Convolution

Fredo Durand
MIT EECS 6.815/6.865
Start Pset 2 early
Convolution

Fredo Durand
MIT EECS 6.815/6.865
Convolution

Fredo Durand
MIT EECS 6.815/6.865
Morphing

- [Link to article](http://www.huffingtonpost.com/2015/02/17/evolution-laptop-gif_n_6698160.html?fb_action_ids=10205121595892118&fb_action_types=og.likes)
Frédo’s office hours

• Mondays 4pm, 32-D424
Blur, sharpen
Image processing motivation

• Sharpen images
• Downsample images
• Fake depth of field
• Smooth out noise, skin blemishes
• ...

• We must understand convolution!
Sharpening
Downsampling

Downsample by a scale of 0.2
Downsampling

• Using bilinear code from Pset 2

Downsample by a scale of 0.2
Downsampling

- Using bilinear code from Pset 2
- Yikes! Herringbone patterns

Downsample by a scale of 0.2
Downsampling

- Using bilinear code from Pset 2
- Yikes! Herringbone patterns

Downsample by a scale of 0.2
Downsampling

- We “randomly” pick a color in the high frequency pattern
Downsampling

- Solution: blur the pattern to get average color over new pixels
Fake tilt shift

- http://www.tiltshiftphotography.net/photoshop-tutorial.php
Blur in optics

- Diffraction
- Lens aberrations
- Object movement
- Camera shake

- Can we remove blur computationally?
  - Inverse the blur equation
  - deconvolution
  - later!
Lens diffraction

- **See also** [http://www.cambridgeincolour.com/tutorials/diffraction-photography.htm](http://www.cambridgeincolour.com/tutorials/diffraction-photography.htm)
Lens diffraction

  (heavily cropped)
- See also [http://www.cambridgeincolour.com/tutorials/diffraction-photography.htm](http://www.cambridgeincolour.com/tutorials/diffraction-photography.htm)
Lens diffraction

- See also [http://www.cambridgeincolour.com/tutorials/diffraction-photography.htm](http://www.cambridgeincolour.com/tutorials/diffraction-photography.htm)
Blur example: spherical aberration

- Pixel value: weighted average of local color
Remove optical artifacts

• Calibrate lenses and remove blur
• e.g. DXO
Questions?
Convolution
101
Blur as convolution

- Replace each pixel by a linear combination of its neighbors.
  - only depends on relative position of neighbors
- The prescription for the linear combination is called the “convolution kernel”.

\[
\begin{bmatrix}
10 & 5 & 3 \\
4 & 5 & 1 \\
1 & 1 & 7 \\
\end{bmatrix}
\quad
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0.5 & 0 \\
0 & 1 & 0.5 \\
\end{bmatrix}
\quad
\begin{bmatrix}
35 & 7 \\
\end{bmatrix}
\]

Local image data  kernel  Modified image data (shown at one pixel)
Linear **shift-invariant** filtering

- Replace each pixel by a linear combination of its neighbors.
  - only depends on relative position of neighbors
- The prescription for the linear combination is called the “convolution kernel”.
  - same kernel for all pixels

<table>
<thead>
<tr>
<th>10</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Local image data

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Kernel

| 7 |

Modified image data (shown at one pixel)
filtering

• Replace each pixel by a linear combination of its neighbors.
  – only depends on relative position of neighbors

• The prescription for the linear combination is called the "convolution kernel".
  – same kernel for all pixels

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local image data

Modified image data (shown at one pixel)
Linear shift-invariant filtering

- Replace each pixel by a linear combination of its neighbors.
- Only depends on relative position of neighbors.
- The prescription for the linear combination is called the "convolution kernel".
- Same kernel for all pixels.

\[
\begin{bmatrix}
1 & 5 & 1 \\
3 & 5 & 3 \\
1 & 5 & 1
\end{bmatrix}
\]

Local image data

\[
p(x) \ast k = \sum_{y} k(y) p(x+y)
\]

Translation

\[
\mathbf{f}(x) \ast \mathbf{k} = \mathbf{g}(x)
\]
Linear **shift-invariant** filtering

- Replace each pixel by a linear combination of its neighbors.
  - only depends on relative position of neighbors
- The prescription for the linear combination is called the “convolution kernel”.
  - same kernel for all pixels

| 10 5 3 | 0 0 0 | 7 |
| 4 5 1 | 0 0.5 0 |
| 1 1 7 | 0 1 0.5 |

Local image data | kernel | Modified image data (shown at one pixel)
Linear shift-invariant filtering

• Replace each pixel by a linear combination of its neighbors.
  – only depends on relative position of neighbors

• The prescription for the linear combination is called the “convolution kernel”.
  – same kernel for all pixels

<table>
<thead>
<tr>
<th>10</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Local image data

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

kernel

| 7 |

Modified image data (shown at one pixel)
Example of linear NON-shift invariant transformation?
Example of linear NON-shift invariant transformation?

- e.g. neutral-density graduated filter (darken high y, preserve small y)
  
  \[ J(x,y) = I(x,y) \times (1 - y/y_{\text{ymax}}) \]
Example of linear NON-shift invariant transformation?

- e.g. neutral-density graduated filter (darken high y, preserve small y)
  \[ J(x,y) = I(x,y) \times (1 - y/\text{ymax}) \]

- Formally, what does linear mean?
Example of linear NON-shift invariant transformation?

• e.g. neutral-density graduated filter (darken high y, preserve small y)
  \[ J(x,y) = I(x,y) \times (1 - y/y_{\text{max}}) \]

• Formally, what does linear mean?
  – For two scalars \(a\) & \(b\) and two inputs \(x\) & \(y\):
    \[ F(ax+by) = aF(x) + bF(y) \]

• What does shift invariant mean?
Example of linear NON-shift invariant transformation?

- e.g. neutral-density graduated filter (darken high y, preserve small y)
  \[ J(x,y) = I(x,y) \times (1 - \frac{y}{y_{\text{max}}}) \]

- Formally, what does linear mean?
  - For two scalars \( a \) & \( b \) and two inputs \( x \) & \( y \):
    \[ F(ax + by) = aF(x) + bF(y) \]

- What does shift invariant mean?
  - For a translation \( T \):
    \[ F(T(x)) = T(F(x)) \]
Example of linear NON-shift invariant transformation?

• e.g. neutral-density graduated filter (darken high y, preserve small y)
  \[ J(x,y) = I(x,y) \times (1-y/ymax) \]

• Formally, what does linear mean?
  – For two scalars a & b and two inputs x & y:
    \[ F(ax+by) = aF(x) + bF(y) \]

• What does shift invariant mean?
  – For a translation T:
    \[ F(T(x)) = T(F(x)) \]
  – If I blur a translated image, I get a translated blurred image
Questions?
Convolution algorithm

• Set output image to zero
• For y, x in output image
  - for y’, x’ in kernel
    - out[y, x] += input[y+y’, x+x’] * kernel[y’, x’]
• (this assumes the kernel coordinates are centered)

Local image data

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

kernel

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

Modified image data (shown at one pixel)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
Questions?

• Set output image to zero
• For y, x in output image
  - for y’, x’ in kernel
    - out[y, x] += input[y+y’, x+x’]*kernel[y’, x’]
• (this assumes the kernel coordinates are centered)

<table>
<thead>
<tr>
<th>10</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Local image data

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

kernel

<table>
<thead>
<tr>
<th>7</th>
</tr>
</thead>
</table>

Modified image data (shown at one pixel)
Convolution (warm-up slide)
Convolution (warm-up slide)

original

Pixel offset

Filtered
(no change)
Convolution (warm-up slide)
Convolution

original

Pixel offset

coefficient

1.0

?
shift

original

pixel offset

coefficient

shifted
Convolution

original

coefficient

0.3

Pixel offset

0

?
Blurring

original

Blurred (filter applied in both dimensions).

coefficient

Pixel offset

0.3
Blur examples

impulse

original

8

impulse

coefficient

Pixel offset

0.3

filtered

2.4
Blur examples

impulse

original

filtered

edge

original

filtered
Questions?
Formally
More formally: Convolution

\((I \otimes g)(x) = \int_{x'} I(x') g(x - x') dx'\)
More formally: Convolution

\[(I \otimes g)(x) = \int_{x'} I(x')g(x - x')\,dx'\]
More formally: Convolution

\[(I \otimes g)(x) = \int_{x'} I(x') g(x - x') \, dx'\]
More formally: Convolution

\[(I \otimes g)(x) = \int_{x'} I(x') g(x - x') \, dx'\]
More formally: Convolution

\[(I \otimes g)(x) = \int_{x'} I(x')g(x - x')dx'\]
More formally: Convolution

\[(I \otimes g)(x) = \int_{x'} I(x') g(x - x') \, dx'\]
More formally: Convolution

\[(I \otimes g)(x) = \int_{x'} I(x') g(x - x') dx'\]
Demo

- http://math.mit.edu/daimp/ConvFlipDrag.html
Questions?

\[(I \otimes g)(x) = \int_{x'} I(x') g(x - x') \, dx'\]
What's up with flipping?
Convolution & probability

- Convolution was first used by Laplace to study the probability of the sum of two random variables.
Random variables

\[ P(-2) = P(X = -1) \cdot P(Y = -1) \]

\[ P(0) = P(X = -1) \cdot P(Y = 1) + P(X = 1) \cdot P(Y = 0) \]

\[ P(X + Y = 0) \]

\[ \sum P(X = x_i) \cdot P(Y = 1 - x_i) \]
Random variables

• How can $X+Y=0$?
  - $X=-1$, $Y=1$
  - $X=0$, $Y=0$
  - $X=1$, $Y=-1$

Probability?

$P(X=-1) \times P(Y=1)$

$P(X=0) \times P(Y=0)$

$P(X=1) \times P(Y=-1)$
Sum of random variables

\[ P(X + Y = k) = \sum_{k'} P(X = k') P(Y = k - k') \]
Questions?

\[ P(X + Y = k) = \sum_{k'} P(X = k') P(Y = k - k') \]
Point Spread Function & flip

\[(I \otimes g)(x) = \int_{x'} I(x') g(x - x') \, dx'\]
Point Spread Function & flip

\[(I \otimes g)(x) = \int_{x'} I(x') g(x - x') \, dx'\]

\[(I \otimes g)(-1) = I(-1)g(-1 - -1) + I(0)g(-1 - 0) + I(1)g(-1 - 1)\]

\[(I \otimes g)(0) = I(-1)g(0 - -1) + I(0)g(0 - 0) + I(1)g(0 - 1)\]

\[(I \otimes g)(1) = I(-1)g(1 - -1) + I(0)g(1 - 0) + I(1)g(1 - 1)\]
Point Spread Function & flip

\[(I \otimes g)(x) = \int_{x'} I(x')g(x - x') \, dx'\]
Point Spread Function & flip

\[(I \otimes g)(x) = \int_{x'} I(x') g(x - x') \, dx'\]

\[
(I \otimes g)(-1) = I(-1)g(-1 - -1) + I(0)g(-1 - 0) + I(1)g(-1 - 1) \\
(I \otimes g)(0) = I(-1)g(0 - -1) + I(0)g(0 - 0) + I(1)g(0 - 1) \\
(I \otimes g)(1) = I(-1)g(1 - -1) + I(0)g(1 - 0) + I(1)g(1 - 1)
\]
Point Spread Function & flip

\[(I \otimes g)(x) = \int_{x'} I(x') g(x - x') \, dx'\]

- Remember: it’s always the inverse transform
- Forward model: light goes from x to x+x’
- Backward model: light at x comes from x-x’
Without the flip

- Called correlation
- Won’t matter most of the time because our kernels are symmetric
Other differences

• Convolution is associative
• Convolution is a little nicer in the Fourier domain
Questions?
Symmetry
Commutativity

\[ I \otimes g = g \otimes I \]
Commutativity

\[ I \otimes g = g \otimes I \]

\[ (I \otimes g)(x) = \int_{x'} I(x') g(x - x') \, dx' \]

\[ \int_y \left| \frac{dy}{dx'} \right| = 1 \mid I(x' = -y + x) g(y) \, dy \]

It doesn't matter which one you flip!
Similar to probability situation, as long as the sum is \( x \)
Movie break
Blur zoo
Demo applet

Box filter
Nice and smooth: Gaussian
Gaussian formula

\[ A e^{-\frac{r^2}{2\sigma^2}} \]

http://en.wikipedia.org/wiki/Gaussian_function
Gaussian formula

\[ ae^{-\frac{r^2}{2\sigma^2}} \]

- \( r \) is the distance to the center
- \( a \) is a normalization constant
  - I usually just normalize my kernels after the fact
- \( \sigma \) is the standard deviation and controls the
Gaussian formula

\[ ae^{-\frac{r^2}{2\sigma^2}} \]

- Gaussians have an infinite support
  - \( >0 \) everywhere
- But are often truncated
  - I recommend \( 3\sigma \)

http://en.wikipedia.org/wiki/Gaussian_function
Truncation

- For computational tractability/efficiency
- Consider Gaussian is zero beyond 3sigma

The bigger the Gaussian kernel, the blurrier the output
Sharpening
Truncation

- For computational tractability/efficiency
- Consider Gaussian is zero beyond 3sigma

The bigger the Gaussian kernel, the blurrer the output. In both directions.
Gaussian formula

\[ ae^{-\frac{r^2}{2\sigma^2}} \]

- Gaussians have an infinite support
  - \( >0 \) everywhere
- But are often truncated
  - I recommend \( 3\sigma \)

http://en.wikipedia.org/wiki/Gaussian_function
Truncation

• For computational tractability/efficiency
• consider Gaussian is zero beyond 3-sigma

in both directions

the bigger the Gaussian kernel
the blurrier the output
Sharpening
How can we sharpen?

• Blurring was easy
• Sharpening is not as obvious
How can we sharpen?

- Blurring was easy
- Sharpening is not as obvious

- Idea: amplify the non-blurring part of an image

- \[ \text{out} = \text{input} + k \times (\text{input-} \ \text{blur} (\text{input})) \]
  
  \[\text{sharp component}\]
Sharpening

Input

- blurred

= High pass

Input

High pass +k*

= Sharpened image
Sharpening: kernel view

• Recall

\[ f' = f + k \star (f - f \otimes g) \]

• \( f \) is the input
• \( f' \) is a sharpened image,
• \( g \) is a blurring kernel
• \( k \) is a scalar controlling the strength of sharpening
Sharpening: kernel view

- Recall

\[ f' = f + k \ast (f - f \otimes g) \]

- Denote \( \delta \) the Dirac kernel (pure impulse)

\[ f = f \otimes \delta \]
Sharpening: kernel view

• Recall

\[ f' = f + k \ast (f - f \odot g) \]

\[ f' = f \odot \delta + k \ast (f \odot \delta - f \odot g) \]

\[ f' = f \odot ((k + 1)\delta - g) \]

• Sharpening is also a convolution
Sharpening kernel

• Note: many other kernels exist (just like we saw multiple blurring kernels)

\[ f' = f \otimes ((k + 1)\delta - g) \]

blue: positive
red: negative
Intuition

- Amplify the difference between a pixel and its neighbors

blue: positive
red: negative
Questions?

\[
\begin{array}{ccc}
\text{original} & - & \text{transformed} \\
\text{transformed} & + k^* & \text{result} \\
\end{array}
\]
Unsharp Mask
Unsharp mask

- Sharpening is often called “unsharp mask” because photographers used to sandwich a negative with a blurry positive film in order to sharpen.

http://www.tech-diy.com/UnsharpMasks.htm
Fig.4: The two examples here show a detail of the brickwork to the left of the church door. The one on the left was printed with the negative alone – the one on the right was printed with both negative and mask as a sandwich. The increase in local contrast and edge sharpness is minute, but clearly visible. Grade 2.5 was used for the straight print but increased to 4.5 for the sandwiched image to compensate for the reduced contrast.

Fig.5: These two examples show a detail of the lower right hand side of the church door. Here the difference in sharpness is clearly visible between the (left) negative and (right) sandwich prints.

http://www.tech-diy.com/images/unsharp2.jpg
Unsharp mask

• http://en.wikipedia.org/wiki/Unsharp_masking
• http://www.largeformatphotography.info/unsharp/
• http://www.tech-diy.com/UnsharpMasks.htm
• http://www.cambridgeincolour.com/tutorials/unsharp-mask.htm
Sharpening++
Problem with excess

- Haloes around strong edges
oversharpening

original

8

Sharpened
(differences are accentuated; constant areas are left untouched).

1.7

-0.3

-0.25

11.2

8
Bells and whistles

- Apply mostly on luminance

- Old Clarity in Adobe Camera Raw
  - As far as I understand, apply only for mid-tones
  - Avoids haloes around black and white points

- Only apply at edges
  - To avoid the amplification of noise

- Sharpening chrominance as well
  - But with very large blur
Lightroom demo
Oriented filters
Gradient: finite difference

- horizontal gradient $\begin{bmatrix} -1, 1 \end{bmatrix}$
- vertical gradient: $\begin{bmatrix} -1 \end{bmatrix}, \begin{bmatrix} 1 \end{bmatrix}$
Gradient: finite difference

- horizontal gradient $[[-1, 1]]$
- vertical gradient: $[[-1], [1]]$
Gradient

- e.g. Sobel
  - [URL](http://en.wikipedia.org/wiki/Sobel_operator)

\[
G_x = \begin{bmatrix}
-1 & 0 & +1 \\
-2 & 0 & +2 \\
-1 & 0 & +1 \\
\end{bmatrix} \ast A \quad \text{and} \quad G_y = \begin{bmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
+1 & +2 & +1 \\
\end{bmatrix} \ast A
\]

Horizontal gradient  \hspace{1cm}  \text{Vertical gradient}

Magnitude