BBM 444 – Week 3
Image Processing Basics

Aykut Erdem
Computer Vision Lab (CVL)
Hacettepe University

Photo from the book
Why Cats Paint
by H. Busch and B. Silver
Image Formation

Illumination (energy) source

Scene element

Imaging system

(Internal) image plane
Sampling and Reconstruction
Sampled representations

- How to store and compute with continuous functions?
- Common scheme for representation: samples
  - write down the function’s values at many points
Reconstruction

• Making samples back into a continuous function
  – for output (need realizable method)
  – for analysis or processing (need mathematical method)
  – amounts to “guessing” what the function did in between
A Digital Image is a Matrix of Pixels

Slide credit: Derek Hoiem
A Digital Image is a Matrix of Pixels
What is in an image?

The input is just an array of brightness values; humans perceive structure in it.
Color Image
Images in Matlab

- Images represented as a matrix
- Suppose we have an $n \times m$ RGB image called “im”
  - $im(1, 1, 1) = \text{top-left pixel value in } R \text{ channel}$
  - $im(y, x, b) = y \text{ pixels down, } x \text{ pixels to right in the } b^{th} \text{ channel}$
  - $im(n, m, 3) = \text{bottom-right pixel in } B \text{ channel}$
- `imread(filename)` returns a `uint8` image (values 0 to 255)
  - Convert to double format (values 0 ... 1) with `im2double`
Basic types of operations

Point operations: range only
\[ g(x, y) = f(t_x(x, y), t_y(x, y)) \]

Domain operations
\[ g(x, y) = f(t_{x,y}(x, y)) \]

Neighborhood operations: domain and range

Slide credit: Derek Hoiem
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, …
Histogram

• For each value (e.g. 0-255) how many pixels have this value?
• Cumulative histogram: for each value $x$, how many pixels have a value smaller than $x$?
Very useful on cameras

- Allows you to tell if you use the dynamic range entirely.
Bad: bright values under-used (underexposure)

Bad: bright values saturate (overexposure)

http://www.luminous-landscape.com/tutorials/understanding-series/understanding-histograms.shtml

Slide credit: Fredo Durand
Histogram equalization

• Given image with given histogram
• monotonic remapping to get a flat histogram

![Histogram equalization diagram](http://en.wikipedia.org/wiki/Histogram_equalization)
Histogram Equalization

Slide credit: Derek Hoiem
Histogram Equalization Algorithm

**Goal:** Given \( n \times m \) image \( f \) with 8 bpp (brightness values 0 to 255), create a new image \( g \) that has about \( nm/255 \) pixels of each brightness value.

- Compute \( f \)’s histogram: \( H(i), 0 \leq i \leq 255 \)
- Compute \( f \)’s cumulative histogram:
- Compute mapping and output image: \( C(i) = \sum_{j=0}^{i} H(j) \)

\[
g(i, j) = \text{round}\left(255 \frac{C(f(i, j)) - C_{\text{min}}}{nm - C_{\text{min}}} \right)
\]

where \( C_{\text{min}} = \) minimum non-zero value in \( C \)
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

• Same moving average operation, expressed mathematically:

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

<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

Some function

| 7 |

Modified image data

Slide credit: Derek Hoiem
Linear Functions

• Simplest: linear filtering
  – *Replace each pixel by a linear combination of its neighbors*

```
<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 | kernel | Modified image data

Slide credit: Derek Hoiem
2D Example: Box Filter

$g[\cdot , \cdot ]$

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

Slide credit: David Lowe
Image Filtering

\[ f[\ldots] \]

\[ h[\ldots] \]

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

Slide credit: Steven Seitz
Image Filtering

\[ f[\ldots] \]

\[ h[\ldots] \]

\[ g[\ldots] \]

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

Slide credit: Steven Seitz
Image Filtering

\[ f[\ldots] \]

\[ h[\ldots] \]

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

Slide credit: Steven Seitz
Image Filtering

\[ f[\ldots] \]

\[ \begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 0 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \]

\[ \begin{array}{cccccccccc}
0 & 10 & 20 & 30 & \\
0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \\
\end{array} \]

\[ g[\ldots] \quad \frac{1}{9} \]

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

Slide credit: Steven Seitz
Image Filtering

\[ f[\ldots] \]

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

Slide credit: Steven 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

\[ f[\ldots] \]

\[ h[\ldots] \]

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

Slide credit: Steven Seitz
Image Filtering

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

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

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

Slide credit: Steven Seitz
Box Filter

What does it do?

• Replaces each pixel with an average of its neighborhood

• Achieves smoothing effect (i.e., removes sharp features)

• Weaknesses:
  • Blocky results
  • Axis-aligned streaks

\[
g[\cdot, \cdot] = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]
Smoothing with Box Filter
Properties of Smoothing Filters

• Smoothing
  – 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
Practice with Linear Filters

Original

?
Practice with Linear Filters

Original

Filtered (no change)

Slide credit: David Lowe
Practice with Linear Filters

Original

Slide credit: David Lowe
Practice with Linear Filters

Original

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

Shifted *left* by 1 pixel

Slide credit: David Lowe
Practice with Linear Filters

Original

(Note that filter sums to 1)
Practice with Linear Filters

Original

Sharpening filter

- Sharpen an out of focus image by subtracting a blurred version

Slide credit: David Lowe
Sharpening

before

after
Sharpening by Unsharp Masking

- $h = f - k(f \ast g)$ where $k$ is a small positive constant and $g =$ \begin{array}{c|c|c} 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}{c|c|c} 0 & 1/5 & 0 \\ 1/5 & 1/5 & 1/5 \\ 0 & 1/5 & 0 \end{array}
- $p \ast g \propto p - (p \ast s) = p - f$
- $h = f - k(f \ast g) \approx f - k((-1/5(p - f))) \approx p$

- Simulates Mach Band effect in human vision
- Called unsharp masking in photography

Slide credit: Derek Hoiem
Sharpening using Unsharp Mask Filter

Original

Filtered result

Slide credit: Derek Hoiem
Unsharp Masking
Other Filters: Edge Detection

Sobel

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

Vertical edges (absolute value)

Slide credit: Derek Hoiem
Other Filters: Edge Detection

Sobel

Horizontal edges (absolute value)

Slide credit: Derek Hoiem
Cross-Correlation vs. Convolution

• 2D filtering / cross-correlation
  - \( h = \text{filter2}(g, f); \) or
  - \( h = \text{imfilter}(f, g); \)

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

• 2D convolution
  - \( h = \text{conv2}(g, f); \)

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

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

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

\[h = g \ast f\]

*Notation for convolution operator*

Slide credit: Kristen 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 operator can be represented as a *convolution* operation
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 \( e = [0, 0, 1, 0, 0] \Rightarrow a \ast e = a \)
Gaussian Filtering

- Weight contributions of neighboring pixels by distance from center pixel

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

- Constant factor at front makes volume sum to 1
- Convolve each row of image with 1D kernel to produce new image; then convolve each column of new image with same 1D kernel to yield output image

\[
\begin{bmatrix}
0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\
0.013 & 0.059 & 0.097 & 0.059 & 0.013 \\
0.022 & 0.097 & 0.159 & 0.097 & 0.022 \\
0.013 & 0.059 & 0.097 & 0.059 & 0.013 \\
0.003 & 0.013 & 0.022 & 0.013 & 0.003
\end{bmatrix}
\]

5 x 5, \( \sigma = 1 \)
Smoothing with a Gaussian

- Smoothing with a box actually doesn’t compare at all well with 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)
What does Blurring take away?

original
What does Blurring take away?

smoothed (5x5 Gaussian)

Slide credit: Derek Hoiem
Smoothing with Gaussian Filter
Smoothing with Box Filter
box average
Gaussian blur

Slide credit: Sylvain Paris
Gaussian Filters

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

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

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

Slide credit: Derek Hoiem
Effect of $\sigma$
Smoothing with a Gaussian

- Parameter $\sigma$ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing

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

Slide credit: Derek Hoiem
Small $\sigma$
Medium $\sigma$
Large $\sigma$
Gaussian Filters

\( \sigma = 1 \) pixel \hspace{1cm} \( \sigma = 5 \) pixels \hspace{1cm} \( \sigma = 10 \) pixels \hspace{1cm} \( \sigma = 30 \) pixels

Slide credit: Derek Hoiem
Spatial resolution and color

original

R

G

B

Slide credit: Derek Hoiem
Blurring the G component
Blurring the $R$ component

original

processed

Slide credit: Derek Hoiem
Blurring the $B$ component

original  processsed

$R$

$G$

$B$

Slide credit: Derek Hoiem
Lab Color Component

- A rotation of the color coordinates into directions that are more perceptually meaningful:
  
  \[ L \]
  \[ a \]
  \[ b \]

  - \( L \): luminance,
  - \( a \): red-green,
  - \( b \): blue-yellow

Slide credit: Derek Hoiem
Blurring $L$

original

processed

Slide credit: Derek Hoiem
Blurring $a$

original

processed

Slide credit: Derek Hoiem
Blurring $b$
Cascading Gaussian Filters

- Removes “high-frequency” components from the image (low-pass filter)

- Convolution of two Gaussians is another Gaussian

\[ * \]  

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

Slide credit: Kristen Grauman
Gaussian Filters

- What parameters matter here?
- **Size** of kernel or mask

\[
\sigma = 5 \text{ with } 10 \times 10 \text{ kernel} \quad \text{and} \quad \sigma = 5 \text{ with } 30 \times 30 \text{ kernel}
\]
How big should the filter be?

- Gaussian function has infinite “support” but need a finite-size kernel
- Values at edges should be near 0
- \(~98.8\%\) of area under Gaussian in mask of size \(5\sigma \times 5\sigma\)
- In practice, use mask of size \(2(k+1) \times 2(k+1)\) where \(k \approx 3\sigma\)
- 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:

\[
\begin{array}{ccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{array}
\]
Matlab Implementation

```matlab
>> hsize = 10;  \ hsize = width of mask
>> sigma = 5;
>> h = fspecial(‘gaussian’ hsize, sigma);
>> mesh(h);
>> imagesc(h);
>> outim = imfilter(im, h);  \ % correlation
>> imshow(outim);
```

Slide credit: Derek Hoiem
Practical Matters

• What is the size of the output?

• MATLAB: \texttt{filter2(g, f, shape)}
  
  – \texttt{shape = ’full’}: output size is sum of sizes of \( f \) and \( g \)
  
  – \texttt{shape = ’same’}: output size is same as \( f \)
  
  – \texttt{shape = ’valid’}: output size is difference of sizes of \( f \) and \( g \)
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
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’)}
Application: Filter Banks for Feature Detection

LM Filter Bank

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)

Slide credit: Derek Hoiem
Filter Banks

• Process image with each filter and keep responses (or squared/abs responses)
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 filter

Slide credit: Derek Hoiem
Median Filter

Salt and pepper noise

Median filtered

Plots of a row of the image

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

- Median filter is edge preserving

Slide credit: Derek Hoiem