More Poisson, Deblurring and Fourier analysis

Fredo Durand
MIT EECS 6.815/6.865

Hi, Dr. Elizabeth?
Yeah, uh... I accidentally took the Fourier transform of my cat...

Meow!
Conjugate Gradient
Behavior of gradient descent

• Zigzag or goes straight depending if we’re lucky
  – Ends up doing multiple steps in the same direction

Unlucky

Lucky
Our residuals

- times 10
- We zigzag between the two same checkerboard patterns
Our residuals

• times 10
• We zigzag between the two same checkerboard patterns
Overview

• Naive iterative solver: Zigzag
  – Ends up doing multiple steps in the same direction

• Conjugate gradient: make sure never go twice in the same direction
  – Don’t go exactly along gradient direction

Good news: the code is simple

```matlab
function [x] = conjgrad(A,b,x0)
    r = b - A*x0;
    w = -r;
    z = A*w;
    a = (r'*w)/(w'*z);
    x = x0 + a*w;
    B = 0;
    for i = 1:size(A);
        r = r - a*z;
        if( norm(r) < 1e-10 )
            break;
        B = (r'*z)/(w'*z);
        w = -r + B*w;
        z = A*w;
        a = (r'*w)/(w'*z);
        x = x + a*w;
    end
end
```

Conjugate gradient

- **Smarter choice of direction**
  - Ideally, step directions should be orthogonal to one another (no redundancy)
  - But tough to achieve
  - Next best thing: make them $A$-orthogonal (conjugate)
    - That is, orthogonal when transformed by $\sqrt{A}$
    - Turn the ellipses into circles

\[
d^T_{(i)} A d_{(j)} = 0
\]

Figure 22: These pairs of vectors are $A$-orthogonal... because these pairs of vectors are orthogonal.
Conjugate gradient overview

• For each step:
  – Take the residual (gradient)
  – Make it A-orthogonal to the previous ones
  – Find minimum along this direction

Figure 30: The method of Conjugate Gradients.
How to make vectors orthogonal

✧ Subtract the non-orthogonal component
  • Use dot product
✧ Make w orthogonal to v:
✧ \( w' = w - \gamma v \)
✧ where \( \gamma = v^T w / v^T v \)
  • denominator needed when \( v \) is not unit length
✧ Gram Schmidt generalizes this to \( n \) vectors
Making vectors A-orthogonal

- Start with residual $r_{(i+1)}$, turn it into $d_{(i+1)}$ which is A-orthogonal to all previous iteration directions

- Perform Gram-Schmidt: subtract the component that are non-A-orthogonal to previous directions
  - more complex dot product to take A into account
  - Turns out we only need to do it for previous iteration direction (lucky!)
Making vectors A-orthogonal

✧ Start with residual $r_{(i+1)}$

✧ Perform Gram-Schmidt:
subtract the non-A-orthogonal component

• turns out we need to take care of only the previous direction $d_{(i)}$

$$d_{(i+1)} = r_{(i+1)} + \beta_{(i+1)} d_{(i)}$$

• where

$$\beta_{(i+1)} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}}$$

Similar to previous formula, but involves r’s to make orthogonal to d’s

• See Shewchuck’s text for derivation
Comparison & recap

- Gradient descent
  - \( r(i) = b - Ax(i) \)
  - \( x(i+1) = x(i) + \alpha(i)r(i) \)
  - \( \alpha = \frac{r^T(i)r(i)}{r^T(i)Ar(i)} \)

- Conjugate gradient
  - \( r(i) = b - Ax(i) \)
  - \( x(i+1) = x(i) + \alpha(i)d(i) \)
  - \( \alpha(i) = \frac{d^T(i)r(i)}{d^T(i)Ad(i)} \)
  - \( \beta(i+1) = \frac{r^T(i+1)r(i+1)}{r^T(i)r(i)} \)
  - \( d(i+1) = r(i+1) + \beta(i+1)d(i) \)
Saving some computation

- Bottleneck: matrix-vector products
- Can avoid one:
  \[ r(i+1) = b - Ax(i+1) \]
  \[ = b - A(x(i) + \alpha(i)d(i)) \]
  \[ = (b - Ax(i)) + \alpha(i)Ad(i) \]
  \[ = r(i) + \alpha(i)Ad(i) \]
- Same as the one needed for \( \alpha(i) \)

\[ r(i) = b - Ax(i) \]
\[ x(i+1) = x(i) + \alpha(i)d(i) \]
\[ \alpha(i) = \frac{d^T(i)r(i)}{d^T(i)Ad(i)} \]
\[ \beta(i+1) = \frac{r^T(i+1)r(i+1)}{r^T(i)r(i)} \]
\[ d(i+1) = r(i+1) + \beta(i+1)d(i) \]
Bells and whistles

- Update $r^{(i)}$ incrementally (previous slide)
  - Compute product $Ad$ once only
  - Pitfall: could drift
  - maybe reset once in a while with full calculation
- Only need to be able to apply matrix $A$ to a vector
  - Often you don’t even store $A$, but use a procedure
- Conjugate gradient is guaranteed to converge in $n$ iterations for $n$ unknowns
  - But we usually want to stop way earlier
The Algorithm

function [x] = conjgrad(A,b,x0)  
  r = b - A*x0;  
  d = -r;  
  z = A*d;  
  a = (r'*d)/(d'*z);  
  x = x0 + a*d;  
  B = 0;  
  for i = 1:size(A);  
    r = r - a*z;  
    if( norm(r) < 1e-10 )  
      break;  
    end  
    B = (r'*z)/(d'*z);  
    d = -r + B*d;  
    z = A*d;  
    a = (r'*d)/(d'*z);  
    x = x + a*d;  
end
Conjugate gradient

• For each step:
  – Take the residual (gradient)
  – Make it A-orthogonal to the previous ones
  – Find minimum along this direction

• Plus life is good:
  – In practice, you only need the previous one
  – You can show that the new residual \( r_{(i+1)} \) is already A-orthogonal to all previous directions \( d \) but \( d_{(i)} \)

Figure 30: The method of Conjugate Gradients.
Convergence
Residuals and direction

• times 10, displayed at 10fps
Compared to gradient descent

gradient descent

conjugate gradient
When use Conjugate Gradient?

- $Ax=b$
- $A$ is positive definite
- $A$ is sparse

- Disadvantage compared to factorization
  + backsubstitution:
    - you start from scratch for every new $b$
    - error if not converged

- Bottomline: use a direct solver when you can afford it, conjugate gradient otherwise
The two references

- An Introduction to the Conjugate Gradient Method Without the Agonizing Pain
  Edition 1 1/4
  Jonathan Richard Shewchuk
  August 4, 1994

- Iterative methods for sparse linear systems (2nd edition)
  Yousef Saad


- http://books.google.com/books?id=Uoe7xBOhS5AC&dq=saad+iterative&printsec=frontcover&source=in&hl=en&ei=Y2GtSerjMdW5twft78CHBg&sa=X&oi=book_result&resnum=11&ct=result#PPR5,M1
PRECONDITIONING
Idea

- We want to solve $Ax=b$
- For any invertible matrix $M$, this is the same as solving $MAx=Mb$
- Maybe some $M$ make the problem easier
- Preconditioning seeks a matrix $M$ that accelerates convergence
- In practice, $M$ does not need to be applied to $A$, only to direction vectors $d$
Preconditioning

✧ At a high level, try to turn the ellipses into circles
  • Then even gradient descent could work well.
Preconditioning

- Perfect preconditioning involves the inverse matrix
  - Probably too costly an acceleration!
- Simplest preconditioning: divide by diagonal elements (good if matrix has strong diagonal)
- Run a solver (e.g. Cholesky decomposition) but only partially
- Or use smart basis functions such as wavelets or pyramids
  - http://portal.acm.org/citation.cfm?id=1142005
MULTIGRID
Motivation

- Laplace equation (minimize square gradient)
- Boundary conditions:
  1 at one corner,
  zero on opposite 2 borders
- Initialize with e.g.
  zero everywhere
- 1st iteration only updates pixels connected to corner
- 2nd iteration only updates their neighbors
- Takes width to reach border: slow
Multigrid

✧ Solve the problem at multiple resolutions
✧ In particular, also solve a lower-resolution version where propagation is faster
✧ Initialize high-resolution version with upsampled-coarser resolution
✧ Also update coarser solution with finer solution

http://www.mgnet.org/mgnet/tutorials/xwb/mg.html
The reference

- https://computation.llnl.gov/casc/people/henson/mgtut/welcome.html
Refs

- http://www.cs.huji.ac.il/~yweiss/Colorization/
Eye candy
Result (eye candy)

source/destination  cloning  seamless cloning
sources  destinations  cloning  seamless cloning
Figure 2: Concealment. By importing seamlessly a piece of the background, complete objects, parts of objects, and undesirable artifacts can easily be hidden. In both examples, multiple strokes (not shown) were used.
Manipulate the gradient

- Mix gradients of g & f: take the max

Figure 8: **Inserting one object close to another.** With seamless cloning, an object in the destination image touching the selected region $\Omega$ bleeds into it. Bleeding is inhibited by using mixed gradients as the guidance field.
Figure 6: **Inserting objects with holes.** (a) The classic method, color-based selection and alpha masking might be time consuming and often leaves an undesirable halo; (b-c) seamless cloning, even averaged with the original image, is not effective; (d) mixed seamless cloning based on a loose selection proves effective.
Figure 7: **Inserting transparent objects.** Mixed seamless cloning facilitates the transfer of partly transparent objects, such as the rainbow in this example. The non-linear mixing of gradient fields picks out whichever of source or destination structure is the more salient at each location.
Seamless Image Stitching in the Gradient Domain

- Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss
  
  http://eprints.pascal-network.org/archive/00001062/01/tips05-blending.pdf

- Various strategies (optimal cut, feathering)

Fig. 1. Image stitching. On the left are the input images. $\omega$ is the overlap region. On top right is a simple pasting of the input images. On the bottom right is the result of the GIST1 algorithm.
Photomontage


Figure 6 We use a set of portraits (first row) to mix and match facial features, to either improve a portrait, or create entirely new people. The faces are first hand-aligned, for example, to place all the noses in the same location. In the first two images in the second row, we replace the closed eyes of a portrait with the open eyes of another. The user paints strokes with the designated source objective to specify desired features. Next, we create a fictional person by combining three source portraits. Gradient-domain fusion is used to smooth out skin tone differences. Finally, we show two additional mixed portraits.
Reduce big gradients

- Dynamic range compression
- See Fattal et al. 2002

Figure 10: **Local illumination changes.** Applying an appropriate non-linear transformation to the gradient field inside the selection and then integrating back with a Poisson solver, modifies locally the apparent illumination of an image. This is useful to highlight under-exposed foreground objects or to reduce specular reflections.
Gradient tone mapping

- Fattal et al. Siggraph 2002
Gradient attenuation

From Fattal et al.
Fattal et al. Gradient tone mapping
Gradient tone mapping


Fig. 1. (a) Mediastinal window of thoracic CT scan. (b) Lung window of thoracic CT scan. (c) Clipped solution of equation (2) for the fusion of (a) and (b). (d) Linearly scaled solution of (2) for the fusion of (a) and (b). (e) Solution of equation (6) for the fusion of (a) and (b).
Gradient tone mapping


Fig. 4. Left: average of images in figure 2. Middle: rendering of the sum of the images in figure 2 through adaptive histogram compression. Right: fusion of images in figure 2 using the obstacle method.
Retinex

• Land, Land and McCann (inventor/founder of polaroid)
• Theory of lightness perception (albedo vs. illumination)
• Strong gradients come from albedo, illumination is smooth
Color2gray

- Use Lab gradient to create grayscale images

**Color2Gray: Salience-Preserving Color Removal**

Amy A. Gooch  Sven C. Olsen  Jack Tumblin  Bruce Gooch
Northwestern University *

Figure 1: A color image (Left) often reveals important visual details missing from a luminance-only image (Middle). Our Color2Gray algorithm (Right) maps visible color changes to grayscale changes. *Image: Impressionist Sunrise by Claude Monet, courtesy of Artcyclopedia.com.*
Gradient camera?


![Diagram of the gradient camera system]

Figure 2. Log-gradient camera overview: intensity sensors organized into 4-pixel cliques share the same self-adjusting gain setting $k$, and send $\log(I_d)$ signals to A/D converter. Subtraction removes common-mode noise, and a linear ‘curl fix’ solver corrects saturated gradient values or ‘dead’ pixels, and a Poisson solver finds output values from gradients.
Poisson-ish mesh editing

- http://portal.acm.org/citation.cfm?id=1057432.1057456
- http://www.cad.zju.edu.cn/home/xudong/Projects/mesh_editing/main.htm
- http://people.csail.mit.edu/sumner/research/deftransfer/

Figure 1: An unknown mythical creature. Left: mesh components for merging and deformation (the arm). Right: final editing result.

Figure 1: Deformation transfer copies the deformations exhibited by a source mesh onto a different target mesh. In this example, deformations of the reference horse mesh are transferred to the reference camel, generating seven new camel poses. Both gross skeletal changes as well as more subtle skin deformations are successfully reproduced.
Deblurring
Sources of blur

- Spherical aberrations
- Axial chromatic aberrations
- Diffraction
- Camera shake
- Object movement
- Defocus
- Camera antialiasing filter
- Atmospheric turbulence
Simple model of blur: convolution

- **Limited**
  - Blur sometimes varies spatially
  - or depends on depth, object, etc.

- **but will often be true locally**
  - and insights usually generalize
Can we remove blur?

original

[1 1 1 1]

Blurred

original

Blurred

original
Problem statement

• We are given a blurry image $y$

• We know it came from the convolution of a sharp image $x$ with a known kernel $k$:

\[ y = x \otimes k \]

• We want to retrieve $x$

• More general (harder problem): blind convolution
  - We don’t know $k$
Convolution is linear!

• We can write it as a big matrix $y = Mx$
• Deconvolution: just solve $Mx = y$!

$y = Mx \quad x = M^{-1}y$

• Three important things:
  - we know the blur kernel perfectly
  - blur is perfect, there is no sensor noise
  - we assume the image encoding is linear
Solver?

- Deconvolution: just solve $Mx = y$!
- Use your favorite conjugate gradient
  - Replace the Laplacian by convolution by $k$
  - Replace $b$ by the input blurry image
  - Forget the mask business

- Poisson image editing was a deconvolution by the Laplacian (+boundary conditions)

- Again: Conjugate gradient is good for such sparse system
  - Doesn’t even need to form the matrix! Just needs the ability to apply it (i.e. to convolve)
Look at the code
Not fully perfect, but not bad

Gaussian blur

Deconvolve

small haloes
Testing the limits
Recall gamma

- Digital images are usually gamma encoded
- Don’t forget to decode before performing linear processing
Non-linear encoding

• Images are usually gamma-encoded
• What if you forget to linearize them?
• Yikes!
Sensor noise

- We usually don’t observe
  \[ y = x \otimes k \]

- Noise \( n \) gets added:
  \[ y = x \otimes k + n \]
Effect of noise

Blurry image
Gaussian, sigma=1

Noise*20

Deblurred
Effect of noise

\[ y = Mx + n \]

\[ \hat{x} = M^{-1}y \]

\[ \hat{x} = M^{-1}(Mx + n) \]

\[ \hat{x} = x + M^{-1}n \]

• Deconvolved noise gets added!
Deconvolving the noise

Noise\(^{20}\)

Deconvolved noise
What’s next?

• Let’s understand the properties of $M$ and $M^{-1}$ better

• Later, we’ll “regularize” the inversion to mitigate the effect of noise

• In some cases, we’ll modify the physical blur (e.g. by changing the lens design) to make sure that deblurring doesn’t introduce as much noise
Fourier
motivation
Studying convolutions

• Convolution is complicated
  – and hard to inverse
• But at least it’s linear
  \[(f+kg) \ast h = f \ast h + k (g \ast h)\]
• We want to find a better expression
  – Let’s study functions whose behavior is simple under convolution
Blurring: convolution

Same shape, just reduced contrast!!!
This is an eigenvector (output is the input multiplied by a constant)
Eigenvectors

• More precisely, the eigenvectors are $e^{i\omega x}$
• aka $\cos \omega x + i \sin \omega x$
Big Motivation for Fourier analysis

• Sine waves are eigenvectors of the convolution operator
• Convolution is diagonal in the Fourier domain
Big Motivation for Fourier analysis

• Sine waves are eigenvectors of the convolution operator

• Convolution is diagonal in the Fourier domain
  – And the diagonal is the Fourier transform of the kernel
  – Convolution theorem:

  Convolution in the primal is a multiplication in Fourier
Back to convolution/deconvolution

• Convolution in space is a multiplication in Fourier
• Note $y$ the observed blurry image and $x$ the original sharp one
• $y = g \otimes x$ in the spatial domain
• $Y = GX$ in the Fourier domain
  - A frequency does not depend on the other ones
Invert the convolution theorem

• Given \( y = g \otimes x \) and \( g \), we seek an estimate \( x' \) of \( x \).

• How do you invert a multiplication?
  - Division!

\[
X'(\omega) = \frac{Y(\omega)}{G(\omega)}
\]

• DECONVOLUTION IS A DIVISION IN THE FOURIER DOMAIN!

• Which means it is also a convolution in the spatial domain, by the inverse Fourier transform of \( 1/G \).
Questions?

- Given \( y = g \otimes x \) and \( g \), we seek an estimate \( x' \) of \( x \).
- How do you invert a multiplication?
  - Division!
- \( X'(\omega) = Y(\omega)/G(\omega) \)

- DECONVOLUTION IS A DIVISION IN THE FOURIER DOMAIN!
- Which means it is also a convolution in the spatial domain, by the inverse Fourier transform of \( 1/G \).
Potential problem?

• Deconvolution is a division in the Fourier domain

• Division by zero is bad!
  - Information is lost at the zeros of the kernel spectrum $G$
Noise problem

- Even when there is no zero, noise is a big problem
- If G has small number, division amplifies noise
- if \( y = g \ast x + v \) where v is additive noise
- \( Y = GX + V \)
- \( X' = (GX + V)/G = X + V/G \)
- V is amplified by 1/G. This is why you typically get more high-frequency noise with deconvolution
Noise problem

blurry, no noise  deconvolved

blurry, with noise  deconvolved

More Fourier Motivation
Other motivation for Fourier analysis: sampling

• The sampling grid is a periodic structure
  – Fourier is pretty good at handling that
  – We saw that a sine wave has serious problems with sampling

• Sampling is a linear process
  – but not shift-invariant
Sampling Density

- If we’re lucky, sampling density is enough.
Sampling Density

- If we insufficiently sample the signal, it may be mistaken for something simpler during reconstruction (that's aliasing!)
Sampling recap

- Sampling has bad properties when undersampling (aliasing)
- But they can be analyzed more easily for sine waves
Something like that, yeah! Nothing too complex! No bending spacetime, no freaking lead balls on some rubber sheet!

Nothing’s wrong with a bit of bidimensionality! A cool world, with natural laws you can understand at first sight!

Oh! Professor Hawking! Thanks for coming! Alas, I’m afraid you won’t have much to study in here!

Ha! Ha! Ha!

I wouldn’t say so, Boulet my friend! I’m studying pixelic relativity, this is fascinating!

Allow me to explain. I discovered that the pixel was relative: it depends of the observer’s point of view. I call that the «Relative Definition».

For example: at close range, an object will appear composed of hundreds of pixels...

... But from some distance, the same object will appear composed by only a few pixels.

Er... Uh?

The pixel is the smallest particle we know, but it contains a lot of information. It’s very strange. Take a look in this microscope, you will see the minimum resolution of the universe.

These creatures are the smallest we know. They are made with only a very few pixels, however they can have very elaborate behaviors. We think they hide a lot of hidden data, beyond their pixels: the famous «bits» whose existence we try to prove in giant pixel accelerators.

But calculations are impossible! At this scale, mathematical laws don’t apply any more! For example, in a square of side = x, we know that the diagonal equals x\sqrt{2}...

But at pixelic scale it doesn’t! The diagonal is made of THE SAME NUMBER OF PIXELS AS ITS SIDE!
Recap: motivation for sine waves

• Blurring sine waves is simple
  – You get the same sine wave, just scaled down
  – The sine functions are the eigenvectors of the convolution operator

• Sampling sine waves is interesting
  – Get another sine wave
  – Not necessarily the same one! (aliasing)

If we represent functions (or images) with a sum of sine waves, convolution and sampling are easy to study
Fourier as change of basis

- Shuffle the data to reveal other information
- E.g., take average & difference: matrix

\[
\begin{bmatrix}
0.5 & 1 \\
0.5 & -1
\end{bmatrix}
\]
Fourier as change of basis

• Same thing with infinite-dimensional vectors
Fourier transform visualization

\[ F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\pi i \left( \frac{km}{M} + \frac{ln}{N} \right)} \]
Fourier as a change of basis

- Discrete Fourier Transform: just a big matrix
- But a smart matrix!

http://www.reindeergraphics.com
To get some sense of what basis elements look like, we plot a basis element — or rather, its real part — as a function of $x,y$ for some fixed $u, v$. We get a function that is constant when $(ux + vy)$ is constant. The magnitude of the vector $(u, v)$ gives a frequency, and its direction gives an orientation. The function is a sinusoid with this frequency along the direction, and constant perpendicular to the direction.

$$e^{-\pi i (ux + vy)}$$

$$e^{\pi i (ux + vy)}$$
Here $u$ and $v$ are larger than in the previous slide.
And larger still...

\[ e^{-\pi i (ux + vy)} \]

\[ e^{\pi i (ux + vy)} \]
Question?
Other presentations of Fourier

• Start with Fourier series with periodic signal
• Heat equation
  – more or less special case of convolution
  – iterate -> exponential on eigenvalues
Motivations

• Insights & mathematical beauty
• Sampling rate and filtering bandwidth
• Computation bases
  – FFT: faster convolution
  – E.g. finite elements, fast filtering, heat equation, vibration modes
• Optics: wave nature of light & diffraction
Questions?
The Fourier Transform

- Defined for infinite, aperiodic signals
- Derived from the Fourier series by “extending the period of the signal to infinity”
- The Fourier transform is defined as
  \[ X(\omega) = \frac{1}{\sqrt{2\pi}} \int x(t)e^{-j\omega t} \, dt \]
- \( X(\omega) \) is called the spectrum of \( x(t) \)
- It contains the magnitude and phase of each complex exponential of frequency \( \omega \) in \( x(t) \)
The Fourier Transform

• The inverse Fourier transform is defined as

\[ x(t) = \frac{1}{\sqrt{2\pi}} \int X(\omega)e^{j\omega t} d\omega \]

• Fourier transform pair

\[ x(t) \xrightarrow{F} X(\omega) \]

• \( x(t) \) is called the *spatial domain* representation

• \( X(\omega) \) is called the *frequency domain* representation
Beware of differences

• Different definitions of Fourier transform
• We use
  \[ X(\omega) = \frac{1}{\sqrt{2\pi}} \int x(t) e^{-j\omega t} \, dt \]
• Other people might exclude normalization or include \(2\pi\) in the frequency
• \(X\) might take \(\omega\) or \(j\omega\) as argument
• Physicist use \(j\), mathematicians use \(i\)
Phase

- Don’t forget the phase! Fourier transform results in complex numbers

- Can be seen as sum of sines and cosines

- Or modulus/phase
Phase is important!
Phase is important!

Figure 6.2 (a) The image shown in Figure 1.4; (b) magnitude of the two-dimensional Fourier transform of (a); (c) phase of the Fourier transform of (a); (d) picture whose Fourier transform has magnitude as in (b) and phase equal to zero; (e) picture whose Fourier transform has magnitude equal to 1 and phase as in (c); (f) picture whose Fourier transform has phase as in (c) and magnitude equal to that of the transform of the picture shown in (g).
Questions?
Duality

Up to details (such as factors of $2\pi$ or signs):
if function $a$ is the Fourier transform of $b$, then $b$ is the Fourier transform of $a$.
For example, the Fourier transform of a box is a sinc,
and the Fourier transform of a sinc is a box.
Duality

Any theorem that involves the primal and Fourier domains is also true when swapping the two domains.

e.g. shift theorem:

Primal

\[ f(x+a) \]

Fourier

\[ e^{-2\pi ia\omega} F(\omega) \]

\[ e^{-2\pi ax} f(x) \]

\[ F(\omega+a) \]
Duality

Any theorem that involves the primal and Fourier domains is also true when swapping the two domains.

e.g. scaling theorem:

Primal

\[ f(ax) \]

Fourier

\[ \frac{1}{a} F\left(\frac{x}{a}\right) \]

\[ \frac{1}{a} f\left(\frac{x}{a}\right) \]

\[ F(\omega a) \]
Convolution/Modulation

A convolution in one domain is a multiplication in the other one.

Recall that Fourier bases are eigenvectors of the convolution.

Primal: \( f \otimes g \)  

Fourier: \( F \otimes G \)
Questions?
Low pass

black means 1, white means 0

http://www.reindeergraphics.com
Filtering in Fourier domain
Analysis of our simple filters

Pixel offset

original

Filtered (no change)

spectrum: \( F(\omega) = 1 \)
(yes, I am now using the definition without \( 1/\sqrt{2\pi} \))
Analysis of our simple filters

spectrum:

\[ F(\omega) = e^{-2\pi j \omega \delta} \]
Analysis of our simple filters

Pixel offset

0.3

0

Low-pass filter

F(\omega) = \text{sinc}(\omega) = \frac{\sin(\omega)}{\omega}
Analysis of our simple filters

original

2.0

0.33

sharpened

spectrum:
\[ F(\omega) = 2 - \text{sinc}(\omega) \]
Questions?
Fast convolution with FFT

• Convolution cost in primal: $O(K^2N^2)$
  – where $K$ is the width of kernel and $N$ the width of image

• Convolution cost in Fourier: $O(N^2)$
  – because it is diagonal (simple multiplication)

• This suggests it might be valuable to perform convolution in the Fourier domain using an FFT
  – But we need to take into account the FFT’s cost
Convolution with FFT

- Perform Fourier transform of image & kernel
- Multiply Fourier coefficients
- Perform inverse Fourier transform of result
Convolution in primal versus FFT

- 2-d FFT: $O(N^2 \log N)$
  - where $N$ is number of pixels on a side
- Convolution: $K^2 N^2$
  - where $K^2$ is number of samples in kernel
- Say $N=2^{10}$, $K^2=100$.
  FFT $\sim 3 \times 20 \ 2^{20}$, while convolution $\sim 100 \ 2^{20}$
- Rule of thumb: if kernel is smaller than 10-20 pixels, FFT is not worth the extra headache
  - But good for larger kernels
Recall: Words of wisdom

• Careful with the FFT: it assumes a cyclic signal
• Oftentimes, the answer you get mostly shows wraparound artifacts
• For convolution: various options such as mirroring the image or duplicating the boundary values
• For analysis: Proper windowing is needed
  – e.g. multiply function by a smooth function that falls off away from the center so that the boundary is zero
Questions?
Sampling and aliasing
In photos too
More on Samples

• In signal processing, the process of mapping a continuous function to a discrete one is called *sampling*

• The process of mapping a continuous variable to a discrete one is called *quantization*

• To represent or render an image using a computer, we must both sample and quantize
  – Now we focus on the effects of sampling and how to fight them
Sampling in the Frequency Domain

original signal

sampling grid

sampled signal

Fourier Transform

(convolution)

(multiplication)
Reconstruction

• If we can extract a copy of the original signal from the frequency domain of the sampled signal, we can reconstruct the original signal!

• But there may be overlap between the copies.
Guaranteeing Proper Reconstruction

• Separate by removing high frequencies from the original signal (low pass pre-filtering)

• Separate by increasing the sampling density

• If we can't separate the copies, we will have overlapping frequency spectrum during reconstruction → aliasing.
Sampling Theorem

• When sampling a signal at discrete intervals, the sampling frequency must be greater than twice the highest frequency of the input signal in order to be able to reconstruct the original perfectly from the sampled version (Shannon, Nyquist, Whittaker, Kotelnikov, Küpfmüller)
Question?