Lab 4
Lab report due by 1:00 PM on Monday, April 17, 2017

This is a MATLAB only lab, and therefore each student needs to turn in her/his own lab report and own programs.

---

1 Introduction

In this lab you will implement the JPEG algorithm for image compression. Moreover, we will review modern sparse representation models in image processing.

1.1 Why compressing images

Images are increasingly being acquired in a digital form. Analog images, such a camera pictures, or a chest X-Rays are more and more often digitized. Images exist not only in their two-dimensional (2-D) form, but also as three-dimensional (3-D) volumes (e.g. medical images), or spatio-temporal sequences (movies). The availability of images in a digital form offers a very large number of advantages:

- images can be stored, and transmitted very efficiently
- images can be processed. Examples of post processing include: enhancement, denoising, zooming, etc.
- Computer assisted diagnosis can be performed on digital images. These higher level processing can involve feature extraction, decision making, classification, etc.
- Images can be stored in large databases, where they can be indexed, and very efficiently searched for.

However the very large size of images, and movies are basic hurdles to any attempt at storing, transmitting, or searching for images. Table 1 show samples of average sizes of image files or movie files. For each application, we calculated the compression ratio required for a realistic usage under normal conditions. It is clear from this table that the ability to decrease the amount of bits necessary to represent an image, or a sequence of images, will have a dramatic impact in a wide variety of domains, including but not limited to: multimedia technology, medical imaging, geophysical exploration, satellite imagery, etc.

Applications	horiz x vert	frame/s	size	ratio
Digital camera	768 x 512		375kB	10
Fax	1728 x 1100		240 kB	20
CD	352 x 240	30	7.6 Mb/s	50
HDTV	1920 x 1080	30	186 Mb/s	80
VideoPhone	176 x 144	15	1.1 Mb/s	300

Table 1: Typical size of images or sequence of images, and the compression ratio required for a realistic usage under normal conditions.

2 Structure of a compression system

A compression system can be decomposed into three components: (1) an encoder, (2) a decoder, and (3) and a channel. The channel is that part of the communication system that is out of the designer's control. Examples of channels are: wireless (radio), satellite, ethernet, CD, modem, optic fiber, ISDN, computer memory, etc.

In this lab we will assume that the channel provides a reliable communication of the compressed data, and we will ignore the channel coding errors. It is possible to combat channel errors, by using error protection codes, but we will not explore these here.

2.1 Encoder

The encoder can be decomposed in a series of elementary stages:

1. Pre-processing. A number of operations can be performed on the image prior the effective compression. The goal of pre-processing is to prepare the image in order to minimize the coding complexity, and increase the compression efficiency. Examples of pre-possessing operations are:
   - filtering,
   - padding with zeros,
   - symmetric extension on the boundaries,
   - tiling (cutting in smaller blocks) to enable the coding of images of arbitrarily large size,
2. Transform. The goal of the transform is to reduce the correlation among image pixels, and describe with a smaller set of significant coefficients, most of the image content.
3. Quantization. The output of the transform are real-valued coefficients. In order to decrease the amount of information needed to describe these coefficients, one represents the coefficients with a much smaller set of values. Quantization is a non reversible process that permits to balance the reconstructed image quality with the amount of bits available to code the image.
4. Entropy coding. After quantization, the image is approximated with a small set of codewords. Entropy coding permits to describe in the most efficient way the sequences of symbols from this alphabet with strings of "0", and "1".

2.2 Decoder

The decoder reconstructs the image by applying in a reverse order the inverse of each of the processes 4), 3), and 2) described in the encoder. While entropy coding is a reversible process, quantization is not invertible. A final post-processing is often performed on the reconstructed image in order to conceal coding artifacts, and improve the visual appearance.

2.3 Important parameters of a compression system

Some of the fundamental parameters of a coding system are:

- Compression efficiency: it measures the compression ratio (often quoted in bits per pixel: bpp),
- Fidelity: this is the distortion due to the coding. While everybody agrees that the PSNR (Peak Signal to Noise Ratio) is a global parameter, often unrelated to the visual quality, there is currently no definition of a quantitative measure of visual distortion. For this reason visual inspection of reconstructed images remains one criterion for the evaluation of a compression method.
- Complexity: this is a key parameter for a number of applications, where real-time compression is required.
- Memory. Some applications will require an encoder that can work with a limited memory.
- Coding delay: real-time applications such as video-phone cannot tolerate a coding delay that is longer than a few frames.
- Robustness. This is a key feature in areas such as wireless communication where channel coding errors can have a catastrophic effect on the reconstructed image.

3 Transform coding

Transform coding consists in applying a linear transformation from the image space to a transformed space. The image in the transformed space is represented by a new vector (of same dimension as the original image) of coefficients. The principle of transform coding is that the transformed coefficients should be less correlated than the original samples. Another way to express the same idea is that a small number of transformed coefficients should carry most of the energy of the image, and the rest of the coefficients can be quantized to zero. Most of the transform operate in the frequency domain. This can be justified by a theoretical argument : if the images are realizations of a wide sense stationary process, then the Fourier transform is the Karhunen-Loève transform that diagonalizes the correlation matrix. As we will see in this section, the KLT is the transform that permits to achieve the smallest distortion of the quantized coefficients for a given rate.

From a practical point of view, because the human visual system is sensitive to artifacts in the frequency domain, it is easier to work directly in this domain in order to control the distortion, and even incorporate cues about the human visual system sensitivity.

In the remainder of this section we demonstrate that an advantage can be gained by applying a linear transform before quantization. Precisely, we will prove this result when the linear transform is the Karhunen-Loève transform (KLT), which is the transform that *decorrelates* the input data.

While this transform depends on the knowledge of the correlation matrix of the input, and is therefore rarely used in practice, this result provides a theoretical justification for the performance of transform coding. For certain inputs the Discrete Cosine Transform provides an approximation to the data-dependent KLT. We will present some application of the DCT to transform coding.

3.1 Discrete Cosine based coders

The importance of the DCT comes from several different reasons. First, the DCT provides a good approximation to the KLT if the image has a Markov behavior. In practice, the DCT provides excellent energy compaction for highly correlated data. From a practical point of view, the DCT can be computed using an FFT, but unlike the FFT, the DCT coefficients are real. In 1988, a DCT based algorithm won in blind subjective assessments, for the definition of a new image compression standard for continuous-tone images (Joint Photographic Experts Group - JPEG).

4 Reading, writing, and displaying images

The following MATLAB commands can be used to read, write and display an image.

```
>> f = imread ('clown.pgm');
>> figure,imagesc(f, [0 255]),colormap gray,axis square, axis off;
>> g = f;
>> imwrite (g,'clown2.pgm', 'pgm');
>>
```

4.1 Test images

In this lab, we will be using standard test images available on D2L.

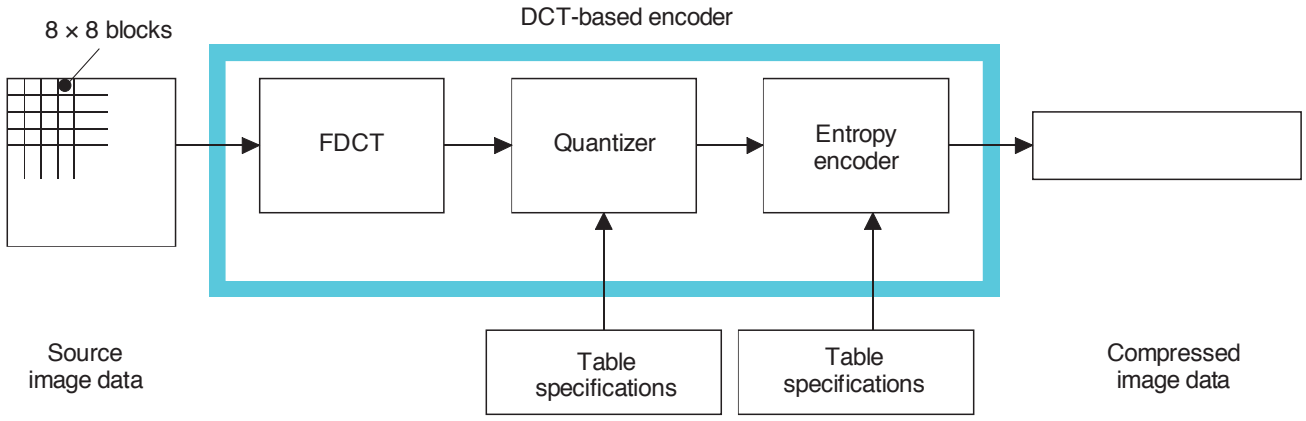

Figure 1: The different stages in the JPEG compression algorithm

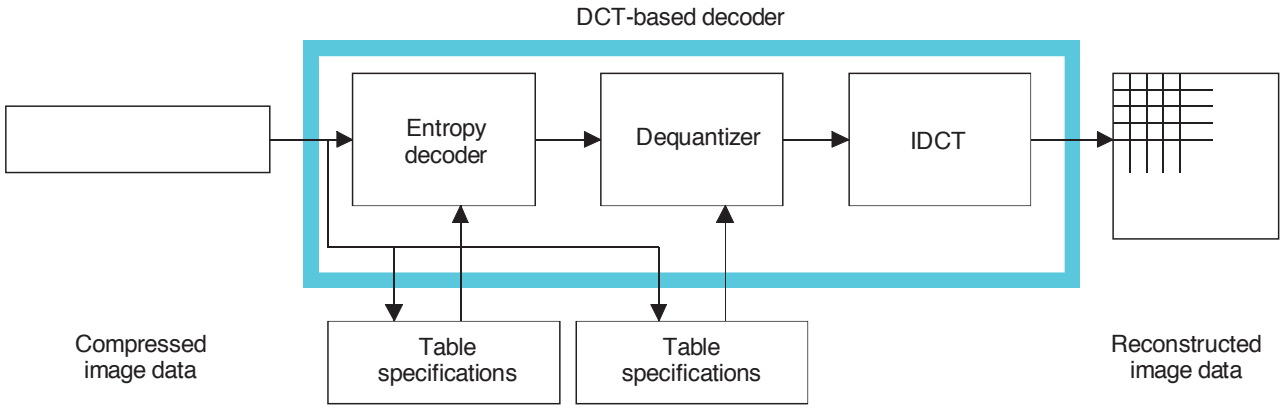

Figure 2: The different stages in the JPEG decoding algorithm

5 The JPEG compression algorithm

Figure 1 shows the three main steps of the JPEG compression algorithm. We will implement the three stages: forward and inverse DCT, forward and inverse quantizer, and entropy coder and decoder. Figure 2 shows the three main steps of the JPEG decoding algorithm. Each step implements the inverse of the corresponding operation performed by the encoder.

5.1 The forward and inverse discrete cosine transform.

The original image is divided into non overlapping blocks of size 8×8. The blocks are scanned horizontally, and then vertically. Each block is transformed using a two-dimensional discrete cosine transform,

$$F_{u,v} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}, \quad u, v = 0, \ldots 7 \tag{1}$$

where

$$C_u = \begin{cases} 1 & \text{if } u \neq 0, \\ \frac{1}{\sqrt{2}} & \text{if } u = 0. \end{cases} \tag{2}$$

The coefficients $F_{u,v}$ are coarsely approximated over a small set of possible values (quantization), and replaced by $\widetilde{F}_{u,v}$. The *quantization* process cannot be inverted and contributes to the loss in image quality.

Given the coefficients $\widetilde{F}_{u,v}$, one can use the inverse Discrete cosine transform to reconstruct an estimate of the block using

$$\widetilde{f}(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u \, C_v \, \widetilde{F}_{u,v} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}, \quad x, y = 0, \ldots 7 \tag{3}$$

where

$$C_u = \begin{cases} 1 & \text{if } u \neq 0, \\ \frac{1}{\sqrt{2}} & \text{if } u = 0. \end{cases} \tag{4}$$

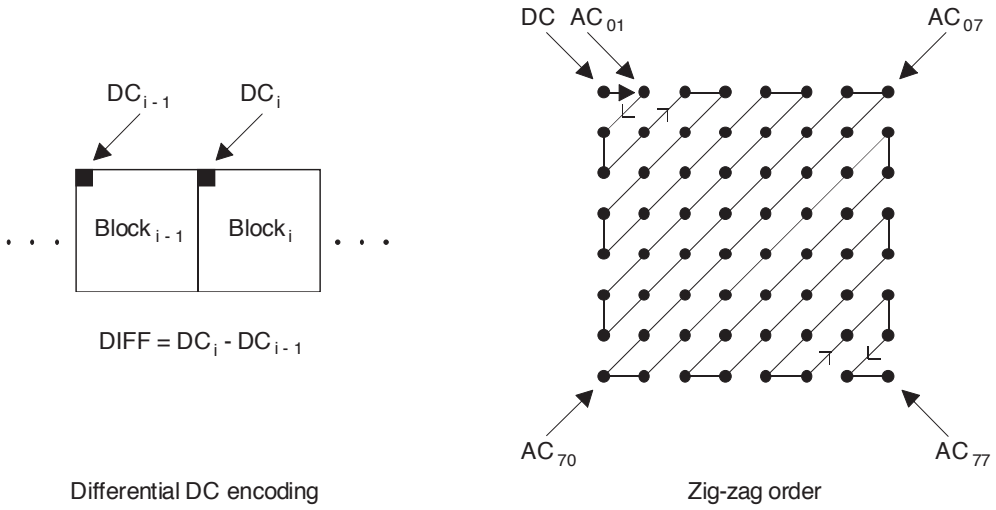

Figure 3: Zig-zag order for mapping the two-dimensional frequencies to a one dimensional array.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figure 4: The quantization table Q

Assignment

1. Write the MATLAB function `dctmgr` that implements the following functionality:

   (a) it takes an luminance (gray-level) image as an input, divides it into non overlapping 8×8 blocks, and DCT transform each block according to (1).

   (b) The DCT coefficients for the entire image are returned into a matrix `coeff` of size $64 \times N_{\text{blocks}}$, where N_{blocks} is the number of 8×8 blocks inside the image.

   (c) For a given block `b`, the coefficients of in the column `coeff(:,b)` are organized according to the zig-zag order shown in Fig. 3. That is,

   `coeff(2,b) = F(1,2), coeff(3,b) = F(2,1), coeff(4,b) = F(3,1),...`

   (d) The zero-frequency coefficients, $F(1,1)$, for each block is encoded using differential encoding,

   `coeff(1,1)` $= F_1(1,1)$,
   `coeff(1,2)` $= F_2(1,1) - F_1(1,1)$,
   `coeff(1,3)` $= F_3(1,1) - F_2(1,1)$,
   ...
   where $F_i(1,1)$ is the zero frequency DCT coefficient of block i.

2. Write the MATLAB function `idctmgr` that implements the following functionality:

   (a) It takes a matrix `coeff` of size $64 \times N_{\text{blocks}}$, where N_{blocks} is the number of 8×8 blocks inside the image and reconstruct a luminance image.

   (b) For each given block `b`, the coefficients in the column `coeff(:,b)` are used to reconstruct the block according to (3).

5.2 Quantization and inverse quantization

The effect of the DCT is to create many small coefficients that are close to zero, and a small number of large coefficients. We can think of the DCT as a rotation of the space $\mathbb{R}^{64}$ that aligns the coordinate system along the direction associated with the largest variance in the distribution of the coefficients. In order to benefit from this redistribution of the energy, we need to simplify the representation of the floating numbers that are used to describe the DCT coefficients. This task is achieved by the quantization step.

The goal of quantization is to represent a continuum of values with a finite (and preferably) small set of symbols. In theory, a scalar quantizer can be defined as a map from the set of real numbers to the set of integers, or any subset thereof. In practice, real numbers are represented in a computer as floating point numbers and have only a fixed number of digits, and thus a fixed precision. In this context a scalar quantizer is a map from a large set of discrete values, to a much smaller set of codewords. Quantization permits to control the performance of the compression by performing a reduction of the set of codewords that are used to represent the initial samples. In principle quantization is the only mechanism in a compression system where irrecoverable loss is introduced.

In the JPEG compression standard, the quantization of the DCT coefficients is performed as follows

$$F^q_{u,v} = \text{round}\left(\frac{F_{u,v}}{\texttt{loss-factor} \times Q_{u,v}}\right). \tag{5}$$

$Q_{u,v}$ is the entry (u, v) of the standard quantization table shown in Fig. 4, and the scaling factor `loss-factor` is a user supplied parameter that measures the quality of the compressed image. For instance, the choice of `loss-factor` $= 2$ should yield an image that is visually identical to the original image, while `loss-factor = 10` produces an image of lower quality.

The inverse quantization of the DCT coefficients is defined by the map

$$\widetilde{F}_{u,v} = F^q_{u,v} \times \texttt{loss-factor} \times Q_{u,v}. \tag{6}$$

We note that the compression becomes lossy during the quantization step performed in (5) that creates many zero coefficients as `loss-factor` becomes larger. The quantization step $Q_{u,v}$ varies as a function of the frequency. As a result, it becomes possible to spend fewer bits for the high frequency coefficients than for the low frequency coefficients. Several quantization tables have been proposed. In this lab we use the table Q defined in Fig. 4.

Assignment

3. Implement the quantizer and inverse quantizer, as defined by (5) and (6), and integrate it with the DCT and IDCT transforms.

4. For the six test images available on D2L, display reconstructed images for `loss-factor = 1`, `loss-factor = 10`, and `loss-factor = 20`. Display also the reconstruction error.

5. For the six test images, compute the Peak Signal to Noise Ratio (PSNR) between the original image, and the reconstructed image,

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\frac{1}{N^2}\sum_{i,j=0}^{N-1}|f(i,j) - \widetilde{f}(i,j)|^2}\right), \tag{7}$$

   for the values of `loss-factor` in $\{1, \ldots, 50\}$. You will plot the values of PSNR as a function of `loss-factor`.

6 Sparse representation of images

Please refer to lecture notes for a comprehensive explanation of the K-SVD algorithm. For the next assignment, download the K-SVD software from http://www.cs.technion.ac.il/~ronrubin/software.html (`KSVD-Box v13` and `OMP-Box v10`)

Assignment

6. Choose the first 100 examples of each digit from the USPS data set. Run the K-SVD algorithm on the selected examples from all ten digits with parameters $d = 30$ and $T = 10$. Please visualize these 30 dictionary atoms and explain.