

Digital Image Compression

By:

Suresh Kumar Dhayal iit2014060

Anupam Jaiswal iit2014038

Shubham Gupta iit2014061

Sarthak Panda iit2014063

Under the supervision of:

Prof. Anupam Agarwal

IIIT Allahabad

Image Compression - Why and How

A digital image is a numeric representation of (normally binary) a two-dimensional image. To efficiently use these images and store them effectively so they consume less space we try to reduce its size and losing as little information as possible. These images are stored as vectors, and the redundancies in these vectors is exploited for use in compression.

Ex: 3504X2336 (full color) image : $3504 \times 2336 \times 24/8 = 24,556,032$ Byte = 23.418 Mbyte

Data compression is defined as the process of encoding data using a representation that reduces the overall size of data. In general, three basic redundancies exist in digital images that follow

Coding Redundancy:

- Inter - pixel Redundancy:
- Psycho visual Redundancy

Image Compression - Why and How

Compression ratio:

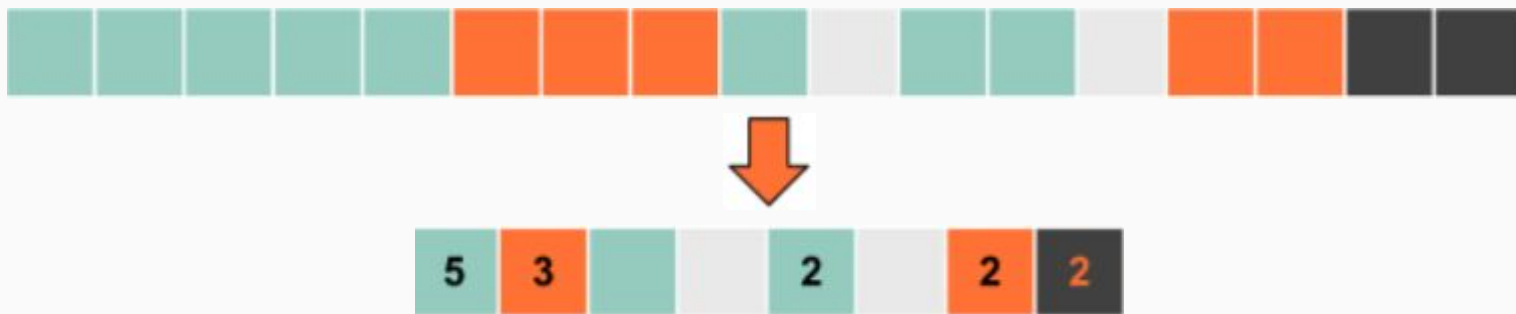
$$C = b/b'$$

Relative data redundancy:

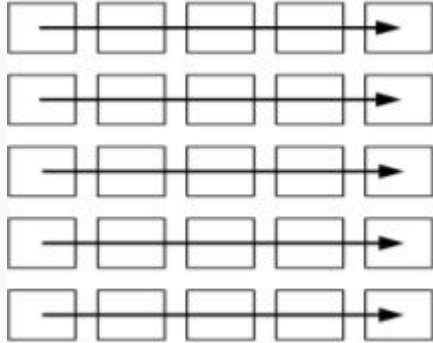
$$(R) = 1 - 1/C$$

Run Length Encoding

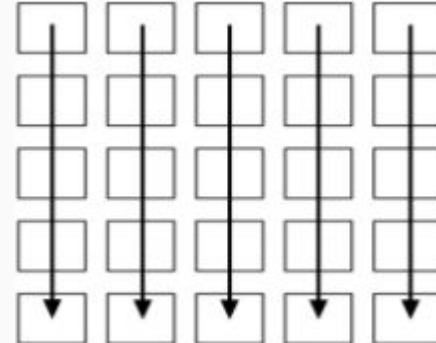
Run-length encoding is a data compression algorithm that helps us encode large runs of repeating items by only sending one item from the run and a counter showing how many times this item is repeated. It is useful when it comes to image compression, because images happen to have long runs of pixels with identical color.



Run Length Encoding



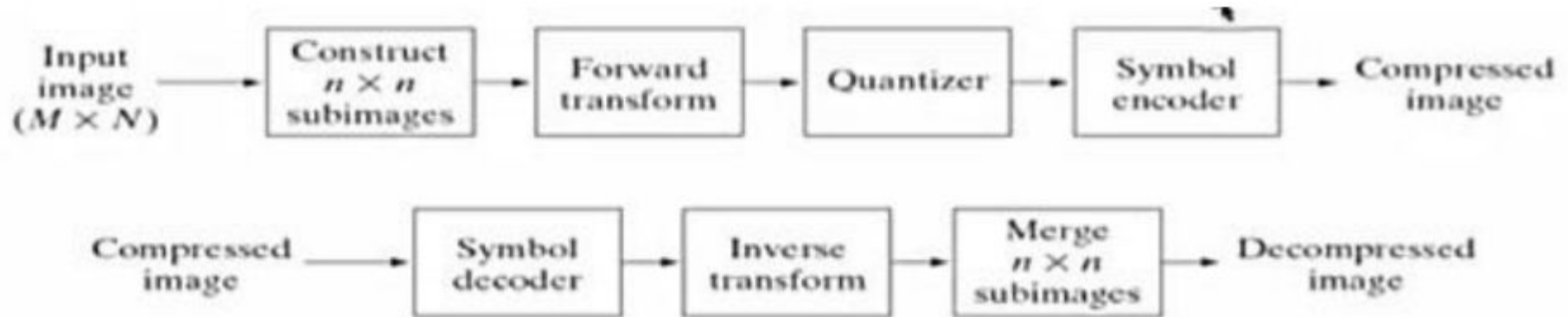
Row by row run length encoding



Column by column run length encoding

Lossy Compression Methods

Lossy compression techniques usually comprise 3 steps -
Transformation->Quantization->Encoding



Transform Coding

It uses a reversible and linear transform to decorrelate the original image into a set of coefficients in transform domain. The coefficients are then quantized and coded sequentially in transform domain. Some transforms used in transformation coding are:

- KLT (Karhunen-Loeve transform) : Optimal but difficult to compute
- DFT (discrete Fourier transform): Approximate energy packing efficiency of KLT and more efficient
- DCT : Requires half the storage space as DFT. Thus the most suitable choice

Transform Coding

Block Transform Coding:

Exploits correlation of the pixels within a number of small blocks that divide the original image. As a result, each block is transformed, quantized and coded separately. Is utilized in the ISO JPEG

Full Frame Transform Coding:

The transform is applied to the whole image as a single block. The tradeoff is increased computational cost

Discrete Cosine Transform:

1D DCT

$$F(u) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \Lambda(i) \cdot \cos \left[\frac{\pi \cdot u}{2 \cdot N} (2i + 1) \right] f(i)$$

Inverse 1D DCT

$$\Lambda(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } i = 0 \\ 1 & \text{otherwise} \end{cases}$$

2D DCT

$$F(u, v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i) \cdot \Lambda(j) \cdot \cos \left[\frac{\pi \cdot u}{2 \cdot N} (2i + 1) \right] \cos \left[\frac{\pi \cdot v}{2 \cdot M} (2j + 1) \right] \cdot f(i, j)$$

Inverse 2D DCT

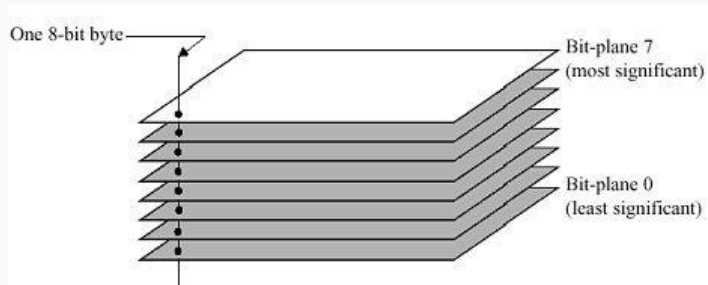
$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

DCT Encoding

- Suppose the input image is N by M and $f(i,j)$ is the intensity of the pixel in row i and column j
- $F(u,v)$ is the DCT coefficient in row k_1 and column k_2 of the DCT matrix.
- For most images, much of the signal energy lies at low frequencies; these appear in the upper left corner of the DCT.
- Compression is achieved since the lower right values represent higher frequencies, and are often small - small enough to be neglected with little visible distortion.
- The DCT input is an 8 by 8 array of integers. This array contains each pixel's gray scale level

Bit Plane Slicing and encoding

In bit plane slicing , for slicing one bit plane we set all other bit to 0 except that particular bit for which we are performing slicing. Each bit plane contains different amount of data. , so removing those bit planes which don't have significant data would help in compressing image



Lossless Image Compression

Lossless compression is a class of data compression algorithms that allows the original data to be perfectly reconstructed from the compressed data.

We will be implementing 3 lossless encoding techniques:

- Huffman Compression
- Run Length Encoding
- Discrete Wavelet Transform

Huffman Compression

We can demonstrate huffman compression with an example:
Suppose we have a 5×5 raster image with 8-bit color, i.e. 256 different colors(fig. h1). The uncompressed image will take $5 \times 5 \times 8 = 200$ bits of storage.

First, we count up how many times each color occurs in the image. Then we sort the colors in order of decreasing frequency. We end up with a row that looks like fig. h2

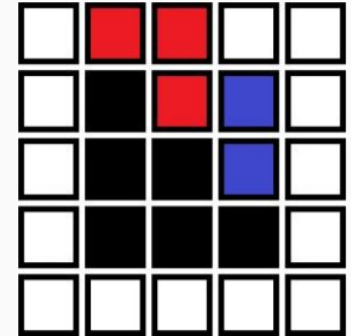


Fig. h1

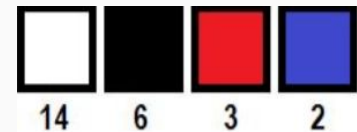
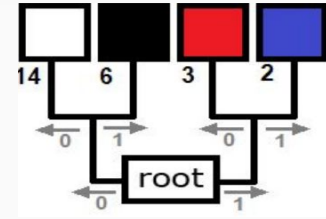


Fig. h2





Huffman Compression

Now we put the colors together by building a tree such that the colors farthest from the root are the least frequent. The colors are joined in pairs, with a node forming the connection.

Because each color has a unique bit code that is not a prefix of any other, the colors can be replaced by their bit codes in the image file. The most frequently occurring color, white, will be represented with just a single bit rather than 8 bits. Black will take two bits. Red and blue will take three. After these replacements are made, the 200-bit image will be compressed to $14 \times 1 + 6 \times 2 + 3 \times 3 + 2 \times 3 = 41$ bits



Huffman Tree

color	freq.	bit code
	14	00
	6	01
	3	10
	2	11

Encoded colors

Discrete Wavelet Transform

The discrete wavelet transform is a very useful tool for signal analysis and image processing, especially in multi-resolution representation. It can decompose signal into different components in the frequency domain

Two-dimensional discrete wavelet transform (2-D DWT) decomposes an input image into four sub-bands, one average component (LL) and three detail components (LH, HL, HH)

LL	HL
LH	HH

Discrete Wavelet Transform

For image compression, Haar wavelet transform is preferable because the operation for Haar DWT is simpler. Haar wavelets are real, orthogonal, and symmetric and the coefficients for high pass and low pass filter are simpler.

Haar Transform

Step 1:

A	B	C	D



A+B	C+D	A-B	C-D
L		H	

Step 2:

A		C	
B		D	
L		H	



A+B			C+D		
LL		HL			
A-B			C-D		
LH		HH			

LL1	HL1
LH1	HH1

First level



LL2	HL2	HL1
LH2	HH2	
LH1		HH1

Second level

Most important
part of the image

LL3	HL3	HL2	HL1
LH3	HH3		
LH2		HH2	
LH1			HH1

Third level

Inverse DWT

Just as a forward transform is used to separate the image data into various classes of importance, a reverse transform is used to reassemble the various classes of data into a reconstructed image. A pair of high pass and low pass filters are used here also. This filter pair is called the Synthesis Filter pair. The filtering procedure is just the opposite - we start from the topmost level, apply the filters column wise first and then row wise, and proceed to the next level, till we reach the first level

Thank You