Bayer's Pattern

Narendiran S

06-06-2020

1 Use Case

A Bayer filter mosaic is a color filter array (CFA) for arranging RGB color filters on a square grid of photosensors. Its particular arrangement of color filters is used in most single-chip digital image sensors used in digital cameras, camcorders, and scanners to create a color image. The filter pattern is half green, one quarter red and one quarter blue.

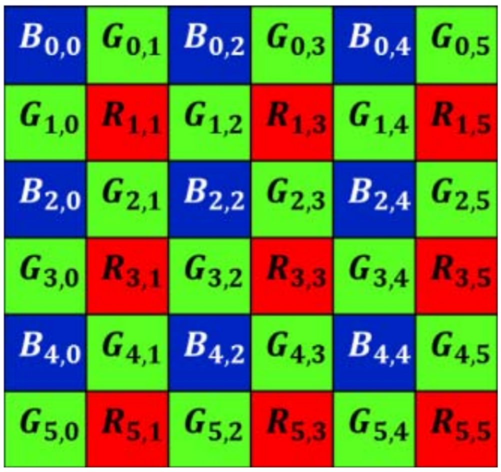


Figure 1: Bayer Pattern

2 Creation of Bayer Pattern Image in PYthon

```
bayerImg = np.zeros((actualImg.shape[0], actualImg.shape[1]),
 ↪  dtype=np.uint8)

# Blue values - odd rows and odd columns
bayerImg[::2, ::2] = actualImg[::2, ::2, 2]
bayerImg2[::2, ::2, 2] = actualImg[::2, ::2, 2]
```

```

# Red values - even rows and even columns
bayerImg[1::2, 1::2] = actualImg[1::2, 1::2, 0]
bayerImg2[1::2, 1::2, 0] = actualImg[1::2, 1::2, 0]

# Green values - BGBGBG....
bayerImg[1::2, ::2] = actualImg[1::2, ::2, 1]
bayerImg2[1::2, ::2, 1] = actualImg[1::2, ::2, 1]

# Green values - GRGRGR....
bayerImg[::2, 1::2] = actualImg[::2, 1::2, 1]
bayerImg2[::2, 1::2, 1] = actualImg[::2, 1::2, 1]
```

3 Example

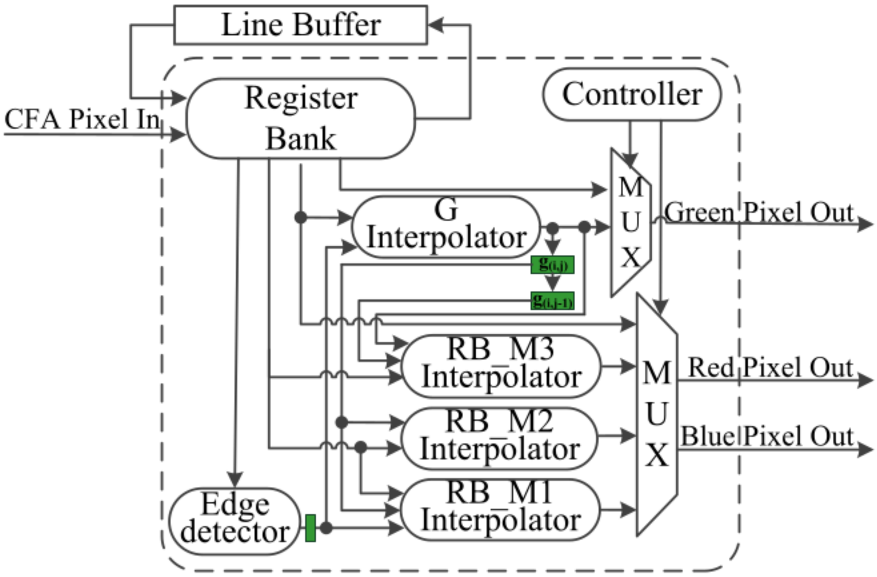
Figure 2: Original Image

Figure 3: Bayer Image with just one channel

Figure 4: Bayer Image with a 3 channel (zeros at other positions)

4 Implementations

Used this paper[1] for Implementations. Two Implementation were done: Vivado-Project/project_1 – without padding of zeros - so we get reduced resolution. Vivado-Project/ReconstructionWithPadding – with Padding of zeros - to get same resolution. The block diagram can be seen below:



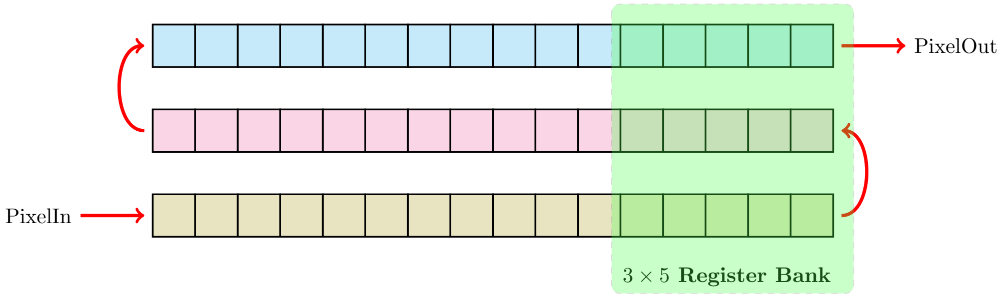
- THe Register Bank (uses line buffers) producess a 3x5 pixel grid.
- This value is gien to Edge Detector to get DH and DV.
- The these values are given to Ginterpolator to get G values.
- The these values are given to RBM1 to get RB when the current pixel is Blue or Red.
- The these values are given to RBM2 to get RB when the current pixel is Green (BGBG.. pattern).
- The these values are given to RBM3 to get RB when the current pixel is Green (GRGR.. pattern).

4.1 Line Buffers

Line Buffers holds the values of a complete Row of an Image. The length of line buffer is equal to the width of the Image. These Line buffers are made from shift registers.

4.2 Register Banks

They are obtained from Line Buffers. Here 3 Line Buffers are used connected serially as shown below:



4.3 EdgeDetector

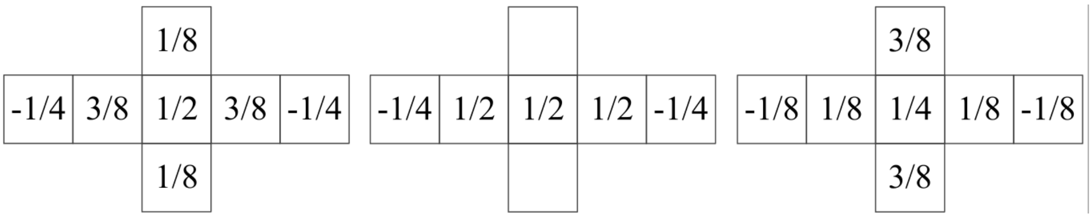
$$DV_{i,j} = \left|P_{i-1,j-1} - P_{i+1,j-1}\right| + \left|P_{i-1,j} - P_{i+1,j}\right| + \left|P_{i-1,j+1} - P_{i+1,j+1}\right| \tag{1}$$
$$DH_{i,j} = \left|P_{i+1,j+1} - P_{i+1,j-1}\right| + \left|P_{i,j+1} - P_{i,j-1}\right| + \left|P_{i-1,j+1} - P_{i-1,j-1}\right| \tag{2}$$

For absoluted differnce - found which is greater and subtracted the lesser value from the greater value.

4.4 G Interpolator

Finding Green when the current pixel is Blue or Red.
The equations can be seen below and their representations can be seen in the figure.

$$G_{i,j} = \begin{cases} \frac{1}{8}(P_{i-1,j} + P_{i+1,j}) + \frac{3}{8}(P_{i,j-1} + P_{i,j+1}) - \frac{1}{4}(P_{i,j-2} + P_{i,j+2}) + \frac{1}{2}P_{i,j}, & \text{DH} = DV \\ \frac{1}{2}(P_{i,j-1} + P_{i,j+1}) - \frac{1}{4}(P_{i,j-2} + P_{i,j+2}) + \frac{1}{2}P_{i,j}, & \text{DH} < DV \\ \frac{3}{8}(P_{i-1,j} + P_{i+1,j}) + \frac{1}{8}(P_{i,j-1} + P_{i,j+1}) - \frac{1}{8}(P_{i,j-2} + P_{i,j+2}) + \frac{1}{4}P_{i,j}, & \text{DH} > DV \end{cases} \tag{3}$$



The pixel values are added an shifted and added.

4.5 RB Interpolator - M1

Finding Red or Blue when the current pixel is Blue or Red.
The equations can be seen below and their representations can be seen in the figure.

$$RBM1_{i,j} = \begin{cases} \frac{1}{4}(P_{i-1,j-1} + P_{i+1,j-1} + P_{i+1,j-1} + P_{i+1,j+1}) - \frac{1}{4}(P_{i-1,j} + P_{i,j-1} + P_{i,j+1} + P_{i+1,j}) + g_{i,j}, & \text{DH} = DV \\ \frac{1}{4}(P_{i-1,j-1} + P_{i+1,j-1} + P_{i+1,j-1} + P_{i+1,j+1}) - \frac{3}{8}(P_{i-1,j} + P_{i+1,j}) - \frac{1}{8}(P_{i,j-1} + P_{i,j+1}) + g_{i,j}, & \text{DH} > DV \\ \frac{1}{4}(P_{i-1,j-1} + P_{i+1,j-1} + P_{i+1,j-1} + P_{i+1,j+1}) - \frac{1}{8}(P_{i-1,j} + P_{i+1,j}) - \frac{3}{8}(P_{i,j-1} + P_{i,j+1}) + g_{i,j}, & \text{DH} < DV \end{cases} \tag{4}$$

$g_{i,j}$ is the updated G interpolated value.

1/4	-1/4	1/4
-1/4	1	-1/4
1/4	-1/4	1/4

1/4	-3/8	1/4
-1/8	1	-1/8
1/4	-3/8	1/4

1/4	-1/8	1/4
-3/8	1	-3/8
1/4	-1/8	1/4

4.6 RB Interpolator - M2

Finding Red or Blue when the current pixel is Green (either BGBG... or GRGR...). The equations can be seen below and their representations can be seen in the figure.

$$RBM2_{i,j} = \frac{1}{2}(P_{i-1,j} + P_{i+1,j}) - \frac{1}{8}(P_{i-1,j-1} + P_{i+1,j-1} + P_{i+1,j-1} + P_{i+1,j+1}) + \frac{1}{2}P_{i,j} \quad (5)$$

-1/8	1/2	-1/8
0	1/2	0
-1/8	1/2	-1/8

4.7 RB Interpolator - M3

Finding Red or Blue when the current pixel is Green (either BGBG... or GRGR...). The equations can be seen below and their representations can be seen in the figure.

$$RBM3_{i,j} = \frac{1}{2}(P_{i,j-1} + P_{i,j+1}) \quad (6)$$

References

[1] S. Chen and E. Ma, "VLSI Implementation of an Adaptive Edge-Enhanced Color Interpolation Processor for Real-Time Video Applications," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 11, pp. 1982-1991, Nov. 2014, doi: 10.1109/TCSVT.2014.2317890.