

General Image Degradation Model

- In general, each pixel of a degraded image, $g(x, y)$, can be modelled as a weighted average of surrounding pixels in the original image $f(x, y)$. I.e.

$$g(x, y) = \iint h(x, y; x', y') f(x', y') dx' dy'$$

where h represents the blurring and is usually called the impulse response or point spread function (PSF)

- Consider the effect of the blurring on a single point: i.e. let the original image be an impulse at the (m, n) 'th pixel $f(x, y) = \delta(x - m, y - n)$, then $g(x, y)$ becomes

$$\begin{aligned} g(x, y) &= \iint h(x, y; x', y') \delta(x' - m, y' - n) dx' dy' \\ &= h(x, y; m, n) \end{aligned}$$

- This tells us that the blur function $h(x, y; m, n)$ describes the blurring of a point at the (m, n) 'th pixel and in general the extent of blurring can be different at different points

- In practice, most blurring functions $h(x, y; x', y')$ *decreases* as the point (x', y') gets *further* away from (x, y) (think of it as trying to smear a drop of ink on a piece of paper, the further away from the drop, the less smearing there will be)
- When the blurring process is the same at every pixel, and that the extent of the blurring depends only on the spatial separation from (x, y) , then we have spatially invariant blurring

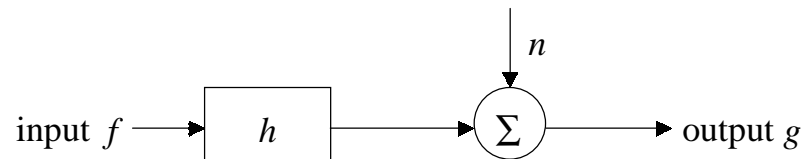
$$h(x, y; x', y') = h(x - x', y - y')$$

- In addition to blurring, another source of degradation is noise, which could be caused by the digitization process, atmospheric disturbance, or dust etc
- A popular model for noise is to treat it as an additive Gaussian random variable, with mean zero and variance σ^2

- Thus we have the spatially invariant degradation model

$$g(x, y) = \iint h(x - x', y - y') f(x', y') dx' dy' + n(x, y) \\ = h \otimes f + n(x, y)$$

- This model can be described symbolically



blurring system

(h is its impulse response)

- The PSF $h(x, y)$ is also known as the blurring function
 - In some applications, the PSF can be calculated theoretically, as in our motion blur model or measured experimentally (e.g. using a guide star)
- ✍ Guide Star: shoot a well focussed beam of laser to the ionosphere. This will create an artificial light source in the sky. A picture of it is the blur function of the camera (plus atmospheric) system

Problem



Suppose we know the PSF $h(x, y)$, and the noise variance σ^2 , can we recover the original image $f(x, y)$ from the degraded image $g(x, y)$? This is the problem of image restoration.

Inverse filter

The degradation model is

$$g = h \otimes f + n$$

Taking Fourier Transform, we have

$$G =$$

Since we know h and hence its Fourier Transform H (called optical transfer function), one “obvious” solution is the inverse filter estimate

$$\hat{F}(u, v) \triangleq \frac{G(u, v)}{H(u, v)} = F(u, v) + \frac{N(u, v)}{H(u, v)}$$

$$\approx F(u, v), \text{ if noise } N \text{ is negligible}$$

☞ Note that we are working in the frequency domain and inverse filtering as presented here requires division by the optical transfer function and should **not** be confused with matrix inversion

☞ Note also that the inverse filter is the best filter when noise is negligible as it recovers $F(u, v)$ exactly. Thus all other filters must at least *act like* the inverse filter in the limit of no noise

☞ Unfortunately, noise has *high frequency* Fourier Transform components while the optical transfer function H becomes rather small (or even zero) at *high frequencies!*

☞ Thus in practice, inverse filtering does not work due to *noise amplification*

☞ Pseudo-inverse filter

Restoration is obtained using the filter H_p^{-1} , i.e.

$$\hat{F} \triangleq H_p^{-1} G$$

where

$$H_p^{-1}(u, v) = \begin{cases} & \text{if } |H| > \varepsilon \\ & \text{if } |H| \leq \varepsilon \end{cases}$$

☞ The small but non-zero threshold ε avoids the problem of noise-amplification, but is difficult to derive

☞ Modified inverse filter

Another solution that reduces noise amplification uses the filter

$$H_M^{-1}(u, v) =$$

[i.e. the restored image is obtained from $\hat{F} \triangleq H_M^{-1} G$]

- ☞ When $r=0$, H_M^{-1} reduces to the inverse filter
- ☞ When $|H|$ is large, r^2 becomes negligible and H_M^{-1} is close to the inverse filter, which means we can expect good restoration
- ☞ When $|H|$ is small, noise amplification is reduced since the denominator is at least $\geq r^2$.
- ☞ In practice, the choice of ϵ in H_p^{-1} , and r^2 in H_M^{-1} are guessed at by trial and error
- ☞ Inverse filters and Pseudo-inverse filters are very sensitive to noise. Modified-inverse filters are good when appropriate values of r^2 are chosen.
- ☞ It is clearly desirable to calculate the optimal value for r^2 . This is the basis for the development of the Wiener filter. Before discussing Wiener filters, we need to cover the following two results:

1) Signal-independent noise

In general, noise $n(x,y)$ is generated independent of the original image $f(x,y)$. Thus noise and the image (or signal) have zero or negligible correlation, i.e.

$$\underline{0 = n \circ f \Leftrightarrow}$$

In such cases, we say the noise is signal-independent.

2) Orthogonality Principle (for 1D case)

Let $e = \int (f(x) - (h_w \otimes g)(x))^2 dx$. We want to find the function h_w which minimizes the mean square error e . This means we need to evaluate some sort of functional derivatives.

A sort of rough "proof" (outside syllabus):

$$\begin{aligned} \frac{\partial e}{\partial h_w(x'')} &= \int 2(f - h_w \otimes g) \frac{\partial}{\partial h_w(x'')} (f - h_w \otimes g) dx \\ &= -2 \int (f - h_w \otimes g) \frac{\partial}{\partial h_w(x'')} \left(\int h_w(x') g(x - x') dx' \right) dx \\ &\quad \text{(this is non-zero only when } x' = x'') \\ &= -2 \int (f - h_w \otimes g) \cdot g(x - x'') dx \\ &= -2g \circ (f - h_w \otimes g), \quad \text{(definition of correlation)} \end{aligned}$$

☞ Hence, e is minimized when $\frac{\partial e}{\partial h_w} = 0 \Rightarrow h_w$ satisfies the orthogonality principle

$$\underline{(f - h_w \otimes g) \circ g = 0}$$

☞ This result also holds in 2D. We are now ready for the Wiener filter

☞ Wiener filter

The restoration given by the filter H_w

$$\hat{F} = H_w G$$

is optimal in *the least squares sense* when

$$H_w = \frac{H^*}{|H|^2 + r^2}$$

where

$$r^2 = \frac{|N|^2}{|F|^2}$$

Proof

$\hat{F} = H_w G$ is the Fourier Transform of the restored image \hat{f} . Let h_w be the inverse FT of H_w , then by the convolution theorem,

$$\hat{f} = h_w \otimes g$$

we can measure how good this restoration is by computing

the error from the original error in the (least) squares sense:

$$\begin{aligned} e &= \int (f - \hat{f})^2 dx dy \\ &= \int (f - h_w \otimes g)^2 dx dy \end{aligned}$$

The orthogonality principle tells us that e is minimized when h_w satisfies

$$(f - h_w \otimes g) \circ g = 0$$

taking Fourier Transform, we have

$$\begin{aligned} (F - H_w G) G^* &= 0 = (F - H_w G)^* G \\ \Rightarrow H_w &= \frac{G^* F}{|G|^2} \end{aligned} \quad (\text{R1})$$

But from the degradation model, we have

$$g = h \otimes f + n \Rightarrow G = H F + N$$

Substituting G into equation (R1) yields

$$\begin{aligned} H_w &= \frac{H^* |F|^2 + N^* F}{(HF + N)(HF + N)^*} \\ &= \frac{H^* |F|^2 + N^* F}{|H|^2 |F|^2 + |N|^2 + HF N^* + H^* F^* N} \end{aligned}$$

For signal-independent noise, $N^* F = 0 = F^* N$, hence

$$H_w = \frac{H^* |F|^2}{|H|^2 |F|^2 + |N|^2} = \frac{H^*}{|H|^2 + r^2}, \quad \text{where } r^2 = \frac{|N|^2}{|F|^2}$$

📺 Note that the Wiener filter is very similar to the modified inverse filter, except that the ratio r^2 is now a function of FT coefficients

📺 In practice, r^2 is difficult to evaluate since the original image is not known, and thus is usually approximated by a constant term and is called the noise-to-signal ratio, e.g.

$$r^2 \approx \frac{\iint |N|^2 dudv}{\iint |F|^2 dudv} = \frac{\iint n^2 dxdy}{\iint f^2 dxdy}$$

$$= \frac{\text{variance of noise}}{\text{variance of signal}} = \frac{1}{\text{SNR}}$$

- 📺 As in the modified-inverse filter case, the Wiener filter is close to the inverse filter when r^2 is small (i.e. when SNR is high). Thus Wiener filter restoration can be expected to be good at high SNR.
- 📺 When noise is significant, the Wiener filter achieves an optimal balance between acting like the inverse filtering (to restore the original image) and low-pass filtering (to suppress noise amplification)
- 📺 Wiener filter restorations are less sharp when more noise is present
- 📺 The determination of the optical transfer function H is non-trivial and is also an important area of research
- 📺 In some cases, it may be possible to restore a blurred image without knowing H ! This is called blind restoration and is a very difficult area of research!

Discretization

Since we are going to work with digital images, we need to develop some discretized version of Fourier Transform. In 1-Dimension, the continuous F.T. is given by

$$F(k) = \int_{-\infty}^{\infty} e^{-i2\pi kx} f(x) dx$$

In discrete case, suppose we have a sequence of N values $f(x)$, $x=0, 1, 2, \dots, N-1$, we define its (unitary) discrete Fourier Transform (DFT) as:

$$F(k) \triangleq \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{-i2\pi kx/N} f(x), \quad k = 0, 1, \dots, N-1$$

- $F(k)$ is now another sequence of N values. By analogy with the continuous case, it is the frequency representation for the spatial sequence $f(x)$, $x=0, 1, 2, \dots, N-1$.
- Note that the normalization factor $\frac{1}{\sqrt{N}}$ will make this

formulation more symmetrical and elegant. There are other formulations of DFT which uses other normalization factors.

- For convenience, let us define the order N (unitary) DFT matrix (i.e. $N \times N$ matrix)

$$W_{kx} \triangleq \frac{e^{-i2\pi kx/N}}{\sqrt{N}}, \quad k, x = 0, 1, 2, \dots, N-1$$

Note that we number rows and columns starting with 0 instead of 1!

- Examples: Let $w \triangleq e^{-i2\pi/N}$

$$\text{Then } W_{kx} = \frac{w^{kx}}{\sqrt{N}}, \quad k, x = 0, 1, \dots, N-1$$

$N = 2$ case

$$w = e^{-i2\pi/2} = -1$$

$$W = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

$N = 4$

$$w = e^{-i2\pi/4} = e^{-i2\pi/2} = -i$$

$$W = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 \\ 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w^9 \end{pmatrix}$$

$$= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}$$

- W is an example of an $N \times N$ unitary matrix, i.e.

$$W^H W = 1 = W W^H$$

where

$$W^H = (W^T)^*, \text{ the Hermitian adjoint of } W$$

- If W is real, then $W^H = W^T$, and Hermitian adjoint is the same as transpose. This means that W is now an

orthogonal matrix. Thus unitarity is just a generalization of orthogonality for complex matrices.

- Check : $N = 2$ case

$$W = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad W^H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

and

$$\begin{aligned} WW^H &= \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \blacksquare \end{aligned}$$

- With our definition of W , we can now express our DFT in matrix notation

$$\begin{aligned} F(k) &= \sum_{x=0}^{N-1} \frac{e^{-i 2\pi kx/N}}{\sqrt{N}} f(x), \quad k = 0, 1, \dots, N-1 \\ &= \sum_{x=0}^{N-1} W_{kx} f(x) \end{aligned}$$

or in matrix form

$$F = Wf, \quad \text{where } f = \begin{pmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{pmatrix} \text{ etc}$$

- The advantage of the matrix formulation is that DFT can now be easily computed. Moreover, the unitarity of W enables us to obtain the inverse DFT very easily.

$$\begin{aligned} F = Wf &\Rightarrow W^H F = W^H Wf = f \\ &\Rightarrow f = W^H F \end{aligned}$$

or

$$f(x) = \sum_{k=0}^{N-1} \frac{e^{i 2\pi kx/N}}{\sqrt{N}} F(k), \quad x = 0, 1, \dots, N-1$$

This is the inverse DFT and is much easier to derive than the continuous case.

Extension to 2D is straightforward. The 2D DFT of $f(x,y)$, $x, y = 0, 1, \dots, N-1$, is

$$\begin{aligned} F(u, v) &= \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \frac{e^{-i 2\pi ux/N}}{\sqrt{N}} \frac{e^{-i 2\pi vy/N}}{\sqrt{N}} f(x, y) \\ &= \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} W_{ux} f(x, y) W_{yv} \end{aligned}$$

$$\Rightarrow \underline{F = W f W}$$

Its inverse DFT can again be obtained easily by multiplying by W^H on both the left and right

$$W^H F W^H = W^H (W f W) W^H = f$$

$$\Rightarrow \underline{f = W^H F W^H}$$

or

$$\underline{f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \frac{e^{i 2\pi ux/N}}{\sqrt{N}} \frac{e^{i 2\pi vy/N}}{\sqrt{N}} F(u, v)}$$

- DFT enjoys virtually the same properties as the continuous Fourier Transform. However, there is one very important difference for convolution in the DFT.
- Recall the continuous convolution was defined by

$$h \otimes f \triangleq \int_{-\infty}^{\infty} h(x-x') f(x') dx'$$

In the discrete case when we have two N -value sequences, $h(x)$ and $f(x)$, $x = 0, 1, \dots, N-1$ the “natural” definition for discrete convolution would be

$$h \otimes f \triangleq \sum_{x'=0}^{N-1} h(x-x') f(x'), \quad x = 0, 1, \dots, N-1$$

However, we immediately see problems when x' takes values greater than x since $h(x-x')$ is not defined for negative indices!

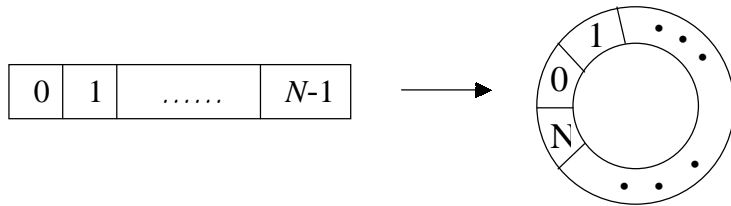
- In order to remedy the problem above, we treat all the negative indices by taking modulo N , which would convert all indices to a value from $0, 1, 2, \dots, N-1$

e.g. $N = 8$

$$-1 \equiv 8 - 1 \equiv 7 \pmod{8}$$

$$10 \equiv 10 - 8 \equiv 2 \pmod{8} \quad \text{etc}$$

- This idea can be easily implemented in C using the syntax $(x - x') \% N$
- The effect of working with indices modulo N is just like joining the two ends of a sequence into a closed loop. We call this the notion of wraparound.



- Using wraparound, we may now define circular convolution between two sequences

$$(h \otimes f)(x) \triangleq \sum_{x'=0}^{N-1} h(x - x' \bmod N) f(x'), \quad x \in [0, 1, \dots, N-1]$$

- The corresponding circular convolution theorem is just like the continuous case:

$$\text{if} \quad g(x) = h \otimes f$$

$$\text{then} \quad G(k) = H(k) F(k), \quad k=0, 1, 2, \dots, N-1$$

where G, H, F are DFT of g, h, f respectively

- The use of circular convolution theorem automatically implies wraparound in both the horizontal and the vertical direction (so that the image is folded up like a torus or a doughnut)
- Thus we may anticipate strange edge effects in some cases whenever we use DFT. This is one disadvantage of working in the frequency domain.
- The ID DFT for a sequence of N values requires $O(N^2)$ operations. The 2D DFT for $N \times N$ images requires $O(N^3)$ operations.



There exists Divide-and-Conquer type algorithm called Fast Fourier Transform[♦] (FFT) which only requires $O(N \log N)$ operations in 1D and $O(N^2 \log N)$ for 2D.

- FFT can be much faster than conventional DFT when N is large.

Example: when $N = 2^{10} = 1024$, the speed up using FFT is of the order of 100 times!

- FFT is considered one of the most important discovery in Applied Mathematics this century and has spurred many similar discovery of conquer and divide type algorithms that revolutionized computing since the 1960's. Examples include quick sort and the DCT (see below).

[♦] This topic is outside our syllabus. If interested, see e.g. GW 3.4.1.

Discrete Cosine Transform (DCT)

DFT involves complex arithmetic. A more convenient image transform is the DCT which does not have complex numbers and can be *loosely* thought of as the “real part” of DFT.

- Define N -order DCT matrix C

$$C_{kx} = \begin{cases} \frac{1}{\sqrt{N}}, & k = 0, 0 \leq x \leq N - 1 \\ \sqrt{\frac{2}{N}} \cos \frac{\pi(2x+1)k}{2N}, & 1 \leq k \leq N - 1, 0 \leq x \leq N - 1 \end{cases}$$

example: $N=2$

$$C = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \text{ the } N=2 \text{ DFT matrix } W!$$

$N=4$

$$C = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} \cos \frac{\pi}{8} & \frac{1}{\sqrt{2}} \cos \frac{3\pi}{8} & \frac{1}{\sqrt{2}} \cos \frac{5\pi}{8} & \frac{1}{\sqrt{2}} \cos \frac{7\pi}{8} \\ \frac{1}{\sqrt{2}} \cos \frac{\pi}{4} & \frac{1}{\sqrt{2}} \cos \frac{3\pi}{4} & \frac{1}{\sqrt{2}} \cos \frac{5\pi}{4} & \frac{1}{\sqrt{2}} \cos \frac{7\pi}{4} \\ \frac{1}{\sqrt{2}} \cos \frac{3\pi}{8} & \frac{1}{\sqrt{2}} \cos \frac{9\pi}{8} & \frac{1}{\sqrt{2}} \cos \frac{15\pi}{8} & \frac{1}{\sqrt{2}} \cos \frac{21\pi}{8} \end{pmatrix}$$

- Note that C is real and orthogonal, i.e.

$$CC^T = 1 = C^T C$$

- The ID DCT of the sequence of N values $f(x)$, $x=0, 1, \dots, N-1$ is defined as

$$F(k) \triangleq \sum_{x=0}^{N-1} C_{kx} f(x), \quad k = 0, 1, \dots, N-1$$

or in the matrix form

$$F = Cf,$$

where the vectors f and F are defined as in DFT

- The orthogonality of the DCT matrix easily leads to the inverse DCT

$$F = Cf \Rightarrow$$

$$\Rightarrow \underline{f = C^T F}$$

or

$$f(x) = \sum_{k=0}^{N-1} C_{xk}^T F(k), \quad x = 0, 1, \dots, N-1$$

$$= \underline{\sum_{k=0}^{N-1} C_{kx} F(k)}$$

- The 2D case is similar to that for DFT. Given a digital image $f(x,y)$, $x, y=0, 1, \dots, N-1$, its DCT is defined by

$$\begin{aligned} F(u,v) &= \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_{ux} C_{vy} f(x,y) \\ &= \sum_x \sum_y C_{ux} f(x,y) C_{yv}^T \end{aligned}$$

in matrix form, this is

$$\underline{F = C f C^T}$$

Multiply on the left by C^T and on the right by C , we easily obtain the inverse DCT

$$C^T F C = \quad = f$$

$$\Rightarrow \underline{f = C^T F C}$$

or

$$\begin{aligned} f(x, y) &= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C_{xu}^T F(u, v) C_{vy} \\ &= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C_{ux} C_{vy} F(u, v) \end{aligned}$$

- As for DFT, there exists a very fast Divide-and Conquer type algorithm for DCT, also of order $O(N \log N)$ for N -order 1D sequences, and $O(N^2 \log N)$ for N -by- N images.
- DCT has excellent energy compaction for highly correlated images, i.e. most of the high frequency DCT components are negligible. This nice property is the reason why DCT-based compression algorithms are so popular today.

- There are many other image transforms. However, for efficient implementations, the transformation *kernels*, $K(x, y, u, v)$, of such transforms are usually designed to satisfy certain nice properties such as:

1) Separable

$$K(x, y, u, v) = K_r(x, u)K_c(y, v)$$

so that we may treat the rows and columns of an image independently

2) Symmetric

$$K_r(\alpha, \beta) = K_c(\alpha, \beta)$$

so that rows and columns are treated in the same way

- Examples of separable and symmetry 2D transforms are DFT, DCT, Discrete Sine Transform, Walsh Transform, Hadamard Transform, Haar Transform.....

¶ This is just about the easiest way to get your name in a textbook: design a transform yourself! But make sure that there is a Divide & Conquer algorithm for your transform.