# Supplementary Material for Splatty- A Unified Image Demosaicing and Rectification Method

Pranav Verma, Dominique E. Meyer, Hanyang Xu, Falko Kuester
University of California San Diego
{pverma, dom, hax032, fkuester}@ucsd.edu

## 1 Appendix A: Dataflow

We present a high-level hardware dataflow implementation that highlights the efficiency of the proposed method. The Bayered input streamed gets mapped using an efficient LUT Decoder block to a set of output buffers given the decay function. As this implementation architecture prioritizes memory savings, the stream can be fully pipelined and demonstrates ideal dataflow. Future work can extend this forward-mapping approach to both ASIC and FPGA design to fully optimize the efficiency in hardware systems.

## 2 Appendix B: Reasoning for Rectification performance comparison

We used the algorithm described by [1] to get estimate the uncompressed LUT from camera parameters and image features. This same algorithm has been used by OpenCv's rectification function to generate their own Lookup Tables. Therefore, the reconstruction errors that we computed between our compressed and uncompressed lookup tables also correspond to the errors between our compressed LUT based rectification function and OpenCV's rectification function.

### 2.1 Demosaicing + Rectification pipeline Complete set of outputs

### 2.2 Rectification of various types of distortions

For the second quantitative comparison of rectification method that we described, we considered lens distortions (radial, tangential, radial and tangential) as well as projective distortions. We consider a high density checkerboard as the ground truth image (Figure 3). We then apply the above mentioned distortions, by using OpenCV's initUndistortRectifyMap to generate the distortion maps, using the following distortion coefficients:

- Radial Distortion Coefficients: [0.152962472, -0.488448363, 0.479883735]
- Tangential Distortion Coefficients: [-0.002962353, -0.000317416]

For projective distortion, we use a projective transformation matrix that has the following elements:

$$H = \begin{bmatrix} cos(-\pi/12) & sin(-\pi/12) & 3.91325630e+01 \\ sin(\pi/12) & cos(-\pi/12) & -4.03624934e+02 \\ 0 & 0 & 2/3 \end{bmatrix}$$

And transforming the pixels using this.

Using these distortion parameters, we create the distortion Look Up Tables (LUTs), and create distorted inputs using OpenCV's remap function. (Figure 4)

Then we invert these LUTs and feed those to our algorithm and to [1] for undistortion of these distorted images. (Figure 5). We then run a corner detector on these undistorted images, and on the ground truth image, and find the matches (based on corner points with least distance), and compute the distance between corresponding points)

## 3 Appendix C: Explanation of memory usage for Debayering and Rectification algorithms

For the rectification step, we consider three rectification implementations. First, we consider the backward mapping approach, which uses 2 $M \times N$ LUTs to store the rectification maps, and image buffers of size $M \times N$ each for the input and output images. Then we consider the approach of Oh and Kim ([4]), which decomposes the two $M \times N$ LUTs to three 1D LUTs of size M,N and C1 (where C1 was empirically determined to be 1024 for a $640 \times 480$ image, and we estimate it to be atleast 2048 for a $1920 \times 1080$ image), but need $M \times N$ space for the output image. Finally, we consider Junger et al.'s approach ([5]), for general rectification on FPGAs, which stores LUT of size $M \times N$, and buffers 1 row of output, and 50 rows of input image.

For the debayering step, we consider all the algorithms in Table 1. For single pass algorithms like bilinear interpolation, or Malvar et al [2], and double pass algorithms like Smooth Hue based demosaicing [6], we only need to store $K \times K$ interpolation filters, and buffer $K$ rows of the input image, and 1 row for each output channel (K=3 for bilinear, K=5 for others). Similarly, because these algorithms use

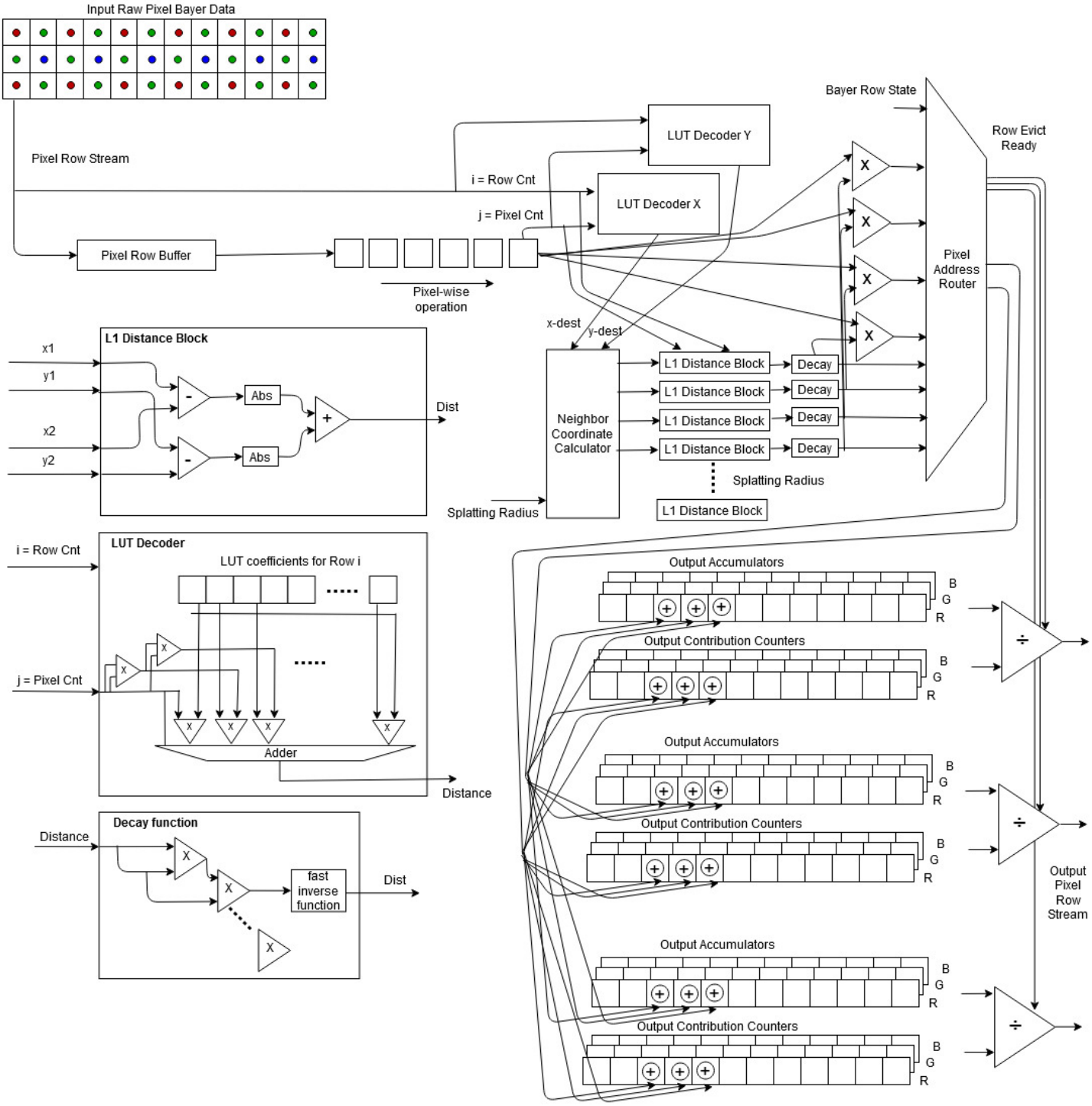



Figure 1: Dataflow

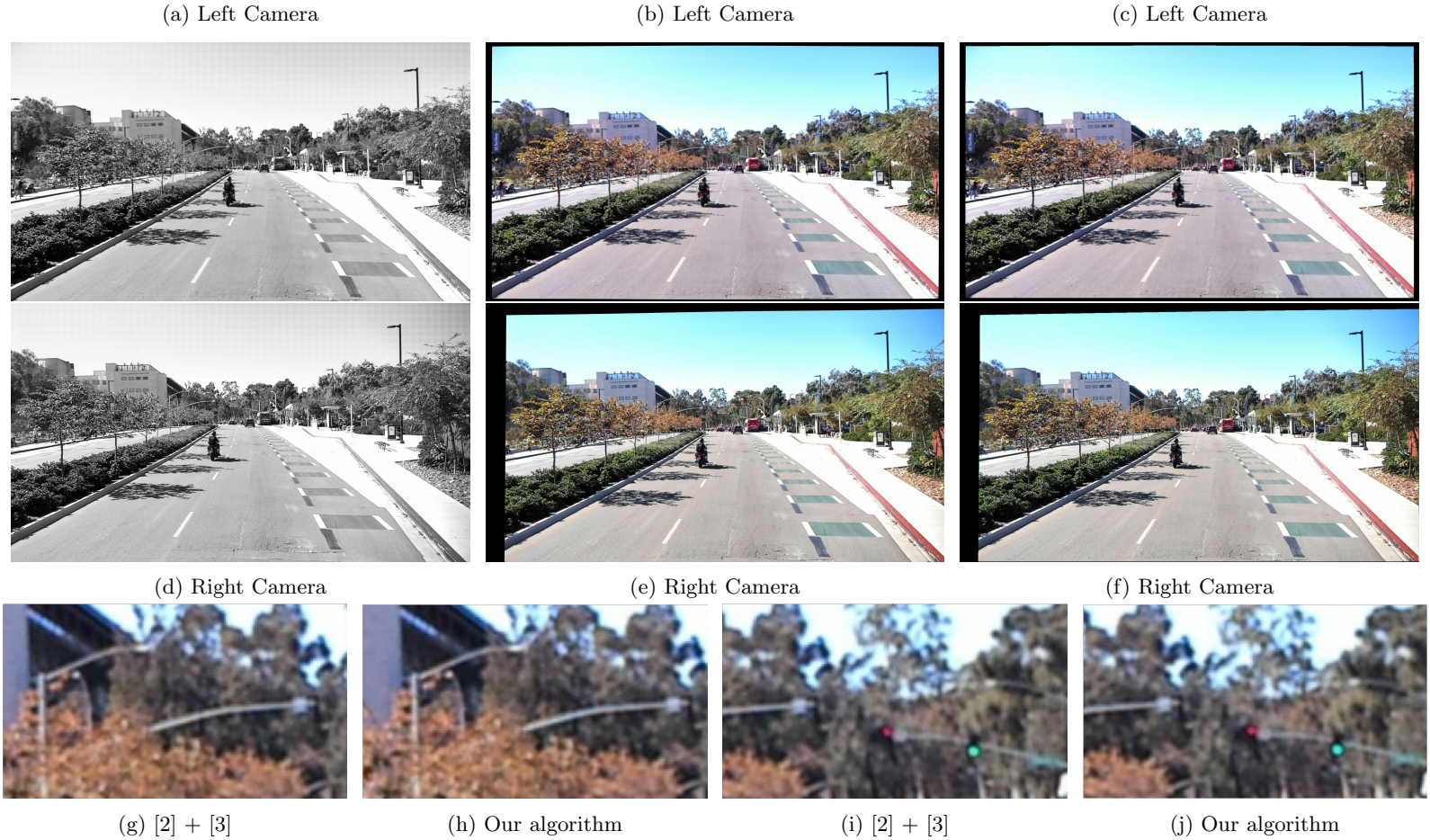



Figure 2: (a),(d) Raw, unrectified image pair from stereo camera, (b),(e) Debayered, then Stereo Rectified Image Pair using [2], and [3] (c),(f) Debayered + Stereo Rectified Output from our algorithm, (g)-(j) Close ups from our method and [2] + [3]

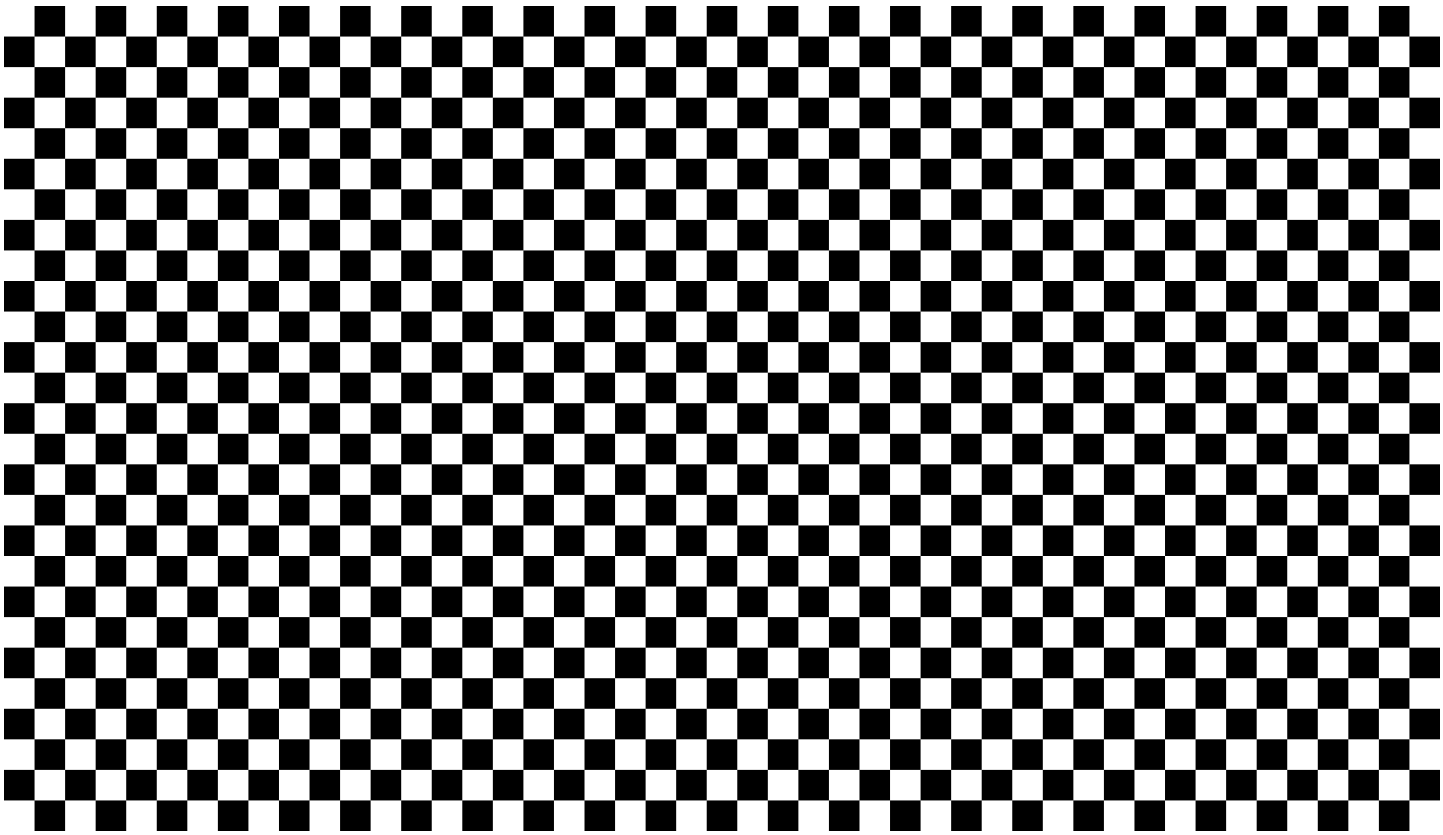
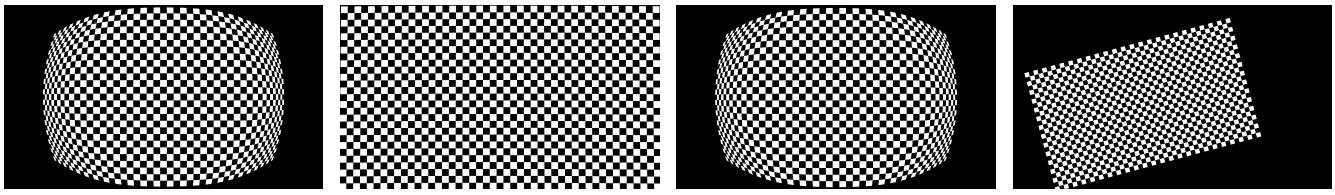


Figure 3: Ground truth image



(a) Radial Distorted Image (b) Tangentially distorted image (c) Radial + Tangentially distorted image (d) Projective Distorted Image

Figure 4: Distorted Images, which are fed to rectification pipelines

| Model | Parameters | Intermediate Feature Maps | Total Memory usage (KB) |
|---|---|---|---|
| SRCNN | 24416 | 4 * MN | 34540.095 |
| VDSR | 668227 | 6 * MN | 53254.98172 |
| Gharbi | 559776 | 5 * MN | 44731.345 |
| EDSR | 1500000 | 2406 * MN | 19496504.09 |
| DRRN | 297216 | 390* MN | 3162205.72 |
| Unet | 1734795 | 109* MN | 891721.263 |
| DPN(lite) | 4416128 | 614* MN | 4992695.22 |
| DMCNN-VD | 668227 | 6* MN | 53254.98172 |

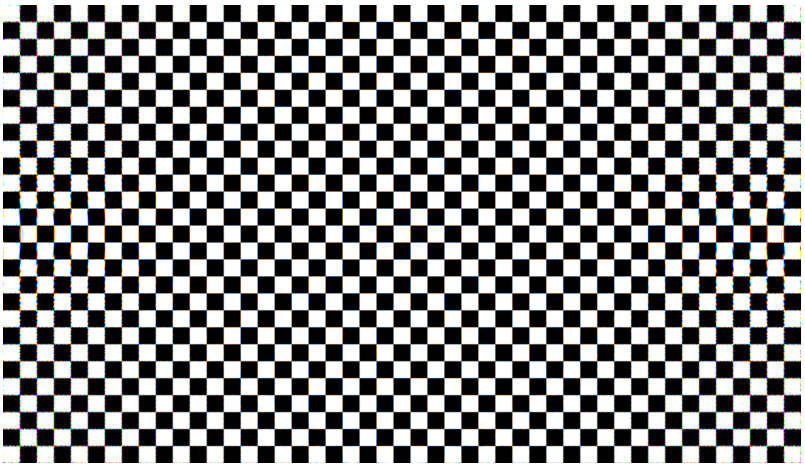
local gradient information, Hamilton's algorithm ([7]), can get by with only $K = 5$ rows per channel in input and output, and another $K$ rows for intermediate gradient computation. Similarly, Li and Orchard's method [8], we only need to buffer $K = 8$ rows for input and 1 per channel for output, and K intermediate rows. For the rest of the algorithms, buffering isn't a viable option, and therefore Gunturk's algorithm ([9]) requires $7.75 * M \times N$ of additional memory to store the intermediate outputs, and Kimmel's algorithm ([10]) needs $7 * M \times N$ additional memory. For the ML based algorithms, we estimate their memory usage by considering the total number of parameters in the model, and the input, output and any intermediate feature maps that would be needed in an optimized FPGA implementation (for eg. feature maps needed for skip connections). All the parameters and intermediate maps are stored as float32, so total memory usage will be (number of parameters + intermediate feature maps) * 4bytes

Finally, for our algorithm, we can compute the memory consumption as follows: from Figure **??**, we see that $6^{th}$ order polynomial is sufficient. Therefore, our lookup tables will have size $min(M, N) * (order + 1) * 2$ (one for $x$, one for $y$). For processing, we use a single row buffer for the input, and a max 50 rows for each channel in the output.

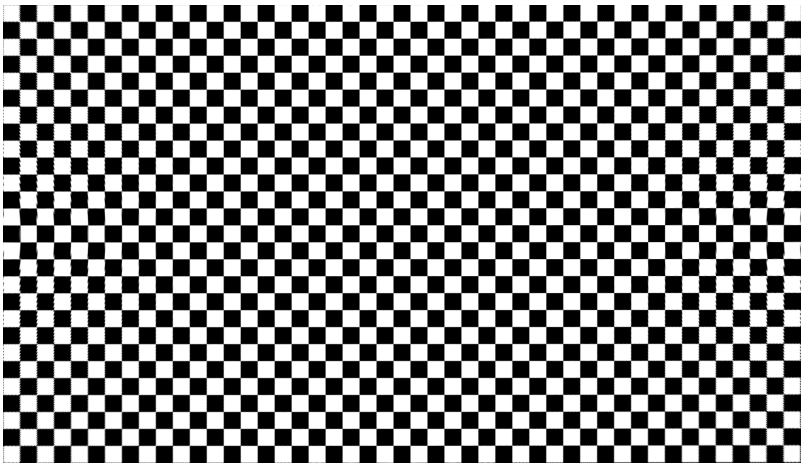
# 4 Append C: PSNR values of various Demosaicing algorithms on selected images from Kodak Dataset

Table 1: Debayering performance of various algorithms, on selected images from Kodak Dataset, measured by PSNR in decibels, the effective number of passes to produce the output, and the order of memory consumed. PSNR scores for all algorithms except ours, and Malvar et al, have been taken from [11], Table 3

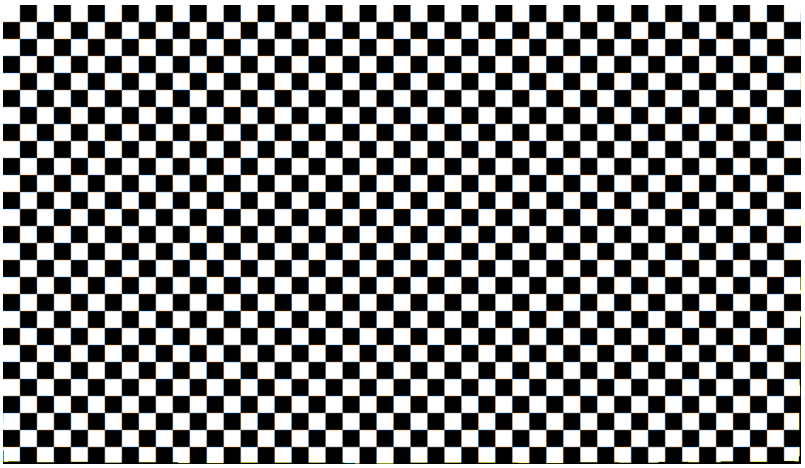
| Image \Algo | Bilinear | Cnst. hue | Malvar et al | Li | Kimmel | Hamilton | Gunturk | Proposed |
|---|---|---|---|---|---|---|---|---|
| 6 | 28.956 | 31.396 | 32.627 | 33.499 | 34.418 | 36.324 | **39.951** | 33.180 |
| 7 | 34.454 | 36.779 | 37.726 | 37.111 | 38.620 | 41.773 | **42.713** | 37.108 |
| 8 | 24.551 | 27.350 | 28.895 | 29.588 | 31.858 | 33.409 | **36.452** | 31.717 |
| 9 | 33.611 | 36.120 | 37.129 | 37.515 | 39.659 | 41.430 | **43.237** | 36.499 |
| 11 | 30.191 | 32.400 | 34.301 | 33.372 | 35.344 | 37.353 | **40.409** | 34.148 |
| 16 | 32.341 | 34.719 | 35.441 | 37.451 | 37.788 | 39.713 | **42.913** | 34.926 |
| 19 | 29.186 | 31.716 | 33.149 | 34.708 | 36.172 | 38.419 | **42.913** | 34.263 |
| 20 | 32.565 | 34.931 | 35.152 | 35.601 | 38.181 | 39.462 | **42.030** | 36.307 |
| 21 | 29.557 | 31.960 | 33.407 | 32.549 | 35.202 | 36.542 | **40.220** | 34.161 |
| 22 | 31.433 | 33.718 | 34.578 | 33.802 | 35.995 | 37.746 | **39.217** | 34.664 |
| 23 | 36.256 | 38.082 | 38.704 | 38.132 | 31.883 | 42.868 | **43.186** | 38.950 |
| 24 | 27.564 | 29.938 | 30.933 | 29.333 | 32.196 | 33.381 | **36.630** | 33.644 |
| Average (12 images) | 30.889 | 33.259 | 34.337 | 34.388 | 35.610 | 38.202 | **40.823** | 34.937 |
| Num Passes | **1** | 2 | **1** | 2 | 2 | 2 | 2 | **1** |
| Memory | O(mn) | O(mn) | O(mn) | O(mn) | O(mn) | O(mn) | O(mn) | **O(min(m,n))** |



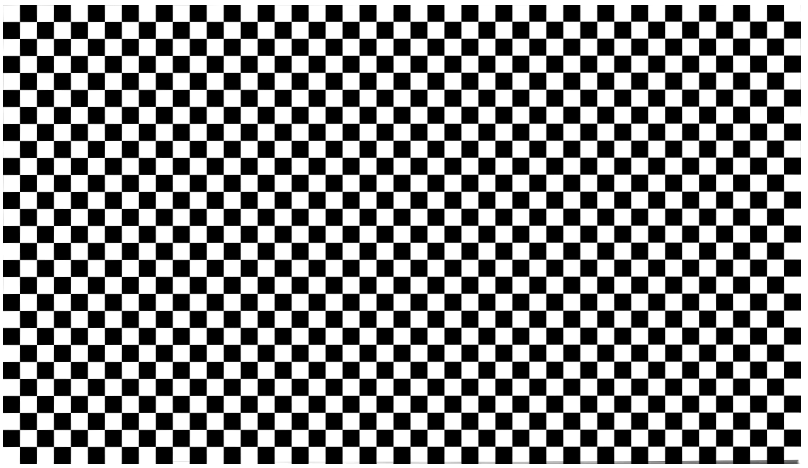
(a) Radial Corrected Image - Ours



(b) Radial Corrected Image - [1]

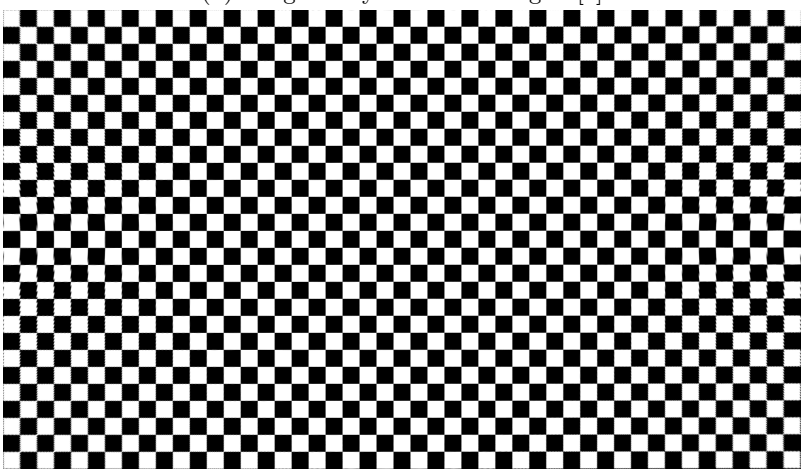


(c) Tangentially corrected image - Ours

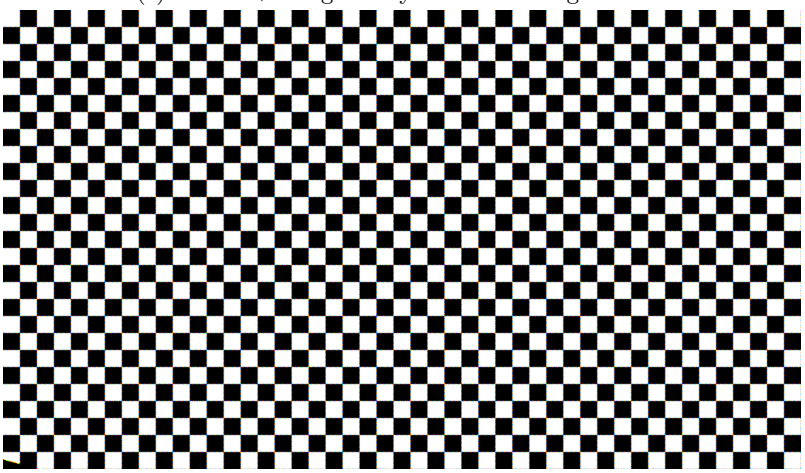


(d) Tangentially corrected image - [1]

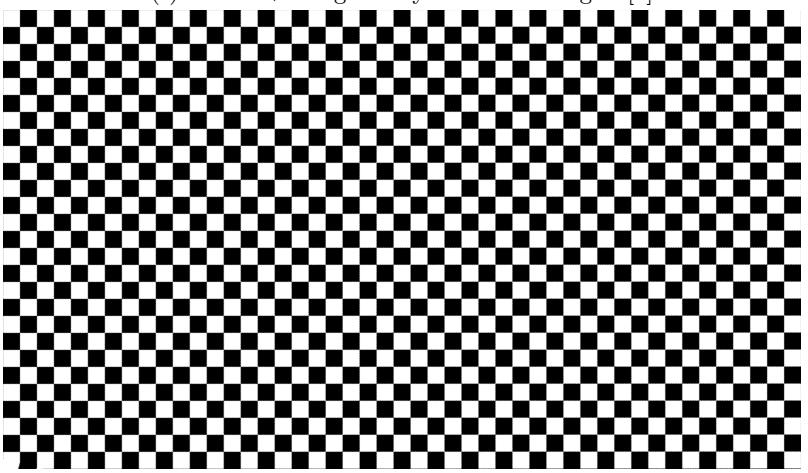
(e) Radial + Tangentially corrected image - Ours



(f) Radial + Tangentially corrected image - [1]



(g) Projective Corrected Image - Ours



(h) Projective Corrected Image - [1]

Figure 5: (a)(c)(e)(g) Corrected Images,from our pipeline and (b)(d)(f)(h) from [1]

# 5 Appendix D: Kodak Dataset Outputs from our algorithm and Malvar He Cutler's Algorithm [2]

We present a comparison of our algorithm's output with the output from Malvar et al.'s [2] algorithm, along with the original color images from Kodak Dataset [12]. We compare against Malvar et al.'s algorithm because it has the highest reconstruction accuracy among existing single pass algorithms, and the lowest memory footprint (as shown in the main paper). It is also the most frequently used debayering algorithm (part of the open-source image processing library, OpenCV).

From this, it is clear that our algorithm produces just as good results as the other algorithm, at a fraction of the memory usage (see the main paper for details).

(a) Original (b) Ours (c) Malvar et al.

Figure 6: Kodak Image 1

(a) Original (b) Ours (c) Malvar et al.

Figure 7: Kodak Image 2


(a) Original (b) Ours (c) Malvar et al.

Figure 8: Kodak Image 3

## References

[1] Richard I Hartley. Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127, 1999.

[2] H. S. Malvar, Li-wei He, and R. Cutler. High-quality linear interpolation for demosaicing of bayer-patterned color images. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages iii–485, May 2004.

(a) Original

(b) Ours

(c) Malvar et al.

Figure 9: Kodak Image 4

(a) Original

(b) Ours

(c) Malvar et al.

Figure 10: Kodak Image 5

(a) Original

(b) Ours

(c) Malvar et al.

Figure 11: Kodak Image 6

(a) Original

(b) Ours

(c) Malvar et al.

Figure 12: Kodak Image 7



(a) Original (b) Ours (c) Malvar et al.

Figure 13: Kodak Image 8



(a) Original (b) Ours (c) Malvar et al.

Figure 14: Kodak Image 9



(a) Original (b) Ours (c) Malvar et al.

Figure 15: Kodak Image 10



(a) Original (b) Ours (c) Malvar et al.

Figure 16: Kodak Image 11

(a) Original

(b) Ours

(c) Malvar et al.

Figure 17: Kodak Image 12

(a) Original

(b) Ours

(c) Malvar et al.

Figure 18: Kodak Image 13


(a) Original


(b) Ours


(c) Malvar et al.

Figure 19: Kodak Image 14

(a) Original

(b) Ours

(c) Malvar et al.

Figure 20: Kodak Image 15

(a) Original (b) Ours (c) Malvar et al.

Figure 21: Kodak Image 16

(a) Original (b) Ours (c) Malvar et al.

Figure 22: Kodak Image 17

(a) Original (b) Ours (c) Malvar et al.

Figure 23: Kodak Image 18

(a) Original (b) Ours (c) Malvar et al.

Figure 24: Kodak Image 19

(a) Original

(b) Ours

(c) Malvar et al.

Figure 25: Kodak Image 20

(a) Original

(b) Ours

(c) Malvar et al.

Figure 26: Kodak Image 21

(a) Original

(b) Ours

(c) Malvar et al.

Figure 27: Kodak Image 22

(a) Original

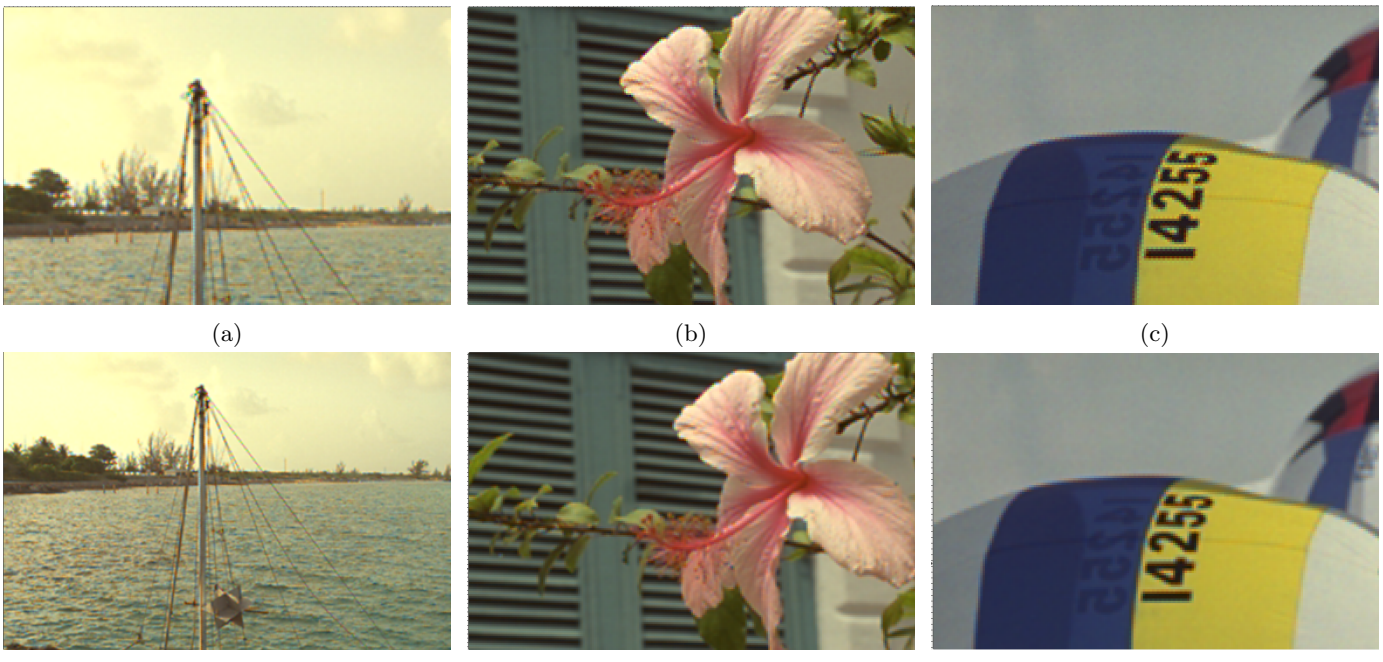
(b) Ours

(c) Malvar et al.

Figure 28: Kodak Image 23

(a) Original (b) Ours (c) Malvar et al.

Figure 29: Kodak Image 24



(a) (b) (c)

(d) (e) (f)

Figure 30: Debayering output on few images from Kodak dataset with (a)-(c) Algorithm proposed by Malvar et al [2]; (d)-(f) our algorithm

[3] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[4] Sungchan Oh and Gyeonghwan Kim. An architecture for on-the-fly correction of radial distortion using fpga. *Proceedings of SPIE - The International Society for Optical Engineering*, 03 2008.

[5] Christina Junger, Albrecht HeA, Maik Rosenberger, and Gunther Notni. FPGA-based lens undistortion and image rectification for stereo vision applications. In Maik Rosenberger, Paul-Gerald Dittrich, and Bernhard Zagar, editors, *Photonics and Education in Measurement Science 2019*, volume 11144, pages 284 – 291. International Society for Optics and Photonics, SPIE, 2019.

[6] David R. Cok. Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal.

[7] James E. Adams Jr. John F. Hamilton Jr. Adaptive color plane interpolation in single sensor color electronic camera.

[8] Xin Li and M. T. Orchard. New edge-directed interpolation. *IEEE Transactions on Image Processing*, 10(10):1521–1527, Oct 2001.

[9] Bahadir K Gunturk, Yucel Altunbasak, and Russell M Mersereau. Color plane interpolation using alternating projections. *IEEE transactions on image processing*, 11(9):997–1013, 2002.

[10] Ron Kimmel. Demosaicing: image reconstruction from color ccd samples. *IEEE Transactions on image processing*, 8(9):1221–1228, 1999.

[11] Olivier Losson, Ludovic Macaire, and Yanqin Yang. Comparison of color demosaicing methods. *Advances in Imaging and Electron Physics - ADV IMAG ELECTRON PHYS*, 162:173–265, 07 2010.

[12] Eastman Kodak Company. *Kodak LossLess True Color Image Suite*, 1999(accessed 23 June 2020).