EE141L: Lab 5
Image Compression

October 10, 2018

1 Image compression

Data compression is an area of Electrical Engineering that studies algorithms for reducing the number of bits required to represent information. This technology is present in most digital devices, thus enabling efficient storage and transmission of information. For example video at 30 frames per second (fps) with resolution 1920×1080, amounts to $\sim 1.5 Gbit/s$. Some applications of compression include

- Video streaming (youtube, netflix).
- Video chat (facetime, duo, hangouts, skype).
- Any device with a camera has built in image/video compression technology (phones, laptops,etc).
- 3D video (point clouds) for virtual/augmented reality (real time communication, entertainment streaming, gaming).

Data compression is possible because real/natural data is very redundant. In this lab you will learn that one of the basic principle behind image compression algorithms involves matrix multiplication.

1.1 Encoder and decoder

Images, video and other types of multimedia content are compressed using a method that can be described as follows. Consider $\mathbf{X} \in \mathbb{R}^{m \times n}$ corresponding to a gray scale image, and $\mathbf{x} \in \mathbb{R}^{mn}$ the vectorized image obtained by stacking its columns (in Matlab this can be done with $\mathbf{x} = \mathbf{X}(:)$).

The algorithm for image compression has two components, first the **encoder**

1. Apply linear transformation $\mathbf{y} = \mathbf{T}\mathbf{x}$
2. Replace small entries of $\mathbf{y}$ with zeros, call it $\hat{\mathbf{y}}$.
3. Store/transmit $\hat{\mathbf{y}}$

The **decoder**

1. Receive $\hat{\mathbf{y}}$
2. Invert linear transformation $\hat{\mathbf{x}} = \mathbf{T}^{-1}\hat{\mathbf{y}}$.

The most important component is the matrix $\mathbf{T}$. Modern image/video compression systems use the Discrete Cosine Transform (DCT)[1], which is close to optimal for most images.

2 Compression of small images

In practice, images are partitioned into small blocks for faster processing. In this section you will compress a small image block

Problem 1. First load your picture using X = im2double(rgb2gray(imread('picture.tiff')));

1. Extract any sub-block of size 32×32, call it Xblock.
2. Vectorize it using xblock = Xblock(:)
3. Construct DCT matrix $\mathbf{T}$ of size $32^2 \times 32^2$ by calling function DCT_for_vectorized_images. Verify that $\mathbf{T}$ is invertible, i.e. check that $\mathbf{T}\mathbf{T}^{-1} = \mathbf{T}^{-1}\mathbf{T} = \mathbf{I}$ (Hint: use function inv to compute inverse, and function matrix_error)
4. Apply DCT transform to xblock, and plot the results (with function plot). How are the magnitudes of the entries of xblock?. Now plot the first 100 elements of the vector, how is the first entry compared to the rest?. Include your plots in the report.

Problem 2. You will implement an algorithm that takes a vector, and only keeps its largest entries. Modify the provided matlab code vector_thresholding, that given a n dimensional vector, it returns a vector that keeps its K largest entries, and zeros-out the remaining. You need to complete lines 11, 17, 19.

Problem 3. You will use results from Problems 1 and 2.

1. Apply the code from Problem 2 to the vector yblock = T xblock, and keep its 20% largest entries. Report the value of K, and plot the resulting thresholded vector.
2. Apply $\mathbf{T}^{-1}$ to yblock, use reshape function to turn the vector back into a square matrix, and display the resulting image.
3. Repeat parts 1 and 2 for $5\%, 10\%, 30\%, 50\%$. Report images and relative errors between original and reconstructed images from part 2.

[1] Ahmed, Nasir, T. Natarajan, and Kamisetty R. Rao. "Discrete cosine transform." IEEE transactions on Computers (1974)

3 Compression of whole image

In this section you will compress the whole image at once. Since the image is so large, it has $512 \times 512 = 262144$ pixels, the transform matrix has size 262144×262144, which might not fit in memory (it's about 260 Mbytes!!!). Fortunately the DCT can be applied to an image very efficiently, with minimal CPU and memory usage with the function dct2.

Problem 4. Whole image compression

1. Apply dct to whole image using Y = dct2(X). Input and output X,Y are matrices of the same size.
2. Vectorize y =Y(:) and repeat part 1 from problem 3.
3. Reshape the resulting vector to be of size 512×512 and apply inverse transform using function idct2. Display result and report relative error.
4. Repeat step 3 for $0.1\%, 1\%, 5\%, 10\%$, report images and relative error.

A Matrix Error and Relative Error

Given two matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{m \times n}$, the matrix error function computes

$$\text{error}(\mathbf{X}, \mathbf{Y}) = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (\mathbf{X}_{ij} - \mathbf{Y}_{ij})^2}. \tag{1}$$

The relative error with respecto to $\mathbf{Y}$ is

$$\text{relative\_error}(\mathbf{X}, \mathbf{Y}) = \frac{\text{error}(\mathbf{X}, \mathbf{Y})}{\sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{Y}_{ij}^2}}. \tag{2}$$

Notice that they have some properties

1. $\text{error}(\mathbf{X}, \mathbf{Y}) = \text{error}(\mathbf{Y}, \mathbf{X})$, and $\text{relative\_error}(\mathbf{X}, \mathbf{Y}) \neq \text{relative\_error}(\mathbf{Y}, \mathbf{X})$
2. $\mathbf{X} = \mathbf{Y} \Leftrightarrow \text{error}(\mathbf{X}, \mathbf{Y}) = 0 \Leftrightarrow \text{relative\_error}(\mathbf{X}, \mathbf{Y}) = 0$
3. $0 \leq \text{relative\_error}(\mathbf{X}, \mathbf{Y}) \leq 1$