CSE 559A: Computer Vision

Fall 2018: T-R: 11:30-1pm @ Lopata 101

Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Course Staff: Zhihao Xia, Charlie Wu, Han Liu

http://www.cse.wustl.edu/~ayan/courses/cse559a/

Sep 6, 2018
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.
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)
OTHER NEIGHBORHOOD OPERATIONS

Median Filter / Order Statistics

\[ Y[n] = \text{Median}\{X[n - n']\}_{N[n'] = 1} \]

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

Other ops: \( Y[n] = \max / \min\{X[n - n']\}_{N[n']>0} \)

Morphological Operations

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

See Szeliski Sec 3.3.2

Figure 3.21  Binary image morphology: (a) original image; (b) dilation; (c) erosion; (d) majority; (e) opening; (f) closing. The structuring element for all examples is a $5 \times 5$ square. The effects of majority are a subtle rounding of sharp corners. Opening fails to eliminate the dot, since it is not wide enough.
Denoising by Smoothing (with a Gaussian filter):

\[ X \]

\[ Y = X \ast G \]

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

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

\[ Y[n] = \sum_{n'} G[n']X[n - n'] \]

\[ Y[n_1] = \sum_{n_2} G'[n_1, n_2]X[n_2] \]
Denoising by Smoothing (with a Gaussian filter):

\[ X \]

\[ Y = X * G \]

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

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

Denoising by Smoothing (with a Gaussian filter):

\[
X
\]

Make the filter weights data dependent!

\[
B[n_1, n_2] \propto \exp \left( -\frac{|n_1 - n_2|^2}{2\sigma^2} - \frac{|X[n_1] - X[n_2]|^2}{2\sigma_1^2} \right)
\]

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

Denoising by Smoothing (with a Gaussian filter):

\[ X \]

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

\[ \sum_{n_2} B[n_1, n_2] = 1 \]
Denoising with a Bilateral Filter

\[ B[n_1, n_2] \propto \exp \left( -\frac{|n_1 - n_2|^2}{2\sigma^2} - \frac{|X[n_1] - X[n_2]|^2}{2\sigma_1^2} \right) \]
BILATERAL FILTERING

Denoising with a Bilateral Filter

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

\( \sigma_I \) High

Gaussian Filter Result
Denoising with a Bilateral Filter

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

\( \sigma_I \) Medium

Gaussian Filter Result
Denoising with a Bilateral Filter

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

Denoising with a Bilateral Filter

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

\(\sigma_I\) Low Repeated

Gaussian Filter Result
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
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

- \( \exp(j\theta) = \cos \theta + j \sin \theta \)
- \( x + jy = M \exp(j\theta) \)
  - \( M = \sqrt{x^2 + y^2}, \theta = \tan^{-1}(y, x) \)
  - \( \theta \) is called the "phase"
- \( M \exp(j\theta) = M \exp(-j\theta) \)
- \((x + jy) \times \exp(j\theta_0) = M \exp(j(\theta + \theta_0)) \)
  - Preserves magnitude, adds to phase
- \( \exp(j0) = 1 \)
- \( \exp(jN\pi) = 1 \) where \( N \) is an even integer, and \( = -1 \) where \( N \) is an odd integer.
  - Real in both cases
The Discrete 2D Fourier Transform

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

\[
\exp(j \theta) = \cos \theta + j \sin \theta
\]

- Defined for a single-channel / grayscale image \( X \).
- \( F \) is a "complex valued" array indexed by integers \( u, v \).
- Each \( F[u, v] \) depends on the intensities at all pixels.
The Discrete 2D Fourier Transform

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

\[
\exp(j \theta) = \cos \theta + j \sin \theta
\]


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

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

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

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

\[ \exp(j \theta) = \cos \theta + j \sin \theta \]

- Therefore, we typically store \( F[u,v] \) for \( u \in \{0, \ldots, W-1\}, v \in \{0, \ldots, H-1\} \).
- Can think of \( F[u,v] \) as a complex-valued "image" with the same number of pixels as \( X \).

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

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

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

- If \( X \) is real-valued, \( F[-u, -v] = F[W-u, H-v] = \bar{F}[u, v] \), 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)

\[ \sum_{u,v} \|F[u, v]\|^2 = \sum_{u,v} F[u, v] \bar{F}[u, v] = \frac{1}{WH} \sum_{n_x, n_y} \|X[n_x, n_y]\|^2 \]
The Discrete 2D Fourier Transform Pair

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

\[ F'[u, v] = F[u, v] \times \exp\left(-j 2\pi \left(\frac{u t_x}{W} + \frac{v t_y}{H}\right)\right) \]

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

for a fixed integers \( t_x, t_y \)

A change in the phase of the Fourier coefficients, that is linear in \( u, v \), leads to a translation in the image.
FOURIER TRANSFORM

DFT as a Co-ordinate Transform

\[ F[u, v] = \frac{1}{\sqrt{WH}} \sum_{n_x=0}^{W-1} \sum_{n_y=0}^{H-1} \tilde{S}_{uv}[n_x, n_y] X[n_x, n_y] \]

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

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

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

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 \( \sqrt{WH} \))

For \( x, y \in \mathbb{C}^n \), \( \langle x, y \rangle = x^*y \)

- \( x^* \) is the Hermitian of \( x \)
  - Transpose + Conjugate (transpose the vector, and take conjugate of each entry)
- \( x^*y = \sum_i \bar{x}_i y_i \)
FOURIER TRANSFORM

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 \( \sqrt{WH} \))
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 \( \sqrt{WH} \))

**Property:** \( \langle S_{uv}, S_{u'v'} \rangle = 1 \) if \( u' = u \) & \( v' = v \), and 0 otherwise.

**Inverse-DFT:**

\[ X[n_x, n_y] = \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} F[u, v] \exp\left(j 2\pi \left( \frac{u n_x}{W} + \frac{v n_y}{H} \right) \right) \]
DFT as a Co-ordinate Transform

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

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 \( \sqrt{WH} \))

Property: \( \langle S_{uv}, S_{u'v'} \rangle = 1 \) if \( u' = u \) & \( v' = v \), and 0 otherwise.

Inverse-DFT:

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

\( X \) is a weighted sum of the \( S_{uv} \) images, weights are given by \( \sqrt{WH}F[u, v] \).
DFT as a Co-ordinate Transform

\[
F[u, v] = \frac{1}{\sqrt{WH}} \left\langle S_{uv}, X \right\rangle, \quad X = \sqrt{WH} \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} F[u, v] S_{uv}
\]

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

DFT as a Co-ordinate Transform

\[
F[u, v] = \frac{1}{\sqrt{WH}} \left\langle S_{uv}, X \right\rangle, \quad X = \sqrt{WH} \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} F[u, v] S_{uv}
\]

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

"Frequency" Locations
Stacked to form Vector

Spatial Locations
Stacked to form Vector
FOURIER TRANSFORM

DFT as a Co-ordinate Transform

\[ F = \frac{1}{\sqrt{WH}} S^* X, \quad X = \sqrt{WH} S F \]

\( S \) 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).
FOURIER TRANSFORM

$X$

$|F|^2$

Zero-centered Co-ordinates for frequencies $[u,v]$
FOURIER TRANSFORM

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

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

Convolution in "matrix" form

\[ Y[n_x, n_y] \rightarrow [\mathbf{Y}] = \mathbf{A}_k \rightarrow X[n_x, n_y] \]

Spatial Locations
Staked to form Vector

- Mostly 0 (sparse)
- Has \( w_k \) non-zero entries per row.
- Same set of values, but at different places in each row

Spatial Locations
Staked to form Vector

\[
Y[n_x, n_y] = \sum_{n'_x} \sum_{n'_y} k[n'_x, n'_y] \cdot X[n_x - n'_x, n_y - n'_y]
\]
$
Y = X * k \Rightarrow Y = A_k X$

$A_k$ is not square for valid / long convolution.

**Question:**

Let $Y = A_k X$ correspond to $Y = X \ast_{\text{valid}} k$. 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 $k$)
CONVOLUTION THEOREM

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

Now if we consider the square \( A_k \) matrix corresponding to 'same' convolution with circular padding, i.e. padding as \( X[W + n_x, n_y] = X[n_x, n_y], X[n_x, -n_y] = X[n_x, H - n_y] \), etc.

Then, \( A_k \) is \textit{diagonalized} by the Fourier Transform!

\[ A_k = S \ D_k \ S^* \]

- Here, \( D_k \) is a diagonal matrix.
- The above equation holds for every \( A_k \)
  - You get different diagonal matrices \( D_k \).
  - But \( S \) is the diagonalizing basis for all kernels.
- In the Fourier co-ordinate system, convolution is a 'point-wise' operation!

\[ Y = A_k X = S \ D_k \ S^* \ X \Rightarrow (S^* Y) = D_k(S^* X) \]
CONVOLUTION THEOREM

Why does this happen?

1. \[ X = \sqrt{WH} \sum_{u,v} F[u, v] S_{uv} \]
2. \[ Y = X \ast k = \sqrt{WH} \sum_{u,v} F[u, v] S_{uv} \ast k \text{(by linearity / distributivity)} \]
3. \[ (S_{uv} \ast k)[n] = \sum_{n'} k[n'] S_{uv}[n - n'] \]
4. \( S_{uv}[n - n'] \), assuming circular padding, is also a sinusoid with the same frequency \((u, v)\) and magnitude, but different phase.
5. Multiplying by \( k[n'] \) changes the magnitude, but frequency still the same.
6. Adding different sinusoids of the same frequency gives you another sinusoid of the same frequency.
7. \( (S_{uv} \ast k)[n_x, n_y] = d_{uv:k} S_{uv}[n_x, n_y] \), where \( d_{uv:k} \) is some complex scalar.

\textit{Sinusoids are eigen-functions of convolution}

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

\[ A_k = S \, D_k \, S^* \]

- What's more, the diagonal elements of \( D_k \) are the \((W_x \times W_y)\) 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'.
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 has to be the same size as the image.

- From same circular, you can always get 'valid' by cropping.
- To get full / same with zero-padding, pad your original image first.

1. Zero-pad
2. Circularly shift to center at (0,0)
Kernel / Fourier Transform (magnitude) Pairs

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

Gaussian Derivatives: Band-pass

For more indepth coverage:
Szeliski Sec 3.4