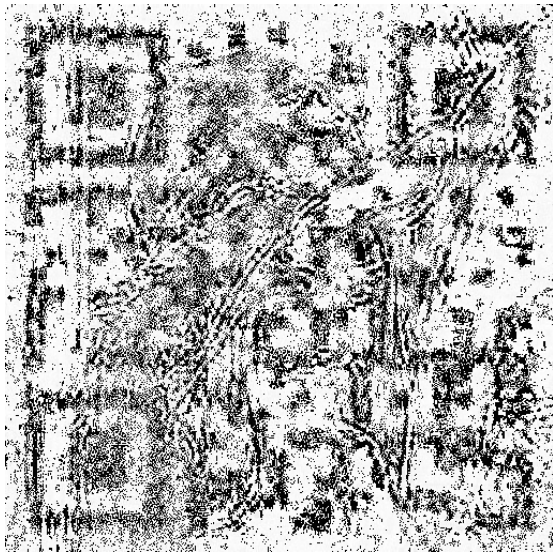
Digital Image Processing: Assignment 1

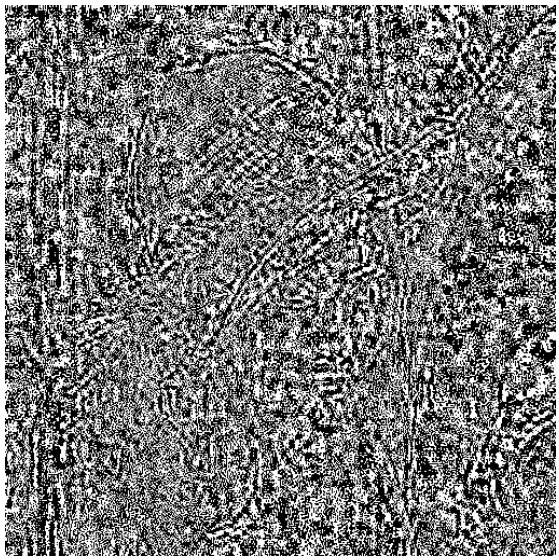
(1) You are given two images (lena1.jpg and lena2.jpg). Find a clue in these images to answer this question??

We split the images into their R,G,B components. Then to find the difference, we subtract the corrosponding channels from lena1 and lena2 to obtain a difference mask. By doing a simple imshow on the difference map, we obtain the following images for each of the channel differences.

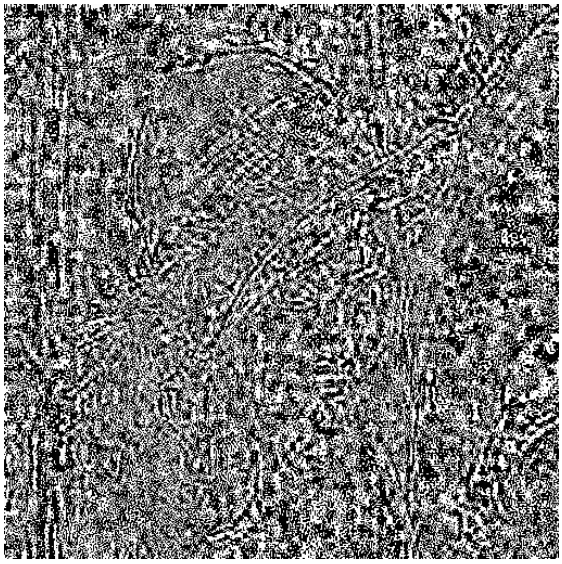
a. Red



b. Green



c. Blue



As you can plainly see, there has been a QR code of sorts encoded on the image, which translates to the Number “09237353”.

(2) Extract the person from the image ‘greens screen.jpg’ and paste him on a realistic background.

In this question, we find try to find the colour green in the image. Now, what makes a part of the picture, green? It’s because the value of the Green intensity, is much higher than that of the Red and Blue.

I claim that if R,G,B denotes the pixel intensity of red, green and blue, then the following will be very high for green pixels.

I = G*(G-R)*(G-B)

If that value is greater than some threshold, we can claim that the color is green. But since we want to extract the man from the green screen, we invert the mask using

Mask = (1-I)

We can then easily just replace all the pixels in the new background image corrosponding to the 1s in the mask, with pixels from the green_screen.jpg

The following is the image obtained from this process.

(3) Write a function to imitate blur effect of a narrow focal length camera capture. Your function can take some appropriate inputs from the user. For example, given a line annotated by the user, can you generate 'car blur1.jpg' from the image 'car.jpg'. Similarly, given a circle and a line from the user, can you generate 'car blur2.jpg' from the image 'car.jpg'.

We make a copy of the given image, let's call this image I2, the original is given by I1. Given a line as an input from the user, we simply construct a rectangular area around the line. Now we blur the whole image. The area outside this rectangle is taken and superimposed over the area outside the rectangle in the original image. Similarly given a circle as input, we perform a similar procedure, along with which, we reduce the pixel intensities with the distance from the center of the circle.

The output of the two processes is given below.

(4) Write a function to convert the image ‘lotus’ to ‘lotus edited’. (Hint: Median or a Mode)

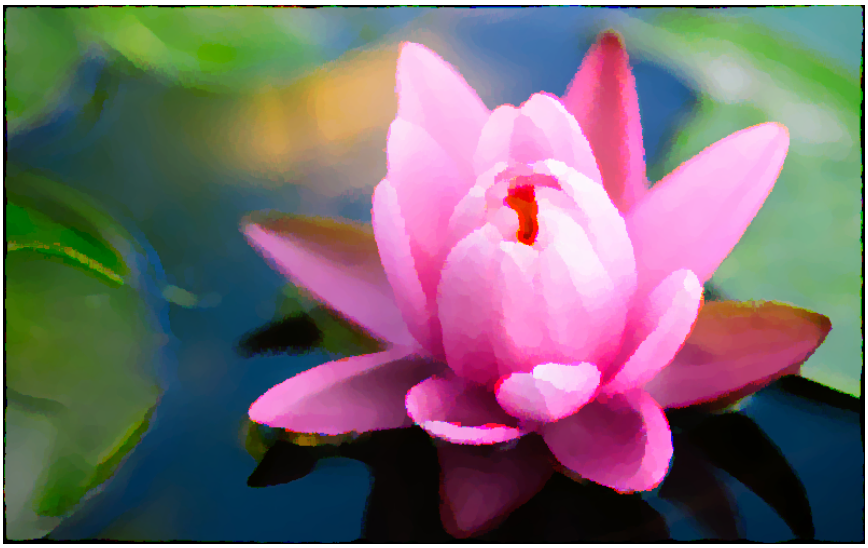
To perform an operation where a function is applied on a window, Matlab/Octave provide a function “colfilt” which given an image performs a transformation on each MxN block of the image by either sliding a window, or considering each MxN block as distinct.

By trying various different functions, we obtain the following outputs

a. Mean

b. Median

c. Mode

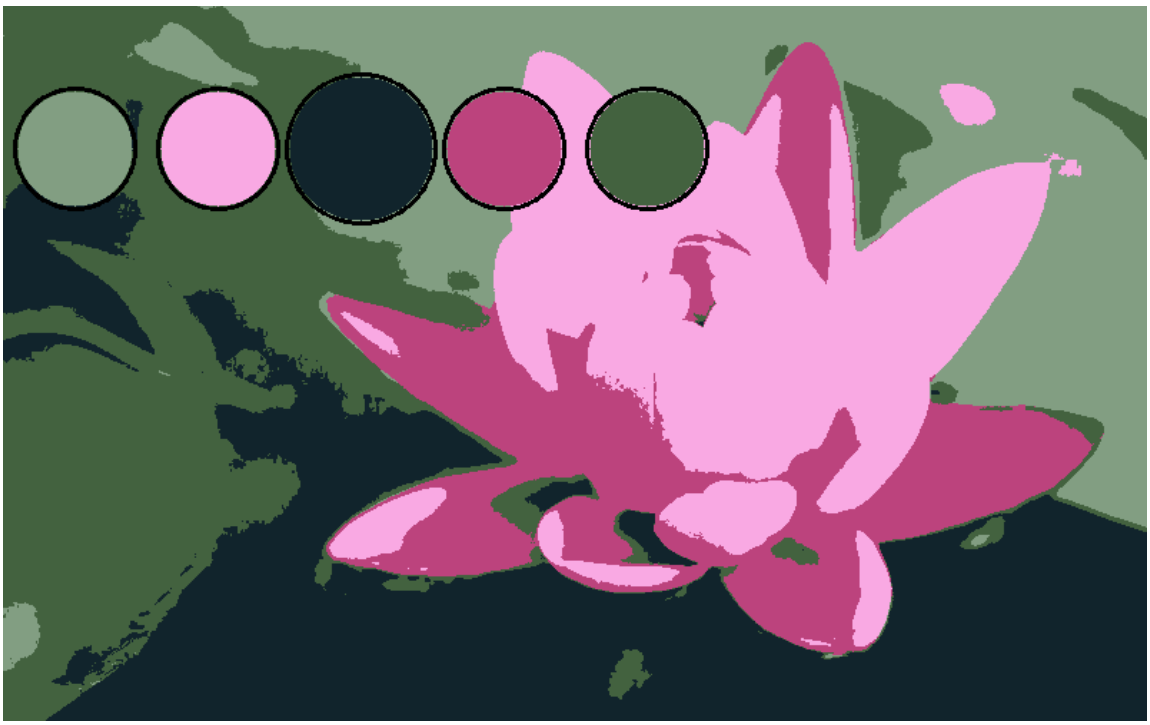


As it can be seen, we are getting a similar effect by using the mode. Now, we can try changing the size of window function. When using a window of 10x10, we obtain an image similar to the one given, as shown above.

(5) An interesting web tool for extracting relevant colors from an image is shown on the following page: 'http://lokeshdhakar.com/projects/color-thief/'. Your task is to do something similar in Matlab or Python. Now use this information for an application (build any application of your choice)?

One of the first algorithms that comes to mind to do this, is K-means. We flatten the image given into a 1x3 matrix, with the RGB values preserved. Then we run the K-means algorithm to cluster these colors based on how close the colors are in the colorspace. Depending on our value of K, the algorithm returns a set of centers for each of the K-clusters, and the classification of which cluster each point belongs to. We can simply count the number of points in each cluster to find the most dominating color.

Now as an application, I have simply replaces the pixels in original image, with the representative cluster color to make a watercolor filter for styling an image. The output of the code is shown below given the same input as the lotus.jpg



Also, the dominant color is shown in a slightly larger circle than the other colors.

(6) Use a variation of the flood fill algorithm to solve the maze in image ‘maze.png’. Your algorithm can take as input the image (with information of grid structure and the size of each block); the starting block coordinates (yellow) and the destination area coordinates (green).

This problem presented a set of very unique challenges. To Tackle this problem, I follow these steps:-

a. Preprocess the image of the maze to get rid of the dotted lines. - I do this by inverting each color channel and then taking the difference between the Red and Blue channels. The intuition behind doing this is that the maze boundaries are red, the dotted dashes are made of equal parts of red and blue, so by subtracting the red channel and blue channel, I am only left with the maze. I work on this newly obtained mask to get my solution to the maze.

b. Define my start and end points :- Theoretically, these could just be autodetected, but I have choosen to hardcode the start and end points.

c. I maintain a Queue to store the entire path that I’ve seen. Starting with the start point, I first check if this is the end point, else I simply build a new path which includes all the points adjecent to the given point, and enqueue these new paths to be processed later. I continue this process until I have successfully reached my end point or the queue becomes empty in which case,there are no other paths to be found. -- Since this was being done pixelwise, the path tended to gravitate toward the first found path, so instead I placed a restriction that the point should be equidistant from any red-boundary.

d. If the end point has been found, then for all the points that I have stored in my queue, I draw a rectangle to denote the path traversed.

The following shows the image of the solved maze.