LAST TIME

- Convolutions
  - Simplest spatial linear operation
  - Output at each pixel is a function of a limited number of pixels in the input
  - Linear Function
  - Same function for different neighborhoods

- Edge & Line Detection: A Stereotypical Vision Algorithm Pipeline
  - Use convolutions to detect local image properties (Gradients)
  - Apply local non-linear processing to get local features (Edges)
  - Aggregate information to find long-range structures (Lines)

OFFICE HOURS

- This Friday (and this Friday only):
  - Zhihao’s Office Hours in Jolley 431 instead of 309.
- Monday Office Hours:
  - 5:30-6:30pm, Collaboration Space @ Jolley 217.

- PSET 0 Due Today by 11:59pm
  - Any issues with submissions, post on Piazza.

OTHER NEIGHBORHOOD OPERATIONS

Median Filter / Order Statistics

\[ Y[n] = \text{Median}\{X[n - n'], N \}\]  \( n' \in \{0,1\} \)

- Neighborhood function \( N[n'] \in \{0,1\} \)
- Often better at removing outliers than convolution.

- Other ops: \( Y[n] = \max / \min\{X[n - n'], N[n']\} \)  \( N[n'] \geq 0 \)
OTHER NEIGHBORHOOD OPERATIONS

Morphological Operations

- Conducted on binary images \( X[n] \in \{0, 1\} \)
- Erosion: \( Y[n] = \text{AND} \ (X[n-n'] |_{n'=1}^M) = 1 \) if all neighbors 1
- Dilation: \( Y[n] = \text{OR} \ (X[n-n'] |_{n'=1}^M) = 1 \) if any neighbor 1
- Opening: Erosion followed by Dilation
- Closing: Dilation followed by Erosion

See Szeliski Sec 3.3.2

Figure 3.32: Binary image morphology: (a) original image; (b) dilation; (c) erosion; (d) majority; (e) opening; (f) closing. The structuring element for all examples is a 3 x 3 square.

See Szeliski Sec 3.3.2

BILATERAL FILTERING

Denoising by Smoothing (with a Gaussian filter):

\[
X \quad Y = X \ast G
\]

\[
G[n_1, n_2] = G[n_1 - n_2] \propto \exp \left( -\frac{|n_1 - n_2|^2}{2\sigma^2} \right)
\]

\[
\sum_{n_2} G[n_1, n_2] = 1
\]

BILATERAL FILTERING

Denoising by Smoothing (with a Gaussian filter):

\[
X \quad Y = X \ast G
\]

\[
B[n_1, n_2] \propto \exp \left( -\frac{|n_1 - n_2|^2}{2\sigma^2} - \frac{|X[n_1] - X[n_2]|^2}{2\sigma^2} \right)
\]

\[
\sum_{n_2} B[n_1, n_2] = 1
\]

Make the filter weights data dependent!
**BILATERAL FILTERING**

Denoising by Smoothing (with a Gaussian filter):

\[ B[n_1, n_2] \propto \exp \left( -\frac{|n_1 - n_2|^2}{2\sigma_x^2} - \frac{|X[n_1] - X[n_2]|^2}{2\sigma_f^2} \right) \]

\[ \sum_{n_2} B[n_1, n_2] = 1 \]

**BILATERAL FILTERING**

Denoising with a Bilateral Filter

\[ B[n_1, n_2] \propto \exp \left( -\frac{|n_1 - n_2|^2}{2\sigma_x^2} - \frac{|X[n_1] - X[n_2]|^2}{2\sigma_f^2} \right) \]

\[ \sum_{n_2} B[n_1, n_2] = 1 \]

**BILATERAL FILTERING**

Denoising with a Bilateral Filter

\[ B[n_1, n_2] \propto \exp \left( -\frac{|n_1 - n_2|^2}{2\sigma_x^2} - \frac{|X[n_1] - X[n_2]|^2}{2\sigma_f^2} \right) \]

\[ \sum_{n_2} B[n_1, n_2] = 1 \]

**BILATERAL FILTERING**

\[ B[n_1, n_2] \propto \exp \left( -\frac{|n_1 - n_2|^2}{2\sigma_x^2} - \frac{|X[n_1] - X[n_2]|^2}{2\sigma_f^2} \right) \]

\[ \sum_{n_2} B[n_1, n_2] = 1 \]
BILATERAL FILTERING

- Guided Bilateral Filter: \( B[n_1, n_2] \) based on a separate image \( Z[n] \): depth, infra-red, etc.
- Far less efficient than convolution
  - Filter also has to be computed, normalized, at each output location.
  - Efficient Datastructures Possible
- Further Reading:
  - Paris et al., SIGGRAPH/CVPR Course on Bilateral Filtering
  - Recent work on using this for inference, best paper runner up at ECCV 2016
  - Barron & Poole, The Fast Bilateral Solver, ECCV 2016

BILATERAL FILTERING

\[
B[n_1, n_2] \propto \exp \left( -\frac{|n_1 - n_2|^2}{2\sigma^2} - \frac{|X[n_1] - X[n_2]|^2}{2\sigma_f^2} \right)
\]

Gaussian Filter Result

FOURIER TRANSFORM

Quick Recap: Complex Numbers

- A complex number \( f = x + jy \) where \( x \) and \( y \) are scalar numbers.
  - \( j = \sqrt{-1} \) (EE convention; we use \( j \) instead of \( i \))
  - \( x \) and \( y \) are called the real and imaginary components of \( f \)

Think of \( f \) as a 2-D vector with special definitions of addition, multiplication, etc.

- \( (x_1 + jy_1) + (x_2 + jy_2) = (x_1 + x_2) + j(y_1 + y_2) \)
- \( (x_1 + jy_1) \times (x_2 + jy_2) = (x_1x_2 - y_1y_2) + j(x_2y_1 + x_1y_2) \)
- \( (x_1 + jy_1) \times x_2 = x_1x_2 + jy_1x_2 \)
- Conjugate: \( x + jy = x - jy = x + j(-y) \)
- Magnitude: \( (x + jy) \times (x + jy) = x^2 + y^2 \)
Quick Recap: Complex Numbers

Euler’s Formula
\[ \exp(i\theta) = \cos \theta + j \sin \theta \]
\[ x + jy = M \exp(i\theta) \]
\[ M = \sqrt{x^2 + y^2}, \theta = \tan^{-1}(y, x) \]
\[ \theta \text{ is called the “phase”} \]
\[ M \exp(i\theta) = M \exp(-i\theta) \]
\[ (x + jy) \times \exp(i\theta_0) = M \exp(i(\theta + \theta_0)) \]
\[ \text{Preserves magnitude, adds to phase} \]
\[ \exp(i0) = 1 \]
\[ \exp(iN\pi) = 1 \text{ where } N \text{ is an even integer, and } = -1 \text{ where } N \text{ is an odd integer.} \]
\[ \text{Real in both cases} \]

The Discrete 2D Fourier Transform
\[ F[X] = F[u, v] = \frac{1}{WH} \sum_{n_x=0}^{W-1} \sum_{n_y=0}^{H-1} X[n_x, n_y] \exp\left(-j 2\pi \left(\frac{u n_x}{W} + \frac{v n_y}{H}\right)\right) \]
\[ \exp(j\theta) = \cos \theta + j \sin \theta \]

\[ \exp\left(-j 2\pi \left(\frac{(u + W) n_x}{W} + \frac{v n_y}{H}\right)\right) = \exp\left(-j 2\pi \left(\frac{u n_x}{W} + \frac{v n_y}{H}\right)\right) \]
\[ = \exp\left(-j 2\pi \left(\frac{u n_x}{W} + \frac{v n_y}{H}\right)\right) \exp\left(-j 2\pi \frac{2n_x}{W}\right) \]
\[ = \exp\left(-j 2\pi \left(\frac{u n_x}{W} + \frac{v n_y}{H}\right)\right) \]

Can be implemented fairly efficiently using the FFT algorithm: \( O(n \log n) \)
(Often, FFT is used to refer to the operation itself.)
The Discrete 2D Fourier Transform Pair

\[ F[X] = F[u, v] = \frac{1}{WH} \sum_{n_u=0}^{W-1} \sum_{n_v=0}^{H-1} X[n_u, n_v] \exp \left( -j 2\pi \left( \frac{u n_u}{W} + \frac{v n_v}{H} \right) \right) \]

\[ F^{-1}[F] = X[n_u, n_v] = \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} F[u, v] \exp \left( j 2\pi \left( \frac{u n_u}{W} + \frac{v n_v}{H} \right) \right) \]

- If \( X \) is real-valued, \( F[-u, -v] = \bar{F}[u, v] \) where \( \bar{F} \) implies complex conjugate.
- \( F[0, 0] \) is often called the DC component. It is the average intensity of \( X \). It is real if \( X \) is real.
- Only \( WH \) independent *numbers* in \( F[u, v] \) (counting real and imaginary separately) if \( X \) is real.
- Parseval’s Theorem: (energy preserving up to constant factor)
  \[ \sum_{u,v} ||F[u,v]||^2 = \sum_{u,v} F[u,v] \bar{F}[u,v] = \frac{1}{WH} \sum_{n_u,n_v} ||X[n_u,n_v]||^2 \]

DFT as a Co-ordinate Transform

\[ F[u, v] = \frac{1}{\sqrt{WH}} \sum_{n_u=0}^{W-1} \sum_{n_v=0}^{H-1} S_{uv}[n_u, n_v] X[n_u, n_v] \]

where each \( S_{uv} \) can be thought of as a different (complex-valued) image:

\[ S_{uv}[n_u, n_v] = \frac{1}{\sqrt{WH}} \exp \left( j 2\pi \left( \frac{u n_u}{W} + \frac{v n_v}{H} \right) \right) \]

For \( x, y \in \mathbb{C^n} \), \( (x, y) = x^* y \)
- \( x^* \) is the Hermitian of \( x \)
  - Transpose + Conjugate (transpose the vector, and take conjugate of each entry)
- \( x^* y = \sum_{i} \bar{x}_i y_i \)
DFT as a Co-ordinate Transform

\[ F[u, v] = \frac{1}{\sqrt{WH}} \langle S_{uv}, X \rangle \]

where each \( S_{uv} \) can be thought of as a different (complex-valued) image:

\[ S_{uv}[n_x, n_y] = \frac{1}{\sqrt{WH}} \exp\left(j 2\pi \left( \frac{u n_x}{W} + \frac{v n_y}{H} \right) \right) \]

\( F[u, v] \) is the inner-product between \( X \) and \( S_{uv} \) (scaled by \( \sqrt{WH} \)).

\[ S_{uv} = \text{image} \]
FOURIER TRANSFORM

DFT as a Co-ordinate Transform

\[ F[u, v] = \frac{1}{\sqrt{WH}} \langle S_{uv}, X \rangle, \]

\[ X = \sqrt{WH} \sum_{v=0}^{H-1} \sum_{u=0}^{W-1} F[u, v] S_{uv} \]

\[ \langle S_{uv}, S_{u'v'} \rangle = 1 \text{ if } u' = u \text{ and } v' = v, \text{ and 0 otherwise.} \]

\[ S \] is a \( WH \times WH \) matrix with each column a different \( S_{uv} \).

So, \( SS^* = S^*S = I \Rightarrow S^{-1} = S^* \).

- This means \( S \) is a unitary matrix.
- Multiplication by \( S \) is a co-ordinate transform:
  - \( X \) are the co-ordinates of a point in a \( WH \) dimensional space.
  - Multiplication by \( S^* \) changes the 'co-ordinate system'.
  - In the new co-ordinate system, each 'dimension' now corresponds to frequency rather than location.
  - \( S \) is a length-preserving matrix (\( ||S^*X||^2 = ||X||^2 \)).
  - It does rotations or reflections (in \( WH \) dimensional space).

Reconstruct with only these frequency components.
FOURIER TRANSFORM

$X$  $|F|^2$  $\angle F$

Reconstruct with only these frequency components

FOURIER TRANSFORM

$X$  $|F|^2$  $\angle F$

Reconstruct with only these frequency components

FOURIER TRANSFORM

$X$  $|F|^2$  $\angle F$

Reconstruct with only these frequency components

FOURIER TRANSFORM

A  Magnitude A  Phase B

Location of edges / structure, defined by phase more than magnitude.

B  Magnitude B  Phase A
**CONVOLUTION THEOREM**

Convolutions in “matrix” form:

\[
Y[n_x, n_y] = \sum_{n'_x} \sum_{n'_y} k[n'_x, n'_y] X[n_x - n'_x, n_y - n'_y]
\]

**CONVOLUTION THEOREM**

\[
Y = X \ast k \Rightarrow Y = A_k X
\]

**CONVOLUTION THEOREM**

\[
Y = X \ast k \Rightarrow Y = A_k X
\]

\[
A_k \text{ is not square for valid / long convolution.}
\]

**Question:**
Let \( Y = A_k X \) correspond to \( Y = X \ast k \). Now, let \( X' = A_k^T Y \). How is \( X' \) related to \( Y \) by convolution?

What operation does \( A_k^T \) represent?

A: Full convolution with \( k[-n_x, -n_y] \) (flipped version of \( k \))

**Why does this happen?**

- \( X = \sqrt{WH} \sum_{u,v} F[u,v] S_{uv} \)
- \( Y = X \ast k = \sqrt{WH} \sum_{u,v} F[u,v] S_{uv} \ast k \) (by linearity / distributivity)
- \( (S_{uv} \ast k)[n] = \sum_{n'} k[n'] S_{uv}[n - n'] \)
- \( S_{uv}[n - n'] \), assuming circular padding, is also a sinusoid with the same frequency \( (u, v) \) and magnitude, but different phase.
- Multiplying by \( k[n'] \) changes the magnitude, but frequency still the same.
- Adding different sinusoids of the same frequency gives you another sinusoid of the same frequency.

\[
(S_{uv} \ast k)[n, n_i] = d_{uv,k} S_{uv}[n, n_i], \text{ where } d_{uv,k} \text{ is some complex scalar.}
\]

**Sinusoids are eigen-functions of convolution**

\[
Y = X \ast k = \sqrt{WH} \sum_{u,v} F[u,v] S_{uv} \ast k = \sqrt{WH} \sum_{u,v} (F[u,v] d_{uv,k}) S_{uv}
\]
CONVOLUTION THEOREM

\[ A_k = S D_k S^* \]

- What's more, the diagonal elements of \( D_k \) are the \((W_y \times W_x)\) Fourier transform of \( k \).

\[ D_k = \text{diag} \left( \frac{1}{\sqrt{WH}} S^* k \right) \]

- This is the convolution theorem.
  - Computational advantage for performing (and inverting!) convolution, albeit under circular padding.
  - Good way of analyzing what a kernel is doing by looking at its Fourier transform.
- Why did we use complex numbers? Like quaternions in Graphics, for convenience!
  - If we used real number co-ordinate transform, convolution would convert to several \( 2 \times 2 \) transforms on pairs of co-ordinates.
  - Complex numbers are just a way of grouping these pairs into a single 'number'.

CONVOLUTION THEOREM

Doing Convolutions in the Fourier Domain:
- DFT, Point-wise multiply with FT of kernel, Inverse DFT
- Need to keep in mind some padding / size issues.

Kernel / Fourier Transform (magnitude) Pairs

1. Zero-pad
2. Circularly shift to center at \((0,0)\)

Gaussian Kernels: Low Pass (attenuate higher frequencies)
Larger spatial support: smaller Fourier support.

For more indepth coverage:
Szeliski Sec 3.4