407	407	407	407
407	407	407	407
407	407	407	407
407	407	407	407

(a) 4x4 Uniform MCU Bit Budget Map

87	151	82	70
321	317	188	184
736	736	657	498
698	640	598	547

(b) 4x4 Non-uniform MCU Bit Budget Map

Figure 39-13 4x4 MCU Bit Budget Map for Image 21

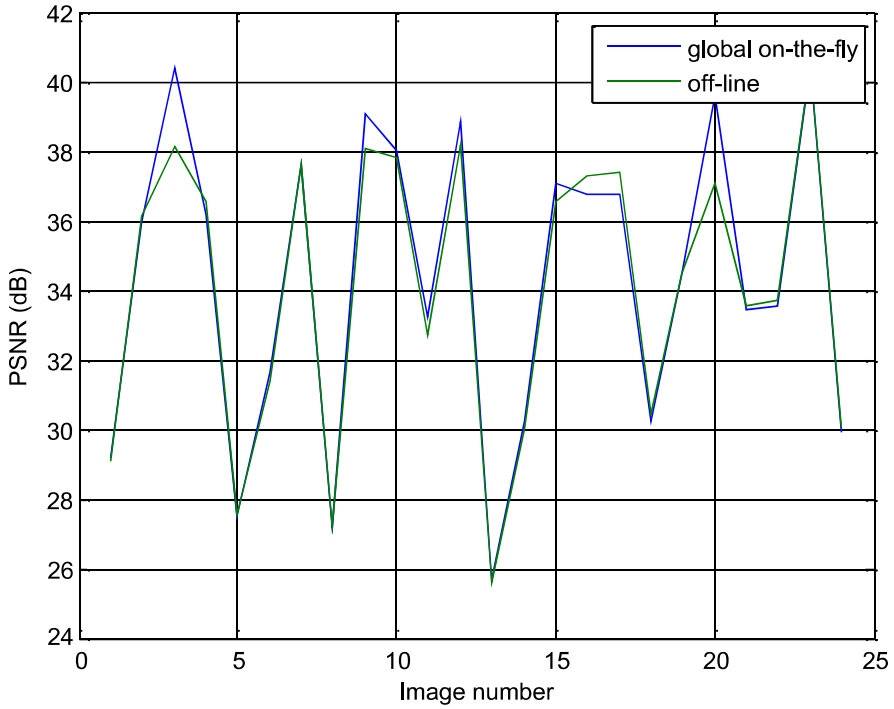


Figure 39-14 Comparison between Off-line Hard Thresholding Rate Control and On-the-fly hard Thresholding Rate Control using 16x16 Bit Budget Map

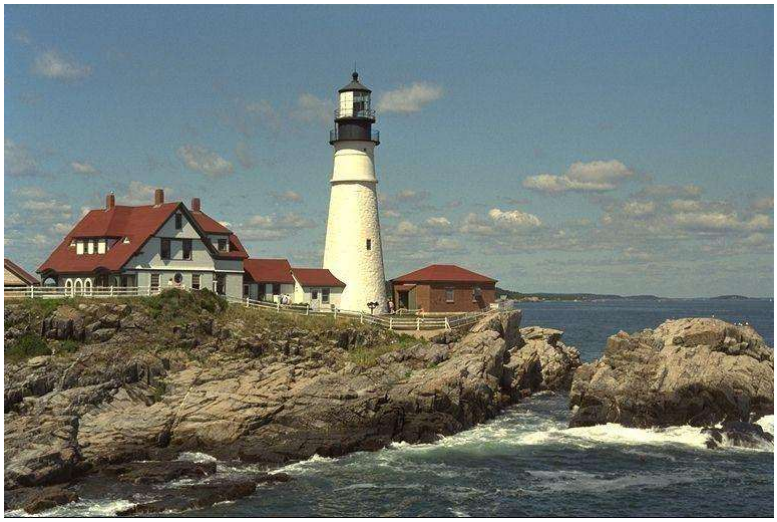


Figure 39-15 Off-Line Quantization Rate Control for Image 21 (80% Quality Factor)

Figure 39-16 On-the-fly Hard Thresholding using the 16x16 MCU Bit Budget Map for Image 21 (80% Quality Factor)

39.2.2.3 Implementation

Because of the JPEG HW pipeline structure, there is a one MCU delay on applying the new threshold. The block diagram is shown below.

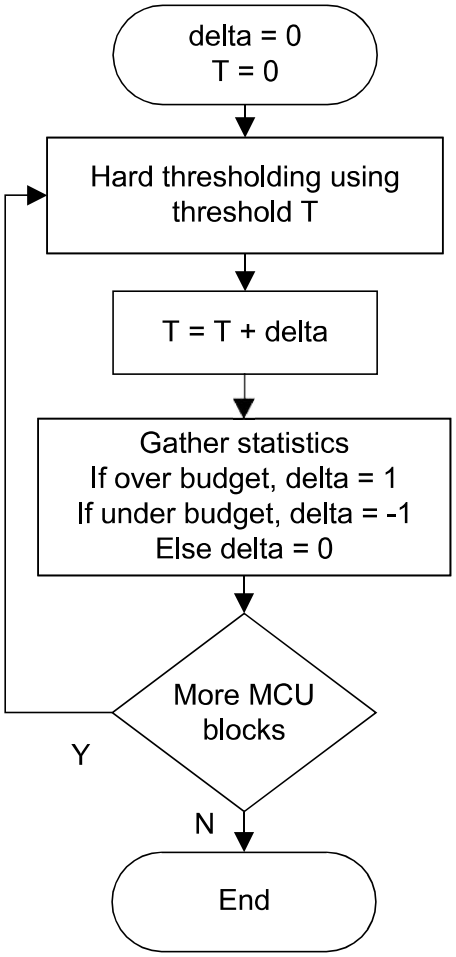


Figure 39-17 File Size Control Block Diagram

39.2.2.4 Output buffer overhead

When using the on-the-fly rate control, there is a possibility that the output buffer will overflow. There are two reasons for overflow. One is that it does not count the number of bits increased when byte stuffing occurs. The other is the tracking error. To get the mean and standard deviation of the buffer overflow percentage, we modeled the data as

, where X and Y are the actual rate and target, repectively. If we have an perfect on-the-fly rate control algorithm, A would be 0.

To get the statistics on A, we used 24 Kodak images for our test images. To increase the size of the image pool, we rotated the 24 Kodak images by 0, 90, 180 and 270 degrees and used them as our test images. We subsampled the image to make is an H2V2 image. We chose data rate (bits/pixel) of 4.625, 2.375, 1.5625, 0.8125, and 0.5 for quality factor 99, 95, 90, 75, and 50%, respectively. These data rates are the smallest data rate for 24 Kodak images and for each quality factor. We obtained mean(A) = 0.0375 and std(A) = 0.0167. The histogram of A is shown in Figure 39-18. By adding 14.7 % overhead (( ) ( )), the probability of overflow is 9.87e-10.

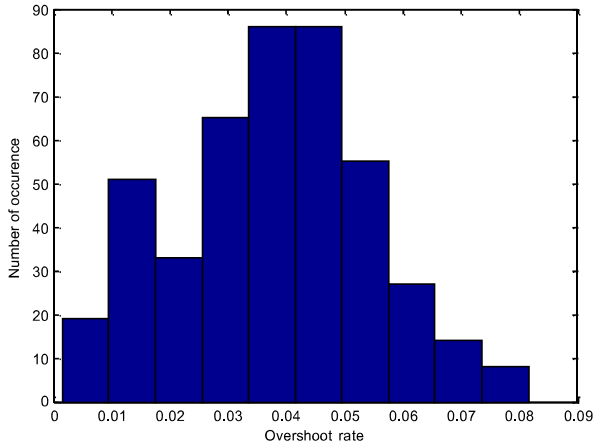


Figure 39-18 Histogram of 'A'

For each quality factor, it would be helpful if we could have a target data rate. But the target data rate will depend on the application requirements. To get an idea for data rates for typical images, we obtained the data rates of the H2V2 format 2009 Nokia ISP 5MP comeptition images. The total number of images was 454. The histogram of the data rates for 90% quality factor are shown in

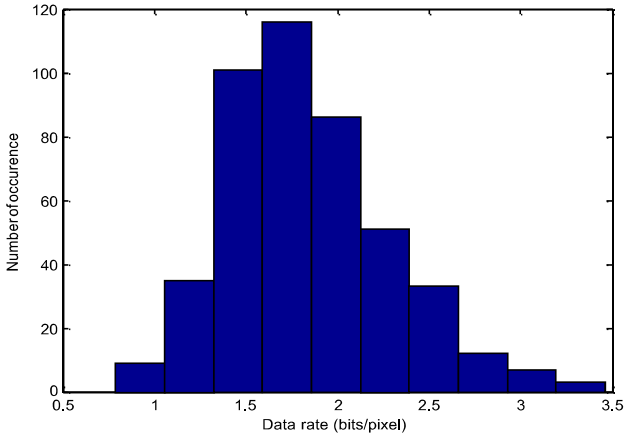


Figure 39-19 Distibution of 90% quality factor data rates for the 2009 Nokia ISP competition images

The mean was 1.835 bits/pixel and the standard deviation was 0.446 bits/pixel.

39.3 System partition

Table 39-1 JPEG Scaler system partition

Operation	Chromatix	ARM	ADSP	JPEG HW
Parameter Setting		X		
On-the-fly file size control				X

40 Lens Roll-Off Correction

40.1 Problem statement

Due to cost-saving considerations and size issues, the lens mounted on the camera of a cell phone tends to be small and inexpensive. Because of its small size and inferior material, the captured image shows brightness attenuation near the edge of the image. Figure 40-1 illustrates a typical example of this brightness attenuation. The luminance of the pixels near the edge is considerably darker than that of the pixels in the center. As can be seen in the example, the color tone can shift as well.

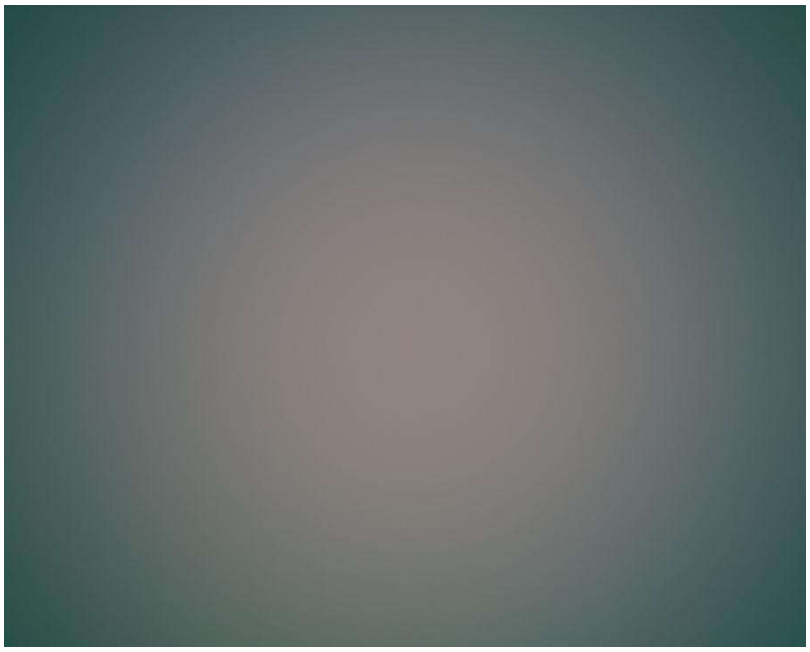


Figure 40-1 Flat field image without roll-off correction

The previously proposed method was to correct the image based on a radial symmetric assumption. This method works very well if the correction is applied to all three color channels (R, G, and B) and if the distortion satisfies the assumption of radial symmetry. However, the mounting of the lens may be imperfect (tilted and off-centered) and the lens surface may not be radially symmetric. Because of this, the previously proposed method does not meet the desired correction capability. Therefore, a surface mesh matching method is proposed that is designed to handle the non-radial symmetric lens roll-off distortion.

The following figures show some example of non-radial symmetric lens roll-off distortion. Figure 40-2 provides an example of non-radial symmetric roll-off distortion (this image was provided by CH Mobile).

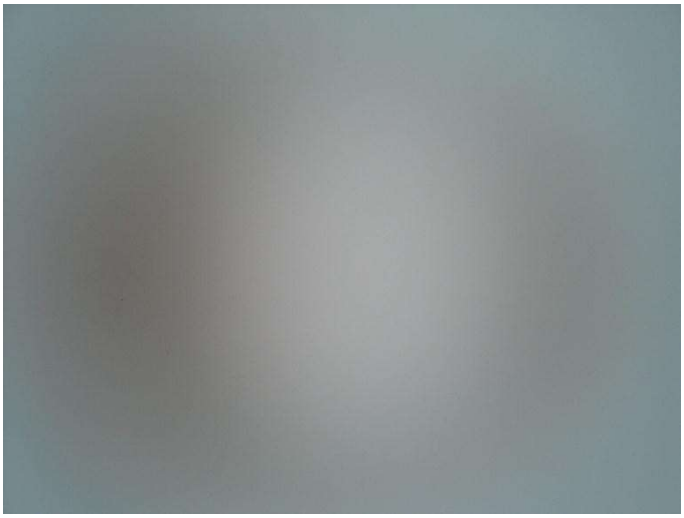


Figure 40-2 Example of non-radial symmetric roll-off distortion

Mesh-based surface matching method uses bilinear interpolation for pixel gains between mesh grid points. This approach would cause grid pattern problem in high contrast boosting such as strong luma adaptation, as shown in Figure 40-3. Figure 40-4 explains how bilinear interpolation would cause the grid pattern. The error between the ideal rolloff correction curve and the linearly interpolated curve becomes a grid pattern in the corrected image as seen in Figure 40-3. For this reason, bicubic interpolation is introduced in VFE4 to interpolate between mesh grids to finer subgrids. Bilinear interpolation is then used to interpolate for each pixel in the subgrids.

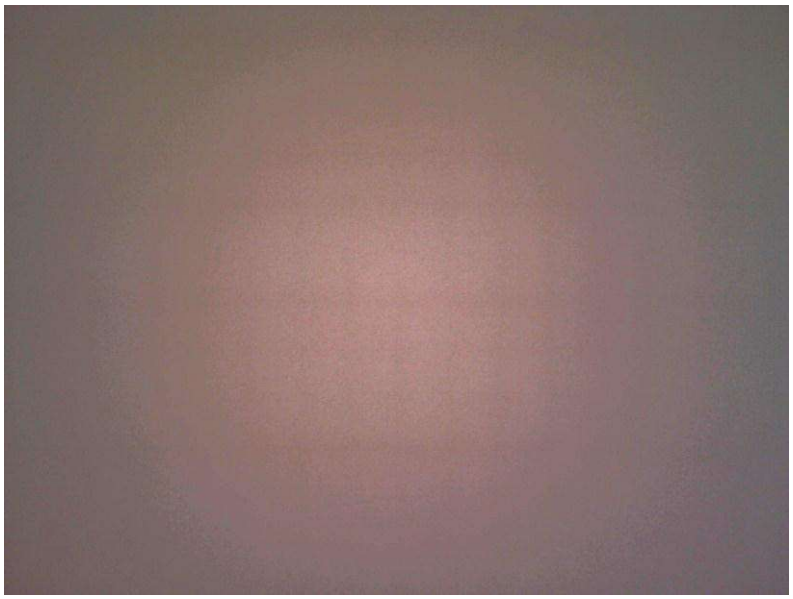


Figure 40-3 Mesh gird pattern problem under high contrast boosting

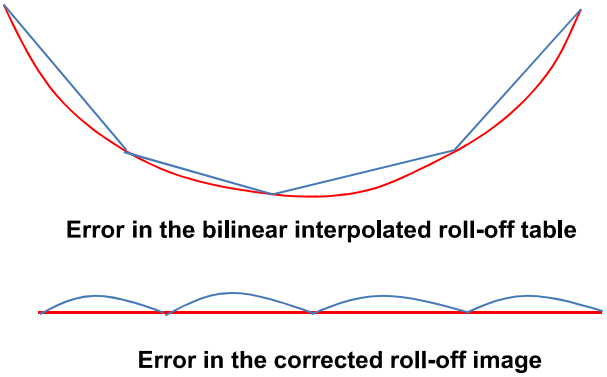


Figure 40-4 Formation of mesh grid pattern due to bilinear interpolation

Table 40-1 summarizes the lens roll-off correction characteristics.

Table 40-1 Summary of lens roll-off correction characteristics

Characteristic	Description
Assumptions	Lens shading is module-independent
Inputs	12-bit Bayer data
Outputs	12-bit Bayer data corrected for lens roll-off
Frequency of use	Every pixel in every frame
Modes of operation	Viewfinder, Capture, Snapshot Processing

40.2 Algorithm

The basic approach to performing lens roll-off correction for any type of distortion pattern is to apply a mesh over the distortion surface as shown in Figure 40-5.

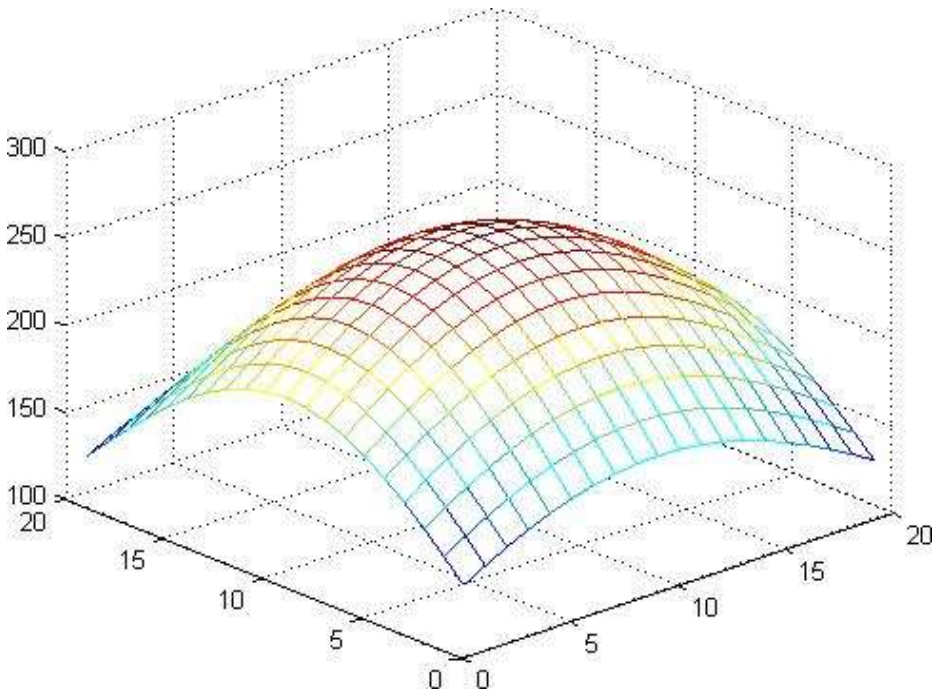


Figure 40-5 Matching the lens roll-off surface with a two-dimensional mesh

No matter what the distortion surface is, it can be subsampled by grid points evenly distributed along the entire surface. Between the grid points, bilinear interpolation is performed to approximate the distortion curve surface. The number of grid points that are needed in order to provide an accurate approximation (no visual artifacts) to the distortion surface depends on the smoothness of the surface. Factors such as memory space consideration also come into play to determine the number of grid points. In software implementation, non-uniformly distributed grid points can be used, but applying evenly distributed grid points is recommended for a hardware implementation.

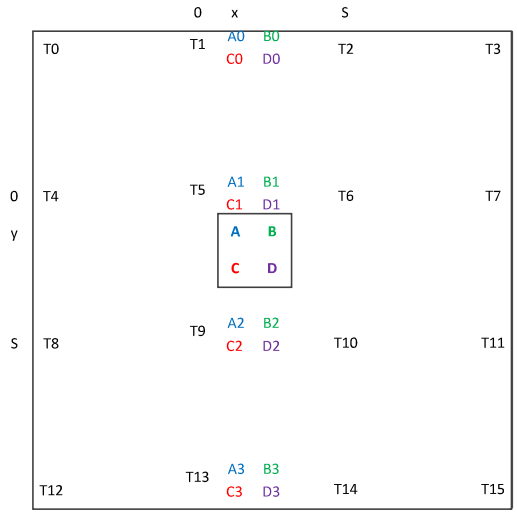
40.2.1 Algorithm details

Lens roll-off correction is applied to Bayer raw images to reduce the computation load. Instead of modeling the curve and performing polynomial calculations, a linear piecewise approximation with the help of LUTs is used. Each color has its own LUT, so the brightness attenuation and color shift problems can be solved simultaneously.

The LUT approach is used to avoid computations such as matrix multiplication, square root, and division. All required LUTs are computed offline and they are never modified unless a different lens or sensor is used. For earlier VFEs, the image is divided into H×V = 16×12 grids, so VFE needs to linearly interpolate between the mesh points of 4 channels, totaling 4×17×13 = 884 mesh points. For VFE4, as stated in section 40.1, bicubic interpolation is used to interpolate finer subgrids between the mesh points. The number of subgrids between horizontal and vertical mesh points can be 1, 2, 4, or 8 subgrids, depending on image dimensions. Having such finer subgrids, the number of mesh grids can be 12×9, thus need 4×13×10 = 520 mesh points to cover the whole

image. This amounts to a 41% saving of parameters. Then within each subgrid, bilinear interpolation is used to interpolate the pixel mesh gains as earlier version VFEs.

Bicubic interpolation from mesh grids to subgrids is shown in Figure 40-6. T0 to T15 are the correction factors (CFs) or digital gains of the mesh points. With these 4×4 CFs, horizontal and vertical bicubic interpolation is done for S×S subgrids, where S = 2^P = 1, 2, 4, or 8 and P = 0, 1, 2, or 3 is the bicubic interpolation factor. As in Figure 40-6, subgrid gains ABCD within (T5, T6, T9, T10) are first interpolated horizontally. AC0 and BD0 are interpolated from T0, T1, T2, and T3, AC1 and BD1 are interpolated from T4, T5, T6, and T7, and so on for AC2, BD2 and AC3, BD3. Then CFs A and C are bicubic interpolated from AC0 to AC3 and B and D are interpolated from BD0 to BD3. The bicubic interpolation coefficients W[9][4] are also shown in Figure 40-6. For different bicubic interpolation factor P, different entries of W are utilized for different subgrid points. If (T5, T6, T9, T10) are against the mesh boundaries or corners, then linear extrapolation is used to extrapolate the missing CFs of T0~T3, T4, T7, T8, T11, and T12~T15.



Programmable number of sub-grids:
P = (0, 1, 2, 3) for S = (1, 2, 4, 8) sub-grids of each 2×2 mesh grid (T5, T6, T9, T10).
Linearly extrapolate grid gains on mesh boundaries.

Bicubic Coefficient LUT:
W[9][4] = {
{ 0, 1024, 0, 0 }
{ -49, 987, 93, -7 }
{ -72, 888, 232, -24 }
{ -75, 745, 399, -45 }
{ -64, 576, 576, -64 }
{ -45, 399, 745, -75 }
{ -24, 232, 888, -72 }
{ -7, 93, 987, -49 }
{ 0, 0, 1024, 0 }
}; (12sQ10 numbers)

LUT Utilization:
For P=0, use W[0,8][0~3];
For P=1, use W[0,4,8][0~3];
For P=2, use W[0,2,4,6,8][0~3];
For P=3; use W[0~8][0~3].

Figure 40-6 Bicubic interpolation from mesh grid gains to subgrid gains

Next, as shown in Figure 40-7, bilinear interpolation is used to compute the CFs for each pixel within the subgrid ABCD.

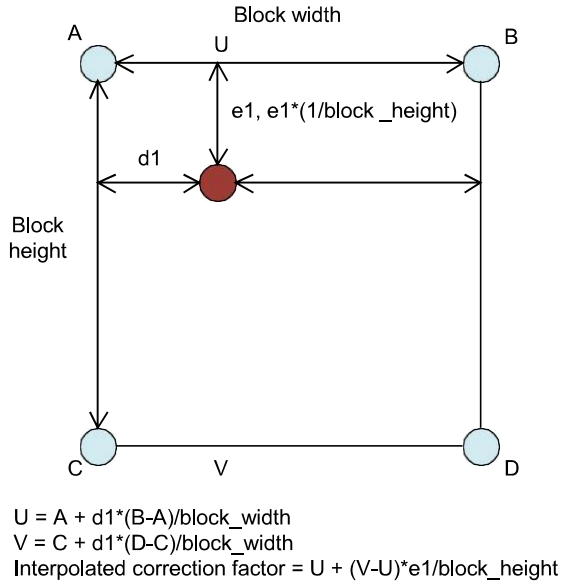


Figure 40-7 Bilinear interpolation in hardware implementation

The detailed derivation is as follows,

$U = (d2 * A + d1 * B)/block\_width$; where $d2 = 1 - d1$

$= A + delta1 * d1$; where $delta1 = (B - A) * (1/block\_width)$

$= A + accumulate(delta1)$

$V = (d2 * C + d1 * D)/block\_width$

$= C + delta2 * d1$; where $delta2 = (D - C) * (1/block\_width)$

$= C + accumulate(delta2)$

Interpolated correction factor $= (U * e2 + V * e1)/block\_height$; $e2 = 1 - e1$

$= U + (V-U) * e1 / block\_height$

$= U + (V-U) * accumulate(delta\_Height)$

Note that the multiplications delta1*d1 and delta2*d1 in the derivation of U and V are replaced by horizontal and vertical accumulations, respectively. delta1 represents the slope of correction factor (CF) along the top edge of the ABCD block, and delta2 along the bottom edge. Thus the lens rolloff correction gain of a certain pixel in the ABCD block can be obtained by hardware accumulations, first pixel by pixel horizontally along the top and bottom edges, then line by line vertically.

In VFE3.2 (MSM8960v1) and earlier, there is a delta_table[] (16sQ20, (Ny + 1) * Nx entries) and an init_table[] (16uQ13, (Ny + 1) entries) for each R/Gr/Gb/B channel, where Nx=16, Ny=12. So it totally costs 14144 bits of RAM in hardware. Double-buffering the rolloff tables would then cost 28288 bits of RAM. In order to save RAM area and allow double buffering, VFE3.3 (MSM8960v2) uses principle component analysis (PCA) decomposition representation of the mesh tables. With double buffering, it costs 9558 bits of RAM, which is 68% of mesh table RAM. For VFE4 (MSM8974) with 4×13×10 entries of mesh gains (13uQ10) with double buffering, it will cost 13520 bits of RAM, which is 96% of the single buffered mesh table RAM. The detailed memory usage is in Table 40-2. Although the RAM of VFE4 bicubic rolloff table seems to be close to mesh rolloff table, note that not only the bicubic mesh table is double

buffered, which allows real-time rolloff change, but the accuracy of mesh interpolation is increased also. In addition, bicubic rolloff table does not need PCA computation to decompose the mesh tables interpolated in software, and dynamic mesh lens rolloff allows dynamic local gain adjustment and dynamic tint correction.

Item	Mesh Rolloff	VFE3.3 PCA-based Rolloff	VFE4 Bicubic Mesh Rolloff
Initial/Coefficient Table Entries	Initial: 13x4 = 52	Coeff.: 4x13x8 = 416	N/A
Delta/Basis/Mesh Table Entries	Delta: 13x16x4 = 832	Basis: 8x17 = 136	Mesh: 10x13x4 = 520
Total Entries	884	552	520
Double Buffered Entries	(1768)	1104	1040
Initial/Coefficient Table Bits	Initial: 13x4x16b = 832b	Coeff.: 4x13x(12b + 9bx2 + 8bx5) = 3640b	N/A
Delta/Basis/Mesh Table Bits	Delta: 13x16x4x16b = 13312b	Basis: (11b+8bx7)x17= 1139b	Mesh: 10x13x4x13b = 6760b
Total Bits	14144b	4779b	6760b
Double Buffered Bits	(28288b)	9558b	13520b

Table 40-2 Parameter and memory comparison of different versions of rolloff tables

Figure 40-8 is the block diagram showing the fetching of R/Gr/Gb/B mesh gains and extrapolation if the grid (T5, T6, T9, T10) is at image boundary or corner. Figure 40-9 shows the horizontal and vertical bicubic interpolation computation for subgrid gains ABCD within mesh grid (T5, T6, T9, T10), as illustrated in Figure 40-6. After the gains of a mesh subgrid are available, Figure 40-10 shows the computation of the horizontal slopes of mesh gain along the upper and lower edges of a block. The inverse of block width is provided by software in 17uQ20 to support input Bayer image sizes from QVGA (320×240) to 21.67MP (5376×4032) with (12×S)×(9×S) subgrids, where S = 1, 2, 4, or 8 is the number of subgrids per mesh block. Thus the subgrid width, W_B, ranges from 14 to 224, and $1/W_B$ in Q20 ranges from 4681 to 74898. Figure 40-11 shows the horizontal bilinear interpolation by accumulating the upper and lower block slopes pixel by pixel. Figure 40-12 shows the vertical bilinear interpolation of lens rolloff correction. Different subgrid widths and heights are allowed for left-right stereo pair images of native 3D support.

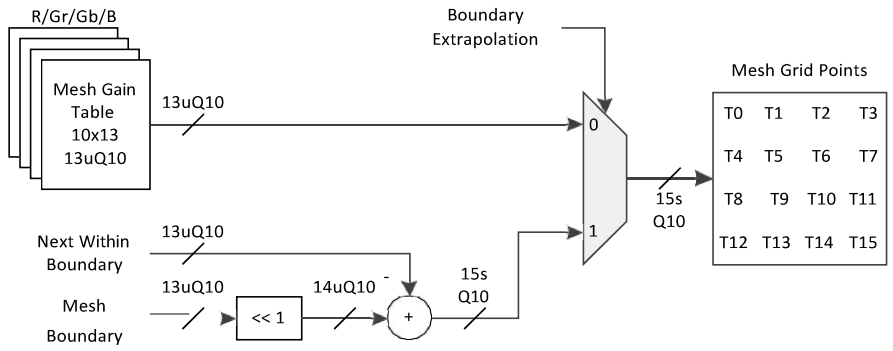


Figure 40-8 Mesh gain extrapolation

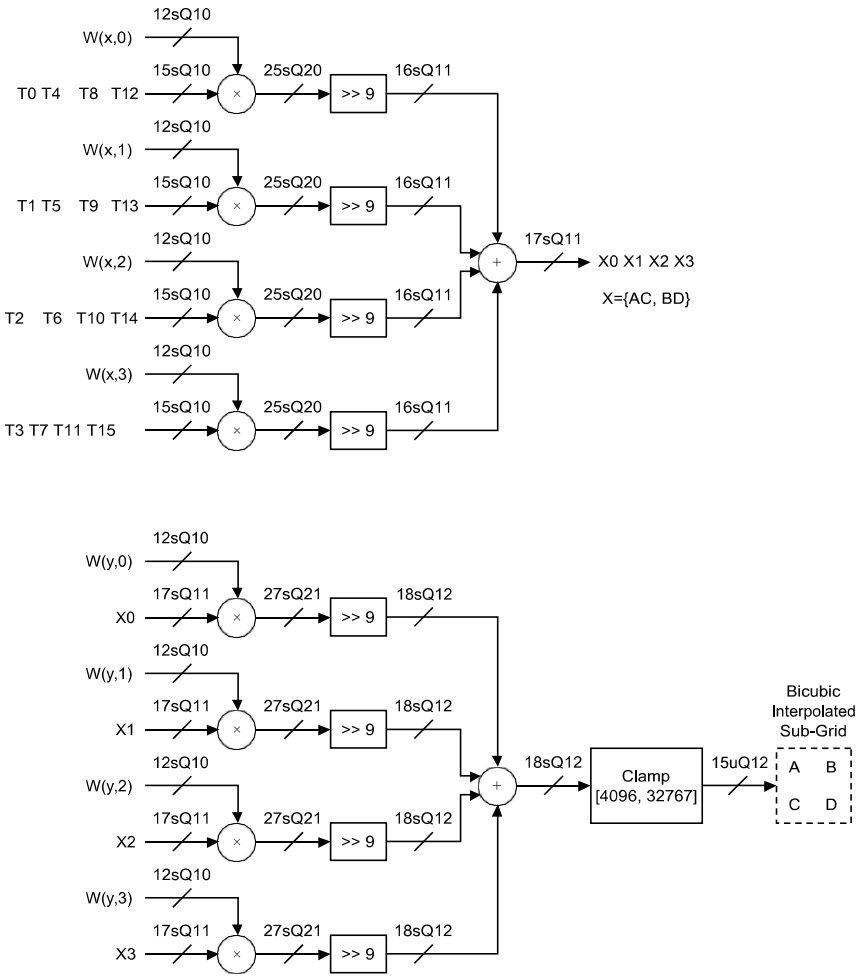


Figure 40-9 Bicubic interpolation of mesh subgrid correction factors

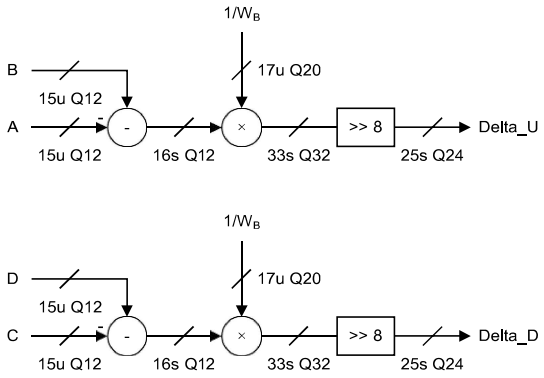


Figure 40-10 Computation of horizontal slopes of correction factors of a mesh subgrid

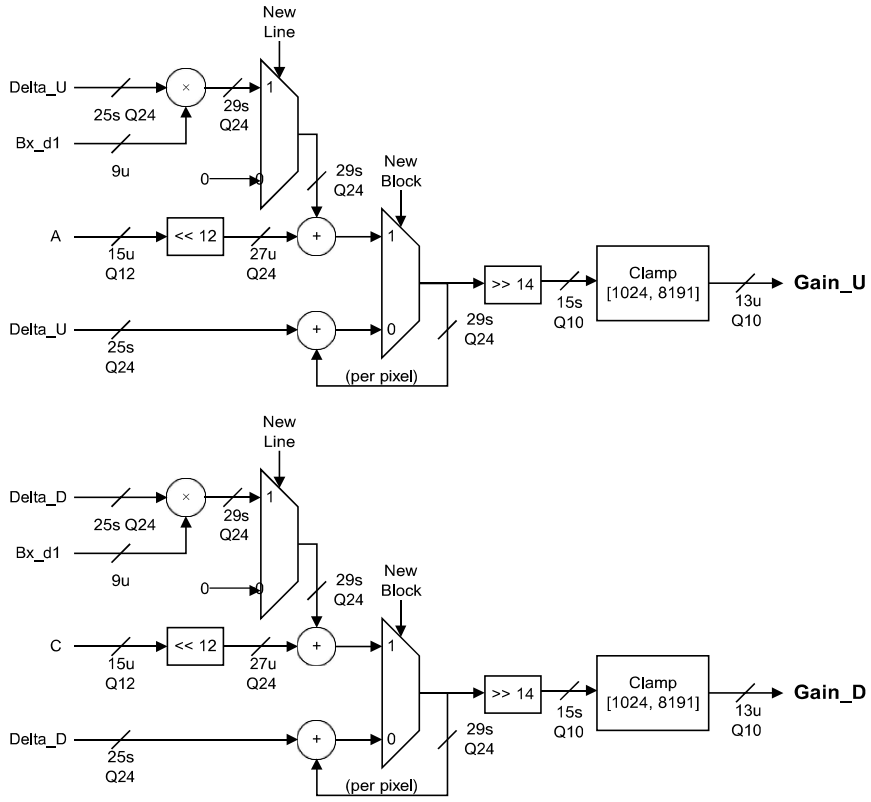


Figure 40-11 Horizontal bilinear interpolation of lens rolloff correction factor

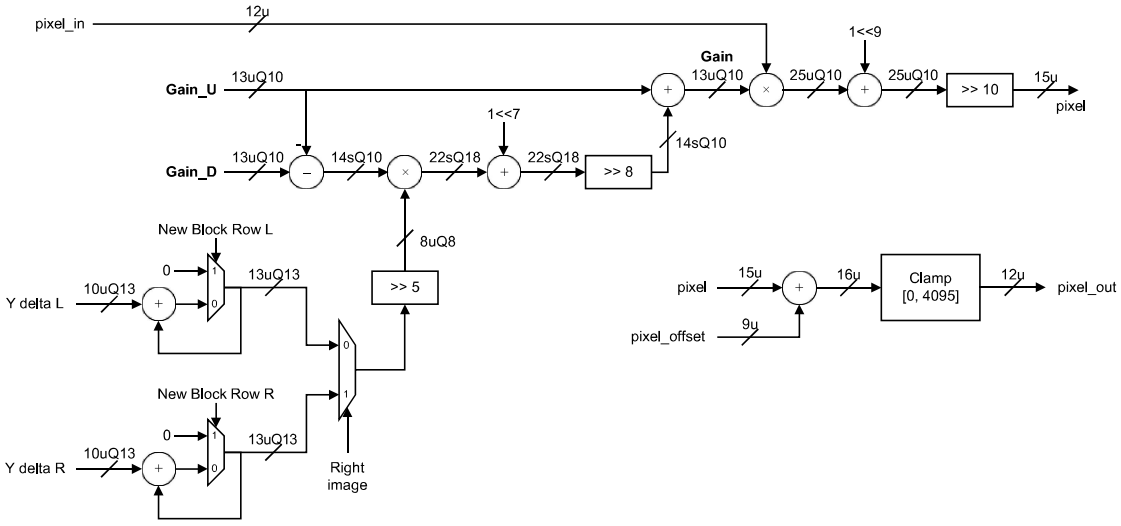


Figure 40-12 Vertical bilinear interpolation of lens rolloff correction factor

The calibration and generation of the tables are done in Chromatix. The calibration process first acquires flat field Bayer raw images under D65, TL84, and A lighting conditions. Then these flat field images are loaded into Chromatix to generate floating point rolloff tables. At run time, software converts these floating point tables into fixed point 13uQ10 tables and loads them into the VFE hardware.

The subgrid interpolation factor P=0, 1, 2, 3 specifies how many subgrids S=1, 2, 4, 8 per mesh grid. Due to hardware timing constraint, there is a minimum subgrid width limitation. Also it is preferable that the overhead of coverage of all the subgrids on the whole image be not exceed the width or height of a subgrid. With these criteria and possibly other constraints, P needs to be reduced from 3 down to 0 for the image widths and heights until all constraints are met. The Matlab code below can be used to decide P values for various image widths:

```
% Rolloff interpolation subgrid setup according to image size
P=3;  % Maximum subgrid factor
HG=12;  % Number of horizontal grids
% Horizontal direction
h=320:2:5376;  % Image size
gridh=h/(HG*2);  % Default grid size w/o interpolation, per channel
Ph=P*ones(1,length(h));  % Subgrid factor
Sh=2.^Ph;  % Number of subgrids
for p=P:-1:1
    subgridh=gridh./Sh;  % Subgrid size in float, per channel
    ceilsubh=ceil(subgridh);  % Subgrid size, integer, per channel
    hwgridh=ceilsubh.*Sh;   % Grid size, per channel
    hwh=hwgridh*(HG*2);  % Mesh size
    deltah=hwh-h;  % Overhead from image size
    fail=((ceilsubh<10) + ...
        (deltah>=hwgridh)) > 0;  % Reduce #subgrids if violate
    Ph=Ph-fail;  % Reduce subgrid factor
    Sh=2.^Ph;  % New number of subgrids
end
subgridh=gridh./Sh;  % Subgrid size in float, per channel
ceilsubh=ceil(subgridh);  % Subgrid size, integer, per channel
hwgridh=ceilsubh.*Sh;   % Grid size, per channel
hwh=hwgridh*(HG*2);  % Mesh size
```

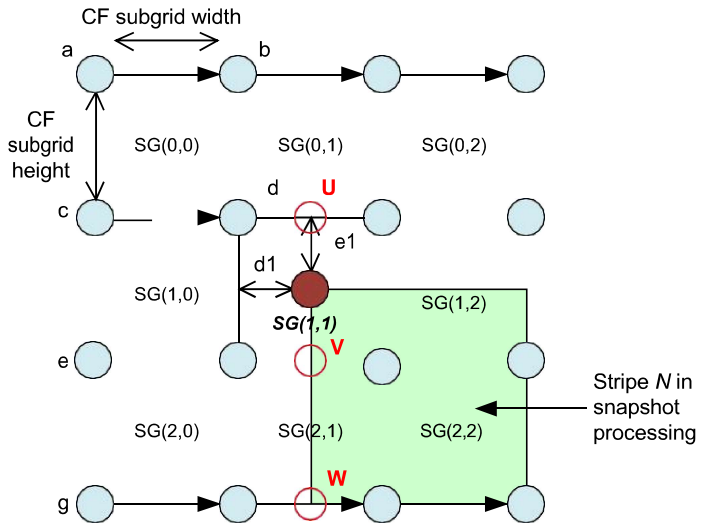


Figure 40-13 Lens roll-off correction in snapshot stripe processing mode

The proposed algorithm allows multi-chunk processing for performing lens rolloff correction. This is achieved by specifying the coordinate of the top left corner of the chunk (relative to the original entire frame). Figure 40-13 illustrates roll-off correction for stripe N in snapshot processing mode. The (x, y) coordinates of the top left pixel (i.e., d1, e1 of the dark circle in Figure 40-13) are loaded into the row and column counter at the beginning of the chunk. The

starting gain of a pixel line is initialized by multiplication of horizontal offset, Bx_d1, with subgrid gain slope for the upper and lower subgrid mesh gains U and V, as shown in Figure 40-11 and Figure 40-13. The (x, y) starting subgrid indices also need to be specified.

40.3 Analysis

In Figure 40-14, a flat field image (2048×1536) from a 3MP Micron sensor with non-radial roll-off distortion is used to test the algorithm. This is a tough test image because, not only is the roll-off not radial symmetric, the brightest spot (roll-off center) is skewed to the top.

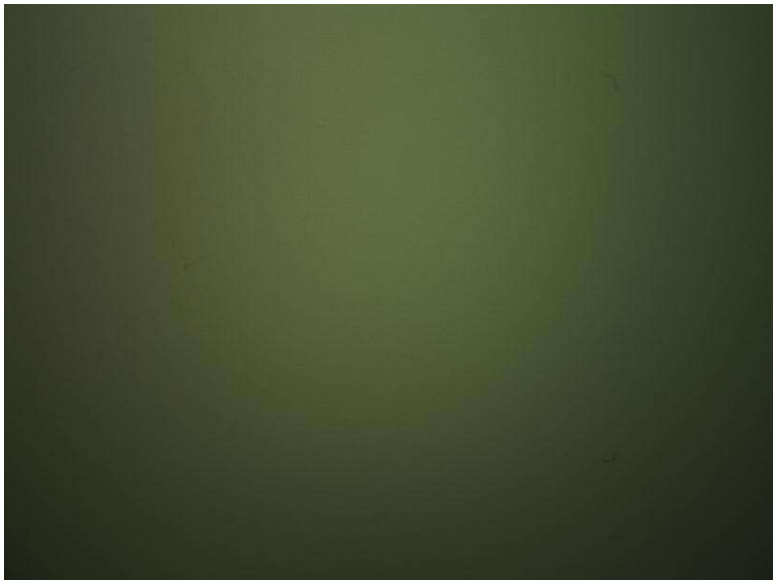


Figure 40-14 A flat field image with non-radial symmetric roll-off distortion

As Figure 40-15 illustrates, applying the roll-off correction method based on the radial symmetry assumption does not fully correct the distortion. This is expected because the basic assumption of radial symmetry is not met. In addition, the green channel mismatch causes the obvious artifacts. The LUT size is 224 in this image.

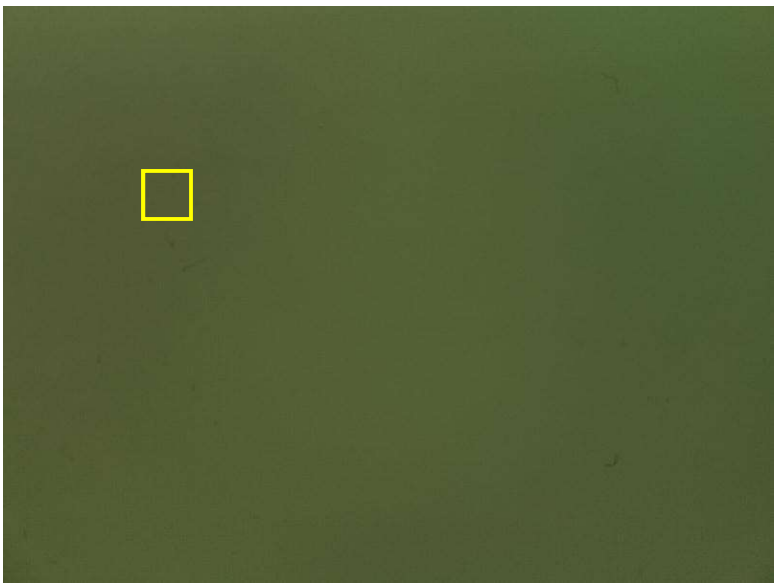


Figure 40-15 Image after roll-off correction based on radial symmetry

Figure 40-16 is a 200% magnified image of the block segment marked in Figure 40-15.

Figure 40-16 Radial symmetry roll-off correction at 200% scaling

However, the new four-channel mesh correction method reduces the channel mismatch and generates a uniform flat image, as shown in Figure 40-17.

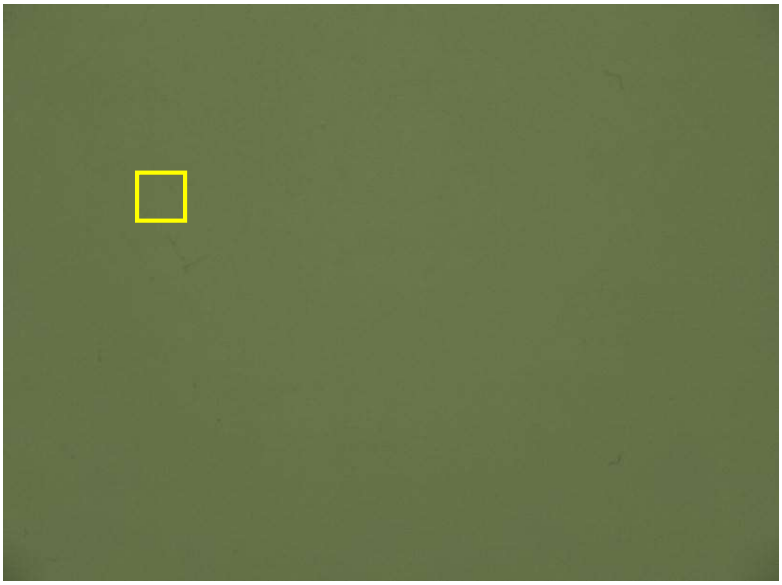


Figure 40-17 Corrected image using four-channel mesh correction

When applying the mesh surface matching method described in Section 40.2, the result in Figure 40-17) is much better than the radial symmetry correction result, but at the cost of increased hardware with a larger LUT size. Note that as the number of LUT entries decrease, the artifacts (vertical and horizontal grid lines) start to become visible because the coarse grids do not provide a good fit to the distortion surface.

Figure 40-18 is a 200% magnified image of the block segment marked in Figure 40-17.

Figure 40-18 Four-channel mesh correction at 200% scaling

Figure 40-19 is a very tough test image the roll-off is not radial symmetric. By applying the radial symmetric roll-off correction, there will be an obvious color shift in the processed image.