Image Filtering

Computer Vision

James Hays

Many slides by Derek Hoiem
Recap: Light and Sensors
Pinhole camera

\[ f = \text{focal length} \]
\[ c = \text{center of the camera} \]

Figure from Forsyth
Some examples of the reflectance spectra of surfaces

<table>
<thead>
<tr>
<th>Wavelength (nm)</th>
<th>Red</th>
<th>Yellow</th>
<th>Blue</th>
<th>Purple</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>700</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

% Photons Reflected

© Stephen E. Palmer, 2002
Physiology of Color Vision

Three kinds of cones:

- Why are M and L cones so close?
- Why are there 3?
The Retina

Cross-section of eye

Cross section of retina

Ganglion axons
Ganglion cell layer
Bipolar cell layer
Receptor layer
Pigmented epithelium

Practical Color Sensing: Bayer Grid

• Estimate RGB at ‘G’ cells from neighboring values

Slide by Steve Seitz
Color Image
Images in Matlab

- Images represented as a matrix
- Suppose we have a NxM RGB image called “im”
  - \( \text{im}(1,1,1) = \) top-left pixel value in R-channel
  - \( \text{im}(y, x, b) = \) \( y \) pixels down, \( x \) pixels to right in the \( b \text{th} \) channel
  - \( \text{im}(N, M, 3) = \) bottom-right pixel in B-channel
- \text{imread(filename)} returns a uint8 image (values 0 to 255)
  - Convert to double format (values 0 to 1) with \text{im2double}
From the 3D to 2D

Let's now focus on 2D
Extract building blocks
Extract useful building blocks
The big picture...

Feature Detection

- e.g. DoG

Feature Description

- e.g. SIFT

Database of local descriptors

Matching / Indexing / Detection

Slide credit Fei Fei Li
Image Filtering

Computer Vision
James Hays
Upcoming classes: three views of filtering

• Image filters in spatial domain
  – Filter is a mathematical operation of a grid of numbers
  – Smoothing, sharpening, measuring texture

• Image filters in the frequency domain
  – Filtering is a way to modify the frequencies of images
  – Denoising, sampling, image compression

• Templates and Image Pyramids
  – Filtering is a way to match a template to the image
  – Detection, coarse-to-fine registration
Image filtering

- Image filtering: compute function of local neighborhood at each position

- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
  - Deep Convolutional Networks
Example: box filter

\[
g[\cdot, \cdot] = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]
Image filtering

\[ f[\ldots] \]

\[ g[\cdot, \cdot] \]

\[ h[\ldots] \]

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

Credit: S. Seitz
Image filtering

\[ f[\ldots] \]

\[ h[\ldots] \]

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

\[ f[\cdot, \cdot] \]

\[ g[\cdot, \cdot] = \frac{1}{9} \]

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

Credit: S. Seitz
Image filtering

\[ f[\cdot, \cdot] \]

\[ h[\cdot, \cdot] \]

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

\[ f[\cdot,\cdot] \]

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

\[
f[\cdot, \cdot]
\]

\[
h[\cdot, \cdot]
\]

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

Credit: S. Seitz
Image filtering

\[ f[\cdot, \cdot] \]

\[ h[\cdot, \cdot] \]

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

Credit: S. Seitz
Image filtering

\[ f[\cdot,\cdot] \]

\[ h[\cdot,\cdot] \]

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

Credit: S. Seitz
What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)
Smoothing with box filter
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

Filtered (no change)

Source: D. Lowe
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

Shifted left
By 1 pixel

Source: D. Lowe
Practice with linear filters

Original

(Note that filter sums to 1)
Practice with linear filters

Original

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Other filters

Sobel

Vertical Edge (absolute value)
Other filters

Sobel

1 2 1
0 0 0
-1 -2 -1

Horizontal Edge (absolute value)
Filtering vs. Convolution

• 2d filtering
  - \( h = \text{filter2}(f, I); \) or
  - \( h = \text{imfilter}(I, f); \)
  \[
  h[m, n] = \sum_{k,l} f[k, l] I[m+k, n+l]
  \]

• 2d convolution
  - \( h = \text{conv2}(f, I); \)
  \[
  h[m, n] = \sum_{k,l} f[k, l] I[m-k, n-l]
  \]
Key properties of linear filters

**Linearity:**
\[
\text{imfilter}(I, f_1 + f_2) = \text{imfilter}(I,f_1) + \text{imfilter}(I,f_2)
\]

**Shift invariance:** same behavior regardless of pixel location
\[
\text{imfilter}(I,\text{shift}(f)) = \text{shift}(\text{imfilter}(I,f))
\]

Any linear, shift-invariant operator can be represented as a convolution

Source: S. Lazebnik
More properties

• Commutative: \(a * b = b * a\)
  – Conceptually no difference between filter and signal
  – But particular filtering implementations might break this equality

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

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

• Scalars factor out: \(k a * b = a * k b = k (a * b)\)

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

Source: S. Lazebnik
Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness

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

Slide credit: Christopher Rasmussen
Smoothing with Gaussian filter
Smoothing with box filter
Gaussian filters

• Remove “high-frequency” components from the image (low-pass filter)
  – Images become more smooth

• Convolution with self is another Gaussian
  – So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  – Convolving two times with Gaussian kernel of width $\sigma$ is same as convolving once with kernel of width $\sigma\sqrt{2}$

• *Separable* kernel
  – Factors into product of two 1D Gaussians

Source: K. Grauman
Separability of the Gaussian filter

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \]

\[ = \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)\right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right)\right) \]

The 2D Gaussian can be expressed as the product of two functions, one a function of \(x\) and the other a function of \(y\).

In this case, the two functions are the (identical) 1D Gaussian.

Source: D. Lowe
Separability example

2D convolution (center location only)

\[
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1
\end{bmatrix}
\ast
\begin{bmatrix}
2 & 3 & 3 \\
3 & 5 & 5 \\
4 & 4 & 6
\end{bmatrix}
\]

The filter factors into a product of 1D filters:

Perform convolution along rows:

\[
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1
\end{bmatrix}
\ast
\begin{bmatrix}
1 \\
2 \\
1
\end{bmatrix}
= \begin{bmatrix}
11 \\
18 \\
18
\end{bmatrix}
\]

Followed by convolution along the remaining column:

Source: K. Grauman
Separability

• Why is separability useful in practice?
Some practical matters
Practical matters

How big should the filter be?

• Values at edges should be near zero
• Rule of thumb for Gaussian: set filter half-width to about $3\sigma$
Practical matters

• What about near the edge?
  – the filter window falls off the edge of the image
  – need to extrapolate
  – methods:
    • clip filter (black)
    • wrap around
    • copy edge
    • reflect across edge

Source: S. Marschner
Next class: Thinking in Frequency