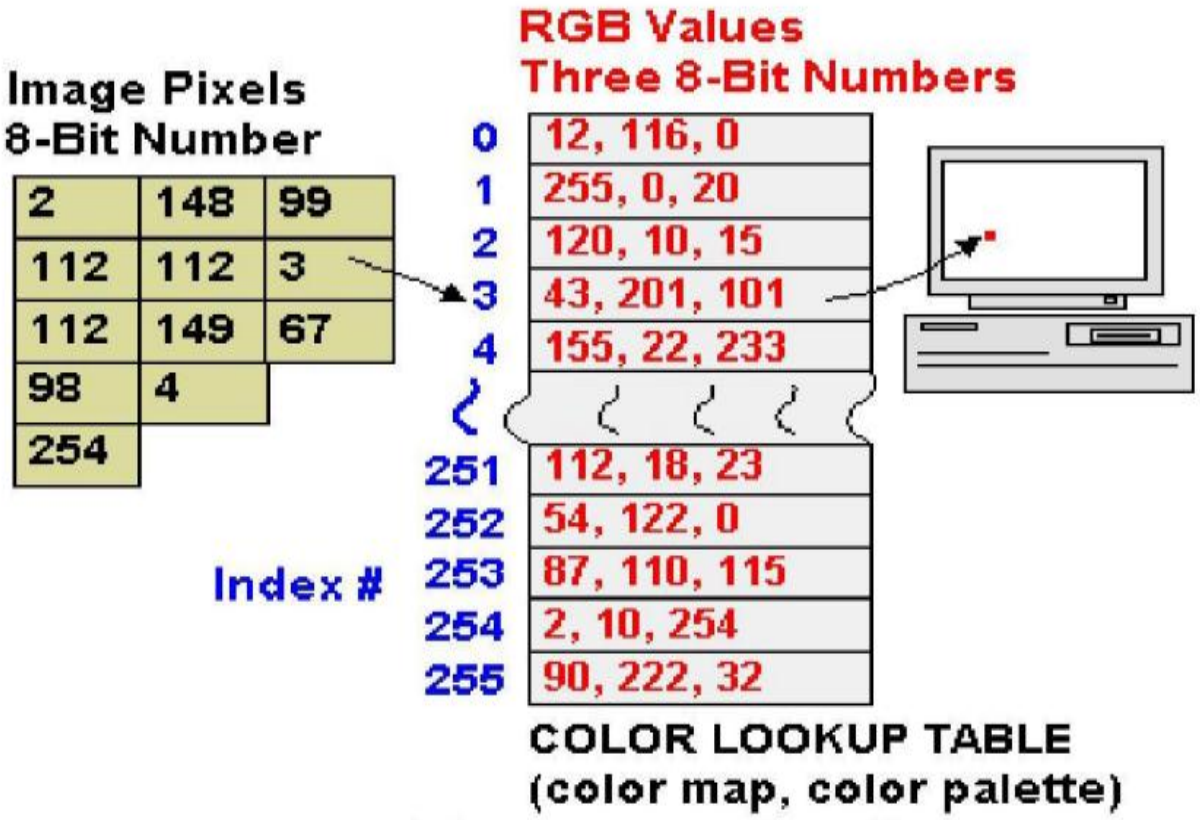
multimedia

Color Lookup Tables (LUT): Idea used in 8-bit color images is to store only index or code value for each pixel. While images are displayed as two dimensional arrays of values, they are usually stored in row-column order as simply a long series of values. LUT is often called a Palette.



Color picker: consists of an array of fairly large blocks of color, such that mouse click will select color indicated. A simple animation process is possible via simply changing color table Known as Color Cycling or Palette Animation. Dithering can also be carried out for color printers, using 1 bit per color channel and spacing out color with R, G, and B dots.

How to devise a Color Lookup Table

It gives idea of clustering to generate most important 256 colors from 24-bit color image. In general, clustering is expensive & slow process. This can be done in two ways: Method 1: it is straight forward way to make 8-bit lookup color out of 24-bit colors by dividing RGB cube into equal slices in each dimension.

- Divide 24 bit color RGB cube into equal slices in each dimension
- Center of each of resulting cube would serve as entries in color LUT
- This will scale RGB of 24 bit into 8 bit code
- Shrinking R range from 0 to 255
- 0to 7 which takes 3 bits only

- Similarly, G range from 0 to 255 0 to 7 which takes 3 bits only
- Finally, B range from 0 to 255 0 to 3 which takes 2 bits only
- Combining all the resulting bits give 8 bit color value.

Method 2: Median Cut Algorithm

This approach derives from computer graphics. The idea is as follows:

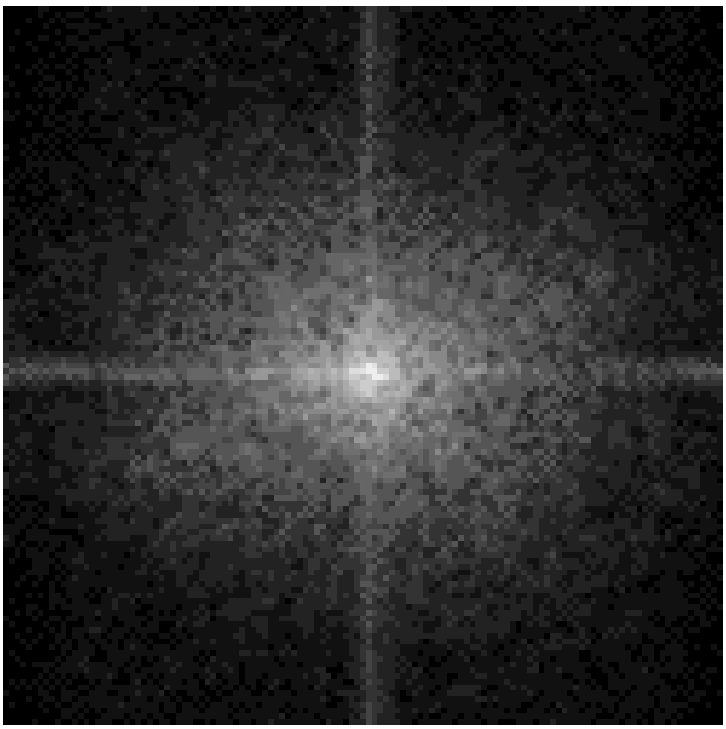
- Sort R byte values & find their median. Then values smaller than median are labeled with 0 bit & values larger than median are labeled with 1 bit.
- Next consider only pixels with 0 label from first step & sort their G values. Again label image pixels with another bit 0 for less than median in greens & 1 for greater
- Carrying on to blue channel, we have 3-bit scheme
- Repeating all steps, R, G, B results 6-bit & cycling through R & G once more results 8-bits.

These bits form out 8-bit color index value of pixels & 24-bit colors can be centers of resulting small color cubes. Accurate version of Median Cut algorithm

- Find smallest box that contains all colors
- Sort enclosed colors along longest dimension of box
- Split box into two regions at median of sorted list
- Repeat 2& 3 until original color space divided to 256 regions
- For every box, call mean of R, G, B in that box representative color for box
- Based on Euclidean distance between pixel RGB & box center assign every pixel to one of representative colors
- Repeat pixel by code in lookup table that indexes representative colors.

DCT , DWT and DFT Techniques of Image

Fourier Transform



Common Names: Fourier Transform, Spectral Analysis, Frequency Analysis

Brief Description

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the *Fourier* or frequency domain, while the input image is the spatial domain equivalent. In

the Fourier domain image, each point represents a particular frequency contained in the spatial domain image.

The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression.

How It Works

As we are only concerned with digital images, we will restrict this discussion to the *Discrete Fourier Transform* (DFT).

The DFT is the sampled Fourier Transform and therefore does not contain all frequencies forming an image, but only a set of samples which is large enough to fully describe the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image, *i.e.* the image in the spatial and Fourier domain are of the same size.

For a square image of size N×N, the two-dimensional DFT is given by:

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\, e^{-\iota 2\pi (\frac{ki}{N} + \frac{lj}{N})}$$

where *f(a,b)* is the image in the spatial domain and the exponential term is the basis function corresponding to each point *F(k,l)* in the Fourier space. The equation can be interpreted as: the value of each point *F(k,l)* is obtained by multiplying the spatial image with the corresponding base function and summing the result.

The basis functions are sine and cosine waves with increasing frequencies, *i.e. F(0,0)* represents the DC-component of the image which corresponds to the average brightness and *F(N-1,N-1)* represents the highest frequency.

In a similar way, the Fourier image can be re-transformed to the spatial domain. The inverse Fourier transform is given by:

$$f(a,b) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l)\, e^{\iota 2\pi (\frac{ka}{N} + \frac{lb}{N})}$$

Note the $\frac{1}{N^2}$ normalization term in the inverse transformation. This normalization is sometimes applied

to the forward transform instead of the inverse transform, but it should not be used for both.$$

To obtain the result for the above equations, a double sum has to be calculated for each image point. However, because the Fourier Transform is *separable*, it can be written as

$$F(k,l) = \frac{1}{N} \sum_{b=0}^{N-1} P(k,b)\ e^{-\imath 2\pi \frac{lb}{N}}$$

where

$$P(k,b) = \frac{1}{N} \sum_{a=0}^{N-1} f(a,b)\ e^{-\imath 2\pi \frac{ka}{N}}$$

Using these two formulas, the spatial domain image is first transformed into an intermediate image using *N* one-dimensional Fourier Transforms. This intermediate image is then transformed into the final image, again using *N* one-dimensional Fourier Transforms. Expressing the two-dimensional Fourier Transform in terms of a series of *2N* one-dimensional transforms decreases the number of required computations.

Even with these computational savings, the ordinary one-dimensional DFT has N^2 complexity. This can be reduced

to $N \log_2 N$ if we employ the *Fast Fourier Transform* (FFT) to compute the one-dimensional DFTs. This is a significant improvement, in particular for large images. There are various forms of the FFT and most of them restrict the size of the input image that may be transformed, often to $N = 2^n$ where *n* is an integer. The mathematical details are well described in the literature.

The Fourier Transform produces a complex number valued output image which can be displayed with two images, either with the *real* and *imaginary* part or with *magnitude* and *phase*. In image processing, often only the magnitude of the Fourier Transform is displayed, as it contains most of the information of the geometric structure of the spatial domain image. However, if we want to re-transform the Fourier image into the correct spatial domain after some processing in the frequency domain, we must make sure to preserve both magnitude and phase of the Fourier image.

The Fourier domain image has a much greater range than the image in the spatial domain. Hence, to be sufficiently accurate, its values are usually calculated and stored in float values.