CIP Assignment-2

1. Search the web for *Colour Halftoning algorithms*. Select one of them and write a detailed report on it **OR** implement the selected algorithm and show results on the test images.

Ans. Find the code in ***colorHalfTone.py*** inside *codes* directory.

Some sample outputs are given below,

halftone_fall-color_cmy.png

halftone_orange-flower_cmy.ppm

The rosette pattern is quite visible inspite of keeping the channels at (15^o, 75^o, 0^0, 45^o) angle respectively.

halftone_orange-flower_cmyk.png

The first output of orange-flower is done through CMY channel only. But the weired diagonal(45^0) checkerbox pattern in appeared when I added Black color i.e the K component which is also alligned 45^O by convension.

halftone_wplane_cmyk_15_45_0_75.png

The diagonal pattern disappeared when i used the following angles (15^o,45^o,0^o,75^o).

halftone_lena_cmy.png ***halftone_lena_cmyk.png***

The left sample is created with CMY dots only whereas the right one has all CMYK dots. It's clear that we get better output if we don't include the black component which is included only to make the process cheaper. But upon zooming in you won't see much diffrence in naked eyes.

2. Try coming up with your own *error diffusion coefficients* and implement the standard error-diffusion algorithm. Compare the performance of your coefficients against Floyd-Steinberg's on this image. Discuss the patterns visible in yours and in Floyd-Steinberg's at the various gray levels.

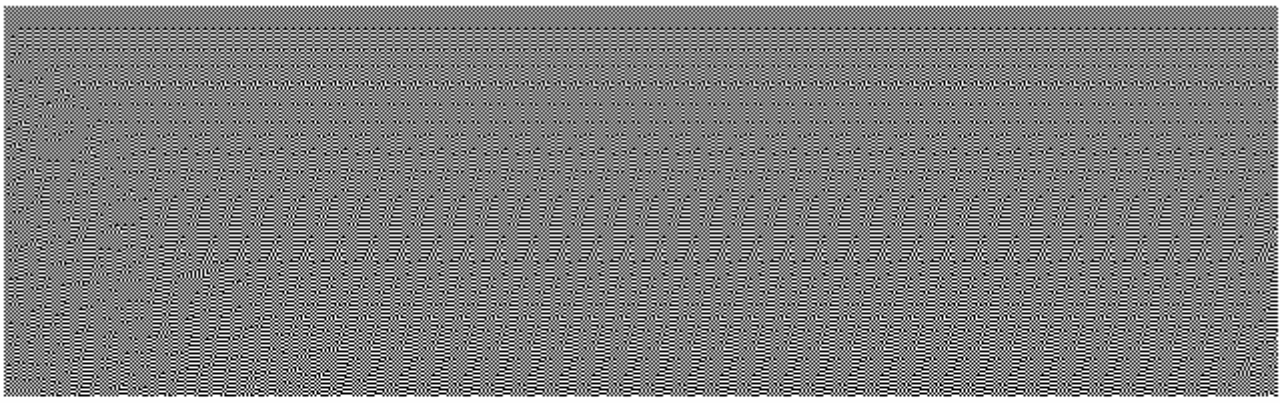
Ans. The coefficients I applied are,

	(x, y)	1/4 (x, y+1)
1/6 (x+1,y-1)	5/12 (x+1, y)	1/6 (x+1,y+1)

I have pushed maximum part of the error downwards. Whereas in Floyd-Steinberg's the bigger part is been pushed to the right pixel. Also I have kept the diagonal error propagation same for both left and right diagonal pixels.

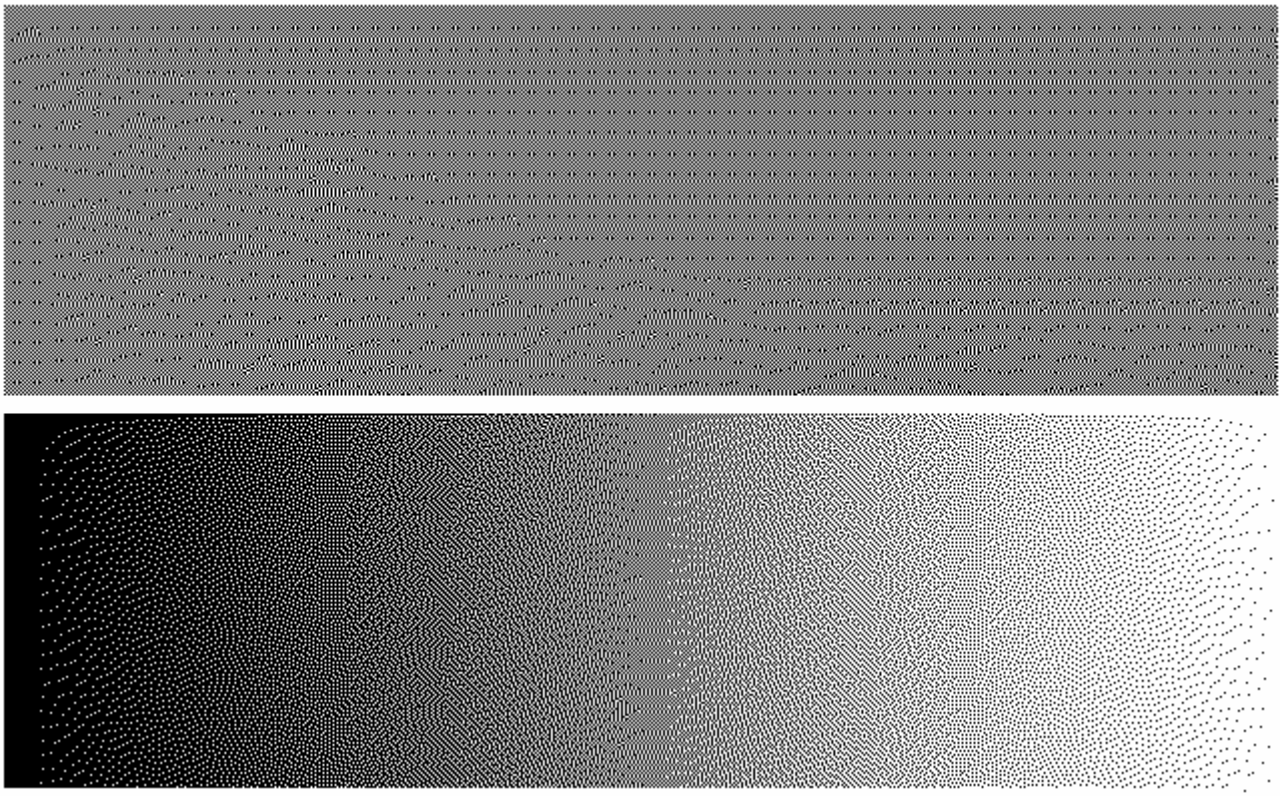
Output for my coeffcients:

half-tone_6_4_10_4.pgm



Output for Floyd-Steinberg :

half-tone_floydSteinberg.pgm



The patterns visible in horizontal gradient for my version of algorithm is more or less same as that of Floyd-Steinberg's. But if we look carefully at the middle part, it seems that worms are more prominent in my version of error diffusion.

In case of vertical gradient, at the bottom left corner, patterns are more symmetric in my version than Floyd-Steinberg's. So we can say that Floyd-Steinberg's algorithm is doing a better job.

3a. Implement an algorithm to simulate the grayscale output from a colour filter array. The function prototype is

```
image colour_filter (image, filter)
```

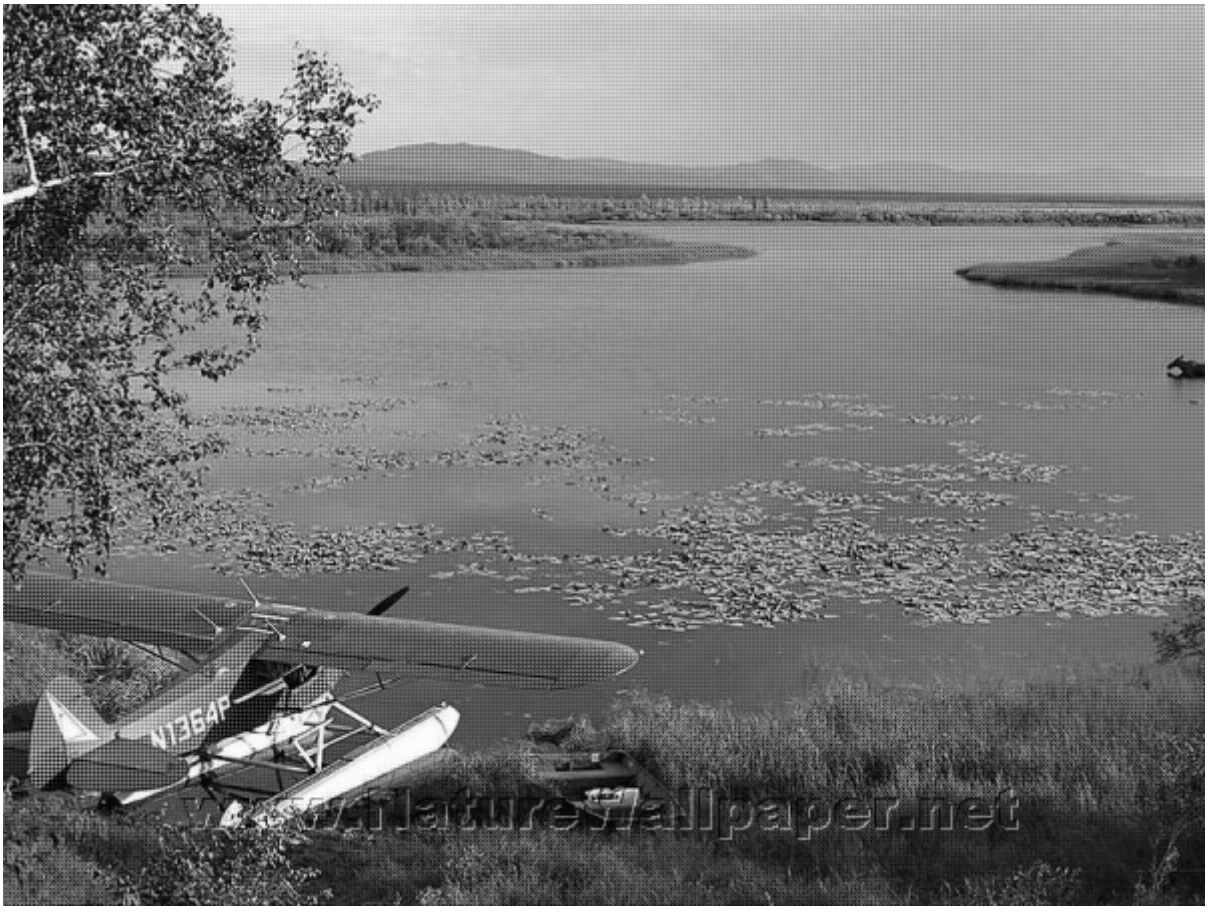
That is, it takes an input colour image and a colour filter as parameters and returns a grayscale image.

Ans. Please find the implementation in '***colorFilter.py'*** inside 'codes' directory.

Here is some sample outputs of the program,

cfa_fall-colours.pgm

cfa_wplane.pgm



cfa_orange-flower.pgm

3b. Implement a demosaicking algorithm with the prototype

```
image demosaic (image, filter)
```

The input image is a grayscale image output by the `colour_filter` algorithm and the corresponding filter array; the output is a colour image.

Ans. Please check ***demosaic.py*** for the implementation in *codes* directory.

Thsese are some results I got by demosaicing the output images from previous question.

demosaiced_fall-colors.ppm

The bleeding effect is quite visible in the leaves and on the ground. The fine details of the edges for smaller section is lost.

demosaiced_wplane.ppm

The gradient of the sky color and and water has been produced quite well but a lot of bleeding effect is visible on the tree leaves and also on the floating herbs on the water.

demosaiced_orange-flower.ppm

Our algorithm has produced best result for this sample image as very little bleeding effect is visible. Though if you notice closely, the smoothness is lost at the edges of the flower.