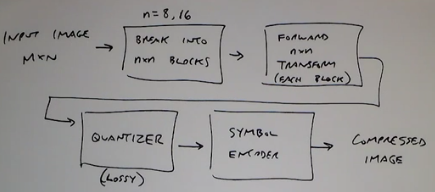
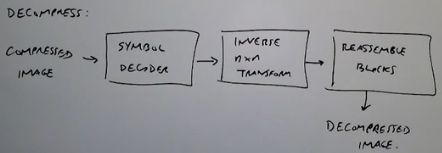
**LOSSY IMAGE COMPRESSION**

More compression gains can be accessed when strategically throwing away information that the human visual systems is not going to notice.

Block Transform Coding:

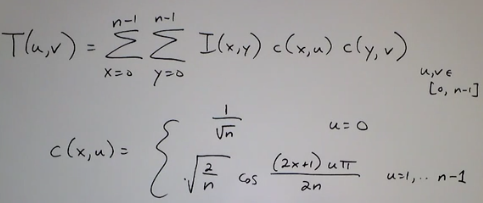




Now what 2D transform can be used?

* DFT (not used because they have complex values)
* DCT (JPEG) favored because it is real valued and compacts energy into low freq.
* Hadamard
* Haar
* Wavelet (JPEG 2000)

DCT:

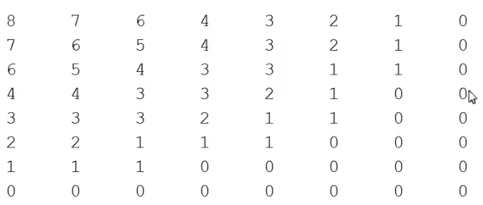


So the image is divided into blocks of 8x8 pixels and each of those is converted to 8x8 DCT coefficients. Now we can perform compression.

Possibilities for compression:

* ***Zonal coding***: tells how many bits per coefficient as a function of (u, v) is needed.

An ideal zonal matrix would look like:



Here the important low frequencies are packed with higher number of bits in top left corner. And as we move away from it the higher frequency coefficients can be stored with less nimber of bits. And majority of other coefficients can be thrown away.

In this case the average bits used per pixel is ~2 bits per pixel.

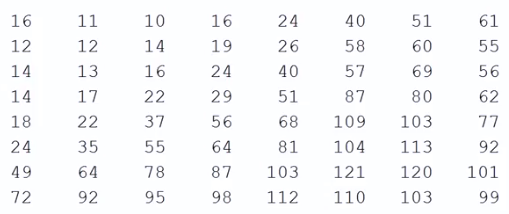
* ***Threshold coding***:
  + Select the ‘**m**’ largest coefficients and throw the others away.
  + Select the coefficients that account for approximately 95% of the total energy in the block
  + Select all coefficients above a threshold ‘**T**’.

So of the above techniques, the threshold coding is better, because it preserves those coefficients having larger signal energy. But zonal coding would throw away coefficients in lower space even if it has high signal energy.

How JPEG actually works:

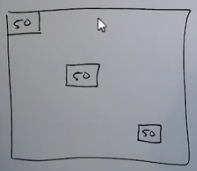
Specify a normalization matrix for a block T(u, v), this is a way of implicitly specifying how many levels for each coefficient.

Consider the following matrix that tells how to go about doing quantization:



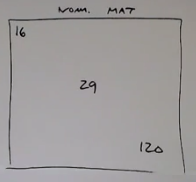
There are smaller numbers in upper corner and larger ones in the bottom corner(inverse of zoning).

Now consider that after applying transformation to a block it came out to be ‘50’.



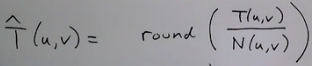
Now each of these would have different number of bit representations.

And looking at the corresponding positions of the normalization matrix:

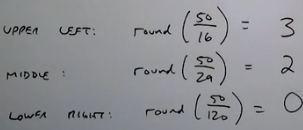


So each of these positions get quantized in a different way.

The quantized value is obtained as follows:



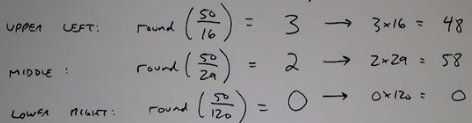
So for each of those three positions it would be:



Now upon reconstruction (decompressed):



So each of those values would be reconstructed as:



So idea is, smaller the value of the normalization matrix, closer is the reconstructed value to original value. The lower right hand corner quantizing with a larger value so that it differs a lot from original value.

So smaller the value in the normalization matrix, the more accurate the reconstructed coefficient.

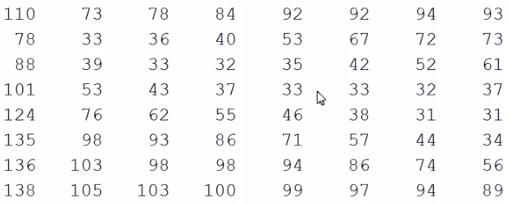
To adjust “JPEG quality” the normalization matrix can be multiplied by a number

* less than 1 that leads to higher quality
* Or a number greater than 1 that leads to worse quality

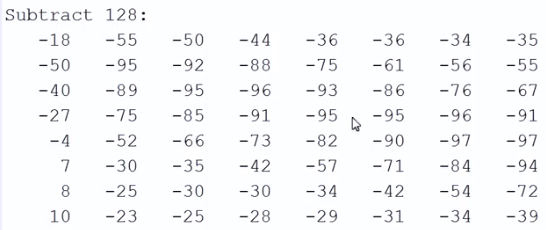
Further details:

1. After quantization, JPEG reorders the coefficients in a zig-zag pattern, and then applies a lossless coder (Huffman, RLE).

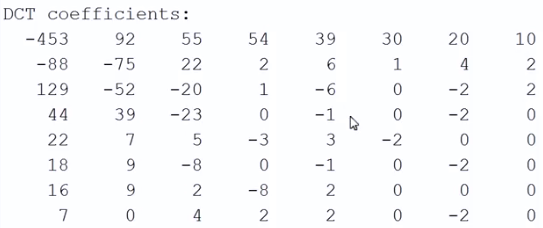
Example: below is the original block of PEG coefficients:



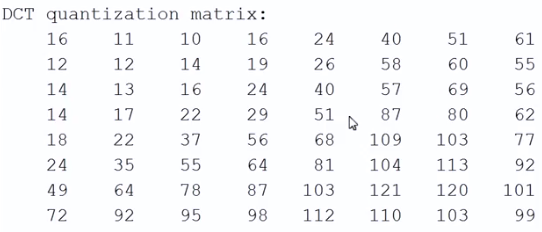
To get 0 average:



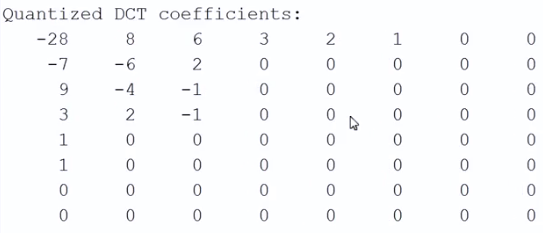
Applying DCT:



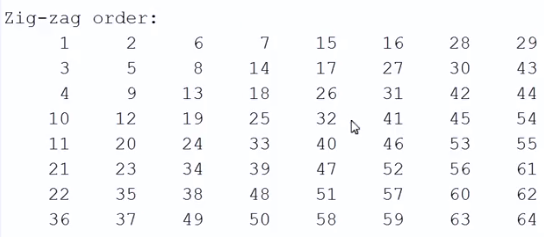
Here is the quantization matrix:



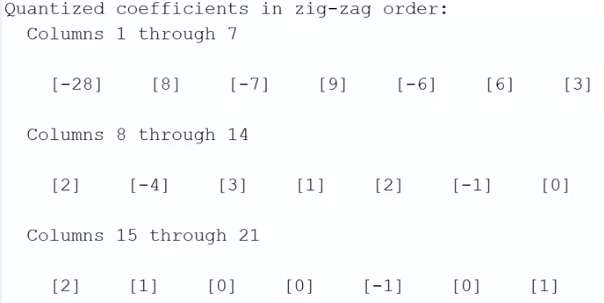
Dividing the previous matrix by the one above we get:



Now performing the zig-zag ordering in this manner:



Following is the result:



To this lossless coding is performed.

Zig-zag is performed because after a certain point all the coefficients become 0. The 0 ‘s are all clustered towards the end, making coding easier. How? So by applying Huffman code to this, we can approximate how many numbers repeated itself many times rather than having 0’s pop in between these numbers.

1. DC coefficient is coded separately w.r.t the DC for the previous sub-block
2. For color, we convert the image to a luminance/chrominance color space (LUV or LAB)

So in these images the intensity channels are coded at higher bit rates but chroma channels at a lower bit rate. This is because; the human eye can perceive intensity changes more than color changes.

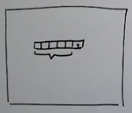
***Predictive Coding***:

In JPEG compression, each block is coded independently. In general for real images, most of the neighboring block has values closer to each other. So having a predictive coding mechanism is fair where in most of the cases it would predict the almost correct code and in few cases it would have errors.

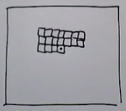
Basic idea: images have a lot of correlation between neighboring pixels, so we will use the previous pixels values to predict the value of the next pixels and code the error in the prediction.

Possibilities:

1. Predict value of one pixel in a row after seeing 5 consecutive pixels in the same row:

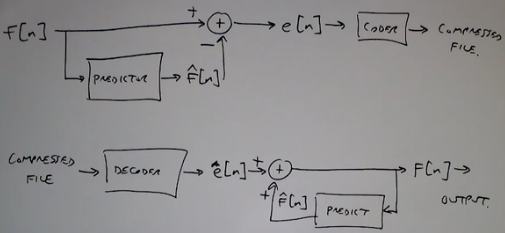


1. Predict the value based on its neighbors

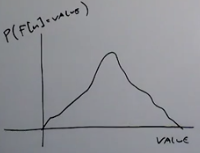


In a way this is causality because we do not have to look into the future blocks to predict the current block.

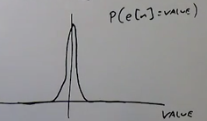
Compression and decompression:



So the distribution of actual values looks like:

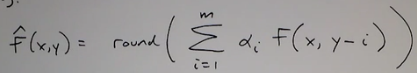


And the errors are plotted like a delta function:

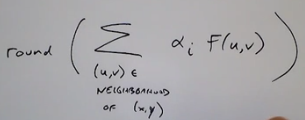


So reason predictive coding reduces the entropy of the distribution and gives great compression gain.

For example, the value of pixel at point (x, y) would be weighted average of previous pixels in that row:

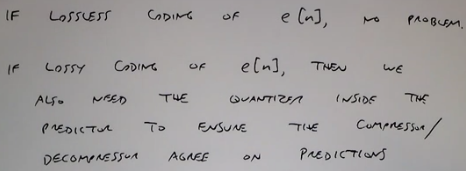


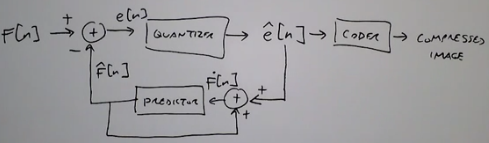
Or it can also be weighted average of collection of pixel values in a neighborhood:



It is also known as an autoregressive process.

**NOTE**: two points



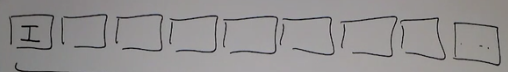


***Video Compression***:

In videos images are closely spaced (twenty fourth or thirtieth of a second). So basically the neighbouring frames contain SIMILAR looking images. There is no need to encode each frame separately.

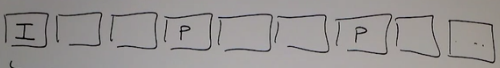
Most video compression algorithms use ***block-based motion compensation***.

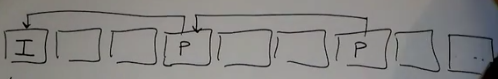
So there are a collection of frames/pictures called group of pictures (GoP)

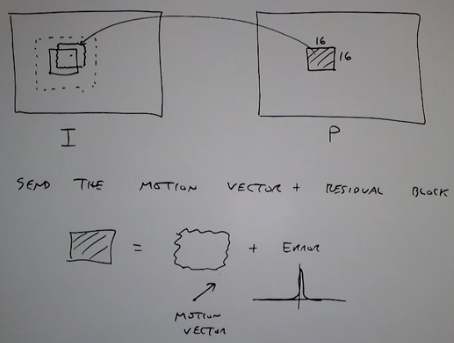


The first frame of each GoP is called the I frame.

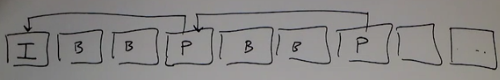
Then every once in a while there are P frames (predictive frames). Here P frames predict macroblocks based on previous P or I frames:



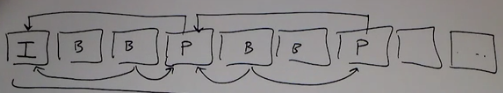




And the remaining frames are called B frames (bi-directional) frames



For these B frames predict the macroblock from P/I frames on either side.



So depending on the position of P frame there would be delay in predicting the B frame. Because if distance between P frames are large there would be a lot of B frames in between and it would takea lot of time to predict the best value for that macroblock.