Figure 1 shows the ratio of measured light intensity levels between the dark and bright regions in the scene. For normal scenes, this ratio is 1. For WDR scenes, the ratio can be larger than 100, and hence we would need more that 12 bits of information per pixel to preserve the entire dynamic range. In this work, we discuss a real-time HDR video camera system that provides 30fps at HD video resolution. The outlined algorithm is designed such that it achieves real-time performance on a resource constrained embedded platform.

1.1. Related Work

HDR Tone mapping operators (TMO) in literature are mainly classified as global and local operators. A thorough evaluation of the operators is provided in [8]. Our focus in this paper is a real-time algorithm that can run on an embedded platform that is constrained in terms of computation, memory and power. The main requirements of an commercial embedded real-time HDR tone mapping algorithm are the following: a) Single pass over the video frame to obtain the output, b) Optimal DMA transfers c) Algorithm designed in fixed point precision arithmetic thereby being SIMD friendly d) Negligible motion and temporal artifacts such as ghosting, jagged edges, etc. e)

Optimal color and contrast reproduction in the final HDR tone-mapped frame and f) No halos in the bright/dark boundaries. Based on the above stringent requirements, the global operators described in literature can work in real-time systems, but they need manual parameter tuning, while others are automatic [1][7] but cannot run in real-time since they are iterative (more than single pass on the frame). To our knowledge, the local operators [2-5] are hardware unfriendly since they are mostly iterative and require significant hardware resources in terms of memory, computation and power. A few of the operators that are hardware friendly have been implemented on FPGAs [6][9][16], on ARM SoC [17] and GPU [18]. Commercially available solutions include a hardware accelerated software solution [10] and a FPGA based solution [11] that provide good HDR image quality.

The solution we describe in the paper is a complete software solution on a Texas Instruments Davinci media processor [12-13] and achieves real-time performance on high definition frame sizes (720p/1080p).

The paper is organized as follows. Section 2 provides the overall HDR imaging pipe and explains the HDR tone mapping in detail. Embedded implementation is explained in section 3. Results and discussion is covered in section 4. Conclusion and future work are in section 5.

2. Algorithm Description

Input images to the HDR pipeline (shown in Figure 2) are Bayer pattern images. Thus each 2x2 pixels have four color components – a red component denoted as R, a blue component denoted as B, two green components denoted as Gr and Gb. Here we assume that the 16-bit HDR image is provided to us and do not focus on the temporal merging of dual exposures. The defect correction component is configured to correct the values of defective pixels in the HDR image. As is well known, an image sensor may have some number of defective pixels which respond to light exposure differently than other pixels due to factors such as manufacturing faults. This correction is performed using a look-up table (LUT) based technique. The defect corrected image is provided to the black level adjustment block. The black level adjustment block is configured to set sensor black to image black in the HDR image. That is needed because image sensors may record some non-zero value at dark pixel locations. The adjusted HDR image is provided to the noise filter component to remove various sources of noise in an HDR image by averaging similar neighboring pixels. The filtered HDR image is provided to the lens shading correction component. Lens shading is the phenomenon that an image is bright in the center and decreases in brightness towards the edges of the image. Lens shading correction is performed by a gain applied on a per-pixel basis to compensate for any decrease in brightness.

The white balance block is configured to adjust the white pixels in an HDR image to compensate for color differences introduced by light sources, such as the differences between incandescent, fluorescent, natural light sources, XE strobe, and W-LED flash, as well as mixed light conditions. This difference may result in a different color appearance that may be seen, for example, as the bluish appearance of a face or the reddish appearance of the sky. Also, the sensitivity of each color channel varies such that grey or neutral colors may not be represented correctly. The 3A analysis block is configured to collect metrics from the HDR image for auto focus, auto white balance, and auto exposure of subsequent images. The design of 3A for HDR data is an active research topic and not the focus of this work. The tone mapping module is configured to perform a method for local tone mapping on the HDR image and will be explained in detail later. The tone mapped image is provided to the remaining components of the image pipeline for further processing to generate the final HDR image, such as RGB blending, gamma correction, conversion to the YCbCr color space, edge enhancement, contrast enhancement, and/or false chroma suppression.

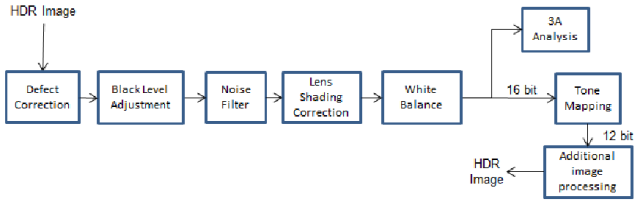


Figure 2. HDR Pipeline

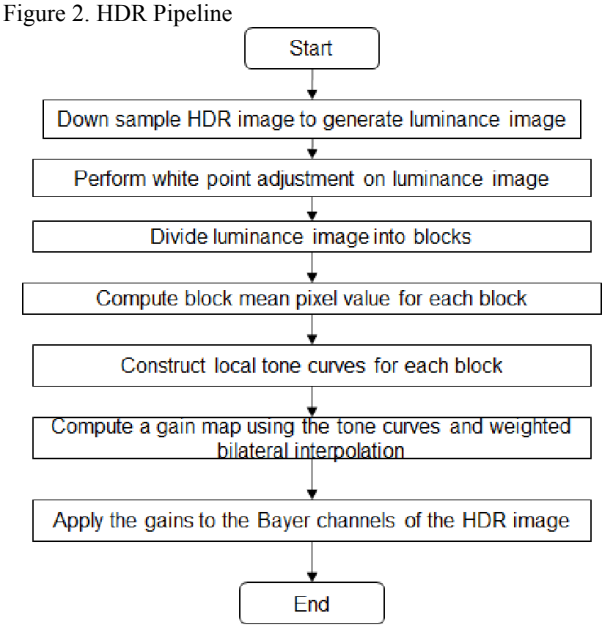


Figure 3. HDR Tone-Mapping Algorithm

2.1. HDR Tone-Mapping

The HDR tone-mapping algorithm is shown in Figure 3. The luminance image is computed from the input Bayer frame. For example, if the size of the Bayer is 1280x736 (720p), the size of the luminance image is 640x368. This is down-sampled to 320x184 to achieve speedup. We will discuss the embedded implementation in the next section. White point adjustment is then performed on the luminance image and is important due to the wide dynamic range. For example, for 16-bit pixel depth, the dynamic range is [0- 65535] where black is represented 0 and white as 65535. If the values of the white pixels in the image are not close to true white, *i.e.*, 65535, low contrast may result in the tone-mapped image. The white point adjustment can be performed as follows. First, a histogram of the pixel values in the image is computed and the whitest pixel value, *e.g.*, 65500, in the image is determined from the histogram. A small percentage of the white pixels with values closest to the whitest pixel value, *e.g.*, 2-5%, are then saturated to white, *i.e.*, the values of these pixels are changed to 65535. Further, after the saturation process, every pixel in the image is scaled by the gain between the whitest original pixel value and true white. The percentage of the whitest pixels to be saturated is an implementation choice.

2.1.1 Tone Curve Generation

The luminance image is then divided into overlapping blocks and a block mean pixel value is computed for each of the blocks. The block size and the amount of overlap between the blocks is implementation dependent and may be chosen empirically, for example, based on a tradeoff between quality and performance. The block mean pixel value may be computed in any suitable way, *e.g.*, as the average of the pixel values in the block or as a weighted average of the pixel values in the block. If a weighted average is used, the weight for each pixel may be based on the inverse of pixel intensity difference between the center pixel of the block and the pixel. Local tone curves are then computed for each block. These tone curves may be of any suitable length. For simplicity of explanation, the length of each local tone curve is assumed to be 256.The computation of a local tone curve of length 256 is illustrated in Figure 5.

To compute a tone curve for a block, three points on the curve are determined as illustrated in Figure 5B. In this figure, CB is the block mean value, *TB = CB + delta_brightness* is the target brightness, and *LC and RC* are, respectively, left and right contrast (*i.e.*, the midpoint between 0 and CB and the midpoint between CB and 255, respectively). The three points of a tone curve for the simple case, *LC =0 and RC=255* or these can be estimated based on CB and computing the gain values using hermite interpolation which results in a 5-point tone curve. Thus for the simple case, the three points of the local tone curve in the *[x,y]* format are *[0 0], [CB TB]* and *[255, 255]* as in the right plot of Figure 5.The 256 points on the tone curve are constructed from these three points using band limited interpolation and Hermite polynomials [19]. The left plot in Figure 5 illustrates the determination of the value of the *delta_brightness* to be added to the block mean pixel value to determine the target brightness TB. In this graph, the *x -axis* is the block mean value scaled from 16 bits to 8 bits and the *y-axis* is *delta_brightness* (gain). As seen from this graph, for scaled block mean pixel values less than the first knee point k_1, the *delta_brightness* is the value of $gain_1$. For scaled block mean pixel values larger than the second knee point k_2, the *delta_brightness* is the value of $gain_2$. For scaled block mean pixel values between the two knee points, the value of *delta-brightness* is chosen along the gain curve between the two knee points. The determination of *delta_brightness* values for particular block mean pixel values may be implemented as a lookup table (LUT) of 255 values derived based on the values of k_1, k_2, $gain_1$, and $gain_2$. The knee points and the gains may have any suitable values, are chosen based on quality preference.

2.1.2 Gain Map Computation

After the local tone curves are constructed for each block, a gain map is computed using the local tone curves and weighted using bilateral interpolation for each pixel since the local tone curves are at the block centers. We chose to calculate the local tone curves for 209 (19 horizontal and 11 vertical) blocks. The resulting gain map contains a gain corresponding to each pixel in the luminance image. The gain *G(x,y)* for a 16-bit luminance pixel *X(x,y)* in the luminance image with the value *L* is computed as

$$G(x, y) = L_{out} / L \quad (1)$$

where L_{out} is computed as a weighted sum of applying the tone curves of the four neighboring blocks having center points closest to *X(x,y)* compared to *L*. The four block centers closest to *X(x,y)* are referred to as the *upper-left (UL)* point, the *upper-right (UR)* point, the *lower-left (LL)* point, and the *lower-right (LR)* point. Figure 4 is an example illustrating the locations of the center points relative to *X(x,y)*. Each of the small circles in the images is a center point of a block. For odd block sizes, the center point is the center pixel in the block. We provide embedded friendly fixed-point implementation equations below. For even block sizes, the center point is between the four center pixels in the block. More specifically, L_{out} is computed as follows,

$$L_{out} = \beta^{UL} L_{out}^{UL} + \beta^{UR} L_{out}^{UR} + \beta^{LL} L_{out}^{LL} + \beta^{LR} L_{out}^{LR} \quad (2)$$

where $NN = \{UL, UR, LL, LR\}$ and L_{out}^{NN} is the result of applying the tone curve of the block containing the nearest neighbors (*NN*) with respect to the center point of *X(x,y)* and β^{NN} is a weight computed for this result, The value of L_{out}^{NN} is computed as,

$$L_{out}^{NN} = \frac{LUT[L_{floor}]^{NN} W_{ceil} + LUT[L_{ceil}]^{NN} W_{floor}}{W_{ceil} + W_{floor}} \quad (3)$$

where $LUT[]^{NN}$ is the tone mapping lookup table for the floor and ceil luminance values for set *NN* from the current center point of *X(x,y)* and $L_{floor}, L_{ceil}, W_{floor}$, and W_{ceil} are computed as follows. The floor and ceiling values of *L* are computed as per

$$L_{floor} = \frac{L}{256}; 0 \le L_{floor} < 254$$
$$L_{ceil} = L_{floor} + 1; 1 \le L_{floor} < 255 \quad (4)$$

and the floor and ceiling weights are computed as per

$$W_{floor} = L - L_{floor} * 256$$
$$W_{ceil} = L_{ceil} * 256 - L \quad (5)$$

The value of β^{NN} in equation (2) is computed as follows,

$$\beta^{NN} = \alpha_{dw}^{NN} \alpha_{iw}^{NN}; 0 \le \alpha_{dw}^{NN}, \alpha_{iw}^{NN} \le 1 \quad (6)$$

where α_{dw}^{NN} is a distance weight based on the distance from the *NN* pixels with respect to the center point *X(x,y)* and α_{iw}^{NN} is an intensity weight based on the difference in intensity between the mean pixel value of the block containing the *NN* neighboring center point and current block mean *L*. This is depicted in Figure 6. Assuming that that the block size is odd and the center point is the center pixel of the block, the distance weight α_{dw}^{NN} is computed as follows,

$$\alpha_{dw}^{NN} = |x - x_{NN}||y - y_{NN}| / \lambda \quad (7)$$

where (x_{NN}, y_{NN}) are the coordinates of the *NN* with respect to the center point coordinates *(x,y)* and λ is a normalization factor to scale the distance weight to be between 0 and 1 inclusive. If the block size is even, the center point is located between the four center pixels of the block, and the distance weight is computed based on the distances from the pixel *X(x,y)* to each of these four center pixels. The intensity weight α_{iw}^{NN} is computed as per

$$\alpha_{iw}^{NN} = LUT[|L - L_{NN}|] \quad (8)$$

where L_{NN} is the mean pixel value of the block containing the *NN* neighboring center point and the LUT is derived as per the graph similar to the left plot in Figure 5.

Figure 6 are examples illustrating, respectively, the computation of the distance weights and the intensity weights for *X(x,y)*. Similar to the left plot in Figure 5, *the x axis for* α_{iw}^{NN} computation *is* $|L-L_{NN}|$ and the *y-axis* is the intensity weight (gain). As in the earlier case for *delta_brightness* computation, for values of $|L-L_{NN}|$ less than the first knee point k_1, the intensity weight is the value of $gain_1$. For values of $|L-L_{NN}|$ larger than the second knee point k_2, the intensity weight is the value of $gain_2$. For values of $|L-L_{NN}|$ between the two knee points, the intensity weight is chosen along the gain curve between the two knee points. The determination of intensity weights for particular values of $|L-L_{NN}|$ may be implemented as a lookup table (LUT) of size 255 values derived based on the values of k_1, k_2, $gain_1$, *and* $gain_2$. The knee points and the gains may have any suitable values, which may be selected empirically. In our case, the values of the knee points and the gains are the same as those used to generate the LUT for determining the value of *delta_brightness* for particular block mean pixel values. Since it is lot of computation overhead to deal with 16-bit pixel values, we use LUT to map them to a smaller range for *delta_brightness* and α_{iw}^{NN} computations.

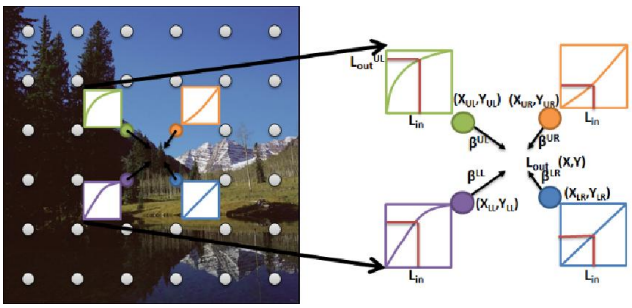
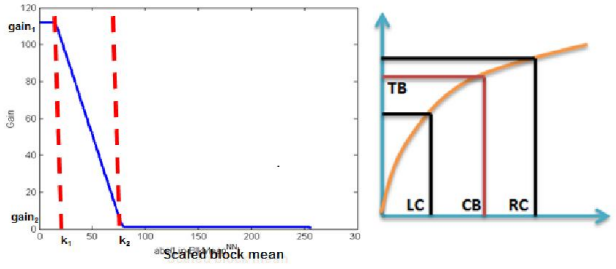


Figure 4. An example illustrating local tone mapping



(A) (B)

Figure 5. Graphs illustrating generation of local tone curves

Referring again to Figure 3, once the gain map is computed, the gains are applied to the four Bayer channels of the HDR image. To apply the gains, the gain map is up-sampled to the original HDR frame resolution and the gains applied to the four Bayer components of the corresponding pixels in the HDR image. The gains may be applied as per

$$Rbayer(x,y) = G(x,y)\ Rbayer(x,y)$$
$$Grbayer(x,y) = G(x,y)\ \ Grbayer(x,y), \quad (9)$$

Gbbayer(x,y) = G(x,y) Gbbayer(x,y)
Bbayer(x,y) = G(x,y) Bbayer(x,y).

We make sure that the final Bayer values are in the 16-bit range using scaling if needed. While up sampling, directional filtering is used to eliminate any artifacts that may arise.

3. Embedded Implementation

The flow diagram for the embedded implementation is shown in Figure 7. The HDR sensor delivers a 12-bit compressed data. The frames are first de-compressed into 16/20-bit, and then a scene adaptive local dynamic range enhancement algorithm explained earlier is applied to convert this HDR data into 12-bit Bayer frames. These frames are processed by a hardware image pipeline (ISP) to produce YCbCr 4:2:0 video frames, which are then encoded into H.264.

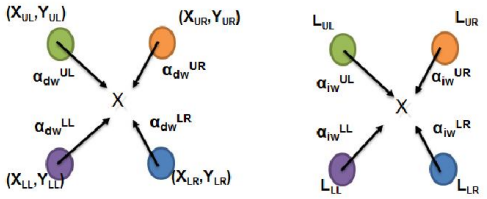


Figure 6. Computation of distance weights and pixel intensity weights for local tone mapping

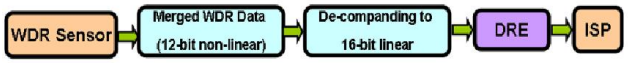


Figure 7. HDR tone-mapping frontend

4. Results and Discussion

Figure 8 depicts the HDR results for various challenging scenes which are actual screen shots from the real-time implementation. The left column image is one without the HDR algorithm and the right column image is with the algorithm. We provide a HDR light-box scene, a lab scene, a light scene and a outdoor scene. The left column image is the original 16-bit HDR image converted to a 12-bit image without tone mapping and the right column image is the 12-bit HDR image with local tone mapping applied. Note the improvement in brightness and contrast after the local tone mapping is applied. In figure 9, we provide some examples of the common issues with HDR scenes namely, halo artifacts around bright/dark boundary and jagged edges due to fixed point arithmetic on embedded platforms. The halo artifact was solved by using down sampling luminance (Y plane in YUV422 frame) using pixel-shift rather than averaging and implementing both distance and intensity based weighting for bilinear interpolation. The jagged edges shown by the "arrows" were eliminated using directional filtering after the gain map was up sampled and applied to the Bayer image.

5. Conclusion and Future Work

An embedded real-time HDR video camera solution is discussed. The fixed point solution achieves 30fps at high definition video resolutions (720p/1080p) without compromising on quality. The solution was evaluated using perceptual tests and against commercially available embedded solutions.

References

[1] E. Reinhard, G. Ward, P. Debevec, P. Sumanta, W. Heidrich,and K. Myszkowski, "Parameter estimation for photographics tone reproduction.," Journal of Graphics Tools, vol. 7(1), pp. 45-51, 2003

[2] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images.," ACM Transactions on Graphics, vol. 21(3), pp. 257–266, 2002.

[3] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression.," ACM Transactions on Graphics, vol. 21(3), pp. 249–256, 2002.

[4] Y. Li, L. Sharan, and E. Adelson, "Compressing and companding high dynamic range images with subband architectures.," ACM Transactions on Graphics, vol. 24(3), pp. 836–844, 2005.

[5] R. Mantiuk, S. Daly, and L. Kerofsky, "Display adaptive tone mapping.,"ACM Transactions on Graphics, vol. 27(3), pp. 68, 2008.

[6] F. Hassan and J. Carletta, "An fpga-based architecture for a local tone-mapping operator.," Real-Time Image Processing, vol. 2, pp. 293–308, 2007.

[7] G. Ward, H. Rushmeier, and C. Piatko, "A visibility matching tone reproduction operator for high dynamic range scenes.," IEEE Transactions on Visualization and Computer Graphics, vol. 3(4), pp. 291–306, 1997.

[8] P. Ledda, A. Chalmers, T Troscianko, and H. Seetzen, "Evaluation of tone mapping operators using a high dynamic range display.,"ACM Transactions on Graphics, vol. 24(3), pp. 640–648, 2005

[9] Chris Kiser, Erik Reinhard, Mike Tocci and Nora Tocci, "Real-time Automated Tone Mapping System for HDR Video.", IEEE International Conference on Image Processing, 2012

[10] www.apical.co.uk/products/iridix-hdr/

[11] http://www.axis.com/files/whitepaper/wp_wide_dynamic_range_49148_en_1210_lo.pdf

[12] http://www.ti.com/tool/dm368ipnc-mt5

[13] http://www.ti.com/tool/dm385-ar0331

[14] http://www.aptina.com/products/image_sensors/mt9m034/

[15] http://www.aptina.com/products/image_sensors/ar0331/

[16] Pierre-Jean Lapray, Barthélémy Heyrman, Matthieu Rossé, and Dominique Ginhac, "Smart camera design for realtime high dynamic range imaging.," ICDSC, page 1-7, 2011

[17] Ching-Te Chiu, Tsun-Hsien Wang, Wei-Ming Ke, Chen-Yu Chuang, Jhih-Siao Huang, Wei-Su Wong, Ren-Song Tsay, Cyuan-Jhe Wu, "Real-Time Tone-Mapping Processor with Integrated Photographic and Gradient Compression using

0.13 µm Technology on an Arm Soc Platform.,"" Signal Processing Systems vol. 64(1), pp. 93-107, 2011
[18] Ahmet Oğuz Akyüz, "High Dynamic Range Imaging Pipeline on the GPU.," Springer Journal of Real-Time Image Processing
[19] http://en.wikipedia.org/wiki/Hermite_polynomials

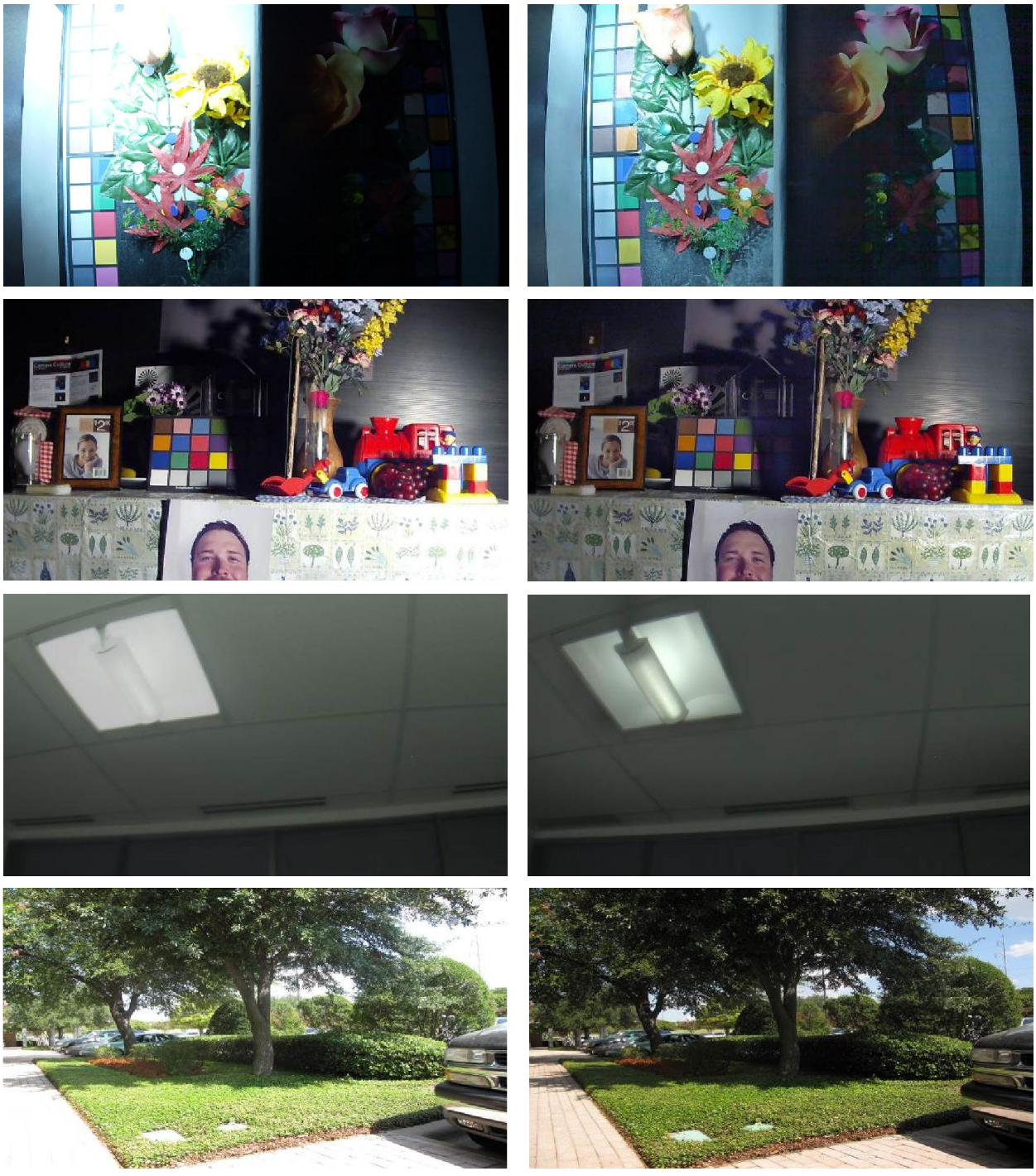


Figure 8. HDR results for HDR light-box scene, a lab scene, a light scene and an outdoor scene. The left column images are the original 16-bit HDR image converted to a 12-bit image without tone mapping and the right column images are the 12-bit image HDR image with local tone mapping applied.

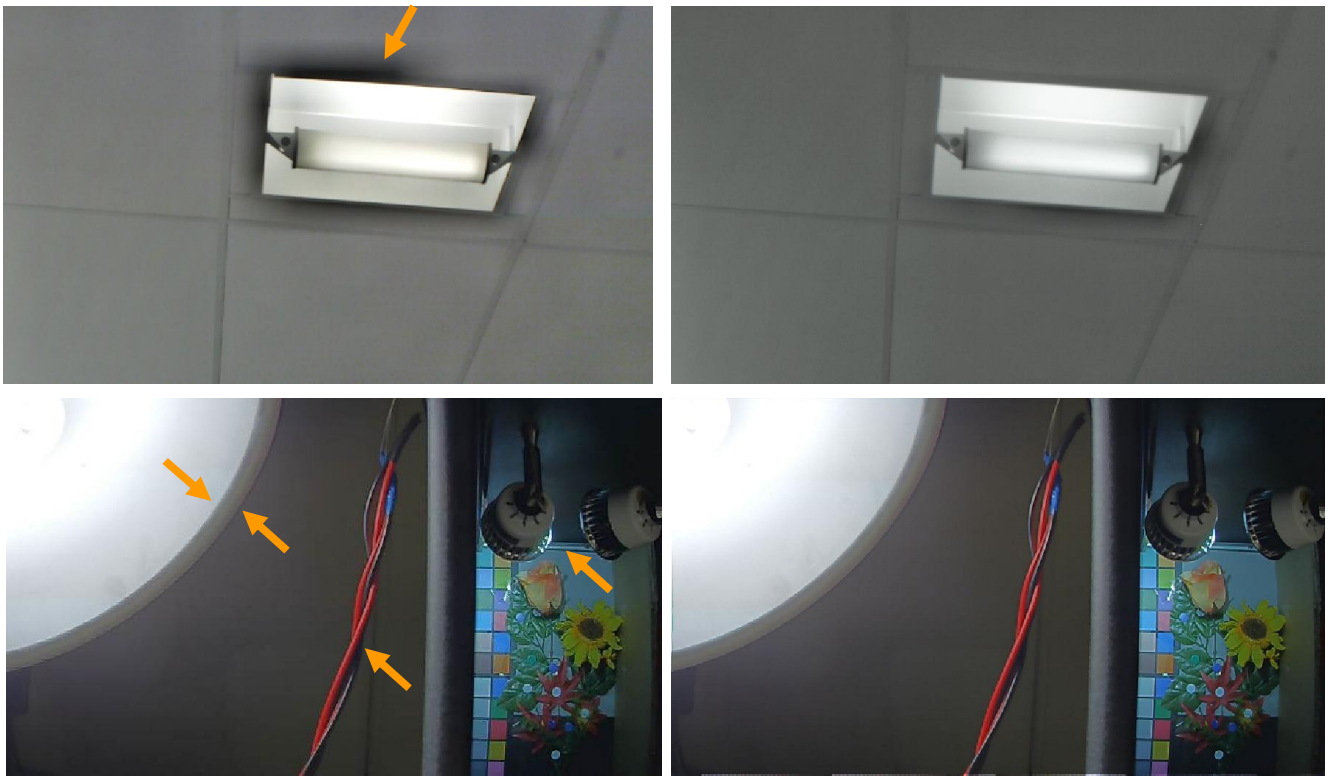


Figure 9. Halo artifacts around bright/dark boundary and jagged edges due to fixed point arithmetic on embedded platforms denoted by arrows in the left column images. The issues are resolved in the right column images.