Filtering & convolution

Wojciech Jarosz
wojciech.k.jarosz@dartmouth.edu
Your Spanish castle illusions
Timelapse photography in the news
Today’s agenda

Linear filtering & convolution
- blurring
- sharpening

Complexity analysis
- Optimizations

Denoising from a single image
- Bilateral filtering
Blur, sharpen
Image processing motivation

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

... 

We must understand convolution!
Sharpening
Downsampling

Yikes! Herringbone patterns

Downsample by a scale of 0.2

After a slide by Frédo Durand
Downsampling

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

Downsample by a scale of 0.2
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?

- invert the blur equation
- deconvolution
Lens diffraction


(heavily cropped)

See also
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
Removing camera shake

Original

Naïve Sharpening

Fergus et al’s algorithm
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”.

After a slide by Frédo Durand
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

```plaintext
local image data
10  5  3
4   5  1
1   1  7

kernel
0   0   0
0   0.5  0
0   1   0.5

modified image
    7
```
Example of linear NON-shift invariant transformation?

e.g. neutral-density graduated filter (darken high y):

- \( J(x,y) = I(x,y) \cdot (1-y/ymax) \)

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 all pixels \((x,y)\) in output image
  for all \((x',y')\) in kernel
    \[
    \text{out}(x,y) += \text{input}(x+x',y+y') \times \text{kernel}(x',y')
    \]
(this assumes the kernel coordinates are centered)

<table>
<thead>
<tr>
<th>local image data</th>
<th>kernel</th>
<th>modified image</th>
</tr>
</thead>
<tbody>
<tr>
<td>10  5  3</td>
<td>0  0  0</td>
<td></td>
</tr>
<tr>
<td>4   5  1</td>
<td>0  0.5 0</td>
<td></td>
</tr>
<tr>
<td>1   1  7</td>
<td>0  1  0.5</td>
<td>7</td>
</tr>
</tbody>
</table>
Questions?

for all pixels \((x,y)\) in
for all \((x',y')\)
\[
\text{out}(x,y) \ += \ \text{input}(x+x',y+y') \times \text{kernel}(x',y')
\]
(this assumes the kernel coordinates are centered)

After a slide by Frédo Durand
Convolution (warm-up slide)

After a slide by Frédo Durand
Convolution (warm-up slide)

After a slide by Frédo Durand

original

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

After a slide by Frédo Durand

\[ f = f \ast \delta \]
Convolution

After a slide by Frédéric Durand
Convolution

After a slide by Frédo Durand
Convolution

After a slide by Frédéric Durand
Blurring

original

×

1/3

pixel offset

= blurred

(applied in both dimensions)
Blur examples

impulse

original

pixel offset

coefficient

filtered

\[ \begin{align*}
8 &\quad \times \quad \frac{1}{3} \\
0 &\quad = \quad \frac{8}{3}
\end{align*} \]
Blur examples

impulse

original

pixel offset

coefficient

filtered

edge

original

pixel offset

coefficient

filtered

After a slide by Frédéric Durand
Questions?
Formally
More formally: Convolution

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

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

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

$$P(X + Y = k) = \sum_{k'} P(X = k') P(Y = k - k')$$

#### How can $X+Y=0$?

- $X=-1, Y=1$
- $X=0, Y=0$
- $X=1, Y=-1$

#### Probability?

- $P(X=-1) \cdot P(Y=1)$
- $P(X=0) \cdot P(Y=0)$
- $P(X=1) \cdot P(Y=-1)$
Questions?

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

How can \(X+Y=0\)?

- \(X=-1, Y=1\)
- \(X=0, Y=0\)
- \(X=1, Y=-1\)

Probability?

- \(P(X=-1)*P(Y=1)\)
- \(P(X=0)*P(Y=0)\)
- \(P(X=1)*P(Y=-1)\)
Compare

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

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

Forward model: light goes from \( x \) to \( x + x' \)

Backward model: light at \( x \) comes from \( x - x' \)
Image processing

I will often use the term “convolution” improperly and fail to flip the kernel

- Called correlation

- Won’t matter most of the time because our kernels are symmetric
Questions?
Movie break
http://graphics.stanford.edu/courses/cs178/applets/convolution.html
Box filter

After a slide by Frédo Durand
Nice and smooth: Gaussian
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 width of the Gaussian

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

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

Gaussians have infinite support

- >0 everywhere

but are often truncated

- consider Gaussian to be zero beyond e.g. 3\(\sigma\)
- for computational tractability/efficiency

http://en.wikipedia.org/wiki/Gaussian_function
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 stuff not in the blurry image

\[ \text{output} = \text{input} + k*(\text{input} - \text{blur}(\text{input})) \]
Sharpening

\[ \text{input} - \text{blurred} = \text{high pass} \]

\[ \text{input} + k^* \text{high pass} = \text{sharpened image} \]
Sharpening: kernel view

Recall

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

\( f \) 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

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

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

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

Sharpening is also a convolution
Sharpening kernel

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

Amplify the difference between a pixel and its neighbors

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

After a slide by Frédo Durand
Alternate interpretation

\[
\text{out} = \text{input} + k(\text{input-blur(input)})
\]

\[
\text{out} = (1 + k)*\text{input} - k*\text{blur(input)}
\]

\[
\text{out} = \text{lerp}(\text{blur(input)}, \text{input}, 1+k)
\]

- linearly extrapolate from the blurred image “past” the original input image
Questions?

\[ \text{Image 1} - \text{Image 2} = \text{Image 3} \]

\[ \text{Image 1} + k* \text{Image 2} = \text{Image 4} \]
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.
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.
Unsharp mask

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

After a slide by Frédo Durand

(original)

Sharpened

(differences are accentuated; constant areas are left untouched).
Bells and whistles

Apply mostly on luminance

Old Clarity in Lightroom/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 \([-1, 1]\]

vertical gradient: \([-1], [1]\]
Gradient: finite difference

horizontal gradient $[[-1, 1]]$

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

After a slide by Frédo Durand
Gradient

e.g. Sobel [http://en.wikipedia.org/wiki/Sobel_operator]

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

Horizontal gradient

Vertical gradient

After a slide by Frédo Durand
Cost
Convolution cost?

set output image to zero
for all pixels \((x,y)\) in output image
  for all \((x',y')\) in kernel
    \[\text{out}(x,y) + = \text{input}(x+x',y+y') \times \text{kernel}(x',y')\]

Cost?
- \(O(\text{input.width} \times \text{input.height} \times \text{kernel.width} \times \text{kernel.height})\)
Separable filters
Separability

Sometimes the 2D kernel can be decomposed into the convolution of a horizontal and a vertical filter.

Example: box

- \( g(x) = \text{const if } (-k \leq x \leq k) , 0 \text{ otherwise} \)
- \( g(x,y) = g(x) \otimes g(y) \)
- (separability doesn’t require the two 1D kernels to be the same, but it’s the case here)
Separable box blur

First blur horizontally using $g(x)$
Then blur vertically using $g(y)$
Separable convolution cost?

for all pixels \((x,y)\) in output image
   for all \(x’\) in kernel
       \(\text{outX}(x,y) += \text{input}(x+x’,y) \times \text{kernel}(x’)\)
for all pixels \((x,y)\) in output image
   for all \(y’\) in kernel
       \(\text{out}(x,y) += \text{outX}(x,y+y’) \times \text{kernel}(y’)\)

Horizontal cost? \(O(\text{input.width} \times \text{input.height} \times \text{kernel.width})\)

Vertical cost? \(O(\text{input.width} \times \text{input.height} \times \text{kernel.height})\)

Total: \(O(\text{input.width} \times \text{input.height} \times (\text{kernel.height}+\text{kernel.width}))\)

Instead of: \(O(\text{input.width} \times \text{input.height} \times (\text{kernel.height} \times \text{kernel.width}))\)
Good news

Gaussians are separable too

See Assignment 4!
Box blur: Can we do even better?

Can we get even better asymptotic complexity?

Very large kernel sizes?
Box blur: Can we do even better?

Since 2D box is separable, let’s focus on the 1D case.

The neighborhoods of pixel $i$ and pixel $i+1$ are very similar.

In fact, they only differ by 2 pixels, so:

$$\text{out}(i+1) = \text{out}(i) + \frac{\text{in}(i+k+1) - \text{in}(i-k+1)}{2k+1}$$

Asymptotically independent of kernel size, depends only on image size!
Box blur cost?

Naïve: \( O(input.\text{width} \times input.\text{height} \times (kernel.\text{height} \times kernel.\text{width})) \)

Separable: \( O(input.\text{width} \times input.\text{height} \times (kernel.\text{height} + kernel.\text{width})) \)

Incremental: \( O(input.\text{width} \times input.\text{height} \times (kernel.\text{height} + kernel.\text{width})) \)

\( O(input.\text{width} \times input.\text{height}) \)
Repeated convolution
Repeated convolution
Repeated convolution
Repeated convolution
Repeated convolution

Convolution of two box kernels yields a tent kernel

\[ \text{Re} \]

\[ \text{Con} \]

\[ \text{vo} \]

\[ \text{lution\ of\ two\ box\ kernels\ yields\ a\ tent\ kernel} \]

\[ \text{Re} \]

\[ \text{Con} \]

\[ \text{vo} \]

\[ \text{lution\ of\ two\ box\ kernels\ yields\ a\ tent\ kernel} \]
Repeated convolution

Yet another convolution with a box yields piecewise quadratic
Repeated convolution

The pattern continues

- Box filtering the piecewise quadratic will yield a piecewise cubic, and so on.

Each time we make the kernel smoother

Taking this to the limit will yield a Gaussian
Photoshops’ Gaussian

not a true Gaussian
Gaussian blur as multi-box blur

Can approximate Gaussian blur with several box blurs

Asymptotically independent of kernel size!

Assignment 4 extra credit

- what is Gaussian’s $\sigma$ for 5 box blurs?
Nitty-gritty stuff
Best input to debug convolution

Impulse
Centering the kernel

Our images are defined with 0,0 in the upper left corner.

Kernels are usually assumed to have origin at the center.
Normalization

As a rule of thumb, you want kernels to be normalized when you want the output to preserve the overall brightness of the image.
Denoising from a single image
Denoising from 1 image

We can’t take average over multiple images
Denoising from 1 image

We can’t take average over multiple images

Idea 1: take a spatial average

- Most pixels have roughly the same color as their neighbor
- Noise looks high frequency => do a low pass

Here: Gaussian blur

Noisy input
Gaussian blur

After a slide by Frédo Durand
Gaussian blur

Noise is mostly gone

But image is blurry

- duh!
Bilateral filtering
Gaussian blur

Noise is mostly gone

But image is blurry

- duh!

Problem: not all neighbors have the same color

Bilateral filter idea: only consider neighbors that have similar values
Bilateral filter

Tomasi and Manduci 1998 http://www.cse.ucsc.edu/~manduchi/Papers/ICCV98.pdf

Developed for denoising

Related to

- SUSAN filter [Smith and Brady 95] http://citeseer.ist.psu.edu/smith95susan.html

Bilateral filtering

Images are often piecewise constant with noise added
- Then nearby pixels are a different noisy measurement of the same value

Simply blurring doesn’t work
- also blurs edges

We should blur only within each constant-colored region
- not across edges between regions
Bilateral filtering

If pixels are similar in intensity, they are probably from the same region of the scene

Perform a “convolution” where the weight applied to nearby pixels falls off with:

- increasing (x,y) distance from the pixel being blurred
- increasing intensity difference from the pixel being blurred

i.e. blur in domain and range dimensions!

[after a slide by Marc Levoy]

[Tomasi and Manduchi 1998]
Start with Gaussian filtering

Here, input is a step function + noise

\[ J = f \otimes I \]

output ← after a slide by Frédéric Durand
Gaussian filter as weighted average

Weight of $\xi$ depends on distance to $x$

$$J(x) = \sum_{\xi} f(x,\xi) \cdot I(\xi)$$

After a slide by Frédo Durand
The problem of edges

Here, $I(\xi)$ “pollutes” our estimate $J(x)$

It is too different

$$J(x) = \sum_{\xi} f(x,\xi) I(\xi)$$

After a slide by Frédo Durand
Principle of Bilateral filtering

Penalty $g$ on the intensity difference

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x, \xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$
Bilateral filtering

Spatial Gaussian $f$

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x, \xi) g(I(\xi) - I(x)) I(\xi)$$

[Tomasi and Manduchi 1998]
Bilateral filtering

Spatial Gaussian \( f \)
Gaussian \( g \) on the intensity difference

\[
J(x) = \frac{1}{k(x)} \sum_{\xi} f(x, \xi) \, g(I(\xi) - I(x)) \, I(\xi)
\]

After a slide by Frédo Durand
Normalization factor

\[ k(x) = \sum_{\xi} f(x, \xi) \ g(I(\xi) - I(x)) \]

\[ J(x) = \frac{1}{k(x)} \sum_{\xi} f(x, \xi) \ g(I(\xi) - I(x)) \ I(\xi) \]

[Tomasi and Manduchi 1998]
Bilateral filtering is non-linear

The weights are different for each output pixel

\[ J(x) = \frac{1}{k(x)} \sum_{\xi} f(x, \xi) g(I(\xi) - I(x)) I(\xi) \]
Bilateral filter

Noisy input

After bilateral filter
Can we do better?

Noisy input  After bilateral filter

chroma noise
Chroma noise

Our visual system has different spatial frequency response to chrominance vs. luminance

Perform Bilateral filtering in YUV

Bigger spatial filter in U & V
Normal RGB Bilateral filter

Noisy input

After bilateral filter
YUV Bilateral filter

Noisy input

After YUV bilateral filter
Comparison

Noisy input  Bilateral filter  YUV bilateral filter
Bilateral filtering

Also used to remove skin blemishes in portraits

- Surface blur in photoshop
  (although box spatial kernel instead of Gaussian)

Useful for lots of other things

- More in future lectures
- In particular, tone mapping for contrast reduction and high-dynamic-range imaging
Photoshop surface blur

Note the radius and threshold controls

- same as $\sigma_{\text{domain}}$ and $\sigma_{\text{range}}$
Assignment 4
Assignment 4

Convolution
Separable
Unsharp mask
Gradient
Denoising
YUV denoising
Other approaches to denoising
Denoising

Bayesian coring in the wavelet domain
- Simoncelli & Adelson

Big heuristics
- BM3D

NL means
- Buades et al.
- Bilateral in the space of patches

Statistics of natural images
References

http://www.cambridgeincolour.com/tutorials/image-noise.htm
http://www.clarkvision.com/imagedetail/does.pixel.size.matter2/
http://www.clarkvision.com/articles/digital.sensor.performance.summary/
http://books.google.com/books?id=OYFYt5C4N94C&pg=PA405&dq=binomial+film+grain+noise#v=onepage&q=binomial%20film%20grain%20noise&f=false
http://www.picturecode.com/noise.htm
http://www.instructables.com/id/Avoiding-Camera-Noise-Signatures/
http://www.photoxels.com/tutorial_noise.html
http://theory.uchicago.edu/~ejm/pix/20d/tests/noise/
http://www.imatest.com/docs/noise/
http://www.cambridgeincolour.com/tutorials/image-averaging-noise.htm
http://www.petapixel.com/2012/02/21/a-simple-explanation-of-how-iso-works-in-digital-photography/
Slide credits

Frédo Durand