**Chapter 7**

**Image Compression**

Image compression reduces the amount of data needed to represent an image and makes it possible to have efficient storage and transmission.

There is considerable redundancy in an image representation which is exploited in compression techniques to reduce the amount of data.

**Redundancies**

**Coding redundancy**: fixed length code is used for all pixels – i.e. 8-bit from 00000000 to 11111111 whereas an image can have fewer graylevels.

**Interpixel redundancy**: Neighboring pixels are strongly correlated – a pixel value is a prediction of its surrounding pixels.

**Psychological redundancy**: Information presented in an image is not of equal importance to the human visual system. The number of graylevels beyond a limit cannot be distinguished by the eye. The same is true for the maximum resolution.

**Interframe redundancy**: This is applicable to video data and is the temporal equivalent of interpixel redundancy. In high rate video frames, the changes in pixel values from one frame to the next are not significant.

**7. 1 Performance Criterion**

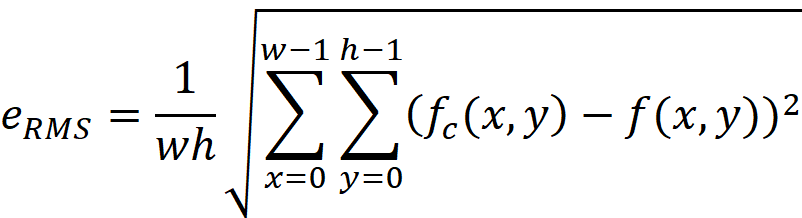
The performance of a compression algorithm is defined by the compression ratio

 (1)

When lossy compression is used, the decompressed image is not the same as the original image. Quantitative measures are:

(a) **RMS (Root Mean square) Error**: Let f(x,y) and fc(x,y) be the original and compressed images, respectively. Then

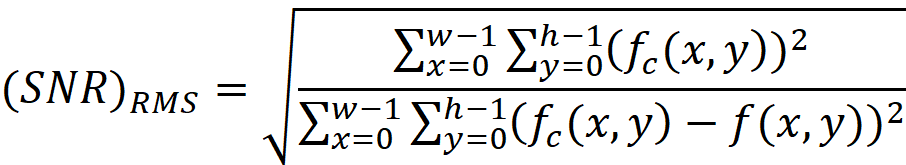
(2)



where *h* and *w* are the image height and width. Lower values are indications of better fidelity.

(b) **Signal to Noise Ratio (SNR) metric**: It is defined as

(3)



Higher values are indication of better compression.

**7.2 Lossless Compression Methods**

1. **Delta or Differential Coding**

Record the values of the first pixel in a row, and difference between greylevel (GL) of each pixel and the previous pixel in that row. The difference can be coded using fewer bits.

215 216 214 217 212 🡪 215 1 -1 2 -3

**(b) Run Length Encoding (RLE)**

RLE exploits high interpixel redundancy in simple images. Provides the count of the number of pixels with the same GL value. It is most effective for binary images

**Example** (binary image)

Raw code

0 0 0 0 0 0 0 0 8

1 1 0 0 0 1 1 1 0,2,3,3

1 1 1 1 1 1 0 1 0,6,1,1

0 0 0 1 1 1 1 1 3,5

Note: The first number is the number of zeros in a row encountered first in a row.

**Example**: Consider greylevel image:



Scan Convert:



Encode:



For non-binary images first, bit plane slicing is applied and then for each bit plane the RLE is performed .

Typical compression ratios are 0.5 to 1.2 so it is not a good compression method for complex images – it is only used for simple images such as graphic files. Alternatively, preprocessing can be applied to reduce graylevel values, but this amounts to lossy compression



0 1 2



3 4 5

1001 01101



6

Fig. 1 Bit-plane slicing of a 7-bit image. Note that lower bit planes provide details of the image compared with higher bit planes that provide overall shape.

Another method to extend RLE to GL images is to use two parameters to characterize a run (GL, RL) where GL is greylevel and RL is the run length.

Example

Raw code

20 20 20 20 15 15 15 15 20,4, 15,4

2 2 2 2 2 7 7 7 2,5, 7,3

0 0 0 1 1 1 1 1 0,3, 1,5

5 5 5 10 10 10 13 13 5,3, 10,3, 13,2

The resulting code is 20,4,15,4,2,5,7,3,0,3,1,5,5,3,10,3,13,2. The number of pixels in a row must be specified.

**(c) Statistical Coding**

This technique removes the coding redundancy that exists when using fixed length (e.g. 8-bit) codewords for every pixel.

Information theory tells us that the amount of information conveyed by a codeword relates to its probability of occurrence. Codewords that occur rarely convey more information than those that occur frequently. A good coding will use more bits for rare codewords and fewer bits for frequent codewords.

To measure the average information content, the concept of entropy is defined as

 (4)

where h(i) is the probability of the occurrence of the i-th GL value, which is the normalized histogram of the image, i.e.

 (5)

E provides a measure for average number of bits per pixels. It is a measure for judging the success of the coding scheme.

**Example**

Let L = 8 and  for i=0,1,…,7; i.e. equalized histogram. Then . This tells us that the theoretical minimum for lossless coding is 3 bits/pixel. There is no better way than the standard coding 000, 001, …, 111. In other words, uniform distribution (equalized histogram) has the highest entropy.

Now if , and  for i=1,3,4,…,7, then  and minimum of 0 bit/pixel is needed. The entire image has a GL of 2, and this is a certain event with a probability of 1.

Notes:

1. The range of  is . Upper value is when h(i)= 1/L.

2. The more randomness in an image, the more evenly distributed GL

and the more bits/pixel are required.

3. We need to assign fewer bits to code with more likely events.

The metric to evaluate coding performance is to use average number of bits/pixel (length), i.e.

 (6)

where  is the length in bits of the code for the i-th GL. The closer  is to E, the better is the code.

**Huffman Coding**

In this coding the codewords are chosen in such a way that  is as close as possible to E. This code is minimum length, though variable length. Typical reduction is 10% to 50%, i.e. 1.1:1 to 1.5:1.

Algorithm

1. Compute normalized histogram
2. Order h(i) from highest to lowest
3. Combine two smallest
4. Go to Step 2 until two probabilities are left
5. Work backward along a Huffman tree to generate the code by alternatively assigning 0 and 1.

Example:

Let h(0) = 0.2, h(1) = 0.3, h(2)= 0.1, h(3) = 0.4. Order as

0.4 0.3 0.2 0.1 Huffman tree:

0.6(0)

0.3

0.4(1) 0.3 0.2 0.1

0.6(0)

0.3(01)

0.4(1) 0.3 (00) 0.2 0.1

0.6(0)

0.3(01)

0.4(1) 0.3 (00) 0.2(010) 0.1(011)

Note that two GL have 3 bits assigned to them, two with 2 bits and one GL has just 1 bit. The latter is the pixel most likely to occur (0.4 or 40% of the time).





This is the average length with Huffman code. The compression ratio is 2.0:1.9, providing 5% compression.

**Example:**



**Comparison of Lossless Methods**

The compression ratios are dependent on the type of image, i.e. synthetic or natural. Typical ratios are given below

Image delta RLE Huffman

Synthetic 1.95 60 6

Natural 1.8 1.1 1.6

Generally the compression ratios are higher with synthetic images since there are much coding and interpixel redundancies in these types of images. In particular RLE is most suitable for synthesis images since there are few greylevels and long runs of the same greylevel.

**7.3 Lossy Compression using Dithering**

The reduced number of bit is possible with dithering, with little loss of information as shown in Fig. 2.



1. (b) (c)

Fig. 2 (a) 24 bit color image, (b) 8 bit color image (c) 8 bit color image with dithering.

**7.4 Lossy Compression Techniques – JPEG**

Lossy compression relies on the fact the human visual system is insensitive to the loss of certain information such as small changes in the greylevel values.

The Joint Photographic Experts Group (JPEG), has specified an algorithm based on a frequency domain transformation, the so called discrete cosine transform (DCT). This transform is similar to Fourier transform but contains only real data.

The DCT is defined as



……………….. (7)

 ; 

The inverse cosine transform is



……………(8)

where a and b are defined as above.

The compression consist of (i) Splitting the image into square subimages or blocks (i.e. 8 pixel by 8 pixel blocks), (ii) applying cosine transform to the sub-images, (iii) quantization and (iv) encoding.

* The goal of compression is to pack as much information as possible into smaller number of transform coefficients C(u,v).
* Quantization selectively eliminates the coarsely quantized transform coefficients that carry least information.
* The encoding is to represents the coefficients into variable length code.
* The division of the image into sub-images reduces the redundancy between adjacent sub-images.

The transform is performed on small blocks of the image since performing the transform on the entire image is computationally intensive. In addition, discarding the high frequency components on the entire image has the effect of low pass filtering which blurs all parts of the image. However, we want to achieve high compression ratios while minimizing perceptible information. Therefore we perform transformation on blocks and remove the high frequency components only in areas of the image where the loss can be tolerated.

Block transformation may suffer from discontinuity effects discussed before. However, the symmetry and other characteristics of the Cosine Transform is such that it is less sensitive to these discontinuities than Fourier transform.

**JPEG Compression Algorithm**

Split image into 8 by 8 blocks of pixels

for each block of pixels  {

1.Shift pixel values (greyleves) by subtracting 128 to get 

2. Compute cosine transform of the block to get 

3. Quantize the coefficients of  to get , see below.

4. Arrange coefficients into a 1-D sequence

5. Encode using delta method the first coefficient, i.e. .

6. Compress zero-values coefficients by RLE

7. Perform Huffman coding of the coefficients

8. Output coded coefficient for the block

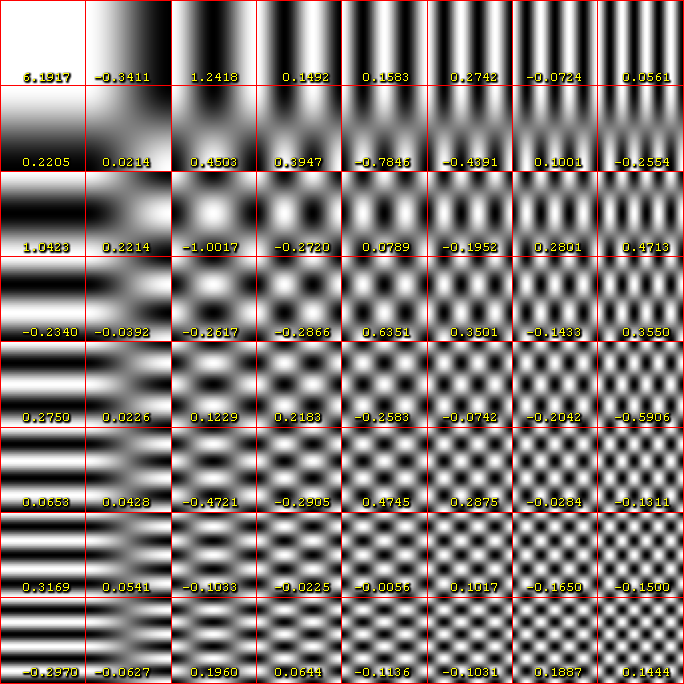
The subimages are processed left to right, top to bottom order. Padding is done if the image is not a multiple of 8.

**Quantization**: Each coefficient is quantized as

 u, v = 0,1,2,…,7 (9)

where Q(u,v) is a quantization table whose values are increase with higher frequencies, and the division is element-wise. This has the effect of reducing  which means that higher frequencies have reduced precision. A typical Q(u,v) is

 (10)



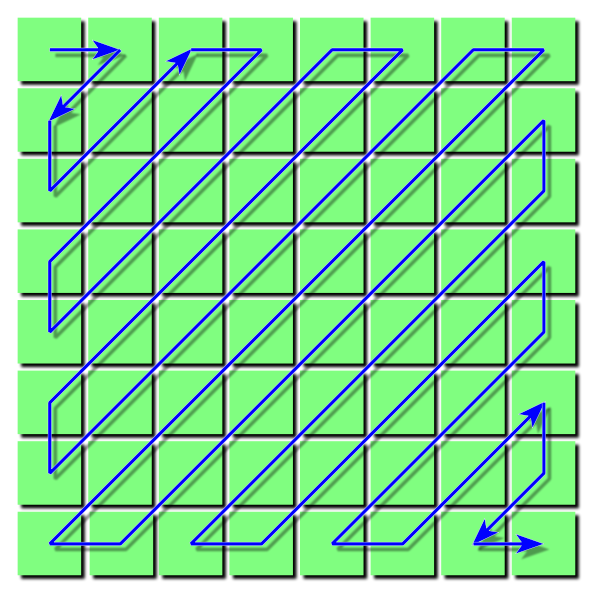
Values in Q(u,v) can be scaled up (or down) to decrease (or increase) the number of coefficients set to zero. This has the effect on the quality of compression (larger number of zeros corresponds to lower quality).

JPEG implementation also allows specifying a quality parameter to generate an appropriate Q(u,v) table instead of the above scaling up/down.

Quantization table also depends on image contents. In places of smoothly varying graylevel values, many DCT coefficient are set to zero, whereas at edges more DCT coefficient are non-zero. In this way JPEG algorithm blurs the image in places where blurring will not be noticeable and preserve the information in interesting (changing) part of the image.

After quantization, DCT coefficient are reordered in a 1-D sequence by following the zigzag order part of which is shown below, where 1, 2, …, 63 indicates the order.

 (11)



The first sequence value (zero frequency coefficient) is usually the highest value and is stored by delta encoding, i.e. the difference between its value and last value from previous block. The remainder of the sequence usually contains long runs of zeros which is compressed by run length coding (RLE). Huffman coding can be applied to compress the coefficients more.

To decompress: Follow the above process in reverse, i.e.

1. Huffman code is decoded and runs of zeros are expanded to give 64 values per block.
2. The 64 values are written into an 8 by 8 array in zigzag pattern and are dequantized using (9), i.e. multiplying them by values in the quantization table

 (12)

Note that  is an approximation of  due to rounding.

1. Inverse DCT is done and 128 is added to array values.

The result of the above is 8 by 8 blocks with values similar to those of original uncompressed image.

Example: Consider the compression of the following 8-bit subimage

 (13)

1. The subimage is shifted in values by -128

 (14)

1. Now we perform DCT on (14), given by (7) to get

 (15)

1. Quantize using (9) and (10), e.g. divide -415 by 16, etc. to get

 (16)

1. Note that transformation and quantization has produced a large number of zero coefficients. When the coefficients are reordered according to the zigzag ordering in (11), the resulting 1-D coefficient sequence is

[-26 -3 1 -3 -2 -6 2 -4 1 1 1 5 0 2 0 0 -1 2 0 0 0 0 0 -1 -1 EOB]

A special end EOB (end of block) Huffman code word is provided to indicate that the remainder coefficients are zero. The above is then coded as the Huffman binary sequence

To decompress a JPEG compressed subimage, the Huffman coded sequence is easily obtained and gives us back (16). After dequantization (12), we get

 (17)

Where for example -26\*16 =416 .

Now taking the inverse DCT according to (8), then shifting (adding 128), we get

 (18)

Comparing (18) with the original subimage (13) shown again below, we see that the errors range from -14 to +11 and the root mean square error is 5.9 graylevels.



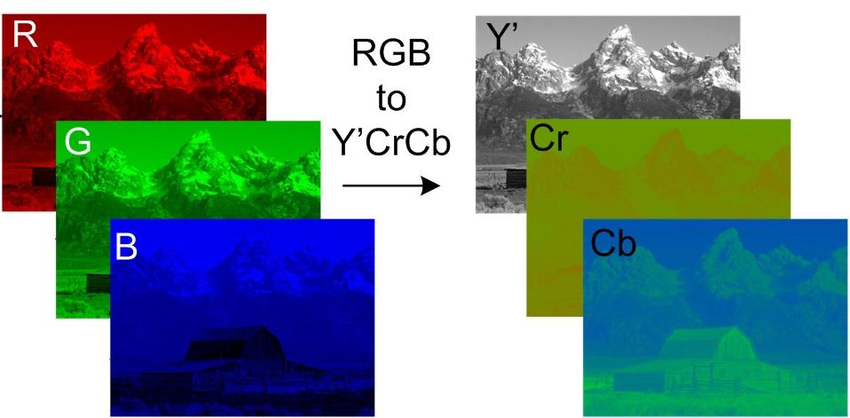
Quality Factor

Quality factor is related to how much of compression is used in JPEG. At quality Q=100, almost no information is lost and the compression is 2.6:1. At lower qualities of 50, the compression ratio is about 15:1. At qualities of Q=25, and 10 the compression ratios are about 23:1 and 46:1, respectively.

 Color Images

The first step is to transform the color space of the image to a space called YCbCr.  The Y component contains the brightness information and is a grayscale image.  The Cb and Cr components contain color information.  The YCbCr image contains exactly the same information as the RGB image, just in a different representation.

The reason for this transformation is that most of the information ends up in the Y image, with much less information in the chroma (Cb and Cr) images.  (Note that that the YCbCr image is the same size as the RGB; they have been downsized to fit the screen.)  Further, the human eye is more sensitive to errors in brightness (the Y image) than chrominance.  The consequence is that we don't really need to store the entire Cb and Cr images.  Instead, they are downsampled by 2.



Original 1/64 of DCT coefficients



3% 34%

Comparison of JPEG Images with different quality factors and file sizes

Example 1



Q=100, size=83KB, compression 2.6:1 Q=50, size=15KB, compression 15:1



Q=25, size=9.5KB, compression 23:1 Q=10, size=4.7KB, compression 46:1

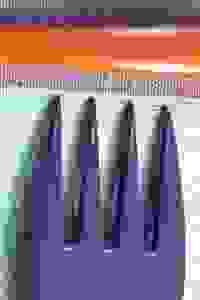
Example 2



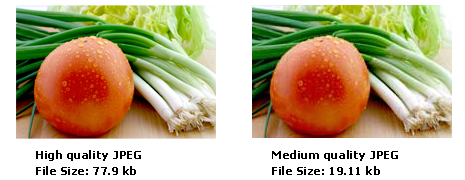
Q=99, 52K Q=90, 21K Q=80, 15K



Q=70, 12K Q=60, 10K Q=30, 6.8K



Q=10, 3.5K Q=1, 2K



Quality: 100; Size: 72.6 KB Quality: 70; Size: 15.2 KB



Quality: 40; Size: 9.3 KB Quality: 10; Size: 3.4 KB