Fourier, filtering, smoothing, and noise

Nuno Vasconcelos

ECE Department, UCSD

(with thanks to David Forsyth)
Plan for today

we have talked about some theoretic of 2D DSP

today we will see examples with real images

we will:

• talk about the discrete-space Fourier transform
• sampling in 2D
• visualize concepts such as frequency spectra and aliasing
• see that very important vision operations are really just filtering, with some post processing
• introduce some important filters that are widely used in practice
• talk about the impact of noise and how to deal with it
The Discrete-Space Fourier Transform

is, once again, a straightforward extension of the 1D Discrete-Time Fourier Transform

\[
X(\omega_1, \omega_2) = \sum_{n_1} \sum_{n_2} x[n_1, n_2] e^{-j\omega_1 n_1} e^{-j\omega_2 n_2}
\]

\[
x[n_1, n_2] = \frac{1}{(2\pi)^2} \iint X(\omega_1, \omega_2) e^{j\omega_1 n_1} e^{j\omega_2 n_2} d\omega_1 d\omega_2
\]

properties:

• basically the same as in 1D (see table in Lim, page 25)

• only novelty is separability (homework)

\[
x[n_1, n_2] = x_1[n_1] x_2[n_2] \leftrightarrow X(\omega_1, \omega_2) = X_1(\omega_1) X_2(\omega_2)
\]
Image spectrum

- two images, the magnitude, and phase of their FTs
Phase and Magnitude

- curious fact
  - all natural images have about the same magnitude transform
  - monotonically decaying with frequency

\[ \chi(\omega_1, \omega_2) \propto \frac{1}{\omega_1^2 + \omega_2^2} \]

- hence, phase seems to matter, but magnitude largely doesn’t

- we have seen this a little bit in the first problem set
- took two pictures, swap the phase transforms, compute the inverse
- here is another example
The importance of phase

Reconstruction with zebra phase, cheetah magnitude

Reconstruction with cheetah phase, zebra magnitude
Sampling in 2D

- consider an analog signal $x_c(t_1,t_2)$ and let its analog Fourier transform be $X_c(\Omega_1,\Omega_2)$
  - we use capital $\Omega$ to emphasize that this is analog frequency
- sample with period $(T_1,T_2)$ to obtain a discrete-space signal

$$x[n_1,n_2] = x_c(t_1,t_2)|_{t_1=n_1T_1; t_2=n_2T_2}$$
Sampling in 2D

- relationship between the Discrete-Space FT of \( x[n_1,n_2] \) and the FT of \( x_c(t_1,t_2) \) is simple extension of 1D result

\[
X(\omega_1, \omega_2) = \frac{1}{T_1 T_2} \sum_{r_1=-\infty}^{\infty} \sum_{r_2=-\infty}^{\infty} X_c \left( \frac{\omega_1 - 2\pi r_1}{T_1}, \frac{\omega_2 - 2\pi r_1}{T_1} \right)
\]

DSFT of \( x[n_1,n_2] \) 
“discrete spectrum”

FT of \( x_c(\omega_1,\omega_2) \) 
“analog spectrum”

- Discrete Space spectrum is sum of replicas of analog spectrum
  - in the “base replica” the analog frequency \( \Omega_1 (\Omega_2) \) is mapped into the digital frequency \( \Omega_1 T_1 (\Omega_2 T_2) \)
  - discrete spectrum has periodicity \((2\pi,2\pi)\)
For example

\[ \Omega' \rightarrow \alpha = \Omega' T_1 \]
\[ \Omega'' \rightarrow \beta = \Omega'' T_2 \]

\( \Rightarrow \) no aliasing if

\[ \begin{cases} 
\Omega' T_1 \leq 2\pi - \Omega' T_1 \\
\Omega'' T_2 \leq 2\pi - \Omega' T_2
\end{cases} \]

\[ \Leftrightarrow \begin{cases} 
T_1 \leq \pi / \Omega' \\
T_2 \leq \pi / \Omega''
\end{cases} \]
Aliasing

- the frequency \( (\Omega'/\pi, \Omega''/\pi) \) is the critical sampling frequency
- below it we have aliasing
- this is just like the 1D case, but now there are more possibilities for overlap
Reconstruction

if there is **no aliasing** we can recover the signal in a way similar to the 1D case

\[ y_c(t_1, t_2) = \sum_{n_1 = -\infty}^{\infty} \sum_{n_2 = -\infty}^{\infty} x[n_1, n_2] \frac{\sin \frac{\pi}{T_1} (t_1 - n_1 T_1)}{\pi \frac{T_1}{t_1 - n_1 T_1}} \frac{\sin \frac{\pi}{T_2} (t_2 - n_2 T_2)}{\pi \frac{T_2}{t_2 - n_2 T_2}} \]

**note:** in 2D there are many more possibilities than in 1D

- e.g. the sampling grid does not have to be rectangular, e.g. hexagonal sampling when \( T_2 = T_1/\sqrt{3} \) and

\[ x[n_1, n_2] = \begin{cases} x_c(t_1, t_2) |_{t_1 = n_1 T_1; t_2 = n_2 T_2} & n_1, n_2 \text{ both even or odd} \\ 0 & \text{otherwise} \end{cases} \]

- in practice, however, one usually adopts the rectangular grid
Sampling and aliasing in 2D

- In summary, sampling is not very different from the 1D case.

- But aliasing is a lot more fun to look at in images.

- It shows up in video too (the wagon wheel effect).
a sequence of images obtained by down-sampling without any filtering

aliasing: the low-frequency parts are replicated throughout the low-res image
The role of smoothing

- too little leads to aliasing
- too much leads to loss of information
Linear Filtering

- smoothing is implemented with linear filters
- given an image $x(n_1,n_2)$, filtering is the process of convolving it with a kernel $h(n_1,n_2)$

$$y(n_1,n_2) = \sum_{k_1,k_2} x(k_1,k_2) h(n_1 - k_1, n_2 - k_2)$$

- some very common operations in image processing are nothing but filtering, e.g.
  - smoothing an image by low-pass filtering
  - contrast enhancement by high pass filtering
  - finding image derivatives
  - noise reduction
Popular filters

- **box function**
  \[ R_{N_1 \times N_2}(n_1, n_2) = \begin{cases} 
  1, & 0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1 \\
  0, & \text{otherwise}
\end{cases} \]

- **Fourier transform of a box** is the sinc, low-pass filter

- **side-lobes** produce artifacts, smoothed image **does not** look like the result of defocusing
Example: Smoothing by Averaging
Camera defocusing

- if you point an out-of-focus camera at a very small white light (e.g. a light-bulb) at night, you get something like this
- the light can be thought of as an impulse
- this must be the impulse response
- well approximated by a Gaussian
- more natural filter for image blur than the box

\[
h(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]
The Gaussian

- the discrete space version is

\[ h(n_1, n_2) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{n_1^2 + n_2^2}{2\sigma^2}\right) \]

- obviously separable

\[ h(n_1, n_2) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{n_1^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{n_2^2}{2\sigma^2}} \]

- \( h(n_1,n_2) \) has Fourier transform

\[ H(\omega_1, \omega_2) = \exp\left(-\frac{\sigma^2 (\omega_1^2 + \omega_2^2)}{2}\right) \]
The Gaussian filter

- The Fourier transform of a Gaussian is a Gaussian $(\sigma_x, \sigma_y) \propto (1/\sigma_{w1}, 1/\sigma_{w2})$

- Note that there are no annoying side-lobes
Smoothing with a Gaussian
Role of the variance

- The variance controls the amount of smoothing.
- Each column shows different realizations of an image of Gaussian noise.
- Each row shows smoothing with Gaussians of different $\sigma$. 

\[
\begin{array}{ccc}
\sigma=0.05 & \sigma=0.1 & \sigma=0.2 \\
\text{no smoothing} & \sigma=1 \text{ pixel} & \sigma=2 \text{ pixels}
\end{array}
\]
Gradients and edges

- for image understanding, one of the problems is that there is too much information in an image.
- just smoothing is not good enough.
- how to detect important (most informative) image points?
- note that derivatives are large at points of great change:
  - changes in reflectance (e.g. checkerboard pattern)
  - change in object (an object boundary is different from background)
  - change in illumination (the boundary of a shadow)
- these are usually called edge points.
- detecting them could be useful for various problems:
  - segmentation: we want to know what are object boundaries
  - recognition: cartoons are easy to recognize and terribly efficient to transmit.
The importance of edges
Gradients

- for a 2D function, \( f(x,y) \) the gradient at a point \((x_0, y_0)\)

\[
\nabla f(x_0, y_0) = \left( \frac{\partial f}{\partial x}(x_0, y_0), \frac{\partial f}{\partial y}(x_0, y_0) \right)^T
\]

\[
= \left( f_x(x_0, y_0), f_y(x_0, y_0) \right)^T
\]

is the direction of greatest increase at that point

- the gradient magnitude

\[
\|\nabla f(x_0, y_0)\|^2 = \left( \frac{\partial f}{\partial x}(x_0, y_0) \right)^2 + \left( \frac{\partial f}{\partial y}(x_0, y_0) \right)^2
\]

measures the rate of change

- it is large at edges!
Derivatives and convolution

Recall that a derivative is defined as

\[
\frac{\partial f(x)}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}
\]

Linear and shift invariant, so must be the result of a convolution.

We could approximate as

\[
\frac{\partial f(n)}{\partial n} = \frac{f(n+1) - f(n)}{1} = f(n+1) - f(n) = f \ast h(n)
\]

Where the derivative kernel is

\[
h(n) = \delta(n+1) - \delta(n)
\]
Finite difference kernels

- in two dimensions we have various possible kernels
- e.g., $N_1=2$, $N_2=3$, derivative along $n_1$, (line $n_2=k$) (horizontal)

\[
\begin{bmatrix}
0 & 0 & 1 & -1 \\
1 & -1 & 1 & -1 \\
0 & 0 & 1 & -1 \\
\end{bmatrix}
\]

- derivative along $n_2$, (line $n_1=k$) (vertical)

\[
\begin{bmatrix}
0 & -1 & 0 & -1 & -1 & -1 \\
0 & 1 & 0 & 1 & 1 & 1 \\
\end{bmatrix}
\]

- derivative along line $n_1=n_2$ (diagonal)

\[
\begin{bmatrix}
0 & 0 & -1 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 \\
1 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
Finite difference kernels

- note that, when
  
  \[
  h(n_1, n_2) = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 0 \end{bmatrix}
  \]
  
- we have
  
  \[
  H(\omega_1, \omega_2) = e^{j\omega_1} - 1
  = \left( e^{\frac{j\omega_1}{2}} - e^{-\frac{j\omega_1}{2}} \right) e^{\frac{j\omega_1}{2}}
  = 2 \sin\left( \frac{\omega_1}{2} \right) e^{\frac{j\omega_1}{2}}
  \]
  
  - derivative is a high-pass filter
  - hw: check that this holds for all others
  - intuitive, because a derivative is a measure of the rate of change of a function
Finite differences

Q: which one do we have here? (gray=0, white=+, dark=-)
Finite differences and noise

because they perform high-pass filtering, finite difference filters respond strongly to noise.

generally, the larger the noise the stronger the response.

for noisy images it is usually best to apply some smoothing before computing derivatives.

what do mean by noise?

we only consider the simplest model:

• independent stationary additive Gaussian noise
• the noise value at each pixel is given by an independent draw from the same normal probability distribution

\[ y(n_1, n_2) = x(n_1, n_2) + \varepsilon(n_1, n_2), \quad \varepsilon \sim \mathcal{N}(0, \sigma^2) \]
\[ \text{sigma}=1 \]
sigma=16
Finite differences responding to noise

Increasing noise variance

- Note that as the noise variance increases, the estimates of the image derivative are also very noisy.
- Q: Would a larger filter do better?
The response of a linear filter to noise

- Suppose we apply filter $h(n_1, n_2)$ to

$$y(n_1, n_2) = x(n_1, n_2) + \varepsilon(n_1, n_2), \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- By linearity

$$h^* y = h^* x + h^* \varepsilon$$

- Consider the noise term

$$\nu(n_1, n_2) = \sum_{k_1, k_2} \varepsilon(k_1, k_2) h(n_1 - k_1, n_2 - k_2)$$

- By linearity of expectation

$$E[\nu(n_1, n_2)] = \sum_{k_1, k_2} E[\varepsilon(k_1, k_2)] h(n_1 - k_1, n_2 - k_2) = 0$$
The response of a linear filter to noise

- and since the $\varepsilon(n_1,n_2)$ are independent

\[
\text{var}[\nu(n_1,n_2)] = \sum_{k_1,k_2} \text{var}[\varepsilon(k_1,k_2)]h^2(n_1 - k_1, n_2 - k_2)
\]

\[
= \sigma^2 \sum_{k_1,k_2} h^2(n_1 - k_1, n_2 - k_2)
\]

- output error has
  - zero mean
  - variance that increases with the size of the filter and the input variance

- increasing the filter size would definitely not help!
Smoothing reduces noise

- noise has a lot of high-frequencies
- strategy:
  1. start by low-pass filtering, to suppress noise
  2. compute derivative on smoothed image
- i.e. for a smoothing filter $g(n_1,n_2)$ compute
  \[ h \ast (g \ast x) \]
- note that, by associativity of convolution, this is equal to
  \[ (h \ast g) \ast x \]
- i.e. filter the image with the filter $h \ast g$
The derivative of a Gaussian

- let’s consider, for example,
  \[ h(n_1, n_2) = \delta(n_1 + 1, n_2) - \delta(n_1, n_2) \]
- in which case
  \[ h \ast g(n_1, n_2) = g(n_1 + 1, n_2) - g(n_1, n_2) \]
  is a difference of two Gaussians
- this is the derivative of a Gaussian (DoG) filter
- for other definitions of \( h \) we have a similar result
Smoothed derivatives

no smoothing

with smoothing
Choosing the right scale

- The scale of the smoothing filter affects:
  - derivative estimates,
  - the semantics of the derivative image
- Trade-off between noise and ability to detect detail

1 pixel | 3 pixels | 7 pixels
Any questions?
Gradients and edges

- by now we have a good idea of how to differentiate images
- remember that edges are points of large gradient magnitude
- edge detection strategy
  1. determine magnitude of image gradient
     
     \[ \left\| \nabla f(x_0, y_0) \right\|^2 = \left( \frac{\partial f}{\partial x} (x_0, y_0) \right)^2 + \left( \frac{\partial f}{\partial y} (x_0, y_0) \right)^2 \]

  2. mark points where gradient magnitude is particularly large wrt neighbours (ideally, curves of such points)
Looks easy but

three major issues (to discuss next class):

• 1) gradient magnitude at different scales is different (see below); which should we choose?

• 2) gradient magnitude is large along thick trail; what are the significant points?

• 3) how do we link the relevant points up into curves?