1. Implementing the dither algorithm.

   The dispersed dot dither is used. The filter used is $\begin{pmatrix} G2 & R1 & B5 & G4 \\ R4 & B3 & G1 & B6 \\ B4 & R2 & B2 & G3 \\ R5 & G5 & B1 & R3 \end{pmatrix}$ for RGB space.

   Each pixel of image is expanded to 4x4 and the dots are put using above filter. The blue dots are 6, green are 5, red are 5. Blue dots are 6 so we divide 0 to 255 in chunks of nearly 6 parts. 0-42, 43-85, 86-128, 129-171, 172-214, 215-255. Red and green dots are 5 in number so we divide 0 to 255 in chunks of nearly 5 parts. 0-50, 51-101, 102-152, 153-203, 204-255. Each pixel is checked for its corresponding color range. If it is in first slot, put one dot of blue for blue pixel following the order in mask. If it is in second slot put two dot of blue for blue pixel following the order in mask, so on.

   Test images

The background is kept black as RGB cannot produce black. So the image seems little dark.

Dithering using CMY dots. The each pixel is subtracted by 255 to get corresponding CMY pixel.

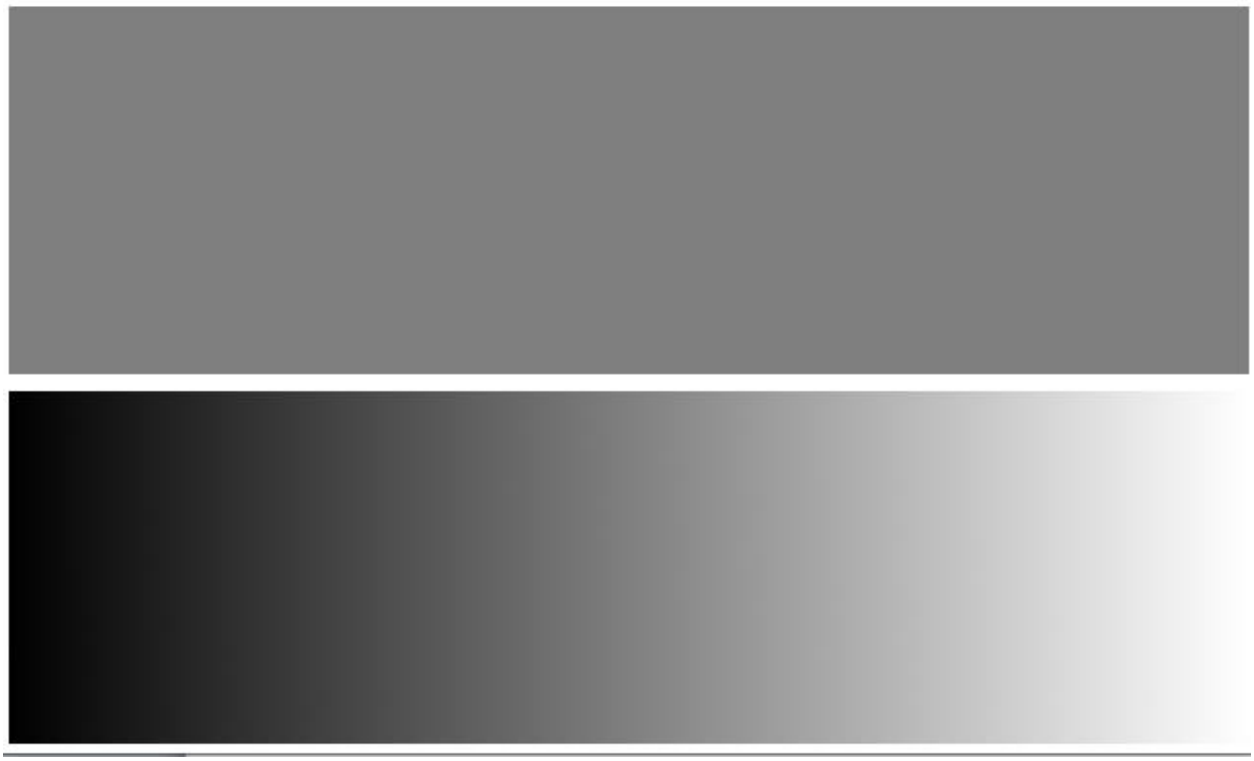
The filter used is $\begin{pmatrix} M2 & C1 & Y5 & M4 \\ C4 & Y3 & M1 & Y6 \\ Y4 & C2 & Y2 & M3 \\ C5 & G5 & Y1 & C3 \end{pmatrix}$ for CMY space. The same procedure is followed to this also, only in place of RGB we use CMY dots.

The background is kept white as CMY cannot produce white. So the image seems little bright.

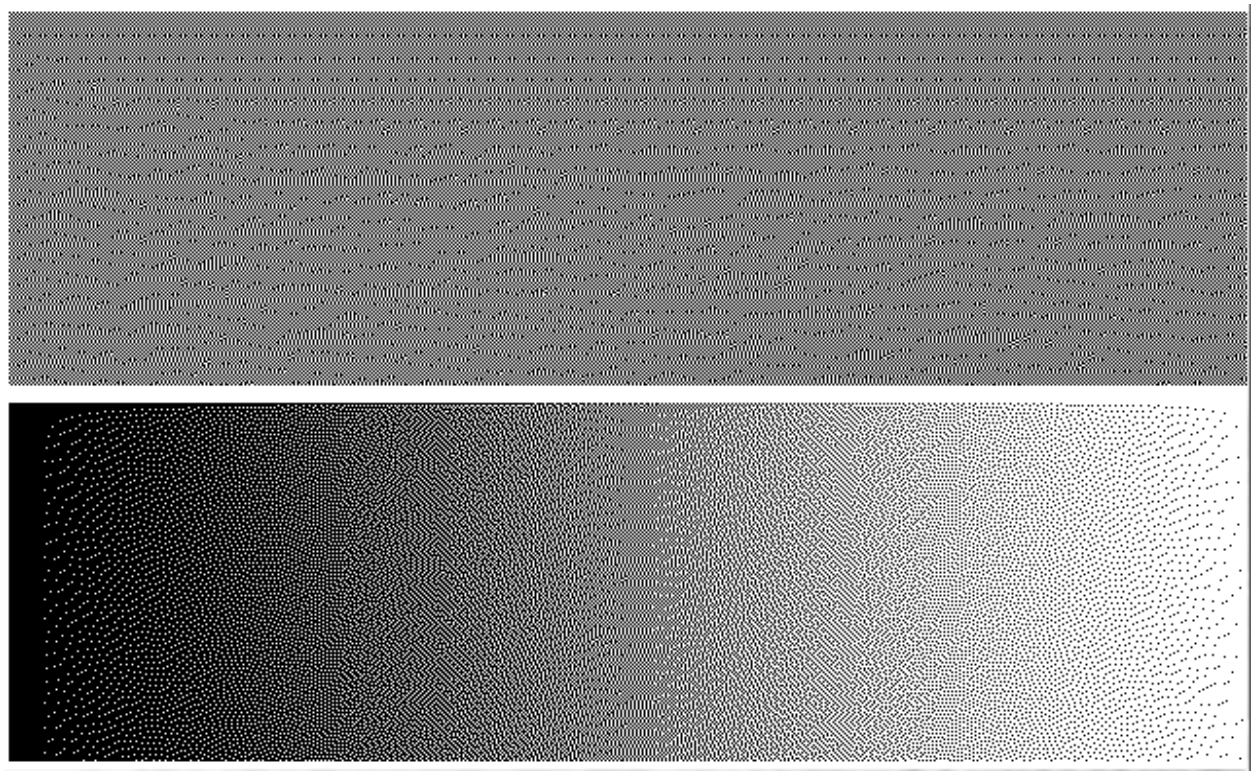
2. Error diffusion: Standard Floyd-Steinberg's error-diffusion algorithm. Floyd-Steinberg's coefficients are diffused like this $\begin{pmatrix} & * & 7/16 \\ 3/16 & 5/16 & 1/16 \end{pmatrix}$.

   Different coefficients that was chosen $\begin{pmatrix} & * & 7/29 \\ 9/29 & 9/29 & 4/29 \end{pmatrix}$.

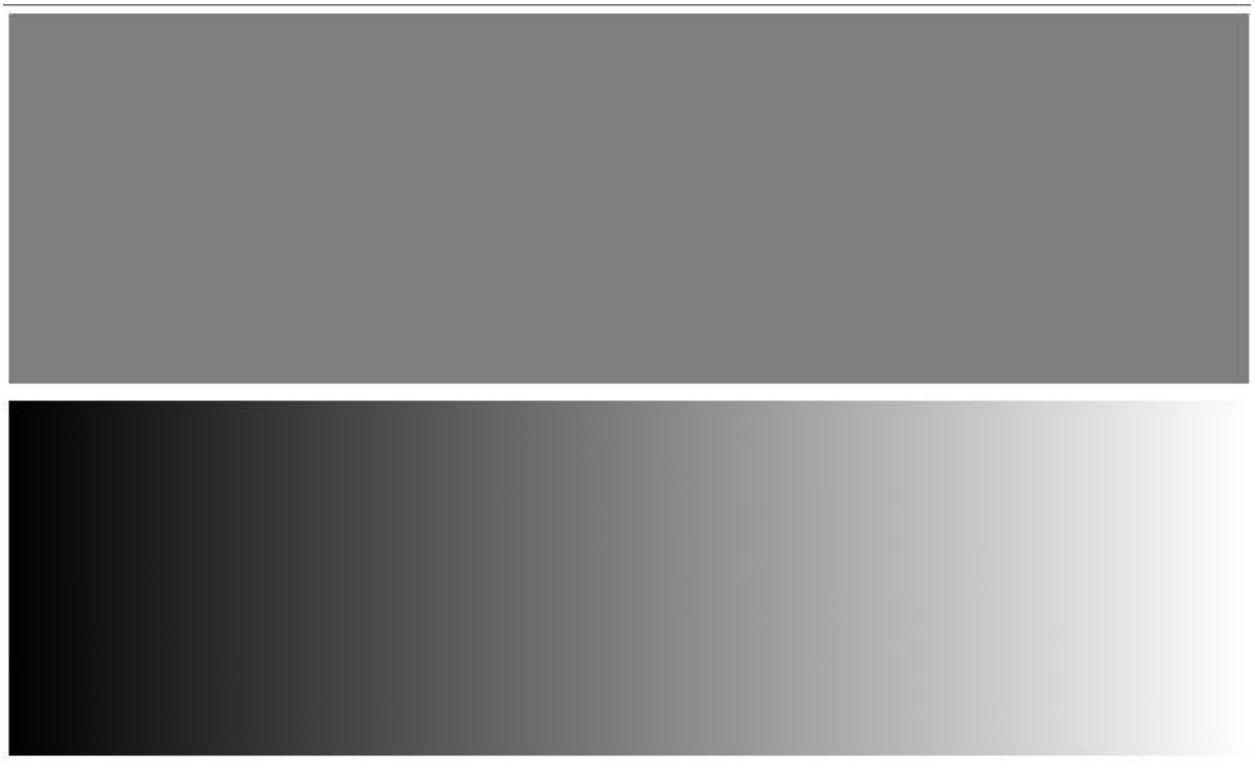
Original image



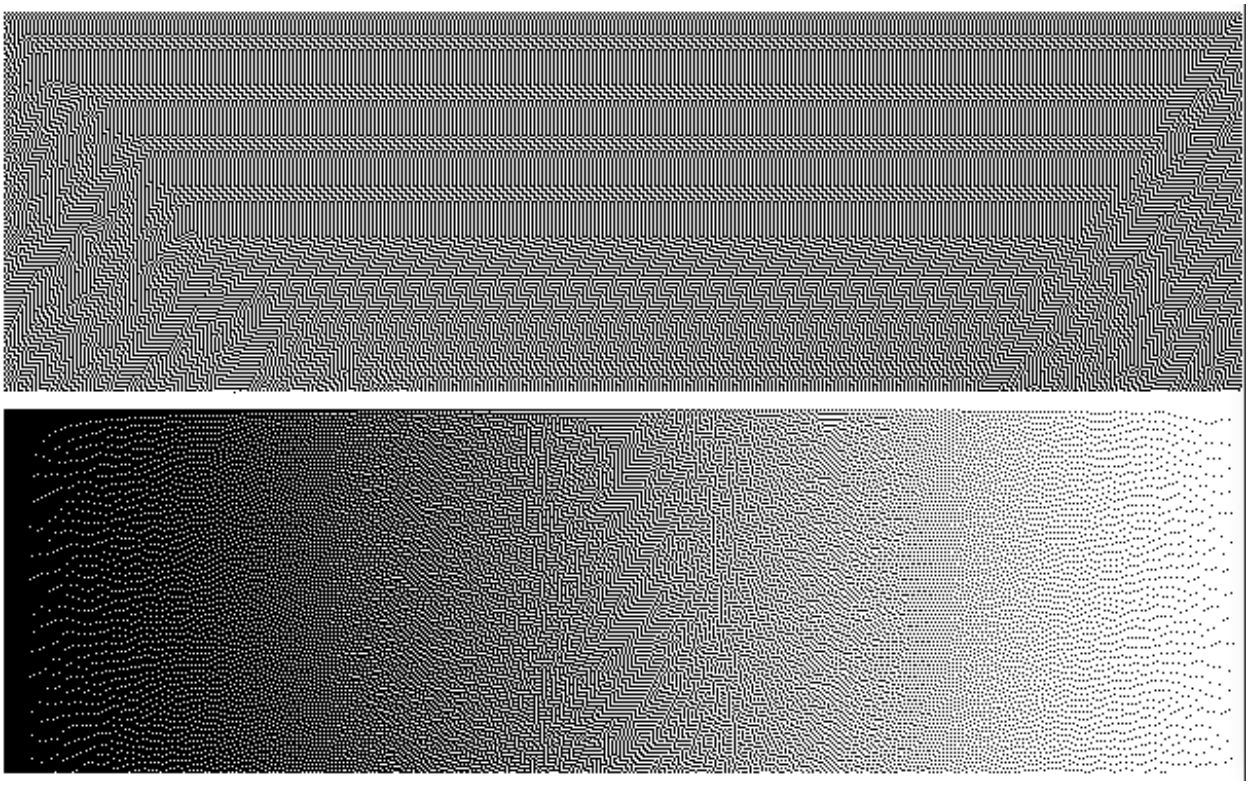
Floyd-Steinberg's error-diffusion



Original image



Different coefficients error-diffusion



In Standard error diffusion algorithm, there is not much of patterns, its just wavy for first pic, then for second pic, there was many patterns. After changing the coefficients, more patterns were visible in both pictures.

3. Color filter array: The mask considered is $\begin{pmatrix} B & G \\ G & R \end{pmatrix}$. The image is considered. According to the mask we take out each pixel value. Suppose the first pixel (R,G,B) we take the Blue value out of that and store it. Second pixel (R,G,B) we take green value out of that and store it. Like this we continue to store one value out of each pixel. This is called color filtering. Original image will be of three dimensions. This obtained color filter from that image will be only two dimensions. Here we go from color image to gray scale image.

   Demosaic: The obtained color filter we take it and we construct the color image. The gray scale image to color image. The filter is shown above. We take 2x2 of the color filter image values. All individual values are taken, if there is more than one individual color we take average of those respective colors. That is, we take all blue colors find average of it, place in blue position (R,G,B). Similarly we consider other colors and average it and place it.

   Then we have to slide our 2x2 window, but now the stored color will be $\begin{pmatrix} G & B \\ R & G \end{pmatrix}$.

   So we have to make it $\begin{pmatrix} B & G \\ G & R \end{pmatrix}$ again. We can use np.roll() function, we can roll the matrix along axis 1 to get it. Horizontally we roll along axis 1. Vertically we roll along axis 0.

   In first image shown below, the orange flower doesn't seem much different from the original one.

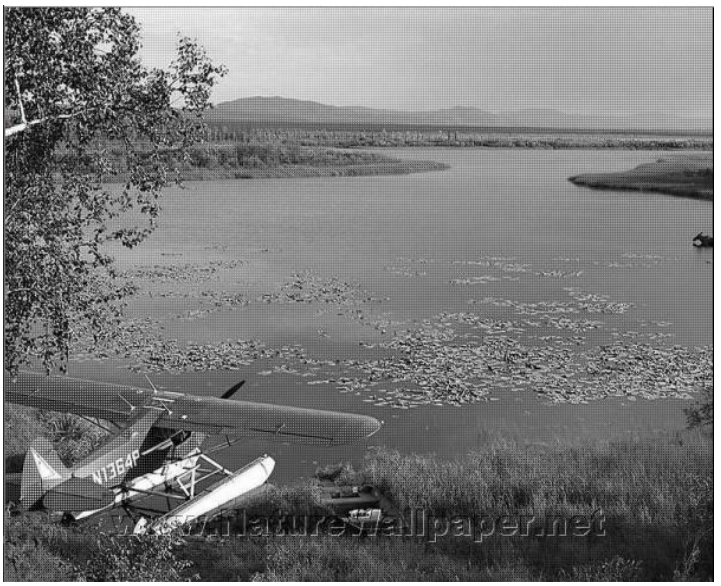
   In Second image shown below, the water plane there is difference from the original one. As there is color bleed.

   In the third image shown below, the chameleon there is little difference from the original one at the eyes, because of color bleed.

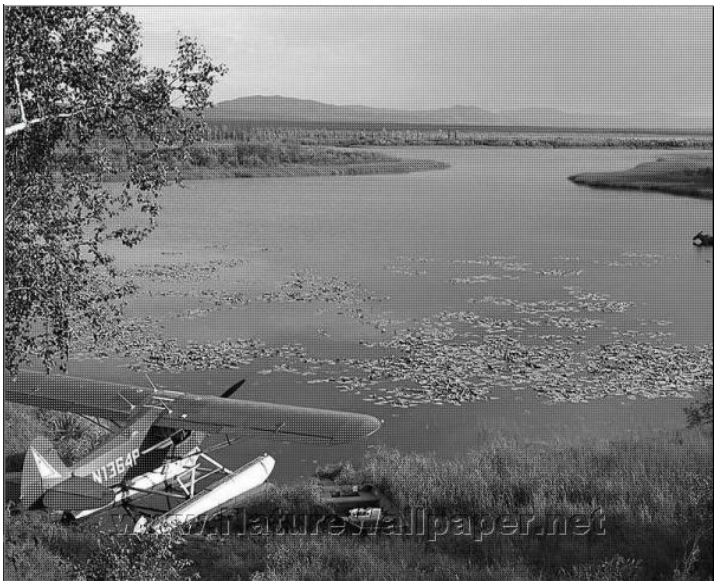
Color filter

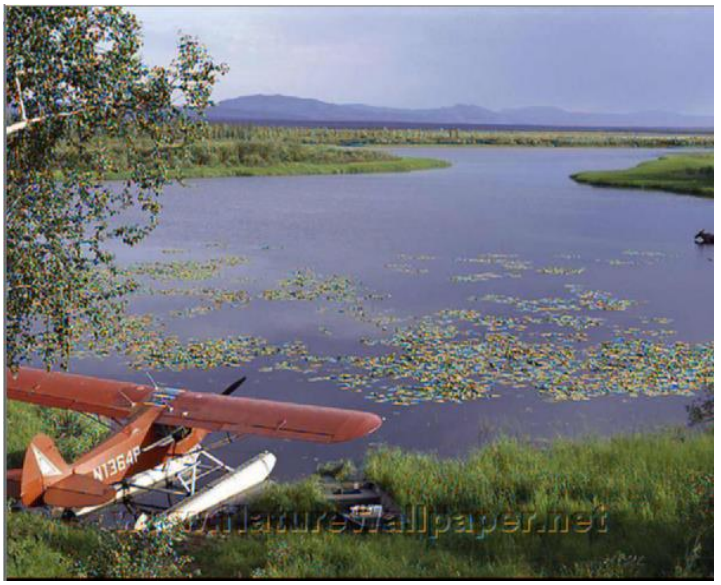
Demosaic

Color filter



Demosaic





Color filter

Demosaic