EECS 442 – Computer vision

Introduction to Image Filters

• Convolution
• Blurring
• Sharpening
• Multi-scale representation
• Aliasing and sampling

Reading: [FP] Chapters: 7,8

Some slides of this lectures are courtesy of prof F. Li, prof S. Lazebnik, and various other lecturers
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
Images as functions

• We can think of an image as a function, \( f \), from \( \mathbb{R}^2 \) to \( \mathbb{R} \):

  – Defined over a rectangle, with a finite range:
    • \( f: [a,b] \times [c,d] \rightarrow [0,255] \)
    – \( f(x, y) \) gives the intensity at position \( (x, y) \)

• A color image:

\[
\begin{bmatrix}
  r(x, y) \\
  g(x, y) \\
  b(x, y)
\end{bmatrix}
\]

Source: S. Seitz
Images as functions

Source: S. Seitz
Images as functions

• Images are usually digital (discrete):
  – Sample the 2D space on a regular grid

• The image can now be represented as a matrix of integer values

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>79</td>
<td>23</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>9</td>
<td>62</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>58</td>
<td>197</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>176</td>
<td>135</td>
<td>5</td>
<td>188</td>
<td>191</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>29</td>
<td>26</td>
</tr>
<tr>
<td>0</td>
<td>89</td>
<td>144</td>
<td>147</td>
<td>187</td>
</tr>
<tr>
<td>255</td>
<td>252</td>
<td>0</td>
<td>166</td>
<td>123</td>
</tr>
<tr>
<td>166</td>
<td>63</td>
<td>127</td>
<td>17</td>
<td>0</td>
</tr>
</tbody>
</table>

Source: S. Seitz
Filters

• **Linear filtering:**
  – Form a new image whose pixels are a weighted sum of original pixel values
  – use the same set of weights at each point

**Goals:**
Extract useful information from the images
  • Features (edges, corners, blobs…)

Modify or enhance image properties
  - super-resolution, in-painting, de-noising
De-noising

Original
Salt and pepper noise

Super-resolution

In-painting

Image inpainting, M. Bertalmio et al.
http://www.isa.upf.es/~mbertalmio/restoration.html

Image Inpainting, M. Bertalmio et al.
http://www.isa.upf.es/~mbertalmio/restoration.html
Convolution

- Let $f$ be the image and $g$ be the kernel. The output of convolving $f$ with $g$ is denoted by $f \ast g$.

$$(f \ast g)[m, n] = \sum_{k,l} f[k, l] g[m - k, n - l]$$

Weighted product of $f(k,l)$ by $g(-(k,l)))$ computed at different locations $m,n$
Box filter

- Kernel $k$ with positive entries, that sum to 1.
- Notice: all weights are equal
Box filter

\[
(f * g)[m, n] = \sum_{k,l} f[k, l] g[m - k, n - l]
\]

Source: S. Seitz
Box filter

\[ F[x, y] \]

\[ G[x, y] \]

\[(f \ast g)[m, n] = \sum_{k,l} f[k, l] g[m - k, n - l] \]

Source: S. Seitz
Box filter

\[ F[x, y] \]

\[ G[x, y] \]

\[ (f \ast g)[m, n] = \sum_{k,l} f[k, l] g[m - k, n - l] \]

Source: S. Seitz
Box filter

\[ F[x, y] \]

\[ G[x, y] \]

\[
(f * g)[m, n] = \sum_{k,l} f[k, l] g[m - k, n - l]
\]

Source: S. Seitz
Box filter

\[ F[x, y] \]

\[ G[x, y] \]

\[(f * g)[m, n] = \sum_{k,l} f[k, l] g[m - k, n - l]\]
Box filter

\[ F[x, y] \]

\[ G[x, y] \]

\[(f \ast g)[m, n] = \sum_{k,l} f[k, l] g[m - k, n - l]\]

Source: S. Seitz
Box filter

- Replaces each pixel with an average of its neighborhood.
- Achieve smoothing effect (remove sharp features)

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

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Slide credit: David Lowe (UBC)
Example: Smoothing with a box filter
Smoothing with a Gaussian

- Weight contributions of neighboring pixels by nearness

![Gaussian filter](image)

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

- Constant factor at front makes volume sum to 1 (can be ignored, as we should normalize weights to sum to 1 in any case).

Slide credit: Christopher Rasmussen
Choosing kernel width

• Rule of thumb: set filter half-width to about $3 \sigma$
Smoothing with a Gaussian
Gaussian noise

- Mathematical model: sum of many independent factors
- Assumption: independent, zero-mean noise

\[
f(x, y) = \hat{f}(x, y) + \eta(x, y)
\]

Gaussian i.i.d. ("white") noise:
\[
\eta(x, y) \sim \mathcal{N}(\mu, \sigma)
\]

Source: K. Grauman
Smoothing with a Gaussian

Smoothing reduces pixel noise:

Each row shows smoothing with Gaussians of different width; each column shows different amounts of Gaussian noise.
Gaussian filters

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

- Convolution with self is another Gaussian
  - Convolving two times with Gaussian kernel of width $\sigma$ is same as convolving once with kernel of width $\sqrt{2} \sigma$

- *Separable* kernel
  - Factors into product of two 1D Gaussians
  - Useful: can convolve all rows, then all columns

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
Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window.

- Is median filtering linear?

Source: K. Grauman
Median filter

- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers
  - Remove salt & pepper noise

Source: K. Grauman
• **Salt and pepper noise**: contains random occurrences of black and white pixels
Filter salt & pepper noise

Salt-and-pepper noise

Median filtered

- MATLAB: medfilt2(image, [h w])

Source: K. Grauman
Median vs. Gaussian filtering

Gaussian

Median
Convolution: Properties

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

• 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 \(e\) =

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

\(a \ast e = a\)
Convolution: Properties

- **Linearity**: \( \text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2) \)

- **Shift invariance**: \( \text{filter}\left(\text{shift}\left(f\right)\right) = \text{shift}\left(\text{filter}\left(f\right)\right) \)
  
  (same behavior regardless of pixel location)

- Theoretical result: any linear shift-invariant operator can be represented as a convolution
Other examples of convolution

Original

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

= ?

Source: D. Lowe
Other examples of convolution

Original

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

Filtered (no change)

Source: D. Lowe
Other examples of convolution

Original

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

=  ?

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{pmatrix}
\]

Shifted left
By 1 pixel

Source: D. Lowe
Other examples of convolution

Original

\[
\begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{pmatrix}
\]

\[
\frac{1}{9}
\]

= ?

Source: D. Lowe
Other examples of convolution

Original

\[
\begin{array}{ccc}
\frac{1}{9} & \cdot 1 & \cdot 1 \\
\cdot 1 & \cdot 1 & \cdot 1 \\
\cdot 1 & \cdot 1 & \cdot 1 \\
\end{array}
\]

Blur (with a box filter)

Source: D. Lowe
Other examples of convolution

(Note that filter sums to 1)

Source: D. Lowe
• What does blurring take away?

  ![Original Image](image1)
  ![Smoothed Image (5x5)](image2)
  ![Detail](image3)

  ![Original Image](image4)
  ![Detail](image5)
  ![Sharpened Image](image6)

• Let’s add it back:
Other examples of convolution

Original

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Sharpening

\[ f \ast ((1 + \alpha) e^{-g}) \]

Impulse function
Differentiation and convolution

- Recall, for 2D function, $f(x,y)$:

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \left( \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon} \right)$$

- This is linear and shift invariant -> convolution

- We could approximate this as

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

2D Kernel

| -1 | 0 | 1 |
|----------------|
| -1 | 0 | 1 |
| -1 | 0 | 1 |

Rudimentary edge detector!
### Differentiation and convolution

#### Directional Derivatives Of A Binary Image

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Arguments: (-w 3 -a 0)</th>
<th>Arguments: (-w 5 -a 0)</th>
<th>Arguments: (-w 7 -a 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Original Image" /></td>
<td><img src="image" alt="2D Kernel" /></td>
<td><img src="image" alt="2D Kernel" /></td>
<td><img src="image" alt="2D Kernel" /></td>
</tr>
<tr>
<td>2D Kernel</td>
<td>2D Kernel</td>
<td>2D Kernel</td>
<td>2D Kernel</td>
</tr>
<tr>
<td>0 0 0</td>
<td>-1 0 1</td>
<td>-2 -1 0 1 2</td>
<td>-3 -2 -1 0 1 2 3</td>
</tr>
<tr>
<td>0 1 0</td>
<td>-1 0 1</td>
<td>-2 -1 0 1 2</td>
<td>-3 -2 -1 0 1 2 3</td>
</tr>
<tr>
<td>0 0 0</td>
<td>-1 0 1</td>
<td>-2 -1 0 1 2</td>
<td>-3 -2 -1 0 1 2 3</td>
</tr>
</tbody>
</table>
Differentiation and convolution
Analyzing the image at different scales
Why is a multi-scale representation useful?

• Find template matches at all scales
  – e.g., when finding hands or faces, we don’t know what size they will be in a particular image
  – Template size is constant, but image size changes

• Efficient search for correspondence
  – look at coarse scales, then refine with finer scales

• Examining all levels of detail
  – Find edges with different amounts of blur
  – Find textures with different spatial frequencies (levels of detail)

Slide credit: David Lowe (UBC)
Sub-sampling the image

• How about taking every second pixel?

Throw away every other row and column to create a 1/2 size image
Sub- sampling the image

Problem: Aliasing

• Sub-sampling may be dangerous….

• Characteristic errors may appear:
  – “Wagon wheels rolling the wrong way in movies”
  – “Checkerboards disintegrate in ray tracing”
  – “Striped shirts look funny on color television”

Source: D. Forsyth
Aliasing

• 1D example (sinewave):

Source: S. Marschner
Aliasing

- 1D example (sinewave):
Aliasing in videos
(stroboscopic effect)
Aliasing in graphics

Disintegrating textures

Source: A. Efros
Sampling and aliasing
Sampling Theorem (Nyquist)

- When sampling a signal at discrete intervals, the sampling frequency must be $\geq 2 \times f_{\text{max}}$;
- $f_{\text{max}} = \text{max frequency of the input signal.}$
- This will allows to reconstruct the original perfectly from the sampled version.
Anti-aliasing

Solutions:

• Sample more often

• Get rid of all frequencies that are greater than half the new sampling frequency
  – Will lose information
  – But it’s better than aliasing
  – Apply a smoothing filter
Anti-aliasing
Algorithm 7.1: Sub-sampling an Image by a Factor of Two

Apply a low-pass filter to the original image
   (a Gaussian with a $\sigma$ of between one
   and two pixels is usually an acceptable choice).
Create a new image whose dimensions on edge are half
   those of the old image
Set the value of the $i, j$’th pixel of the new image to the value
   of the $2i, 2j$’th pixel of the filtered image
The Gaussian pyramid

• Create each level from previous one:
  – smooth and sample

• Smooth with Gaussians, in part because
  – Gaussian*Gaussian = another Gaussian
  – G(x) * G(y) = G(sqrt(x^2 + y^2))
With Gaussian pre-filtering

1/2

1/4  (2x zoom)

1/8  (4x zoom)

Source: Steve Seitz
Super-resolution

EECS 442 – Computer vision

Next lecture:
Filters & Feature detectors
Filters are templates

- filters look like the effects they are intended to find
- filters find effects they look like

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector
- Filtering the image is a set of dot products

Slide credit: David Lowe (UBC)
Normalized correlation

- Think of filters as a dot product of the filter vector with the image region
  - Now measure the angle between the vectors
  
  \[ a \cdot b = \| a \| \| b \| \cos \theta \]

  - Angle (similarity) between vectors can be measured by normalizing length of each vector to 1.
  - Normalized correlation: divide each correlation by square root of sum of squared values (length)

Slide credit: David Lowe (UBC)
Multi-Scalar feature (edge) extraction

Images courtesy of Tony Jebara
The Steerable Pyramid

http://www.cns.nyu.edu/~eero/STEERPYR/


Pag. 204-205, FP (add figure)