**BILATERAL FILTERING**

Denoising by Smoothing (with a Gaussian filter):

\[ X \]

\[ Y = X \ast G \]

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

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

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

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

**OFFICE HOURS**

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>5:40pm-6:30pm</th>
<th>Jolley 431</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jarett Gross</td>
<td>Wed 9:30am-10:30am</td>
<td>Jolley 205</td>
<td></td>
</tr>
<tr>
<td>Ayan Chakrabarti</td>
<td>Fri 10:00am-11:00am</td>
<td>Jolley 420</td>
<td></td>
</tr>
<tr>
<td>Abby Stylianou*</td>
<td>9/8,15</td>
<td>Jolley 309</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9/22-</td>
<td>9/22-</td>
<td></td>
</tr>
</tbody>
</table>

---

Sep 7, 2017
**BILATERAL FILTERING**

Denoising by Smoothing (with a Gaussian filter):

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

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

Make the filter weights data dependent!

---

**BILATERAL FILTERING**

Denoising with a Bilateral Filter

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

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

- \( \sigma_f \) High
- Gaussian Filter Result

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

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

- \( \sigma_f \) Medium
- Gaussian Filter Result
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

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

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

The Discrete 2D Fourier Transform

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

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

We follow EE convention and use $j = \sqrt{-1}$ instead of $i$.

- Defined for a single-channel / grayscale image $X$.
- $F$ is a "complex valued" array indexed by integers $u$, $v$.
- 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 (often, FFT is used to refer to the operation itself).
FOURIER TRANSFORM

The Discrete 2D Fourier Transform Pair

\[ F'[X] = F[u, v] = \frac{1}{WH} \sum_{n_x=0}^{W-1} \sum_{n_y=0}^{H-1} X[n_x, n_y] \exp(-j2\pi \left( \frac{u n_x}{W} + \frac{v n_y}{H} \right)) \]

\[ F^{-1}[F] = X[n_x, n_y] = \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} F[u, v] \exp(j2\pi \left( \frac{u n_x}{W} + \frac{v n_y}{H} \right)) \]

- If \( X \) is real-valued, \( F[-u, -v] = \bar{F}[W - u, H - v] = \bar{F}[u, v] \) 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)
  \[ \sum_{u,v} ||F[u,v]||^2 = \sum_{u,v} X[n_x,n_y] \sum_{u,v} ||X[n_x,n_y]||^2 \]

FOURIER TRANSFORM

DFT as a Co-ordinate Transform

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

(Remember for \( u, v \in \mathbb{C}^n \), \( \langle u, v \rangle = u^*v \)).

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

\[ S_{uv}[n_x, n_y] = \frac{1}{\sqrt{WH}} \exp(j2\pi \left( \frac{u n_x}{W} + \frac{v n_y}{H} \right)) \]

\( F[u, v] \) 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:

\[ X = \sqrt{WH} \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} F[u, v] \bar{S}_{uv} \]
**FOURIER TRANSFORM**

DFT as a Co-ordinate Transform

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

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

So, \(S\) is a unitary matrix.

- 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**
FOURIER TRANSFORM

Reconstruct with only these frequency components

Reconstruct with only these frequency components

Reconstruct with only these frequency components

Reconstruct with only these frequency components
**Fourier Transform**

A is not square for valid/long convolution.

**Convolusion Theorem**

**Convolution in “matrix” form**

\[ Y[n_x, n_y] = X \ast k \Rightarrow Y = A_k X \]

**Question:**

Let \( Y = A_k X \) correspond to \( Y = X \ast_k k \). 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**

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 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?**

- \[ X = \sqrt{W^2 + H^2} \sum_{u,v} F[u,v] S_{uv} \]
- \[ Y = X \ast k = \sqrt{W^2 + H^2} \sum_{u,v} F[u,v] S_{uv} \ast k \]
- 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.
- \[ \alpha \cos \theta + \beta \sin \theta \]
- \[ (S_{uv} \ast k)[n_x, n_y] = d_{uv,k} S_{uv}[n_x, n_y], \] where \(d_{uv,k}\) is some complex scalar.

**Sinusoids are eigen-functions of convolution**

\[ Y = X \ast k = \sqrt{W^2 + H^2} \sum_{u,v} F[u,v] S_{uv} \ast k = \sqrt{W^2 + H^2} \sum_{u,v} (F[u,v] d_{uv,k}) S_{uv} \]

**CONVOLUTION THEOREM**

\[ A_k = S D_k S^* \]

- What's more, the diagonal elements of \(D_k\) are the \((W_x \times W_y)\) Fourier transform of \(k\).
- \[ D_k = \text{diag} \left( \frac{1}{\sqrt{W^2 + H^2}} S^* k \right) \]
- 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.

**CONVOLUTION THEOREM**

Kernel has to be the same size as the image.
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 \(0,0\)

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

For more indepth coverage:
Szeliski Sec 3.4