Assignment 1

Part 1:

Popularity

In this algorithm, we have first created an array of tuples in which a tuple contains a color in the image and the frequency of the said color. Then we sorted the array by the second term of each tuple i.e., the array was sorted on the basis of frequency of colors. The top k colors with maximum frequencies were chosen and stored in a selectedColors array. A function has been created which calculates the euclidean distance of color with all the colors in the selectedColors array and returns the color to which the input color is closest to. Using this distance function, at each pixel the closest color is found and a new image is created with the new colors belonging to the palette.

Median Cut

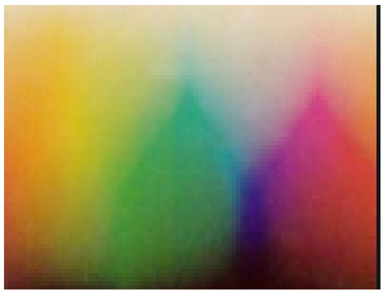
In this algorithm, all the colors found in the image are first noted and the range of each channel is calculated. Now, the colors are sorted on the basis of color having the greatest range. Now this sorted array is divided into k buckets by diving the array in halves in each iteration. From every bucket, the average color of the bucket is calculated and put in selectedColors array. The way of making the new image from this selectedColors array is same as in the popularity algorithm.

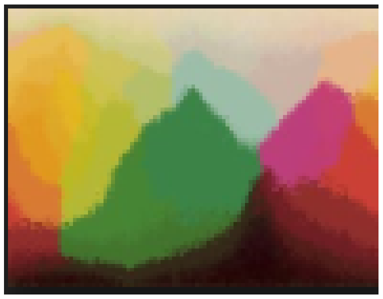
Floyd

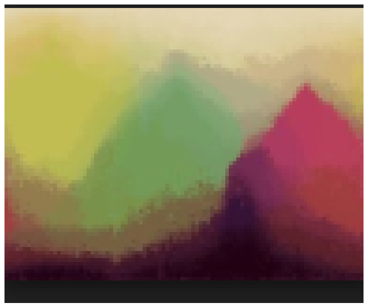
We have implemented floyd with the color palette obtained from median cut. While applying those colors in the new image, in each iteration the error is spread out in four neighbouring pixels with the values 5/16,7/16,3/16 and 1/16.

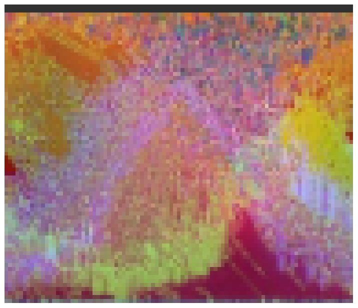
Results

The results of all three parts are shown below:


Original Image


Popularity


Median Cut


Floyd

TRANSFERRING COLOUR TO GREYSCALE IMAGES

1. Without using swatches :-

Step 1 : First the b-g-r colours of the coloured image are noted and stored.

- This is done using the funtion finding_colors(source)

Step 2: The stored b-g-r colours of the image is changed into l-α-β space

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} \mathrm{l} \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

We used these tranformations to change b-g-r into l-α-β space.
The functions made for this are :

- convert_to_labspace(bgr_colorspresent)

Step 3 : Then some random samples in a particular locality from the coloured image are choosen which is the jittered sampling. To do this we divide the images into 18x18 grids and then choose a random colour from each grid and then store the colour as well as its standard deviation from the 5x5 neighbours.
The functins used in this are :-

- jittered_sample(lab_space_colored)
- find_stddev(target,rowstart,rowend,colstart,colend,l)

Step 4 : Now taking the grey image into account, in the grey image the luminous is the l of l-α-β, hence first we find the standard deviation of every pixel's luminous from its 5x5 neighbours and then we move on to the jittered sample and look for the pixel for which |luminous of grey_pixel – luminous of coloured pixel | + | standard deviation of grey_pixel – standard deviation of coloured pixel| is minimum. And then lets say a is the colured pixel which gives the minimum value of the grey pixel b

, then the α-β of a is transferred to b.
The functins used in this are :-

- find_stddev(arr,rowstart,rowend,colstart,colend,num)
- lab_space_noncolor(target, lab_space_colored)

Step 5 : In the last step the modified l-α-β of the grey image are changed into b-g-r using the following transformations :

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}}{3} & 0 & 0 \\ 0 & \frac{\sqrt{6}}{6} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4.4679 & -3.5873 & 0.1193 \\ -1.2186 & 2.3809 & -0.1624 \\ 0.0497 & -0.2439 & 1.2045 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

The functions used in this are :-

- changing_to_bgr(target, final_lab_space_noncolor)

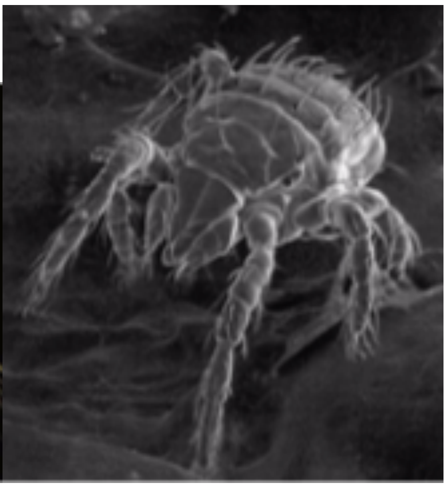
Step 6 : Finally this b-g-r image is stored as result.jpg

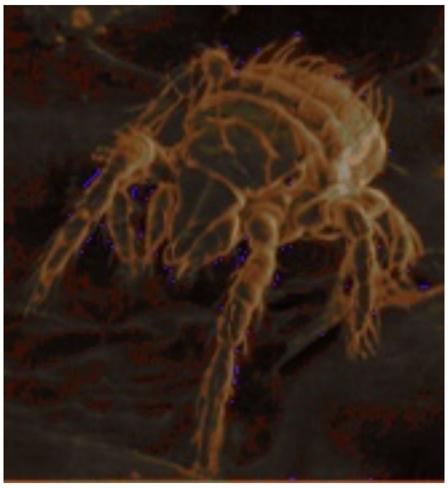
2. Using swatches :

In this we will give the program the location of the swatches in the grey image and the pixels in those swatches will be coloured using the refrence from the given coloured image in the above described way.

But for colouring the rest of the pixels in the grey image instead of taking refrerence from the colured image , we will take into account only the coloured swatches.
For selecting the perfect swatch to colour the pixel , we will find the minimum error swatch where error = sum((l(grey_scale)-l(each pixel in the swatch)^2).
Now to colour the pixel we will only take into account the colours of the perfect swatch and then colour it in the way described above.

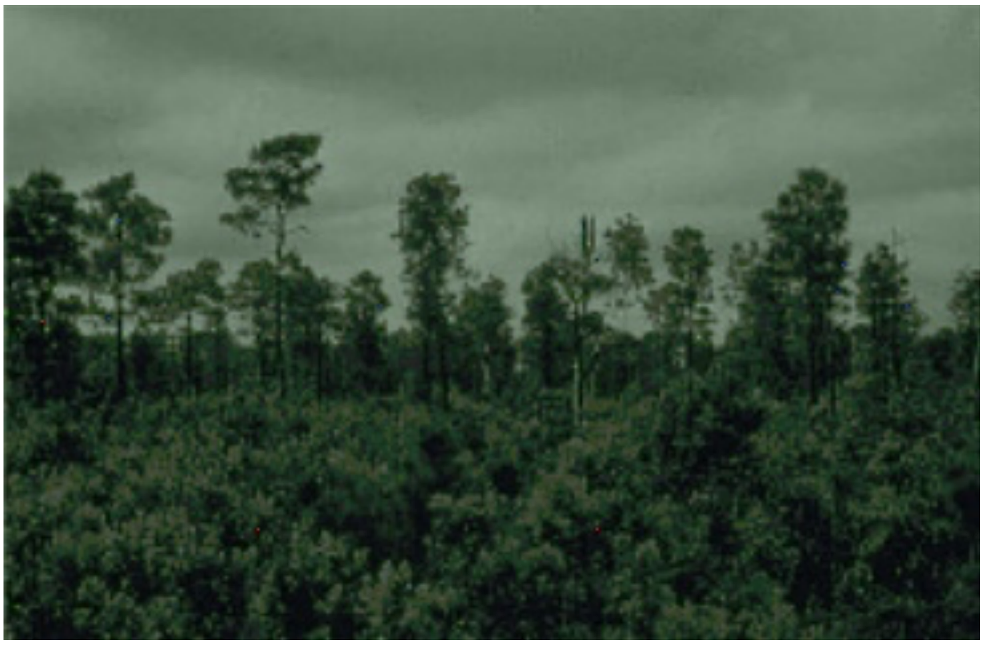
Example 1 :-





Example 2 :

a. Without swatch



b. With swatch :