Coding and Data Compression

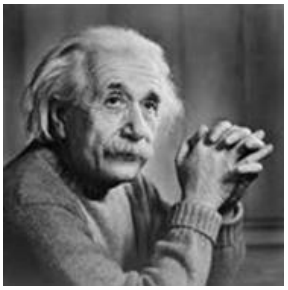
Lec. 5

Image Compression

A digital image is a rectangular array of dots, or picture elements, arranged in *m* rows and *n* columns. The expression *m*×*n* is called the *resolution* of the image, and the dots are called *pixels* (except in the cases of fax images and video compression, where they are referred to as *pels*). The term "resolution" is sometimes also used to indicate the number of pixels per unit length of the image. Thus, dpi stands for dots per inch.

Pixels

Images are all around us. We see them in color and in high resolution. Many objects (especially artificial objects) seem perfectly smooth, with no jagged edges and no graininess. Computer graphics, on the other hand, deals with images that consist of small dots, pixels. The term pixel stands for "picture element".



In the above picture, there may be thousands of pixels, that together make up this image.

We will zoom that image to the extent that we are able to see some pixels division. It is shown in the image below.

The value of the pixel at any point denotes the intensity of image at that location, and that is also known as gray level. Each pixel can have only one value and each value denotes the intensity of light at that point of the image.

We will now look at a very unique value 0. The value 0 means absence of light. It means that 0 denotes dark, and it further means that when ever a pixel has a value of 0, it means at that point, black color would be formed. Have a look at this image matrix

0	0	0
0	0	0
0	0	0

Now this image matrix has all filled up with 0. All the pixels have a value of 0. If we were to calculate the total number of pixels form this matrix, this is how we are going to do it. Total no of pixels = total no. of rows X total no. of columns. = 3 X 3= 9. It means that an image would be formed with 9 pixels, and that image would have a dimension of 3 rows and 3 column and most importantly that image would be black. The resulting image that would be made would be something like this:

Now why is this image all black? Because all the pixels in the image had a value of 0.

Image Types

For the purpose of image compression, it is useful to distinguish the following types of images:

1. **A bi-level (or monochromatic) image:** This is an image where the pixels can have one of two values, normally referred to as black and white. Each pixel in such an image is represented by one bit, making this the simplest type of image.
2. **A grayscale image:** Grayscale is a range of monochromatic shades from black to white. Therefore, a grayscale image contains only shades of gray and no color.A pixel in such an image can have one of the n values 0 through $n - 1$, indicating one of 2n shades of gray (or shades of some other color). The value of n is normally compatible with a byte size; i.e., it is 4, 8, 12, 16, 24, or some other convenient multiple of 4 or of 8. The set of the most-significant bits of all the pixels is the most-significant bitplane. Thus, a grayscale image has n bitplanes.
3. **A continuous-tone image:** This type of image can have many similar colors (or grayscales). When adjacent pixels differ by just one unit, it is hard or even impossible for the eye to distinguish their colors. As a result, such an image may contain areas with colors that seem to vary continuously as the eye moves along the area. A pixel in such an image is represented by either a single large number (in the case of many grayscales) or three components (in the case of a color image). A continuous-tone image is normally a natural image (natural as opposed to

artificial) and is obtained by taking a photograph with a digital camera, or by scanning a photograph or a painting. Next figure is typical examples of continuous-tone images.

4. **A discrete-tone image:** (also called a graphical image or a synthetic image). This is normally an artificial image. It may have a few colors or many colors, but it does not have the noise and blurring of a natural image. Examples are an artificial object or machine, a page of text, a chart, a cartoon, or the contents of a computer screen.

   Compression methods for continuous-tone images often do not handle sharp edges very well, so special methods are needed for efficient compression of these images. Notice that a discrete-tone image may be highly redundant, since the same character or pattern may appear many times in the image. Next figure is a typical example of a discrete-tone image.

5. **A cartoon-like image:** This is a color image that consists of uniform areas. Each area has a uniform color but adjacent areas may have very different colors. This feature may be exploited to obtain excellent compression.

It is clear that each type of image may feature redundancy, but they are redundant in different ways. This is why any given compression method may not perform well for all images, and why different methods are needed to compress the different image types. There are compression methods for bi-level images, for continuous-tone images, and for discrete-tone images. There are also methods that try to break an image up into continuous-tone and discrete-tone parts, and compress each separately.

Computer graphics is used in many areas in everyday life to convert many types of complex information to images. Thus, images are important, but they tend to be big.

Modern hardware can display many colors, which is why it is common to have a pixel represented internally as a 24-bit number, where the percentages of red, green, and blue occupy 8 bits each.

Videos are also commonly used in computers, making for even bigger images. This is why image compression is so important. An important feature of image compression is that it can be lossy. An image, after all, exists for people to look at, so, when it is compressed, it is acceptable to lose image features to which the eye is not sensitive. This is one of the main ideas behind the many lossy image compression methods described in this lecture.

In general, information can be compressed if it is redundant. It has been mentioned several times that data compression amounts to reducing or removing redundancy in the data. With lossy compression, however, we have a new concept, namely compressing by removing irrelevancy. An image can be lossy-compressed by removing irrelevant information even if the original image does not have any redundancy.

Error Metrics

Developers and implementers of lossy image compression methods need a standard metric to measure the quality of reconstructed images compared with the original ones. The better a reconstructed image resembles the original one, the bigger should be the value produced by this metric. Such a metric should also produce a dimensionless number, and that number should not be very sensitive to small variations in the reconstructed image. A common measure used for this purpose is the *peak signal to noise ratio* (PSNR). It is familiar to workers in the field, it is also simple to calculate, but it has only a limited, approximate relationship with the perceived errors noticed by the human visual system. This is why higher PSNR values imply closer resemblance between the reconstructed and the original images, but they do not provide a guarantee that viewers will like the reconstructed image.

Denoting the pixels of the original image by *Pi* and the pixels of the reconstructed image by *Qi* (where $1 \leq i \leq n$), we first define the *mean square error* (MSE) between the two images as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(P_i - Q_i)^2.$$

It is the average of the square of the errors (pixel differences) of the two images. The *root mean square error* (RMSE) is defined as the square root of the MSE, and the PSNR is defined as:

$$\text{PSNR} = 20 \log_{10} \frac{\max_i |P_i|}{\text{RMSE}},$$

Intuitive Methods

1. Subsampling

Subsampling is perhaps the simplest way to compress an image. One approach to subsampling is simply to delete some of the pixels. The encoder may, for example, ignore every other row and every other column of the image, and write the remaining pixels (which constitute 25% of the image) on the compressed stream. The decoder inputs the compressed data and uses each pixel to generate four identical pixels of the reconstructed image. This, of course, involves the loss of much image detail and is rarely acceptable. Notice that the compression ratio is known in advance.

A slight improvement is obtained when the encoder calculates the average of each block of four pixels and writes this average on the compressed stream. No pixel is totally deleted, but the method is still primitive, because a good lossy image compression method should lose only data to which the eye is not sensitive.

Better results (but worse compression) are obtained when the color representation of the image is changed from the original (normally RGB) to luminance and chrominance. The encoder subsamples the two chrominance components of a pixel but not its luminance component. Assuming that each component uses the same number of bits, the two chrominance components use 2⁄3 of the image size. Subsampling them reduces this to 25% of 2⁄3, or 1⁄6. The size of the compressed image is therefore 1⁄3 (for the uncompressed luminance component), plus 1⁄6 (for the two chrominance components) or 1⁄2 of the original size.

2. Quantization

Scalar quantization has already been mentioned. This is an intuitive, lossy method where the information lost is not necessarily the least important. Vector quantization can obtain better results, and an intuitive version of it is described here.

The image is partitioned into equal-size blocks (called *vectors*) of pixels, and the encoder has a list (called a *codebook*) of blocks of the same size. Each image block *B* is compared to all the

blocks of the codebook and is matched with the "closest" one. If *B* is matched with codebook block *C*, then the encoder writes a pointer to *C* on the compressed stream. If the pointer is smaller than the block size, compression is achieved.

Image Transforms

The mathematical concept of a transform is a powerful tool that is employed in many areas and can also serve as an approach to image compression. An image can be compressed by transforming its pixels (which are correlated) to a representation where they are decorrelated. Compression is achieved if the new values are smaller, on average, than the original ones. Lossy compression can be achieved by quantizing the transformed values. The decoder inputs the transformed values from the compressed stream and reconstructs the (precise or approximate) original data by applying the inverse transform.

The term decorrelated means that the transformed values are independent of one another. As a result, they can be encoded independently, which makes it simpler to construct a statistical model. An image can be compressed if its representation has redundancy. The redundancy in images stems from pixel correlation. If we transform the image to a representation where the pixels are decorrelated, we have eliminated the redundancy and the image has been fully compressed.

Image transforms are designed to have two properties: (1) reduce image redundancy by reducing the sizes of most pixels and (2) identify the less important parts of the image by isolating the various frequencies of the image. Thus, this section starts with a short discussion of frequencies. We intuitively associate a frequency with a wave. Water waves, sound waves, and electromagnetic waves have frequencies, but pixels in an image can also feature frequencies.

Image frequencies are important because of the following basic fact: Low frequencies correspond to the important image features, whereas high frequencies correspond to the details of the image, which are less important. Thus, when a transform isolates the various image frequencies, pixels that correspond to high frequencies can be quantized heavily, while pixels that correspond to low frequencies should be quantized lightly or not at all. This is how a transform can compress an image very effectively by losing information, but only information associated with unimportant image details.