Lecture 2

Gidon Rosalki

2025-10-26

1 Properties of histogram equalisation

Monotonic transformation: Does not reverse intensity order. If we apply an equalisation twice, then nothing happens. The new intensity is approximately equal to the cumulative probability. Sometimes we need to be adaptive in histogram equalisation, for example if we have different intensity distributions in an image like sunny areas, and shadowed areas, but how do we segment the image? We compute the histogram of local regions around the pixel. So, for each pixel, we compute the equalisation LUT in the local region, and transform **only the centre pixel** by the LUT. We then do this for each pixel. This is easily optimisable from n^2 by subtracting and adding **only the pixels that change** between each pixel.
We do need to consider what size window we should use around each pixel, consider:

Figure 1: Adaptive histogram equalisation

As we can see, different sized windows have different effects on the final picture.

2 Fourier transformation

Let us consider how to save data from a $10,000$ pixel image. with 1 pixel, well we do not see much. With 1000, we see a tenth, and 5000 we can see half. However, if we instead blur the image a bit, well 1 still does not show much, but 1,000 gives us a blurry image, and 5,000 gives an image that is not trivially distinguishable from the original.

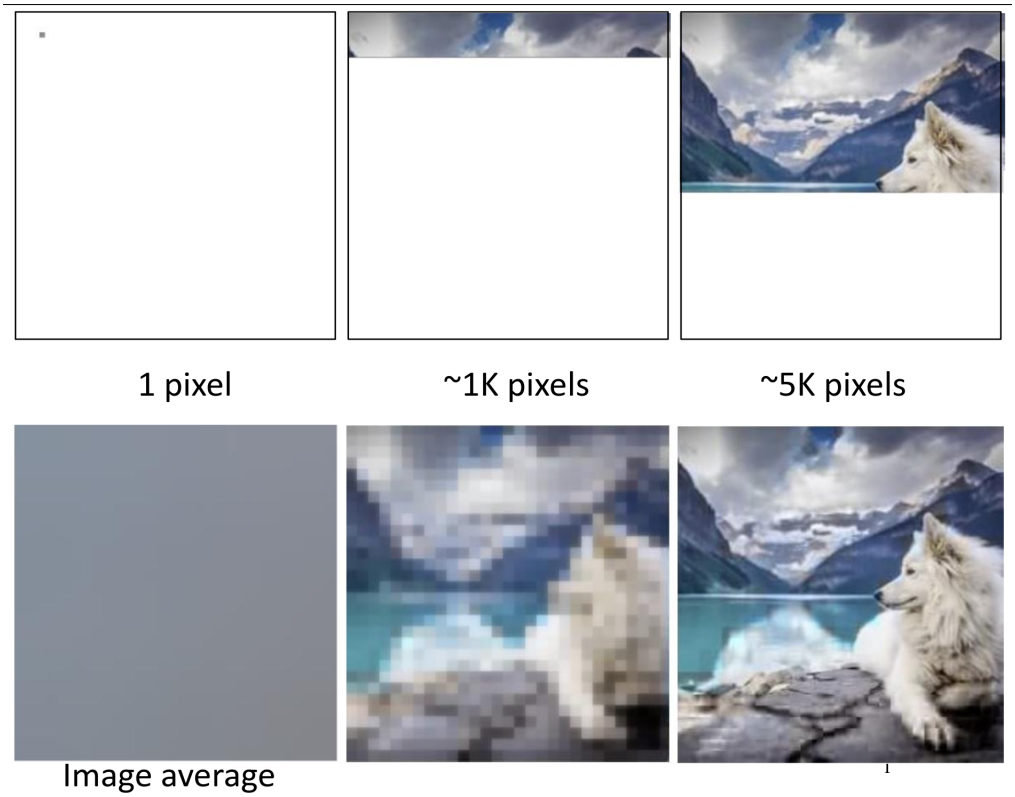

Figure 2: Blurring images

2.1 Basis of a vector space

Every vector in a vector space is a linear combination of basis vectors:

$$\begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} = 2\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + 1\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + 1\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + 0\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

A standard (natural) basis is a local representation, of 1 pixel. In a k dimensional vector space, every set of k *independent* vectors form a basis. An **orthogonal** basis is when every two basis vectors are orthogonal, and an **orthonormal** basis is an orthogonal basis, with all the absolute values of the basis vectors being 1.

2.2 Fourier transformation

The Fourier transformation is comprised of **easy** equations, that are **hard** to understand. It is a representation of signals, and images. The standard representation, is a collection of samples, or perhaps pixels, where one value gives the grey level at one pixel (**local** representation). The *Fourier* representation is the weighted sum of sin waves (of frequency ω), where each wave ω is assigned an *amplitude* and *phase* (the phase is the starting point of the wave). **Any** periodic function can be rewritten as a weighted sum of sines, and cosines of different frequencies. This is called a *Fourier Series*. Pictures do not appear to be periodic, but if we tile them, then they may be considered such.

2.3 Mathematical revision

A complex number is comprised of 2 real numbers, where one is multiplied by i such that $i^2 = 1$:

$$a + bi : a, b \in \mathbb{R} \wedge i^2 = 1$$

This may also be represented as

$$a + bi = R \cdot e^{i\alpha} \wedge e^{i\alpha} = \cos(\alpha) + i\sin(\alpha)$$

From this the absolute value is

$$R = \sqrt{(a^2 + b^2)}$$

and the phase is

$$\alpha = \tan^{-1}\left(\frac{b}{a}\right)$$

To multiply together $a + bi$ can be a bit complex (har har), but should we use the other representation:

$$R_1 e^{i\alpha_1} \cdot R_2 e^{i\alpha_2} = R_1 \cdot R_2 \cdot e^{i(\alpha_1+\alpha_2)}$$

We may gain certain physical numbers from this:

$$\sin(2\pi\omega x) \implies \lambda = \frac{1}{\omega}$$
$$\sin(2\pi\omega x) \implies f = \frac{1}{\omega}$$

(Remember, frequency is 1 over wavelength).

2.4 1D Discrete Fourier Transform

The 1D Fourier transformation is defined to be

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\, e^{\frac{-2\pi iux}{N}}$$

So, for example

$$F(0) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\, e^0 = f$$

The 1D inverse fourier transform

$$f(x) = \frac{1}{1}\sum_{u=0}^{N-1} F(u)\, e^{\frac{-2\pi iux}{N}}$$

Where f is a weighted sum of sines and cosines. The complexity of the base algorithm is $O\left(N^2\right)$, since we have to do N calculations, for N items. However, there exists the FFT, which may be done in $O(N\log(N))$. This was done through clever recursion, where we divide and conquer the input.

2.4.1 Fourier basis vectors

To compute f from F, we carry out

$$f(x) = \sum_{u=0}^{N-1} F(u)\, e^{\frac{-2\pi iux}{N}}$$

So this complex part with e is the Fourier basis, such that

$$e^{\frac{-2\pi iux}{N}} = \cos\left(\frac{2\pi ux}{N}\right) + i\sin\left(\frac{2\pi ux}{N}\right)$$

So for each frequency $0 \leq u \leq N-1$, we can calculate the above basis vector.

All of this gives us some very nice properties, particularly in periodicity and symmetry:

$$F(u) = F(u+N)$$
$$F(u) = F^*(-u) = F^*(N-u) : (a+bi)^* = (a-bi)$$
$$|F(u)| = |F(-u)|$$

Let $N = 256$, so

$$F(6) = F(262)$$
$$F(6) = F^*(-6) = F^*(250)$$
$$|F(6)| = |F(-6)| = |F(250)|$$

All the above properties are immediate from the Fourier Transform. Fourier of N real numbers gives N complex Fourier Coefficients. This is $2N$ real numbers. Given all of $f(0)$ to $f(N-1)$, we only need $\frac{N}{2}$ coefficients, from $F(0)$ to $F\left(\frac{N}{2}-1\right)$, thanks to the symmetry $F(N-u) = F^*(u)$.

2.5 Sound

Most video has both images, and sound. 1D is easier to understand, and since sound vibrations generated by waves, so Fourier analysis is a standard tool. The hearing range of humans is roughly 50Hz - 20,000Hz. To digitise sound, transducers convert air pressure into voltage, and digitiser convert the voltage to a sequence of numbers.

2.5.1 Sampling rate

The sampling rate is how frequently we sample the waveform. The image equivalent is the pixels per line, or bits per pixel. The sound quality depends on the sampling rate (samples per second). The sampling rate should be at least double the maximum frequency. However, this is a lot of data. Telephones have a sampling rate of 8kHz, with each sample being 8 bits. AM radio uses 22.05 kHz, and audio CDs 44.1kHz, with each sample being 16 bits. Sound is often stored in the "wav" file format, with PCM (lossless). The data can be represented as integers, or floats (0 to 1). There are some Python libraries for handling this, such as [columns=fixed]scipy.io.wavfile for io, matplotlib for visualisation, and librosa for io, processing, and visualisation.

2.6 Short Time Fourier Transform

The Short Time Fourier Transform (STFT) computes the FT at narrow time intervals (i.e., narrow enough to be considered stationary). This is called a *window*. Each FT provides the spectral information of a separate time slice of the signal, providing simultaneous time (window location) and frequency (FT in window) information. We do this by:

1. Choosing a window $w(n)$ of finite length
2. Place the window on top of the signal at $t = 0$
3. Truncate the signal by multiplying with this window
4. Compute the FT of the truncated signal, saving the results
5. Incrementally slide the window to the right
6. Go back to step 3, until the window reaches the end of the signal

STFT: For every n

$$F(n,u) = \sum_{m=0}^{N-1} f(m+nH)\, w(m)\, e^{-\frac{2\pi i u m}{N}}$$

Where n is time, N is the number of frequencies, and H is the hope size. The inverse ISTFT is as follows:

$$f(n) = K \sum_{u=0}^{N-1} F(n,u)\, e^{\frac{2\pi i u n}{N}}$$

For a carefully selected window w, hop H, and normalisation K.

The window shape is symmetrical, and unimodal (ie Gaussian, triangular). Its length W, is the number of non zero coefficients in w. L is the number of samples between successive windows, and the window overlap is $W - L$.

2.7 Spectrograms

So, the signal is simply the frequencies. Fourier gives us the frequencies discovered, but no location data. A *spectrogram* will show us both the frequencies discovered, and at what location. They are 2D arrays of complex numbers in a time/frequency array. They are visualised, and often processed, using magnitude and energy. They localise energies in **time** and **frequency**.

This is useful for fast forwarding speech. A naïve method to do this is to just play it back faster, but this will compress the sound waves, and increase both the frequency, and the pitch. We could resolve this by throwing away every other sample point, but I think that there is a better way. We could create a spectrogram, sample it, and reconstruct the speech. This way, frequencies and pitch of sound will remain unchanged.

It used to be that the second exercise (before ChatGPT) was to change the speed of speech, without changing the pitch. ChatGPT made this too trivial, and it has been discontinued.