Bayer Decode

Submit Solution

Write a C++ command line program that accepts the following options:

```
1. bayer_decode <input file .PGM> <output prefix>
```

The program must manage the command line and load a file in PGM format (according to the specifications attached in the `PGM Format Specification.html` file) in which a 16 bit per pixel RAW file is stored, acquired with a Bayer pattern of type:

R	G
G	B

For the input, you have to manage only the case of binary data at 16 bits per pixel with Maxval equal to 65535 and without comments.

1. The program must load the file (**pay attention to the endianness**, read the specifications carefully), convert it to 8 bits per pixel (dividing by 256), save it as `<output prefix>_gray.pgm`, again in binary, at 8 bits per pixel with Maxval equal to 255. Check that the file opens correctly with a viewer, such as XnView.

2. The second step is to assign each pixel value to the correct color channel, so you must create a color image with 3 channels at 8 bits each. If the color corresponds to an R pixel, make the color pixel `(R,0,0)`, if it is a G pixel make it `(0,G,0)`, if it is a B pixel make it `(0,0,B)`. Save the resulting image as `<output prefix>_bayer.ppm`, again in binary, at 8 bits per channel with Maxval equal to 255 (according to the specifications attached in the `PPM Format Specification.html` file). Check that the file opens correctly with a viewer, such as XnView.

Once this is done, you need to interpolate the missing colors using the two-pass algorithm seen in class.

In the first pass the green channel is reconstructed for pixels R and B. We call the pixels in the following way:

		X1		
		G2		
X3	G4	X5	G6	X7
		G8		
		X9		

where X is an R or B pixel, while G is a green pixel. The algorithm calculates the following two values:

$$\Delta H = |G4 - G6| + |X5 - X3 + X5 - X7|$$
$$\Delta V = |G2 - G8| + |X5 - X1 + X5 - X9|$$

and interpolates the green in position G5 as follows:

$$G5 = \begin{cases} \frac{G4 + G6}{2} + \frac{X5 - X3 + X5 - X7}{4} & \Delta H < \Delta V \\ \frac{G2 + G8}{2} + \frac{X5 - X1 + X5 - X9}{4} & \Delta H > \Delta V \\ \frac{G2 + G4 + G6 + G8}{4} + \frac{X5 - X1 + X5 - X3 + X5 - X7 + X5 - X9}{8} & \Delta H = \Delta V \end{cases}$$

Save the resulting image as `<output prefix>_green.ppm`, again in binary, at 8 bits per channel with Maxval equal to 255. Check that the file opens correctly with a viewer, such as XnView.

Finally, in the second pass, the remaining values are interpolated as follows:

1. for the green pixels, R and B are interpolated as the average of the two neighboring R and B. There are two cases: i) red is left and right and blue is top and bottom, or ii) red is top and bottom and blue is left and right.
2. for the red and blue pixels a grid is defined:

1	2	3

4	5	6
7	8	9

if pixel 5 is R, B must be interpolated and vice versa. The algorithm calculates the following two values:

$$\Delta N = |X1 - X9| + |G5 - G1 + G5 - G9|$$
$$\Delta P = |X3 - X7| + |G5 - G3 + G5 - G7|$$

and interpolates the value in position 5 as follows:

$$X5 = \begin{cases} \frac{X1 + X9}{2} + \frac{G5 - G1 + G5 - G9}{4} & \Delta N < \Delta P \\ \frac{X3 + X7}{2} + \frac{G5 - G3 + G5 - G7}{4} & \Delta N > \Delta P \\ \frac{X1 + X3 + X7 + X9}{4} + \frac{G5 - G1 + G5 - G3 + G5 - G7 + G5 - G9}{8} & \Delta N = \Delta P \end{cases}$$

Be careful to **saturate** the values produced with this algorithm between 0 and 255 **before** putting them in a byte.

Save the resulting image as `<output prefix>_interp.ppm`, again in binary, at 8 bits per channel with Maxval equal to 255. Check that the file opens correctly with a viewer, such as XnView.

Suggestion for solving the exam: initially perform the interpolation described by ignoring a 2-pixel border around the image and setting it to all zeros. In this way you will not have to worry about accessing outside the source image limits. Then manage the border, assuming that all the pixels outside the image are black (0).