Pixels and Image Filtering

Computational Photography
Derek Hoiem

Graphic: http://www.notcot.org/post/4068/
Administrative stuff

• Any questions?

• Matlab/algebra tutorial
  • Thursday (9/1): 5-6:30pm, SC 3403
Today’s Class: Pixels and Linear Filters

• What is a pixel? How is an image represented?

• What is image filtering and how do we do it?

• Introduce Project 1: Hybrid Images
Next three classes

• Image filters in spatial domain
  – Smoothing, sharpening, measuring texture

• Image filters in the frequency domain
  – Denoising, sampling, image compression

• Templates and Image Pyramids
  – Detection, coarse-to-fine registration
Image Formation

Illumination (energy) source

Imaging system

Scene element

(Internal) image plane
Digital camera

A digital camera replaces film with a sensor array

- Each cell in the array is light-sensitive diode that converts photons to electrons
- Two common types: Charge Coupled Device (CCD) and CMOS
Sensor Array

**FIGURE 2.17** (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.
The raster image (pixel matrix)
The raster image (pixel matrix)

<table>
<thead>
<tr>
<th>0.92</th>
<th>0.93</th>
<th>0.94</th>
<th>0.97</th>
<th>0.62</th>
<th>0.37</th>
<th>0.85</th>
<th>0.97</th>
<th>0.93</th>
<th>0.92</th>
<th>0.99</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.95</td>
<td>0.89</td>
<td>0.82</td>
<td>0.89</td>
<td>0.56</td>
<td>0.31</td>
<td>0.75</td>
<td>0.92</td>
<td>0.81</td>
<td>0.95</td>
<td>0.91</td>
</tr>
<tr>
<td>0.89</td>
<td>0.72</td>
<td>0.51</td>
<td>0.55</td>
<td>0.51</td>
<td>0.42</td>
<td>0.57</td>
<td>0.41</td>
<td>0.49</td>
<td>0.91</td>
<td>0.92</td>
</tr>
<tr>
<td>0.96</td>
<td>0.95</td>
<td>0.88</td>
<td>0.94</td>
<td>0.56</td>
<td>0.46</td>
<td>0.91</td>
<td>0.87</td>
<td>0.90</td>
<td>0.97</td>
<td>0.95</td>
</tr>
<tr>
<td>0.71</td>
<td>0.81</td>
<td>0.81</td>
<td>0.87</td>
<td>0.57</td>
<td>0.37</td>
<td>0.80</td>
<td>0.88</td>
<td>0.89</td>
<td>0.79</td>
<td>0.85</td>
</tr>
<tr>
<td>0.49</td>
<td>0.62</td>
<td>0.60</td>
<td>0.58</td>
<td>0.50</td>
<td>0.60</td>
<td>0.58</td>
<td>0.50</td>
<td>0.61</td>
<td>0.45</td>
<td>0.33</td>
</tr>
<tr>
<td>0.86</td>
<td>0.84</td>
<td>0.74</td>
<td>0.58</td>
<td>0.51</td>
<td>0.39</td>
<td>0.73</td>
<td>0.92</td>
<td>0.91</td>
<td>0.49</td>
<td>0.74</td>
</tr>
<tr>
<td>0.96</td>
<td>0.67</td>
<td>0.54</td>
<td>0.85</td>
<td>0.48</td>
<td>0.37</td>
<td>0.88</td>
<td>0.90</td>
<td>0.94</td>
<td>0.82</td>
<td>0.93</td>
</tr>
<tr>
<td>0.69</td>
<td>0.49</td>
<td>0.56</td>
<td>0.66</td>
<td>0.43</td>
<td>0.42</td>
<td>0.77</td>
<td>0.73</td>
<td>0.71</td>
<td>0.90</td>
<td>0.99</td>
</tr>
<tr>
<td>0.79</td>
<td>0.73</td>
<td>0.90</td>
<td>0.67</td>
<td>0.33</td>
<td>0.61</td>
<td>0.69</td>
<td>0.79</td>
<td>0.73</td>
<td>0.93</td>
<td>0.97</td>
</tr>
<tr>
<td>0.91</td>
<td>0.94</td>
<td>0.89</td>
<td>0.49</td>
<td>0.41</td>
<td>0.78</td>
<td>0.78</td>
<td>0.77</td>
<td>0.89</td>
<td>0.99</td>
<td>0.93</td>
</tr>
</tbody>
</table>
Perception of Intensity

from Ted Adelson
Perception of Intensity

from Ted Adelson
Digital Color Images

Bayer filter

© 2000 How Stuff Works
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 \( b \)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}
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
Example: box filter
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

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

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

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

Credit: S. Seitz
Image filtering

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

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

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

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

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

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

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

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

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

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

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

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

Credit: S. Seitz
Image filtering

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

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

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

Credit: S. Seitz
Box Filter

What does it do?

• Replaces each pixel with an average of its neighborhood

• Achieve smoothing effect (remove sharp features)

Slide credit: David Lowe (UBC)
Smoothing with box filter
One more on board...
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

Filtered
(no change)

Source: D. Lowe
Practice with linear filters

Original

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

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)

Source: D. Lowe
Practice with linear filters

Original

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Other filters

Sobel

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

Vertical Edge (absolute value)
Other filters

Sobel

Horizontal Edge (absolute value)
theta = 30; len = 20;
fil = imrotate(ones(1, len), theta, 'bilinear');
fil = fil / sum(fil(:));
figure(2), imshow(imfilter(im, fil));
Filtering vs. Convolution

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

- 2d convolution
  - \( h = \text{conv2}(g, f); \)
  
  \[
  h[m, n] = \sum_{k,l} g[k, l] f[m - k, n - l]
  \]
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 operator can be represented as a convolution

Source: S. Lazebnik
More properties

• Commutative: \( a * b = b * a \)
  – Conceptually no difference between filter and signal

• 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: \( ka * b = a * kb = k (a * b) \)

• 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

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

5 x 5, \( \sigma = 1 \)

0.003 0.013 0.022 0.013 0.003
0.013 0.059 0.097 0.059 0.013
0.022 0.097 0.159 0.097 0.022
0.013 0.059 0.097 0.059 0.013
0.003 0.013 0.022 0.013 0.003

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
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 function.
Separability example

2D convolution (center location only)

The filter factors into a product of 1D filters:

Perform convolution along rows:

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 is the size of the output?

• MATLAB: filter2(g, f, \textit{shape})
  
  – \textit{shape} = ‘full’: output size is sum of sizes of \textit{f} and \textit{g}
  
  – \textit{shape} = ‘same’: output size is same as \textit{f}
  
  – \textit{shape} = ‘valid’: output size is difference of sizes of \textit{f} and \textit{g}

Source: S. Lazebnik
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
Practical matters

– methods (MATLAB):
  • clip filter (black): \texttt{imfilter}(f, g, 0)
  • wrap around: \texttt{imfilter}(f, g, ‘circular’)
  • copy edge: \texttt{imfilter}(f, g, ‘replicate’)
  • reflect across edge: \texttt{imfilter}(f, g, ‘symmetric’)

Source: S. Marschner
Application: Representing Texture
Texture and Material

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/
Texture and Orientation

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/
Texture and Scale

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/
What is texture?

Regular or stochastic patterns caused by bumps, grooves, and/or markings
How can we represent texture?

- Compute responses of blobs and edges at various orientations and scales
Overcomplete representation: filter banks

LM Filter Bank

Code for filter banks: www.robots.ox.ac.uk/~vgg/research/texclass/filters.html
Filter banks

• Process image with each filter and keep responses (or squared/abs responses)
How can we represent texture?

• Measure 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 response?

Filters

1

2

3

Mean abs responses

A

B

C
Representing texture by mean abs response

Filters

Mean abs responses
Project 1: Hybrid Images

Gaussian Filter!

Laplacian Filter!


Project Instructions:
http://www.cs.illinois.edu/class/fa10/cs498dwh/projects/hybrid/ComputationalPhotography_ProjectHybrid.html
Take-home messages

• Image is a matrix of numbers

• Linear filtering is sum of dot product at each position
  – Can smooth, sharpen, translate (among many other uses)

• Be aware of details for filter size, extrapolation, cropping

• Start thinking about project (read the paper, create a test project page)
Take-home questions

1. Write down a 3x3 filter that returns a positive value if the average value of the 4-adjacent neighbors is less than the center and a negative value otherwise.

2. Write down a filter that will compute the gradient in the x-direction:

   \[
   \text{grad}_x(y, x) = \text{im}(y, x+1) - \text{im}(y, x) \quad \text{for each } x, y
   \]
Take-home questions

3. Fill in the blanks:

a) \(_ = D \times B\)

b) \(A = _ \times _\)

c) \(F = D \times _\)

d) \(_ = D \times D\)
Next class: Thinking in Frequency