Introduction to Fourier methods

- Very brief introduction. We don't have time to go through the math!
- Fourier methods give insight into image processing
- Provides a principled way to think about reversing the effect of a convolution (e.g., deblurring).
- Provides a way to speed up convolution (depending on the work flow).

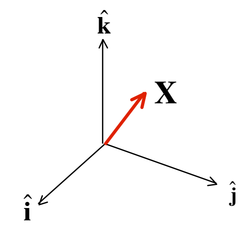
Change of Basis

Recall that $\mathbf{X}=(x_i,x_j,x_k)$ means that

$$\mathbf{X} = x_i\hat{\mathbf{i}} + x_j\hat{\mathbf{j}} + x_k\hat{\mathbf{k}}$$

Note that

$$x_i = \mathbf{X}\bullet\hat{\mathbf{i}} \qquad x_j = \mathbf{X}\bullet\hat{\mathbf{j}} \qquad x_k = \mathbf{X}\bullet\hat{\mathbf{k}}$$



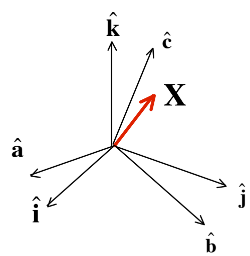
Change of Basis

Suppose that we want to express **X** relative to a different basis.

$$\mathbf{X} = x_a\hat{\mathbf{a}} + x_b\hat{\mathbf{b}} + x_c\hat{\mathbf{c}}$$

Again,

$$x_a = \mathbf{X}\bullet\hat{\mathbf{a}} \qquad x_b = \mathbf{X}\bullet\hat{\mathbf{b}} \qquad x_c = \mathbf{X}\bullet\hat{\mathbf{c}}$$



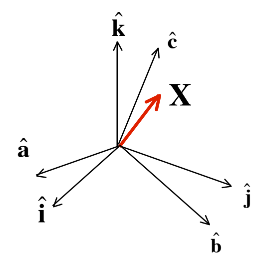
Change of Basis

$$\begin{aligned} x_a &= \mathbf{X}\bullet\hat{\mathbf{a}} \\ &= x_i\hat{\mathbf{i}}\bullet\hat{\mathbf{a}} + x_j\hat{\mathbf{j}}\bullet\hat{\mathbf{a}} + x_k\hat{\mathbf{k}}\bullet\hat{\mathbf{a}} \\ &= (x_i,\ x_j\ \ x_k)\bullet(\hat{\mathbf{i}}\bullet\hat{\mathbf{a}},\ \hat{\mathbf{j}}\bullet\hat{\mathbf{a}},\ \hat{\mathbf{k}}\bullet\hat{\mathbf{a}}) \end{aligned}$$

(Similarly for b and c)

So, the change of basis can be done by a matrix multiplication

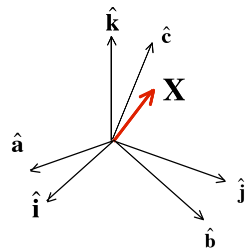
$$\begin{pmatrix} x_a \\ x_b \\ x_c \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{i}}\bullet\hat{\mathbf{a}} & \hat{\mathbf{j}}\bullet\hat{\mathbf{a}} & \hat{\mathbf{k}}\bullet\hat{\mathbf{a}} \\ \hat{\mathbf{i}}\bullet\hat{\mathbf{b}} & \hat{\mathbf{j}}\bullet\hat{\mathbf{b}} & \hat{\mathbf{k}}\bullet\hat{\mathbf{b}} \\ \hat{\mathbf{i}}\bullet\hat{\mathbf{c}} & \hat{\mathbf{j}}\bullet\hat{\mathbf{c}} & \hat{\mathbf{k}}\bullet\hat{\mathbf{c}} \end{pmatrix} \begin{pmatrix} x_i \\ x_j \\ x_k \end{pmatrix}$$



Change of Basis

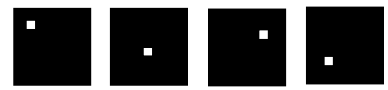
Main points for what follows

- We can express a vector with respect to many basis
- We get the weights (coordinates) by dot products of the vector with the basis vectors



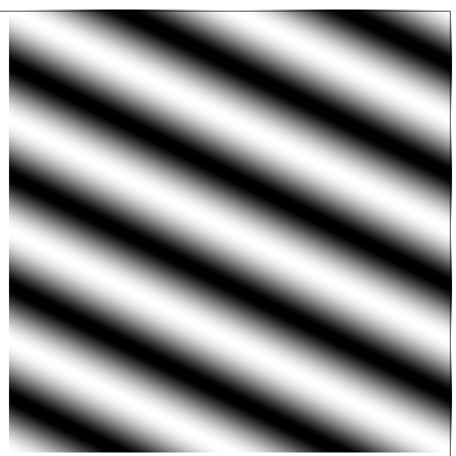
Bases for Images

- Represent function (image) with respect to a new basis
  - Think of functions (images) as vectors with many components
  - This means that they are a weighted sum (linear combination) of basis vectors
  - We can represent the same entity as a linear combination over sets of different basis vectors
  - In canonical/usual form the basis vectors are box(i,j) (discrete) or delta functions (continous).

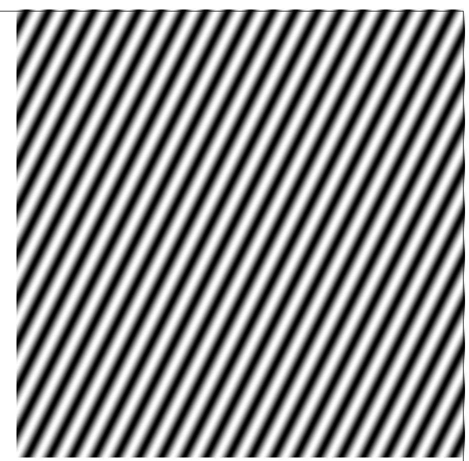


  - In Fourier analysis, the basis vectors are **sinusoids**

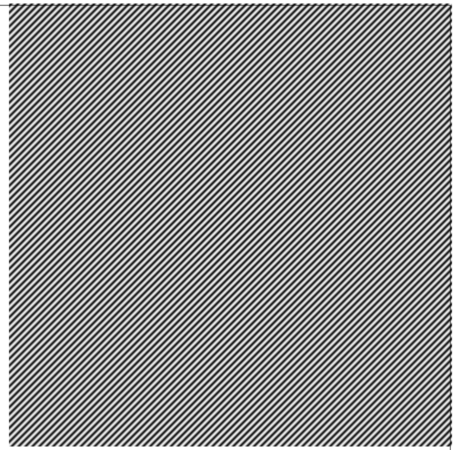
Example 2D Fourier basis function



Another example

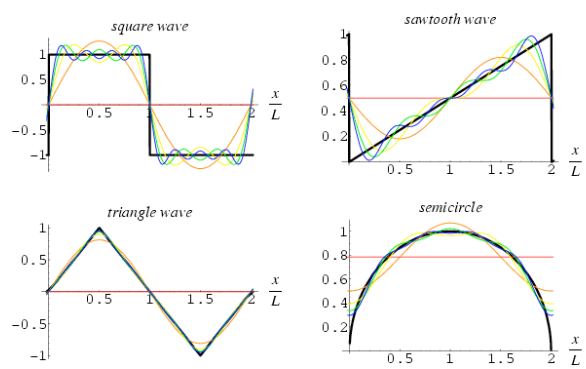


Yet another



Introduction to Fourier methods

- A periodic function (vector) can be decomposed into a sum of sines and cosines
- Sines and cosines are **orthogonal**
- This forms a new basis for the function (vector)



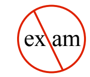
http://mathworld.wolfram.com/FourierSeries.html

Introduction to Fourier methods

- Because the basis functions (sines/cosines) are orthogonal, we find their coefficients by integrating against them (or, in the discrete case, taking dot products)---recall the **change of basis** review
- A discrete signal (e.g. image) is “band limited”-->frequences higher than 1/2 cycle per pixel are lost
- Sampling theorem: We can reconstruct a “band limited” signal from a limited number of samples
  - This is why adding even more bits to the digital representation of music does not help--you can only hear up to certain frequency; sampling more than that rate does not do any good.

The 2D Fourier Transform

- Need both sines and cosines (in the general case)
- In 1D the frequency (a single number) tells us which sine (or cosine)
- In 2D we have frequency and orientation (period and direction)
- Encode these with a pair of numbers, (u,v)



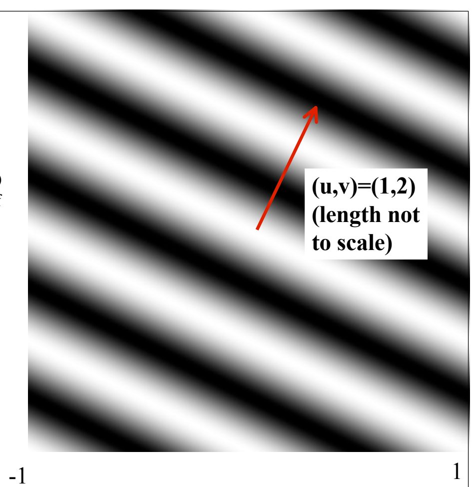
The 2D Fourier Transform

- We use complex numbers for convenient representation
- Recall that $e^{-i\theta} = \cos(\theta) + i\sin(\theta)$
- And the basis function

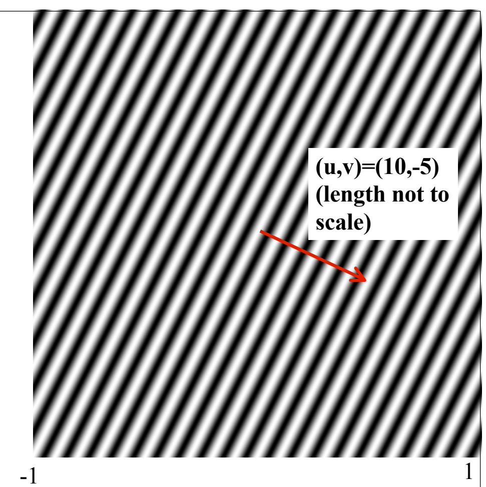
$$e^{-i2\pi(ux+vy)} = \cos(2\pi(ux+vy)) + i\sin(2\pi(ux+vy))$$

- (u,v) gives the frequency and orientation of the sinusoids

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.



Another example



The 2D Fourier Transform

- The Fourier transform expresses the image in the sinusoidal basis denoted by:

$$e^{-i2\pi(ux+vy)}$$

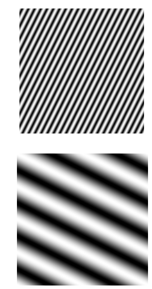
- To get the weights (coefficients) we integrate (continuous case) or take dot products (discrete case)
- The transform (continuous case) is given by

$$F(g(x,y))(u,v) = \iint_{\mathbb{R}^2} g(x,y) e^{-i2\pi(ux+vy)} dxdy$$

The Fourier Transform

- Have both cosines (gives real part) and sines (imaginary part)
- Recall that for an even (symmetric) function f(-x)=f(x), and for an odd (anti-symmetric) one f(-x)=-f(x)
- Sine gives odd part of function, cosine even part
- If the function is even there are only cosine terms, and the result is real (cosine transform)

Example bases with different (u,v)



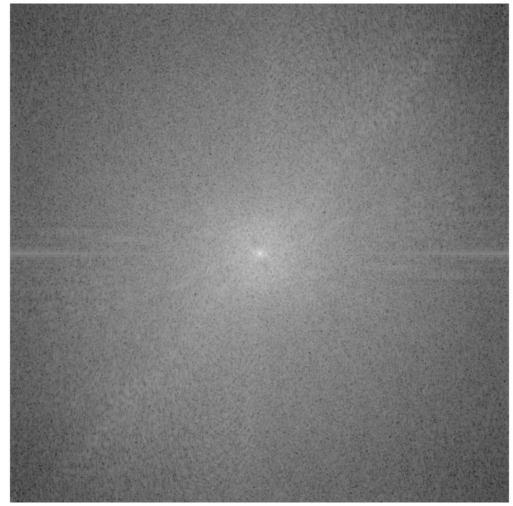
Phase and Magnitude

- Fourier transform of a real function is complex valued
  - transform of image is becomes two images (real and imaginary part)
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- z = a + bi
  - Phase angle: theta=arctan(b/a)
  - Magnitude: |z|=sqrt(a^2 + b^2)
- Magnitude combines both cosine (real) and sine (imaginary) terms
  - Large magnitude means large energy for that (u,v)
- Phase is the relation between with cosine and sine terms

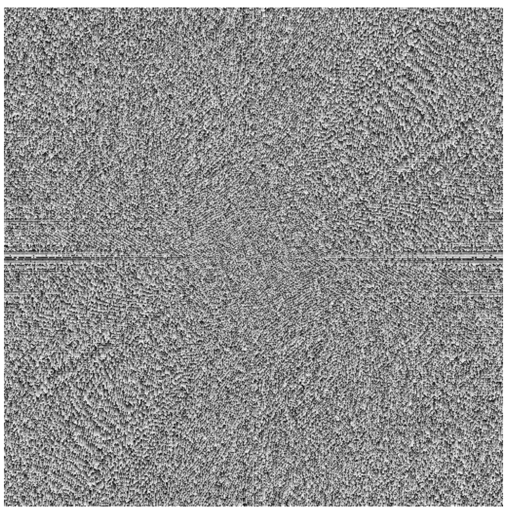
Phase and Magnitude

- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn't
- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?

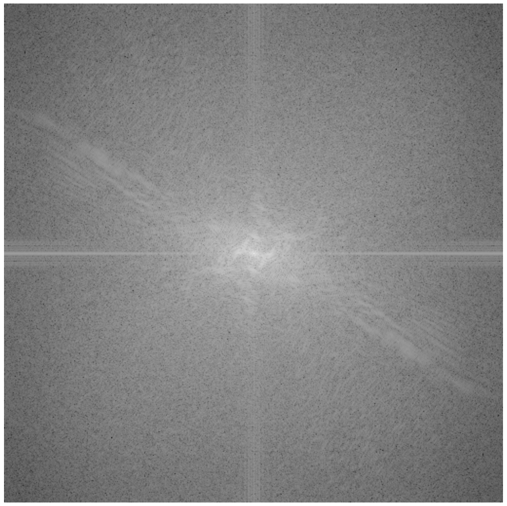
This is the magnitude transform of the cheetah pic



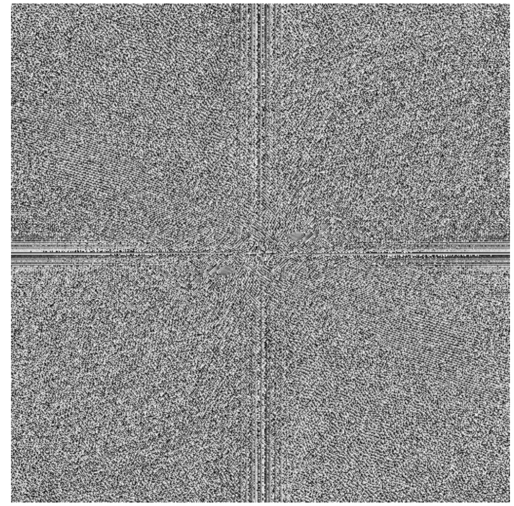
This is the phase transform of the cheetah pic



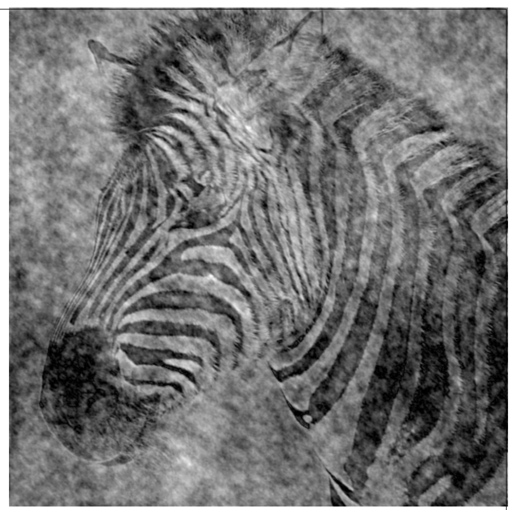
This is the magnitude transform of the zebra pic



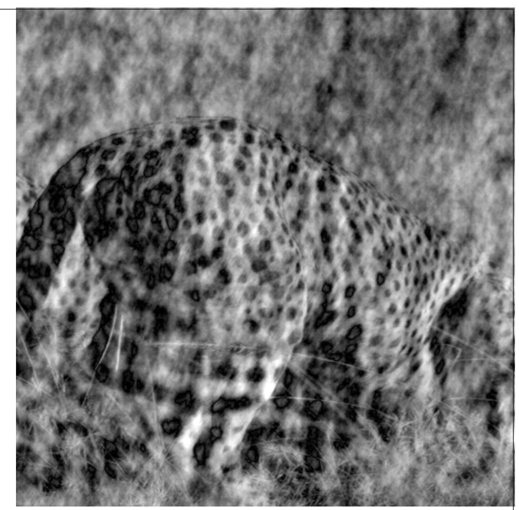
This is the phase transform of the zebra pic



Reconstruction with zebra phase, cheetah magnitude



Reconstruction with cheetah phase, zebra magnitude



Fourier Transform (continued)

- Important facts
  - The Fourier transform is linear
  - There is an inverse FT
- Important observation
  - The Fourier transform is global--the value for each (u,v) is a function of the **entire** image.
  - (This is why it is difficult to visualize/understand)
- Relationship to noise and smoothing
  - Noise is generally high frequency
  - Smoothing strategy
    - Take FT
    - Threshold higher frequency
    - Invert

The Convolution Theorem

- Important result which can have practical impact (convolution theorem)

$$F(a \otimes b) = F(a)F(b)$$

- (Depending on your workflow, using the DFT for convolution can save time).
- A strategy for inverting the effect of a convolution

$$a = F^{-1}\big(F(a)\big) = F^{-1}\left(\frac{F(a \otimes b)}{F(b)}\right)$$

Fourier Transform (practice)

- Because of the convolution theorem, the FT gives a convenient way to invert the effect of convolution.
  - For example, often blurring can be modeled as a convolution, and the FT gives a convenient way to think about de-blurring.
- Fast (O(n*log(n)) methods exist to compute discrete version of Fourier transform (DFT2 in Matlab, IDFT2 for the inverse).
- If we assume that the image is periodic and symmetric then only the cosine terms count and we can avoid imaginary components which can speed up and simplify some tasks (cosine transform; DCT2 in Matlab, IDCT2 for the inverse).