Lossless Image Compression using MATLAB

Authors:

- Prakash Nadgeri
- Abhishek Nair
- Parth Nakar
- Nikhil Jain

Introduction

Images are important documents nowadays; images include various information e.g., human bodies in medical images which are used for different purpose such as medical security and other plans. Compression of images is used in some applications such as profiling data and transmission system. To work with them in a few applications they need to be compressed more or less depending on the purpose of the application. Regard to importance of images information lossless or loss compression is preferred. Lossless compressions are JPEG, JPEG-LS and JPEG2000 are some well-known methods for lossless compression. Image compression plays a key role in many important applications, including image database, image communications, remote sensing. The image to be compressed are gray scale with pixel values between 0 to 255. Because of their large capacity, storing and transmitting are not easy and they need large storage devices and high bandwidth network systems. In order to alleviate these requirements, compression techniques and standards such as JPEG, JPEG2000, MPEG-2 and MPEG-4 have been used and proposed. The JPEG standard constructed from several functions such as DCT, quantization, and entropy coding. Huffman and arithmetic coding are the two most important entropy coding in image compression standards.

1.Need for Compression:

Compression is a technique to reduce the quantity of data without excessively reducing the quality of the multimedia data. The transition and storing of compressed multimedia data are much faster and more efficient than original uncompressed multimedia data.

2.Principle behind Compression:

a. Redundancies reduction aims at removing duplication from the signal source (image/video).

b. Irrelevancy reduction omits parts of the signal that will not be noticed by the signal receiver, namely the Human Visual System.

3.Types of Compression:

(i) Lossless Compression (ii) Lossy Compression

Lossless Compression: Data is compressed and can be reconstituted (uncompressed) without loss of detail or information. These are referred to as bit-preserving or reversible compression systems also Lossless compression frequently involves some form of entropy encoding and are based in information theoretic techniques.

Lossy Compression: The aim is to obtain the best possible fidelity for a given bit-rate or minimizing the bit-rate to achieve a given fidelity measure. Video and audio compression techniques are most suited to this form of compression. If an image is compressed it clearly needs to uncompressed (decoded) before it can viewed/listened to. Some processing of data may be possible in encoded form however. Lossy compression use source encoding techniques that may involve transform encoding, differential encoding or vector quantization.

A. Lossless Coding Techniques –

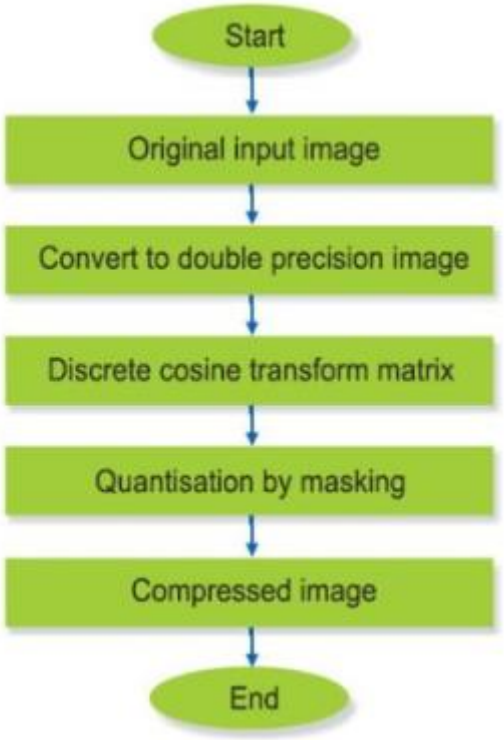
- Run length encoding
- Huffman encoding
- Arithmetic encoding
- Entropy coding
- Area coding

B. Lossy Coding Techniques –

- Predictive coding
- Transform coding (FT/DCT/Wavelets)

Code & Concepts

We have implemented discrete cosine transform algorithm, which compresses the image with a good compression ratio. All equations are implemented in MATLAB in the form of functions.



Flowchart showing compression process

The image is read through MATLAB to capture its pixels. After obtaining the compressed image, peak-signal-noise ratio (PSNR) and mean-square error (MSE) are calculated using the following relationships:

$$MSE = \frac{\sum_{M,N}(Image1(m,n) - Image2(m,n))^2}{mxn}$$

where m and n are the number of rows and columns. Image1 and Image2 are the original and compressed images, respectively.

After compression, there should not be much change in the quality of the image. MSE indicates an error between the original image and compressed image. It should be as small as possible.

Where R is the maximum fluctuation in the input image data type (maximum possible pixel value of image). PSNR is related to MSE and it gives the amount of noise in a compressed image. PSNR should be as high as possible.

$$PSNR = 10 \log_{10} \left(\frac{R^2}{MSE} \right)$$

❖ Why is image converted to a double precision image?

MATLAB has two ways of representing RGB and grayscale images.

- Images can be represented as double precision numbers (or single precision) in the range 0 to 1.
- Images can also be represented as integer data classes, especially uint8 (unsigned 8-bit integers), in which case the values are the integers 0, 1, 2, ... up to the maximum representatable in the integer class, such as 255. The conversion from integer class to double precision is to switch the numbers to floating point and then divide by the maximum allowed for the class, such as double(ImageData)/255 so 0 still corresponds to 0.0 and 255 corresponds to 1.0

Most of the image file formats store only integers, or "prefer" to store integers.

Sometimes operations on images are easier when the images are represented in floating point. For example, if you want to subtract one image from another, if the images are in unsigned integer format, everywhere the second image was greater than the first you would get 0 as the result because unsigned integers cannot represent negative numbers.

There are also some parts of the graphics system, such as AlphaData, that *must* be in floating point format.

It is therefore not uncommon to want to switch from integer representation to floating point representation. im2double() does that. im2double() knows about all the various integer storage classes and what the proper conversion formula is -- it is not necessarily a "difficult" conversion, but you can count on im2double() to get it right. This is especially valueable when you are allowing the user to select the images to be read in: if you want to be able to support a variety of image file formats that might use different ranges of values, it is better to call a routine that is sure to get the conversion right than to do the conversion yourself and risk an incompatibility.

❖ What is discrete cosine transform matrix?

The discrete cosine transform (DCT) represents an image as a sum of sinusoids of varying magnitudes and frequencies. The dct2 function computes the two-dimensional discrete cosine transform (DCT) of an image. The DCT has the property that, for a typical image, most of the visually significant information about the image is concentrated in just a few coefficients of the DCT. For this reason, the DCT is often used in image compression applications.

The input image is divided into 8-by-8 or 16-by-16 blocks, and the two-dimensional DCT is computed for each block. The DCT coefficients are then quantized, coded, and transmitted. For typical images, many of the DCT coefficients have values close to zero. These coefficients can be discarded without seriously affecting the quality of the reconstructed image.

❖ What is quantization?

Quantization, involved in image processing, is a lossy compression technique achieved by compressing a range of values to a single quantum value. When the number of discrete symbols in a given stream is reduced, the stream becomes more compressible. For example, reducing the number of colors required to represent a digital image makes it possible to reduce its file size.

❖ What is masking?

Masking involves setting some of the pixel values in an image to zero, or some other "background" value.

Masking can be done in one of two ways:

1. Using an image as a mask. A mask image is simply an image where some of the pixel intensity values are zero, and others are non-zero.

Wherever the pixel intensity value is zero in the mask image, then the pixel intensity of the resulting masked image will be set to the background value (normally zero).

2. Using a set of ROIs as the mask. The ROIs for each slice are used to define the mask.

❖ Why have we defined three masking matrices?

Matlab reads image in RGB format. Using array indexing we separated the R, G, B channels of image and then we converted them to double precision images individually.

❖ What is the significance of blkproc()?

First, we separated the R, G, B channels and defined discrete cosine transform matrix. blkproc() processes the image I by applying the function to each distinct m-by-n block of I. It is used for implementing distinct block processing for an image and it returns a matrix or vector or scalar.

Conclusion and Future Scope

As for future, more focus can be given on improvement of compression ratio using newer techniques. The proposed technique can be experimented on different kinds of data sets like audio, video or text, but as for the scope of this project, it is restricted to images. New methods can be combined and proposed that decreases the time complexity incurred in creating dictionary. Further the work can be extended to video compression. Video data is basically multiple frames of three-dimensional array of colour pixels, that contains spatial and temporal redundancy. Similarities can thus be encoded by registering differences within a frame where data frame is a set of all pixels that correspond to a single time moment. With the advancements in compression technology, it is now very easy and efficient to compress video files.