Image Filtering

Reading: Sections 3.1, 3.2 and 10.3.1 in Szeliski book

Pixels Measure Brightness over an Area

A Digital Image is a Matrix of Pixels

Digital Images

- Sample the 2D space on a regular grid (the “pixels”)
- Quantize each sample (often, 8 or 24 bits per pixel) (the “brightness”, “intensity” or “gray level”)
- Image represented as a (row, column) matrix of integer values (in Matlab)
Perception of Intensity

Simultaneous Contrast Effect

An articulated background enhances the illusion.

Color Image

Source: E. Adelson

Source: D. Hoiem
Suppose we have an $n \times m$ RGB image called “im”
- $i$m[1,1] = top-left pixel value in R-channel
- $i$m[y, x, b] = y pixels down, x pixels to right in the bth channel
- $i$m[n, m, 3] = bottom-right pixel in B-channel

imread(filename) returns a uint8 image (values 0 to 255 for each color)
- Convert to double format (values 0 … 1) with im2double

\[ \begin{array}{cccccc}
0.79 & 0.96 & 0.86 & 0.49 & 0.71 & 0.96 \\
0.89 & 0.95 & 0.89 & 0.95 & 0.92 & 0.94 \\
0.73 & 0.49 & 0.67 & 0.81 & 0.96 & 0.81 \\
0.49 & 0.67 & 0.66 & 0.85 & 0.58 & 0.60 \\
0.71 & 0.96 & 0.89 & 0.95 & 0.92 & 0.91 \\
\end{array} \]

\[ \begin{array}{cccccc}
0.89 & 0.90 & 0.56 & 0.54 & 0.74 & 0.60 \\
0.88 & 0.51 & 0.82 & 0.94 & 0.49 & 0.93 \\
0.94 & 0.73 & 0.49 & 0.71 & 0.96 & 0.89 \\
0.49 & 0.67 & 0.66 & 0.85 & 0.58 & 0.60 \\
0.71 & 0.96 & 0.89 & 0.95 & 0.92 & 0.91 \\
\end{array} \]
Point Operations

• Map each pixel’s value to a new value
• Neighborhood is $1 \times 1$
• $g(i,j) = h(f(i,j))$ where $f$ is the input image, $g$ is the output (i.e., transformed) image, and $h$ is the point operator / transformation
• Examples
  – $g(i,j) = af(i,j) + b$ where $a > 0$ is a gain parameter and $b$ controls the brightness
  – Mapping one color space to another, e.g., RGB $\Rightarrow$ HSV
  – Image rotation, translation, scale change, ...

Instagram Filters

• How do they make those Instagram filters?

“It’s really a combination of a bunch of different methods. In some cases we draw on top of images, in others we do pixel math. It really depends on the effect we’re going for.” — Kevin Systrom, co-founder of Instagram

Example Instagram Steps

1. Perform an independent RGB color point transformation on the original image to increase contrast or make a color cast

Example Instagram Steps

2. Overlay a circle background image to create a vignette effect
Example Instagram Steps

3. Overlay a background image as decorative grain

Result

Javascript library for creating Instagram-like effects, see: http://alexmic.net/filrr/

Example Instagram Steps

4. Add a border or frame

Histogram Equalization / Flattening

• Transform image by modifying its histogram
  – create a lookup table defining $h$
• Why?
  – Image normalization
  – Comparing images
  – Contrast enhancement
• When an image’s histogram is transformed so that all gray levels occur about equally often, the result tends to produce an image with higher contrast. Why?
Histogram Equalization


Histogram Equalization Algorithm

Goal: Given an \( m \times n \) image, \( f \), with 8 bpp (gray levels 0 . . . 255), create a new image, \( g \), that has about \( mn/256 \) pixels with each gray level value

1. Compute \( f \)'s histogram: \( h(i) = \text{number of pixels with gray level } i \text{ for } i = 0..255 \)
2. Compute \( f \)'s cumulative histogram: \( c(i) = \sum_{j=0}^{i} h(j) \)
   Ideally, image \( g \) has \( c(k) = (mn/256) \times k \)
3. Compute transformation \( k = t(i) = (256/mn) \times c(i) \)
   for \( i = 0..255 \)
4. Create output image: \( g(i, j) = t(f(i, j)) \)

Note: Max value in Step 3 above is 256, but legal values are 0 . . . 255, so subtract 1

Color Transfer

- Goal: Change the colors of a given “source” image to match the color “palette” of another image in order to transfer the “mood” or “style” of one image to another
- One simple method: Shift and scale the pixel values of the target image to match the mean and standard deviation of the source image
- Color space representation of images affects quality of results. One good one: “Lab” color space.

“Lab” Color Representation

L. A transformation of the colors into a color space that is more perceptually meaningful:
   L: luminance,
   a: red – green,
   b: blue – yellow
Color Transfer

Let $I =$ Source image, i.e., one being recolored
Let $J =$ Palette image, i.e., one colors taken from
1. Convert $I$ and $J$ to $Lab$ color space; call result images $S$ and $P$, respectively
2. Compute $L$ channel output image matrix:
   \[
   L_{out} = \frac{\sigma_P}{\sigma_S} (L_S - \mu_S) + \mu_P
   \]
3. Similarly, compute $a$ and $b$ channels
4. Combine $L$, $a$ and $b$, and convert back to RGB

Local / Neighborhood Operations

• Value of pixel in output image is a function of the corresponding pixel in the input image plus other nearby pixels (usually defined by a square or rectangular “window” centered on the given pixel)

Linear Filtering

• basic idea: define a new function by averaging over a sliding window
• a simple example to start off: smoothing
Linear Filtering

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing

\[ \Sigma \]

original

smoothed

\[ \Sigma \]

original

smoothed
Linear Filtering

- Same moving average operation, expressed mathematically:

\[ b_{\text{smooth}}[i] = \frac{1}{2r + 1} \sum_{j=-r}^{r} b[j] \]

Original image

Linear Filtering

- Simple averaging:

\[ b_{\text{smooth}}[i] = \frac{1}{2r + 1} \sum_{j=-r}^{r} b[j] \]

- Convolution: same idea but with weighted average

\[ (a \ast b)[i] = \sum_j a[j] b[i - j] \]

- This is all convolution is: it is a moving weighted average

Image Filtering

- Modify the pixels in an image based on some function of a local neighborhood of the pixels
- Simplest: linear filtering
  - Replace each pixel by a linear combination of its neighbors

\[
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 1 \\
1 & 1 & 7 \\
\end{array}
\]

Some function

\[
\begin{array}{ccc}
7 \\
\end{array}
\]

Original, local image data  Output image data

Linear Functions

- Simplest: linear filtering
  - Replace each pixel by a linear combination of its neighbors

\[
\begin{array}{ccc}
10 & 5 & 3 \\
4 & 5 & 1 \\
1 & 1 & 7 \\
\end{array}
\]

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 0.5 & 0 \\
0 & 0 & 0.5 \\
\end{array}
\]

Local image data  kernel  Output image data
2D Example: Box Filter

\[
g[\cdot, \cdot] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]

Image Correlation Filtering

- Center filter \( g \) at each pixel in image \( f \)
- Multiply weights by corresponding pixels
- Set resulting value in output image \( h \)
- \( g \) is called a filter, mask, kernel, or template
- Linear filtering is sum of dot product at each pixel position
- Filtering operation called cross-correlation, and denoted \( h = f \circledast g \)

\[
h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]
\]
Image Filtering

\[ f[.,.] \quad g[.,.] \]

\[ h[.,.] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]

Credit: S. Seitz

\[ g[.,.] \]

\[ h[.,.] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]

Credit: S. Seitz
Box Filter

What does it do?
• Replaces each pixel with an average (arithmetic mean) of its neighborhood
• Achieves smoothing effect (i.e., removes sharp features)
• Weaknesses:
  • Blocky results
  • Axis-aligned streaks

\[
g_{\text{Box Filter}}(\cdot, \cdot) = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]

Credit: D. Lowe

Properties of Smoothing Filters

• Values all positive
• Sum to 1 ⇒ constant regions same as input
• Amount of smoothing proportional to mask size
• Removes “high-frequency” components
• “low-pass” filter

Smoothing with Box Filter

Practice with Linear Filters

Credit: D. Lowe
Practice with Linear Filters

Original

Filtered (no change)

Credit: D. Lowe

Practice with Linear Filters

Original

Shifted left by 1 pixel

Credit: D. Lowe

Practice with Linear Filters

Original

(Note that filter sums to 1)

Credit: D. Lowe
Practice with Linear Filters

Image Sharpening filter
• Sharpen an out of focus image by subtracting a blurred version

Image Sharpening

Image Sharpening by Unsharp Masking
• \( h = f - k(f * g) \) where \( k \) is a small positive constant and \( g = \)

\[
\begin{array}{ccc}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0 \\
\end{array}
\]

called a Laplacian mask

• Why does it work?
• Say \( f \) is a blurred image produced from an ideal image \( p \) by convolving it with a box filter \( s = \)

\[
\begin{array}{ccc}
0 & 1/5 & 0 \\
1/5 & 1/5 & 1/5 \\
0 & 1/5 & 0 \\
\end{array}
\]

• Simulates \textit{Mach Band} effect in human vision
• Called \textit{unsharp masking} in photography

Sharpening using Unsharp Mask Filter
Unsharp Masking

Mach Bands

Actual brightness

Perceived by you

Mach Bands and the Cornsweet Illusion

Center/Surround Cells in Retina

Organization of ganglion cells in retina respond to discontinuities of light (S. Kuffler, 1953)

On-Center

- 

+ 

- 

Off-Center

+ 

- 

+ 

Credit: E. Adelson
E. Lingelbach’s version of “Hermann Grid”

Lateral Inhibition Explanation

Inhibited less by white stripes
Inhibited more by white stripes

Lateral Inhibition in HVS

Simultaneous Contrast

In Van Gogh’s cafes, the contrast between yellow & blue, and red & green, cause both colors to seem more vivid. Color perception always depends on the other colors nearby.
Cross-Correlation vs. Convolution

- \( g = \text{filter} \)  
- \( f = \text{image} \)

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]

- 2D convolution

\[ h[m,n] = \sum_{k,l} g[k,l] f[m-k,n-l] \]

Example of 1D Convolution

\[ h(4) = \sum_{i=0}^{2} g(i)f(4-i) \]
\[ = g(-2)f(6) + g(-1)f(5) + g(0)f(4) + g(1)f(3) + g(2)f(2) \]
\[ = (1)(12) + (3)(23) + (5)(22) + (3)(8) + (1)(7) \]
\[ = 222 \]
**ConvoluZon**

ConvoluZon is equivalent to: Flipping the image, \( f \), in both dimensions (bottom to top, right to left), and then performing cross-correlaZon with \( g \); (or, flipping filter, \( g \), and then applying cross-correlaZon with \( f \))

\[
h[m,n] = \sum_{k,l} g[k,l] f[m-k,n-l]
\]

\[h = g \ast f\]

Credit: K. Grauman

**Key Properties of Linear Filters**

**Linearity:**

\[\text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2)\]

**Shift invariance:** same behavior regardless of pixel location

\[\text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f))\]

Any linear, shift-invariant (LSI) operator can be represented as a convolution operation

Credit: S. Lazebnik

**More Properties**

- **Commutative:** \( a \ast b = b \ast a \)
  - Conceptually no difference between filter and image

- **Associative:** \( a \ast (b \ast c) = (a \ast b) \ast c \)
  - Often apply several filters one after another: \( ((a \ast b_1) \ast b_2) \ast b_3 \)
  - This is equivalent to applying one filter: \( a \ast (b_1 \ast b_2 \ast b_3) \)

- **Distributes over addition:** \( a \ast (b + c) = (a \ast b) + (a \ast c) \)

- **Scalars factor out:** \( ka \ast b = a \ast kb = k (a \ast b) \)

- **Identity:** unit impulse \( e = [0, 0, 1, 0, 0] \) \( \Rightarrow a \ast e = a \)

Credit: S. Lazebnik

**Gaussian Filtering**

- Weight contributions of neighboring pixels by distance from center pixel

\[
G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}
\]

\[
(x, y) \text{ are coordinates of pixel, where center pixel is at (0, 0)}
\]

\[
5 \times 5, \sigma = 1
\]

Constant factor in front makes it sum to 1

Credit: C. Rasmussen
Smoothing with a Gaussian

- Smoothing with a box filter doesn’t model well a defocused lens
- Most obvious difference is that a single point of light viewed in a defocused lens looks like a fuzzy blob; but the averaging process would give a little square
- Gaussian is isotropic (i.e., rotationally symmetric)

A Gaussian gives a good model of a fuzzy blob
- It closely models many physical processes (the sum of many small effects)

Credit: D. Forsyth

What does Blurring take away?

original

smoothed (5 x 5 Gaussian)

Smoothing with Gaussian Filter
Smoothing with Box Filter

input

box average

Gaussian blurred

Credit: S. Paris
Gaussian Filters

- What parameters matter here?
- **Standard deviation**, \( \sigma \), of Gaussian: determines amount of smoothing

\[
\sigma = 2 \text{ with } 30 \times 30 \text{ kernel} \quad \sigma = 5 \text{ with } 30 \times 30 \text{ kernel}
\]

Credit: D. Hoiem

Smoothing with a Gaussian

Parameter \( \sigma \) is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing

for sigma=1:3:10
    h = fspecial('gaussian', hsize, sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end

Credit: D. Forsyth
Gaussian Filters

Medium $\sigma$

Large $\sigma$

Spatial Resolution and Color

original

Credit: D. Forsyth
Blurring the G Component

Blurring the R Component

Blurring the B Component

Cascading Gaussian Filters

- Removes “high-frequency” information from the image (“low-pass” filter)
- Convolution of two Gaussians is another Gaussian

\[ \ast \]

- Convolving two times with Gaussian of size \( \sigma \) is same as convolving once with Gaussian of size \( \sigma \sqrt{2} \)

Credit: K. Grauman
**Separability**

- A function \( g(x, y) \) is separable if \( g(x, y) = g_1(x) g_2(y) \)
- The Gaussian function is separable:
  \[
  e^{-[(x^2 + y^2)/2\sigma^2]} = e^{-[x^2/2\sigma^2]} \cdot e^{-[y^2/2\sigma^2]}
  \]
- First convolve the image with a 1D horizontal filter
- Then convolve the result of the first convolution with a 1D vertical filter
- For a \( k \times k \) Gaussian filter, 2D convolution requires \( k^2 \) multiplications and \( k^2-1 \) additions per pixel
- But using the separable filters, we reduce this to \( 2k \) multiplications and \( 2k-1 \) additions per pixel
- Matlab: \( \text{h} = \text{conv2}(g_1, g_2, f) \);

**Separable filtering**

\[
a_2[i, j] = a_1[i] a_1[j]
\]
Separable filtering

\[ a_2[i, j] = a_1[i]a_1[j] \]

\[ \sum_{i'} a_1[i'] \left( \sum_{j'} a_1[j'] b[i - i', j - j'] \right) \]

Which is faster? \( O(2(n + 1)) \) or \( O((n + 1)^2) \)

Gaussian Filters

- What other parameter(s) must be set?
- Size of kernel or mask

\( \sigma = 5 \) with 10 x 10 kernel

\( \sigma = 5 \) with 30 x 30 kernel

How Big should the Filter be?

- Gaussian function has infinite “support” but need a finite-size kernel
- Values at edges should be near 0
- \( \sim 98.8\% \) of area under Gaussian in mask of size \( 5\sigma \times 5\sigma \)
- In practice, use mask of size about \( 2k+1 \times 2k+1 \) where \( k = 3\sigma \)
- Multiply real values of Gaussian by a scale factor (\( = \min \) real value) to obtain integer weights
- Normalize output by dividing by sum of all weights
Gaussian Filter

3 x 3 approximation of a Gaussian:

```
  1  2  1
1/16
  2  4  2
  1  2  1
```

1D separable kernel for \( \sigma = 1 \): \( \frac{1}{38} [1, 9, 18, 9, 1] \)

Matlab Implementation

```
>> hsize = 10;
>> sigma = 5;
>> h = fspecial('gaussian', hsize, sigma);

>> mesh(h);
>> imagesc(h);

>> outim = imfilter(im, h);  % correlation
>> imshow(outim);
```

What is the Size of the Output Image?

- MATLAB: `filter2(g, f, shape)`
  - `shape = 'full'`: output size is sum of sizes of \( f \) and \( g \)
  - `shape = 'same'`: output size is same as \( f \)
  - `shape = 'valid'`: output size is difference of sizes of \( f \) and \( g \)

What about Near the Image Border?

- the filter window falls off the border of the image
- need to extrapolate
- methods:
  - clip filter (black)
  - wrap around
  - copy border
  - reflect across border
What about Near the Image Border?

- methods (MATLAB):
  - clip filter (black): \(\text{imfilter}(f, g, 0)\)
  - wrap around: \(\text{imfilter}(f, g, \text{’circular’})\)
  - copy border: \(\text{imfilter}(f, g, \text{’replicate’})\)
  - reflect across border: \(\text{imfilter}(f, g, \text{’symmetric’})\)

Credit: S. Marschner

Application: Filter Banks for Feature Detection

Filter Bank (48 filters)

Code for filter banks: [www.robots.ox.ac.uk/~vgg/research/texclass/filters.html](http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html)

Filter Banks

Process image with each filter and keep responses (or squared or abs value of responses)

Application: Representing Texture

Source: Forsyth
How can we Represent Texture?

- Textures are characterized by the material, orientation, scale and other regular or statistical properties
- Describe a region’s texture as a feature vector defined by measuring the responses of blobs and edges at various orientations and scales
- Record simple statistics (e.g., mean, std) of absolute filter responses

Can you Match the Texture to the Responses?

Application: Hybrid Images


Project Instructions:
http://www.cs.illinois.edu/class/fa10cs498/dah/projects/hybrid/ComputationalPhotography_ProjectHybrid.html
**What Makes the Mona Lisa Smile?**

The smile only becomes apparent if a viewer looks at her eyes or elsewhere on her face; the smile disappears when looking directly at her mouth.

Peripheral vision is low resolution and blurs, picking up shadows from the Mona Lisa's cheekbones, which suggests the curvature of a smile.

*The elusive quality of the Mona Lisa's smile can be explained by the fact that her smile is almost entirely in low spatial frequencies, and so is seen best by your peripheral vision.*

---

— Margaret Livingstone
Non-Linear Filtering

Median Filter

- Replace pixel by the median value of its neighbors
- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Nonlinear operation

Median Filter

Salt and pepper noise

Median filtered

Plots of 1 row in the images

Matlab: output im = medfilt2(im, [h w]);

Credit: M. Hebert

Median Filter

- Median filter is edge preserving
Bilateral Filter for Image Smoothing and “Tooning”

- Idea: Edge-preserving smoothing filter, i.e., smooth image but preserve edges, using a weighted average of pixels

- Weight pixels by spatial distance and intensity difference

Basis for many apps such as ToonPaint
In weighting the neighbors of pixel $p$, we would like to preserve the "step" in brightness. Smoothing using Gaussian weights is given by:

$$W_{\sigma_s}(q) = \exp\left(-\frac{||p - q||^2}{2\sigma_s^2}\right)$$

The Gaussian filtered result is calculated as:

$$I(q) = I(p) \cdot W_{\sigma_s}(q)$$

Credit: J. Chai
Edges Are Smoothed

$W_{c_e}(q) = \exp\left(-\frac{||p-q||}{2\sigma_f^2}\right)$

What Causes the Problem?

$W_{c_e}(q) = \exp\left(-\frac{||p-q||}{2\sigma_s^2}\right)$

What Causes the Problem?

Same weights for these two pixels

$W_{c_e}(q) = \exp\left(-\frac{||p-q||}{2\sigma_f^2}\right)$

The Kernel Weights

Green weights:

$W_{c_e}(q) = \exp\left(-\frac{(I(p)-I(q))^2}{2\sigma_t^2}\right)$

$W_{c_e}(q) = \exp\left(-\frac{||p-q||}{2\sigma_s^2}\right)$
Influence of Pixels

Only pixels close in space and in brightness are used.

\[ h(p) = \frac{1}{W_p} \sum_{q \in N(p)} G_{\sigma_s}(|p-q|) G_{\sigma_r}(|I(p)-I(q)|) I(q) \]

- \( h(p) \) is the output of the bilateral filter.
- \( p = (x_p, y_p) \) is the input pixel.
- \( q = (x_q, y_q) \) are the pixels in the neighborhood of \( p \).
- \( N(p) \) is the set of pixels in the neighborhood of \( p \).
- \( G_{\sigma_s} \) and \( G_{\sigma_r} \) are Gaussian functions for spatial and brightness weighting, respectively.
- \( W_p \) is the normalization factor.

Credit: J. Chai
Bilateral Filter for Image Smoothing

Original  Bilateral filtered  Bilateral filter applied to log-intensity image

Bilateral filter implemented in Photoshop as “Surface Blur” tool

Top row: original; sharpened with Gaussian; sharpened with median (note fewer halo artifacts.) Middle row: Filtered at 20th; 50th [median]; and 80th percentiles. Bottom row: “High Pass” using median; bilateral smoothing filter; logarithmic bilateral filter.

Image Abstraction and “Tooning”

[Winnebmöller, Olsen, Gooch, 2006]
Cartoon Rendition
(Winnemöller 98)

Examples

Real-Time Video Abstraction