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

$P = [x, y, z]$

$P = [x, y, z]$

Let’s now focus on 2D

Extract building blocks
Extract useful building blocks
The big picture...

Feature Detection

e.g. DoG

e.g. SIFT

Feature Description

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:
    • $f: [a,b] \times [c,d] \to [0,255]$
    – $f(x, y)$ gives the intensity at position $(x, y)$

• A color image:

  $$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

Source: S. Seitz
Images as functions
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

```
  62  79  23  119  120  05  4  0
  10  10   9   62   12   78  34  0
 10  58 197   46   46    0   0  48
176 135   5  188  191   68   0  49
  2   1   1   29   26   37   0  77
  0  89 144  147  187  102  62 208
255 252   0  166  123  62   0  31
166  63 127  17   1   0  99  30
```

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

Super-resolution

In-painting

Original
Salt and pepper noise

Inpainting, M. Bertalmio et al.
http://www.ina.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 $f * g$.

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

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

• MATLAB: `conv2` vs. `filter2` (also `imfilter`)
Box filter

- Kernel \( k \) with positive entries, that sum to 1.
- Notice: all weights are equal

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

Slide credit: David Lowe (UBC)
Box filter

\[ F[x, y] \]

\[ G[x, y] \]

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

\[ F[x, y] \]

\[ G[x, y] \]

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

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

\[ F[x, y] \]

\[ G[x, y] \]

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

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

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

- 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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.003</td>
<td>0.013</td>
<td>0.022</td>
<td>0.013</td>
<td>0.003</td>
</tr>
<tr>
<td>0.013</td>
<td>0.059</td>
<td>0.097</td>
<td>0.059</td>
<td>0.013</td>
</tr>
<tr>
<td>0.022</td>
<td>0.097</td>
<td>0.159</td>
<td>0.097</td>
<td>0.022</td>
</tr>
<tr>
<td>0.013</td>
<td>0.059</td>
<td>0.097</td>
<td>0.059</td>
<td>0.013</td>
</tr>
<tr>
<td>0.003</td>
<td>0.013</td>
<td>0.022</td>
<td>0.013</td>
<td>0.003</td>
</tr>
</tbody>
</table>
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
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.
Median filtering

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

![Median filtering diagram]

- 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

- **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

3x3  5x5  7x7

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 $e = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
  $a \ast e = a$
Convolution: Properties

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

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

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

Original

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

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

= 

Shifted left
By 1 pixel

Source: D. Lowe
Other examples of convolution

Original

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

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

= ?

Source: D. Lowe
Other examples of convolution

Original

Blur (with a box filter)

Source: D. Lowe
Other examples of convolution

Original

(Note that filter sums to 1)

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

[Images of an original image, a blurred (smoothed) image, and a detail image]

= detail

• Let’s add it back:

[Images of an original image, a detail image, and a sharpened image]

= sharpened
Other examples of convolution

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Sharpening

unit impulse

Gaussian

Laplacian of Gaussian

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

Impulse function
Differentiation and convolution

• Recall, for 2D function, $f(x,y)$:

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

• This is linear and shift invariant $\rightarrow$ 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="image1.png" alt="Image 1" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
<td><img src="image4.png" alt="Image 4" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2D Kernel</th>
<th>2D Kernel</th>
<th>2D Kernel</th>
<th>2D Kernel</th>
</tr>
</thead>
<tbody>
<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>

2D Kernel:
- For -w 3, the 2D Kernel is: 
  
  \[
  \begin{array}{ccc}
  0 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 0 \\
  \end{array}
  \]
  
  
- For -w 5, the 2D Kernel is: 
  
  \[
  \begin{array}{ccc}
  -1 & 0 & 1 \\
  -1 & 0 & 1 \\
  -1 & 0 & 1 \\
  \end{array}
  \]
  
  
- For -w 7, the 2D Kernel is: 
  
  \[
  \begin{array}{ccc}
  -2 & -1 & 0 \\
  -2 & -1 & 0 \\
  -2 & -1 & 0 \\
  \end{array}
  \]
  
  
Differentiation and convolution...
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)
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 allow 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
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
  – $G(x) \ast G(y) = G(\sqrt{x^2 + y^2})$

Slide credit: David Lowe (UBC)
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)
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)