Image Demosaicing (CSL-461 Project)

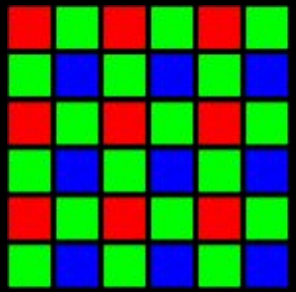
Abhinav Jindal
Chirag Khurana

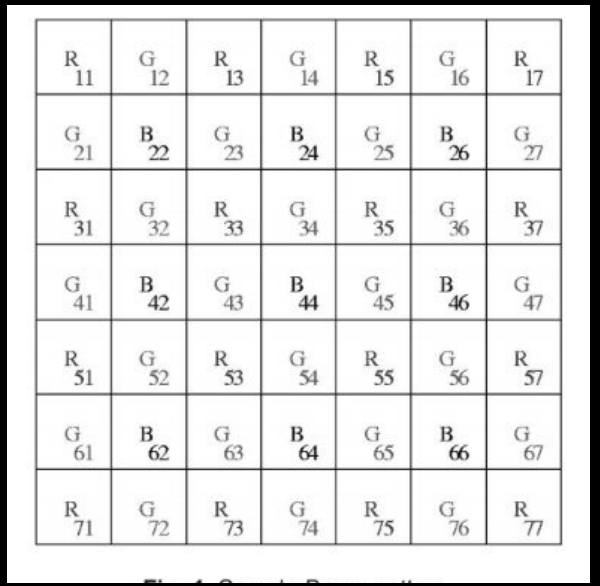
What is Image Demosaicing?

A demosaicing algorithm is a digital image process used to reconstruct a full color image from the incomplete color samples output from an image sensor overlaid with a color filter array (CFA). It is also known as CFA interpolation or color reconstruction.

Bayer Filter

A Bayer filter is a color filter array (CFA) for arranging RGB color filters on a square grid of photosensors. It used twice as many green elements as red or blue, because human eye is more sensitive to the green.





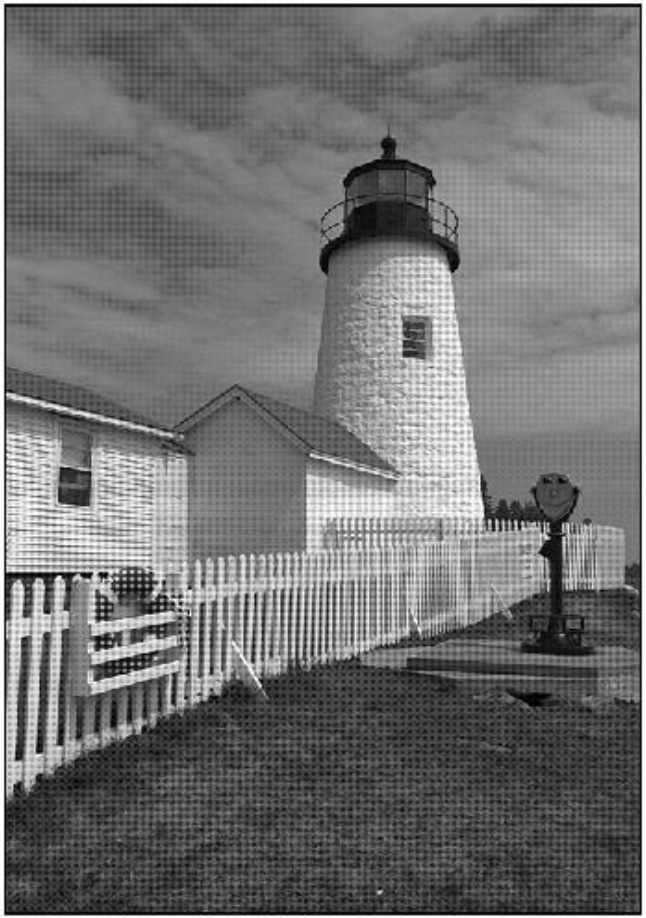
R 11	G 12	R 13	G 14	R 15	G 16	R 17
G 21	B 22	G 23	B 24	G 25	B 26	G 27
R 31	G 32	R 33	G 34	R 35	G 36	R 37
G 41	B 42	G 43	B 44	G 45	B 46	G 47
R 51	G 52	R 53	G 54	R 55	G 56	R 57
G 61	B 62	G 63	B 64	G 65	B 66	G 67
R 71	G 72	R 73	G 74	R 75	G 76	R 77

Algorithms

- Bilinear Interpolation
- Constant Hue based Interpolation
- Combination of NEDI and Kimmel
- Gradient based Interpolation
- Adaptive Color Plane Interpolation
- Zhang & Wu Directional LMMSE Demosaicing

original

bayer



Bayer Filter

Bilinear Interpolation

1. $G(i,j) = (G(i+1,j) + G(i-1,j) + G(i,j+1) + G(i,j-1))/4$

2. $R(i,j) = (R(i+1,j+1)+R(i-1,j-1)+R(i+1,j-1)+R(i-1,j+1))$ for blue nodes.

3. $R(i,j)=(R(i+1,j)+R(i-1,j))/2$ or $(R(i,j+1)+R(i,j-1))/2$ for green nodes.

4. Similarly for Blue.

Hue Based Interpolation

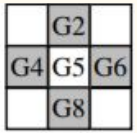
1. G is first interpolated by bilinear.
2. Then instead of using R itself in bilinear interpolation, we used the average of the ratios of R/G of neighbour pixels and multiplied with value of current G.
3. Similar for blue.

NEDI (New Edge Directed Interpolation)

1. 16 diamond neighbours are taken for the given pixel (c).
2. Then for each neighbour 4 diamond neighbours are taken (C).
3. R = (1/M*M)*transpose(C)*C and r = (1/M*M)*transpose(C)*c
4. alpha = inv(R)*r
5. Then alpha is used to calculate the weighted mean of the 4 nearest neighbours.

Kimmel Algorithm

1. Kimmel algorithm for green includes interpolating the values by taking weighted means of four nearest neighbours of the required pixel.



2. The weights are calculated by using the formula

   1/sqrt(1+D(P5)^2+D(Pi)^2)

   Where D(Pi)= $D_{xd}(P_5) = \max\left\{\left|\frac{P_3 - P_5}{\sqrt{2}}\right|, \left|\frac{P_7 - P_5}{\sqrt{2}}\right|\right\}$, $D_{yd}(P_5) = \max\left\{\left|\frac{P_1 - P_5}{\sqrt{2}}\right|, \left|\frac{P_9 - P_5}{\sqrt{2}}\right|\right\}$. ,the direction is chosen as the relative direction of Pi.

.

3. Then the R and B values are calculated by taking weighted means of ratios of red and green or blue and green from the calculated values of green for the nearest neighbours.

4. The weights are calculated by the same formula as that of green.

$$B_5 = G_5 \frac{\sum E_i \frac{B_i}{G_i}}{\sum E_i}, \quad i \neq 5,$$

$$R_5 = G_5 \frac{\sum E_i \frac{R_i}{G_i}}{\sum E_i}, \quad i \neq 5.$$

Combination of NEDI and Kimmel

1. Green pixels are first interpolated using NEDI method.

2. Then the obtained green pixels are used for the interpolation of red and blue pixels.

Gradient Based Interpolation

1. We use the average of values along the direction with less gradient value for green calculation.

2. After that,the red values are calculated by taking the averages of difference between green and red values and then adding the green value at that point.

3. Similarly for blue.

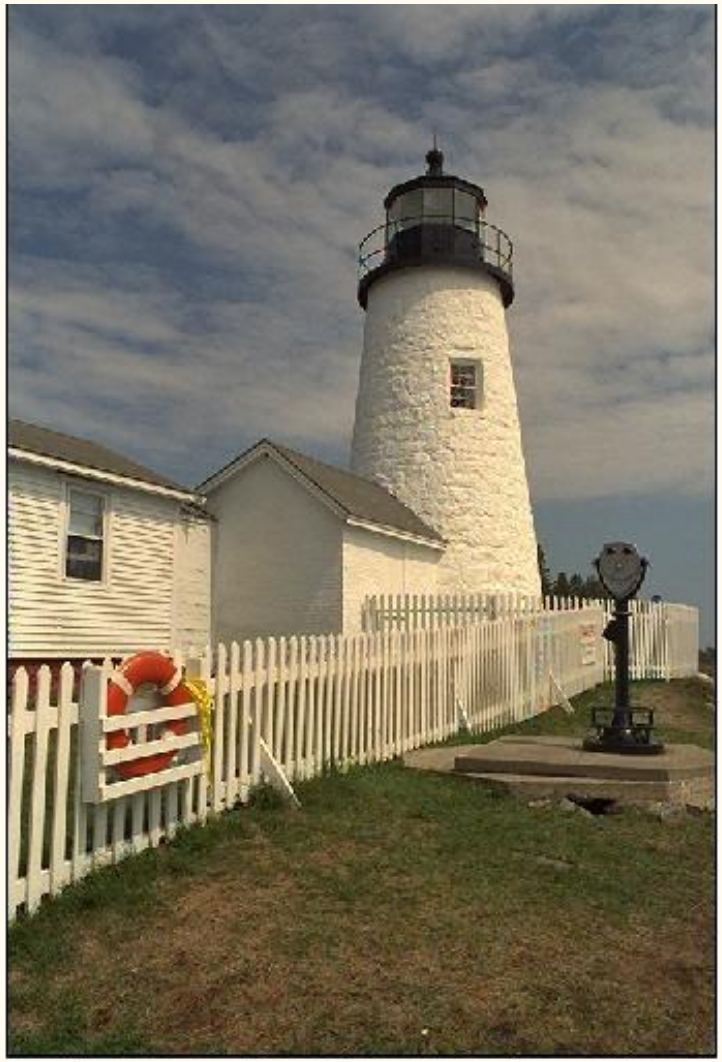
Adaptive Color Plane Interpolation

1. It's like the gradient based method but it also uses the second derivative in addition to first derivative.

2. For green calculation :
   α=abs(-A3+2A5-A7)+abs(G4-G6) and
   β=abs(-A1+2A5-A9)+abs(G2-G8).

$$G_5=\begin{cases}\frac{G_4+G_6}{2}+\frac{-A_3+2A_5-A_7}{2} & \text{if } \alpha<\beta \\ \frac{G_2+G_8}{2}+\frac{-A_1+2A_5-A_9}{2} & \text{if } \alpha>\beta \\ \frac{G_2+G_4+G_6+G_8}{4}+\frac{-A_1-A_3+4A_5-A_7-A_9}{8} & \text{if } \alpha=\beta\end{cases} \quad (7)$$

3. After calculating green we calculate red and blue using a similar concept.



Zhang-Wu's LMMSE Demosaicing

LMMSE(Linear Minimum Mean Square Error) estimation of PDS (Primary Difference Signal) is calculated .

LMMSE Estimation

Suppose X and Y are scalar random variables, and we wish to use the (assumed known) value of Y to estimate the (unknown) value of X. The minimum mean-square error (MMSE) estimate is

$$\hat{X}_{\text{MMSE}} := \arg\min_F \mathbb{E}\left[(F(Y) - X)^2\right],$$

$$\hat{X}_{\text{LMMSE}} := a(Y - \mathbb{E}Y) + b. \tag{2}$$

The optimal coefficients a and b minimize the mean-square error

$$\min_{a,b} \mathbb{E}\left[\left((a(Y - \mathbb{E}Y) + b) - X\right)^2\right]. \tag{3}$$

So the LMMSE estimate is

$$\hat{X}_{\text{LMMSE}} = \frac{\operatorname{Cov}(X,Y)}{\operatorname{Var} Y}(Y - \mathbb{E}Y) + \mathbb{E}X, \tag{6}$$

and its mean-square error is

$$\begin{aligned}\mathbb{E}\left[(\hat{X}_{\text{LMMSE}} - X)^2\right] &= a^2 \operatorname{Var} Y - 2a \operatorname{Cov}(X,Y) + \operatorname{Var} X \\ &= \operatorname{Var} X - \frac{\operatorname{Cov}(X,Y)^2}{\operatorname{Var} Y}.\end{aligned} \tag{7}$$

PDS(Primary Difference Signal)

We make a assumption that the difference images between the green and blue channels are low-pass signals, which are referred in the sequel as primary difference signals and denoted by

$$\Delta_{g,r}(n) = G_n - R_n; \quad \Delta_{g,b}(n) = G_n - B_n$$

where n is the position index of pixels.

Algorithm Outline

1. The primary difference signals are estimated in the horizontal direction and denoised using LMMSE estimation.
2. Similarly, the primary difference signals are estimated and denoised in the vertical direction.
3. The two denoised directional estimates are fused to obtain a full resolution green channel.
4. The primary difference signals are interpolated and then subtracted from green to obtain the red and blue channels.

Directional Estimates

The unknown pixel value are estimated using Second-order Laplacian interpolation.

$$\hat{G}_n = \tfrac{1}{2}(G_{n-1} + G_{n+1}) - \tfrac{1}{4}(R_{n-2} - 2R_n + R_{n+2}),$$
$$\hat{R}_n = \tfrac{1}{2}(R_{n-1} + R_{n+1}) - \tfrac{1}{4}(G_{n-2} - 2G_n + G_{n+2}),$$

which provides an estimate of the difference signal,

$$\hat{\Delta}^h_{g,r}(i) = \begin{cases} \hat{\mathrm{G}}^h_i - \mathrm{R}^h_i, & \mathrm{G \text{ is interpolated}} \\ \mathrm{G}^h_i - \hat{\mathrm{R}}^h_i, & \mathrm{R \text{ is interpolated}} \end{cases} \quad \text{and}$$
$$\hat{\Delta}^v_{g,r}(i) = \begin{cases} \hat{\mathrm{G}}^v_i - \mathrm{R}^v_i, & \mathrm{G \text{ is interpolated}} \\ \mathrm{G}^v_i - \hat{\mathrm{R}}^v_i, & \mathrm{R \text{ is interpolated.}} \end{cases}$$

The estimation errors associated with $\hat{\Delta}^h_{g,r}$ and $\hat{\Delta}^v_{g,r}$ are

$$\begin{cases} \varepsilon^h_{g,r} = \Delta_{g,r} - \hat{\Delta}^h_{g,r} \\ \varepsilon^v_{g,r} = \Delta_{g,r} - \hat{\Delta}^v_{g,r}. \end{cases}$$

Directional Estimates(Continue)

To simplify the notations, we denote by x the true PDS signal and by y the associated observation and by v, the associated demosaicking noise, namely

$$y(n) = x(n) + v(n).$$

The LMMSE of x is computed as

$$\hat{x} = E[x] + \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(y)}(y - E[y]).$$

We can simplify to

$$\hat{x} = \mu_x + \frac{\sigma_x^2}{(\sigma_x^2 + \sigma_v^2)}(y - \mu_x)$$

Directional Estimates(Continue)

The response sequence of a Gaussian low-pass filter, we have

$$y_s(n) = (y * h)(n) = \sum_{k=-\infty}^{\infty} y(n-k) \cdot h(k) \qquad (3.4)$$

where "*" is the convolution operator. In this paper, we set $\{h(k)\}$ to be the Gaussian smooth filter, whose coefficients are

$$h(k) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{k^2}{2\sigma^2}} \qquad (3.5)$$

The low-pass filter output is a weighted average of y(n) and its neighbors, and it is much closer to x(n) than y(n)

$$Y_n^s = [y_s(n-L) \ \cdots \ y_s(n) \ \cdots \ y_s(n+L)] \qquad (3.6)$$

the $2L+1$-dimensional vector centered at $y_s(n)$, we estimate $\mu_x(n)$ as

$$\mu_x(n) = \frac{1}{2L+1} \sum_{k=1}^{2L+1} Y_n^s(k) \qquad (3.7)$$

and then estimate $\sigma_x^2(n)$, the variance of $x(n)$, by

$$\sigma_x^2(n) = \frac{1}{2L+1} \sum_{k=1}^{2L+1} (Y_n^s(k) - \mu_x(n))^2. \qquad (3.8)$$

Directional Estimates(Continue)

$$Y_n = [y(n-L) \ \cdots \ y(n) \ \cdots \ y(n+L)] \tag{3.9}$$

the $2L+1$-dimensional vector centered at $y(n)$. Since $y_s(n)$ is an approximation of $x(n)$, it follows that $y(n) - y_s(n)$ is an approximation of $v(n)$; thus, we can estimate $\sigma_v^2(n)$, the variance of $v(n)$, by

$$\sigma_v^2(n) = \frac{1}{2L+1} \sum_{k=1}^{2L+1} (Y_n^s(k) - Y_n(k))^2. \tag{3.10}$$

For each sample $x(n)$ to be estimated, the corresponding parameters $\mu_x(n)$, $\sigma_x^2(n)$ and $\sigma_v^2(n)$ are computed and substituted into (2.12) to yield $\hat{x}(n)$, the nearly LMMSE estimate of $x(n)$. Let $\tilde{x}(n)$ be the estimation error of $x(n)$: $\tilde{x}(n) = x(n) - \hat{x}(n)$, the variance of $\tilde{x}(n)$ is

$$\sigma_{\tilde{x}}^2(n) = E[\tilde{x}^2(n)] = \sigma_x^2(n) - \frac{\sigma_x^2(n)}{(\sigma_x^2(n) + \sigma_v^2(n))}. \tag{3.11}$$

Optimal Fusion of the Directional LMMSE

$$\hat{x}_w(n) = w_h(n) \cdot \hat{x}_h(n) + w_v(n) \cdot \hat{x}_v(n) \qquad (4.2)$$

where $w_h(n) + w_v(n) = 1$. The weights $w_h(n)$ and $w_v(n)$ are determined to minimize the MSE of $\hat{x}_w(n)$

$$w_h(n) = \frac{\sigma^2_{\tilde{x}_v}(n)}{\sigma^2_{\tilde{x}_h}(n) + \sigma^2_{\tilde{x}_v}(n)}, \quad w_v(n) = \frac{\sigma^2_{\tilde{x}_h}(n)}{\sigma^2_{\tilde{x}_h}(n) + \sigma^2_{\tilde{x}_v}(n)}, \qquad (4.7)$$

Interpolation of pixels

For Green pixel:

$$\hat{G}_n = R_n + \Delta_{g,r}(n) \text{ or } \hat{G}_n = B_n + \Delta_{g,b}(n).$$

For Missing Red (Blue) Samples at the Blue (Red) Sample Positions:

$$\Delta_{n,gr} = \frac{\Delta^{nw}_{n,gr} + \Delta^{se}_{n,gr} + \Delta^{ne}_{n,gr} + \Delta^{sw}_{n,gr}}{4}.$$

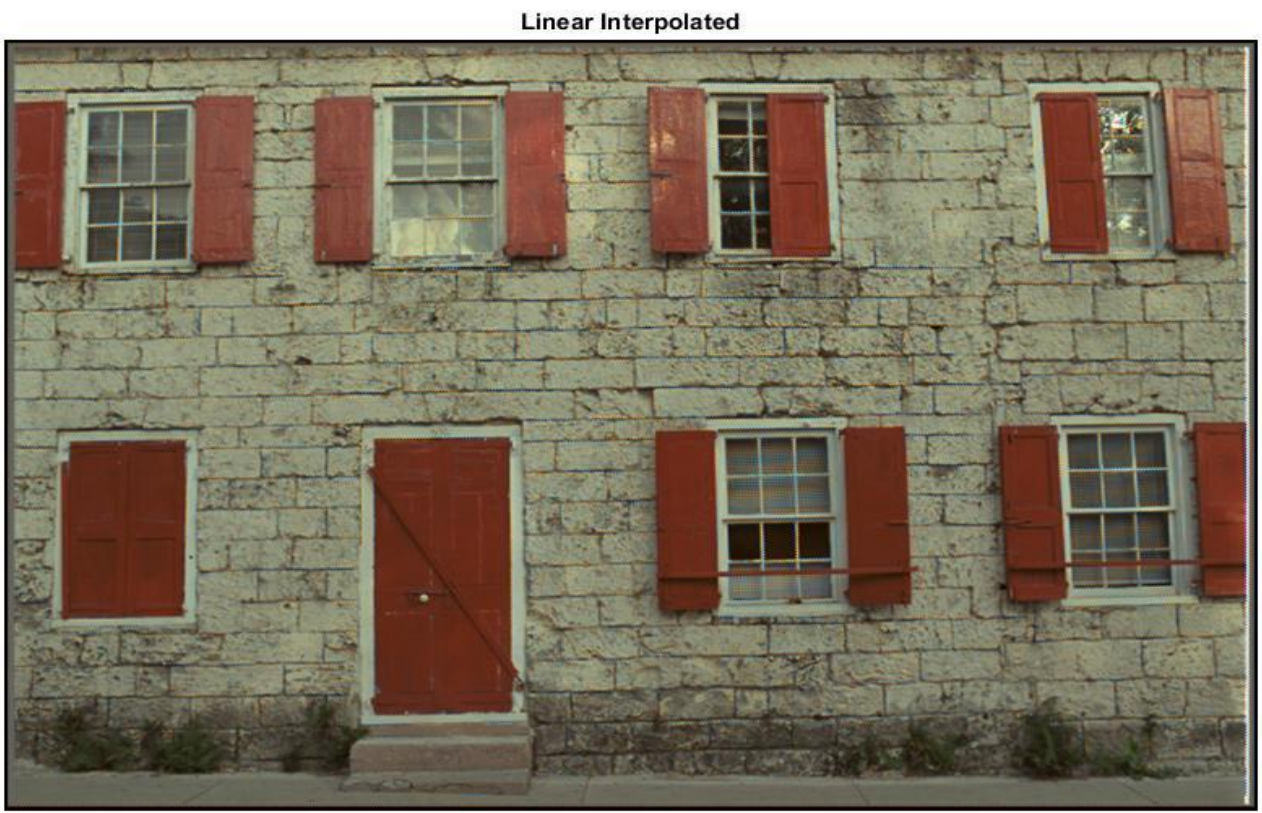
$$\hat{R}_n = \hat{G}_n - \Delta_{n,gr}.$$

$$\hat{B}_n = \hat{G}_n - \Delta_{n,gb}.$$

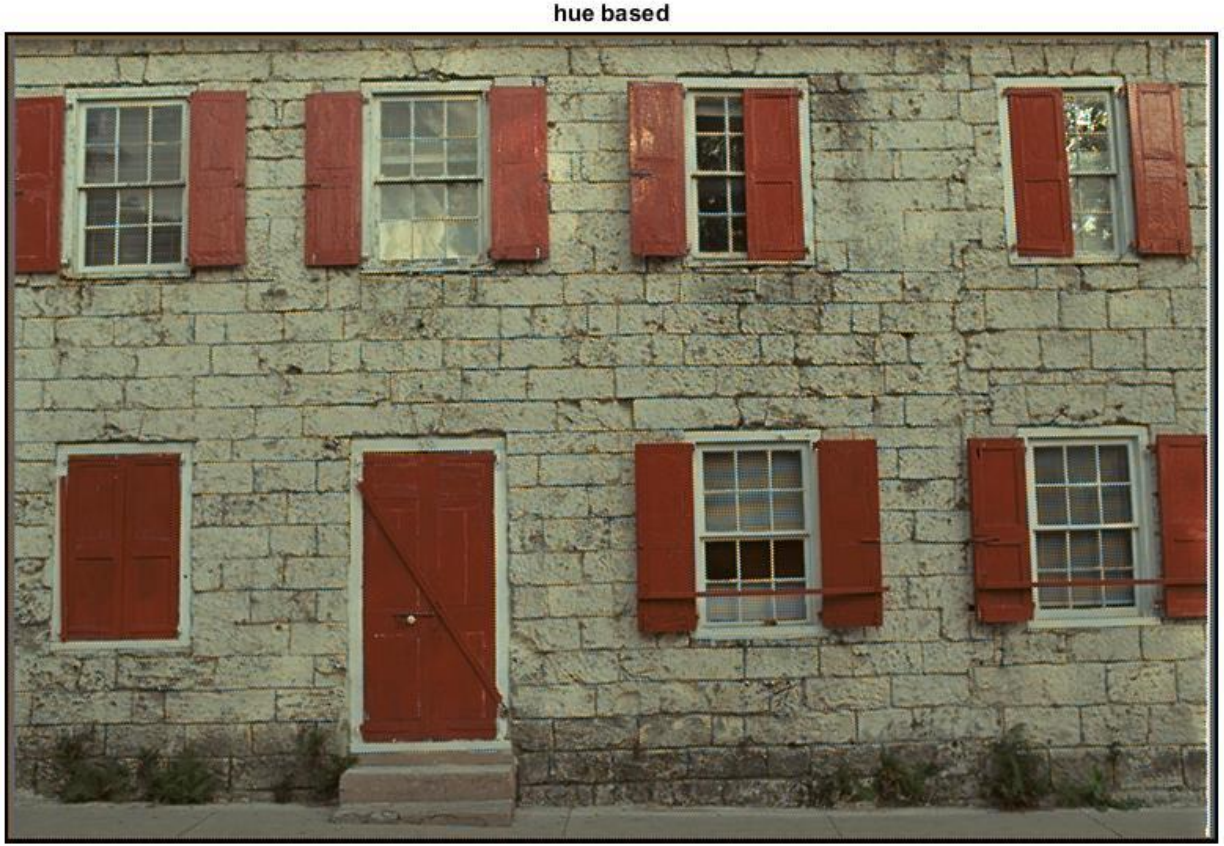
And similarly at green positions.

Sample Image 1

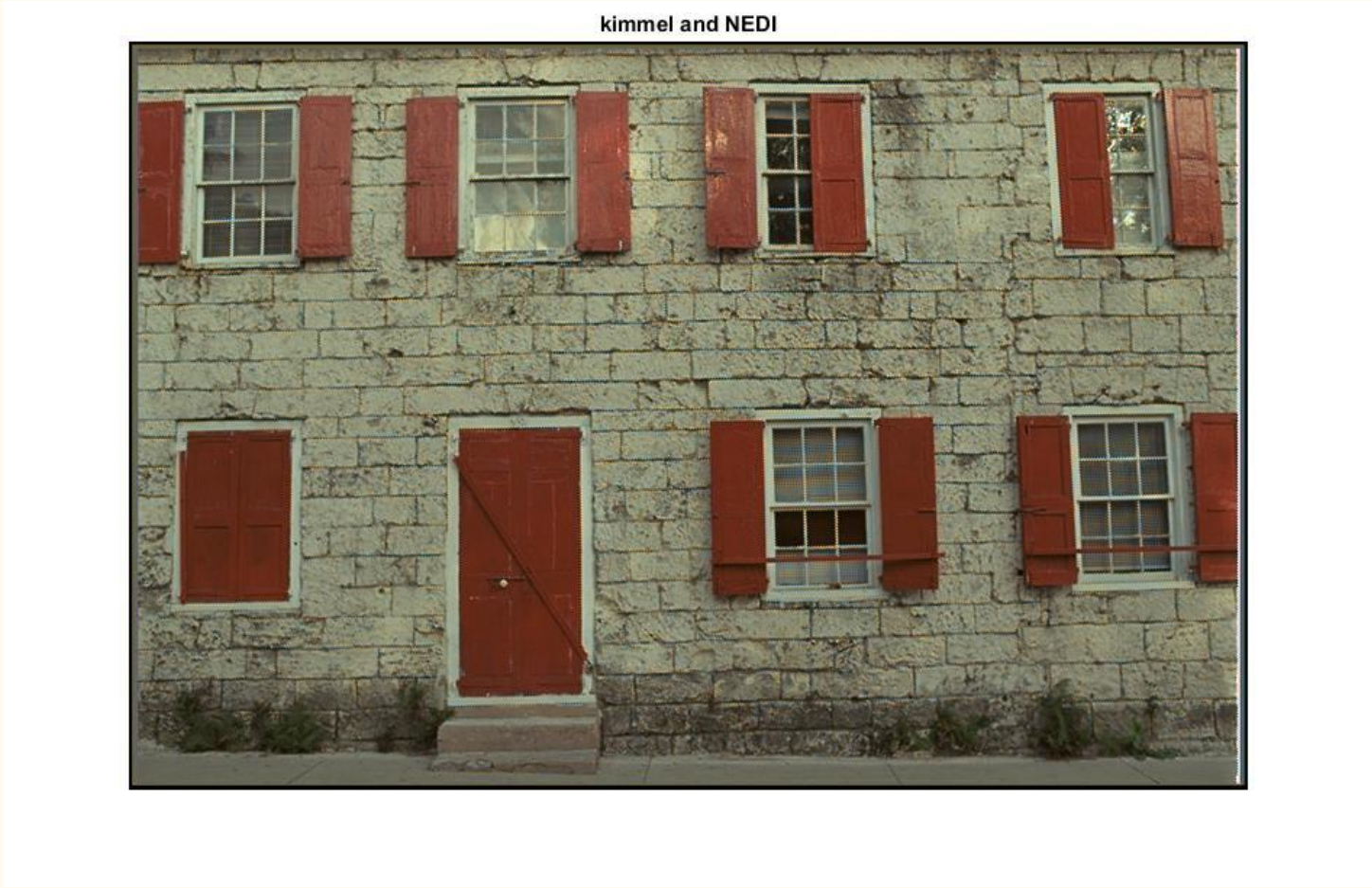
Sample Image 1



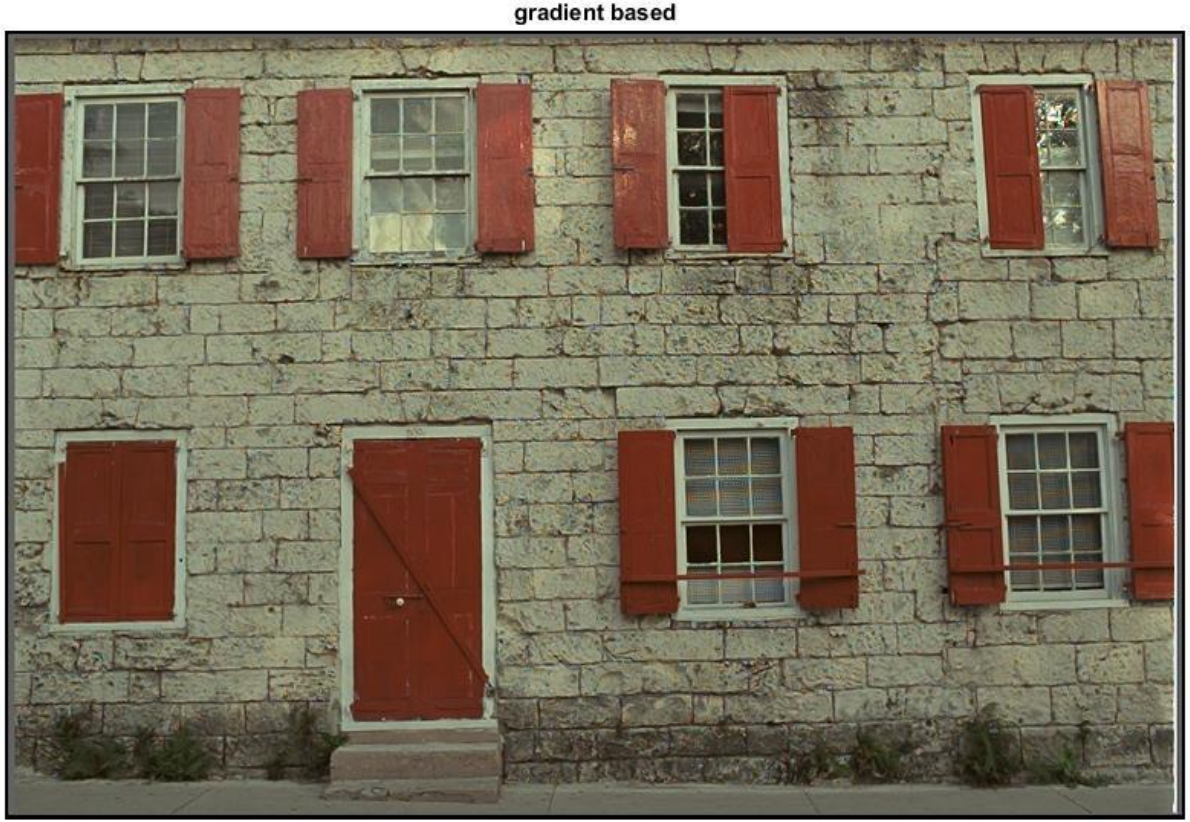
Sample Image 1



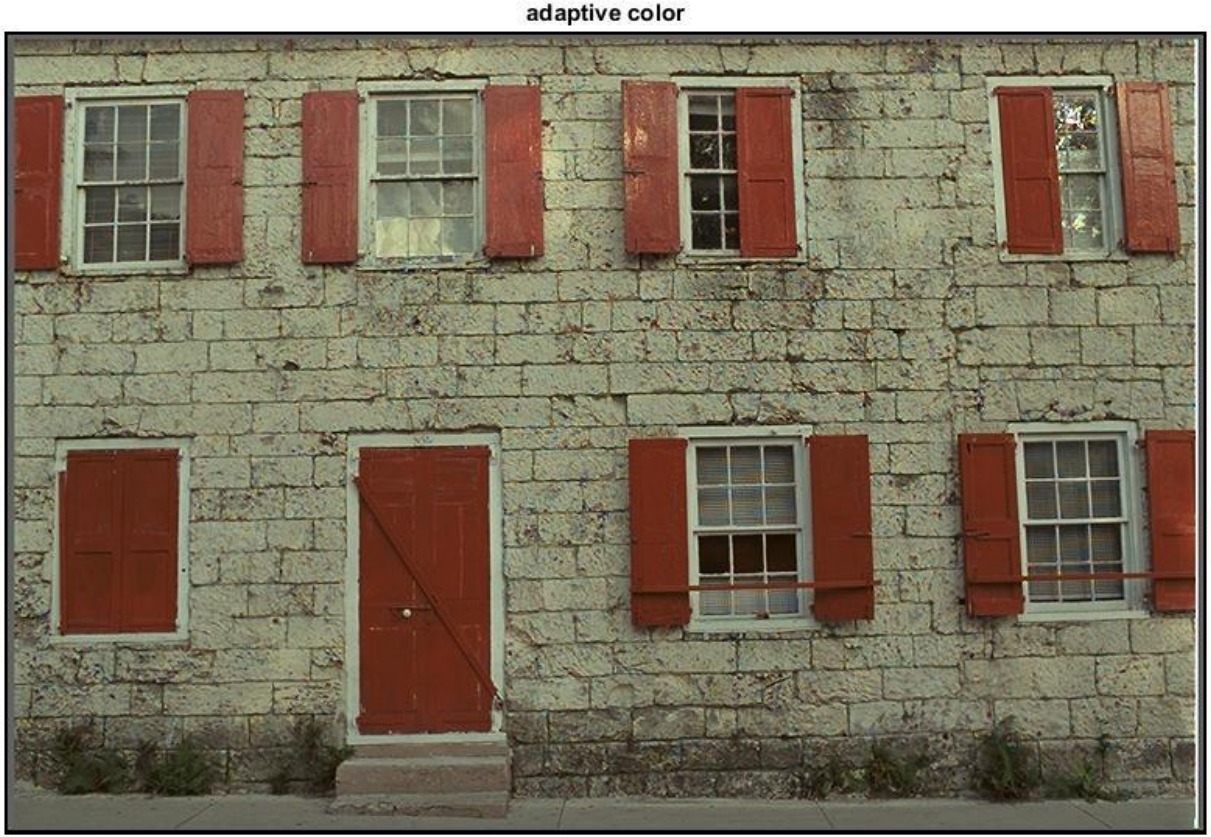
Sample Image 1



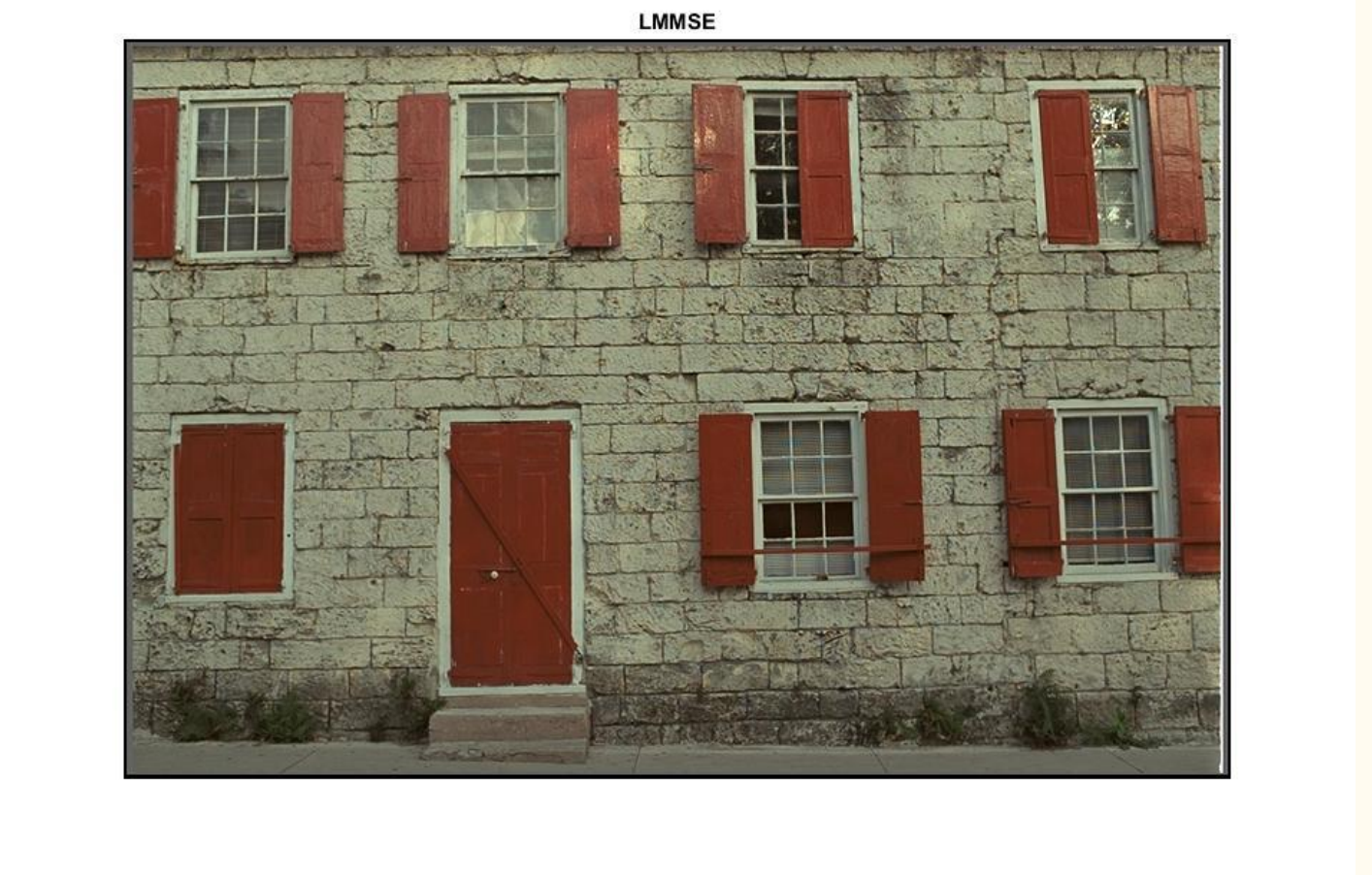
Sample Image 1



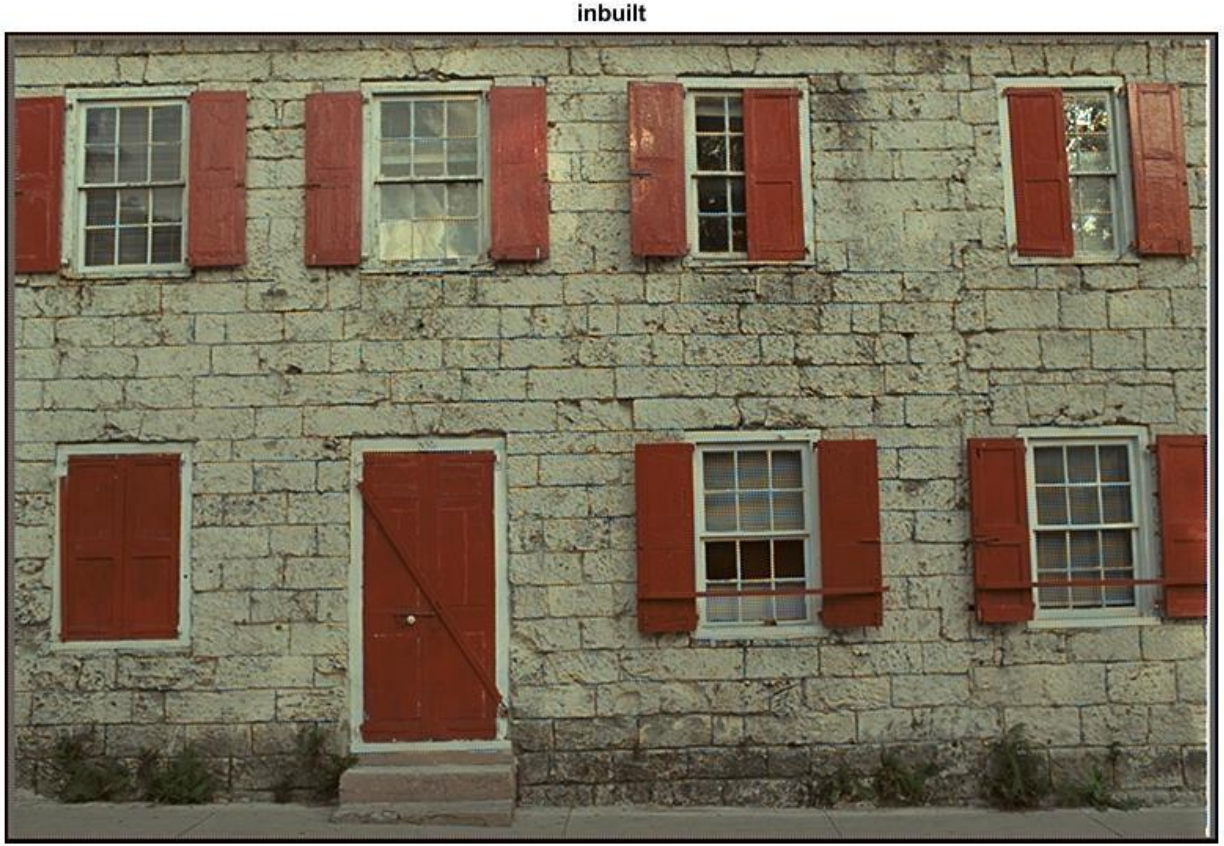
Sample Image 1



Sample Image 1



Sample Image 1



Sample Image 1

S.No.	Method	PSNR
1	Linear Interpolation	25.02
2	Hue Based	28.39
3	Kimmel and NEDI	28.63
4	Gradient Based	30.91
5	Adaptive Color	31.16
6	Wu’s LMMSE	**38.39**
7	Inbuilt	30.99

Sample Image 2

Linear Interpolated

hue based

kimmel and NEDI

gradient based

adaptive color

LMMSE

inbuilt

Sample Image 2

S.No.	Method	PSNR
1	Linear Interpolation	25.57
2	Hue Based	28.29
3	Kimmel and NEDI	28.99
4	Gradient Based	31.32
5	Adaptive Color	30.73
6	Wu’s LMMSE	**38.75**
7	Inbuilt	32.83

References

1. Demosaicking methods for Bayer color arrays. Link
2. High-Quality Algorithm for Bayer Pattern Interpolation. Link
3. Color demosaicking via directional linear minimum mean square-error estimation. Link
4. http://www.ipol.im/pub/art/2011/g_zwld/article.pdf
5. http://www.cis.upenn.edu/~danielkh/files/2013_2014_demosaicing/demosaicing.html
6. Kodak standard data set.


The END