CEE598 – Visual Data Sensing and Analytics for Civil Infrastructure Eng. & Mgmt.

Session 9 – Pixels and Image Filtering

Mani Golparvar-Fard
Department of Civil and Environmental Engineering
Department of Computer Science
3129D, Newmark Civil Engineering Lab
e-mail: mgolpar@illinois.edu

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
# REAL2016

# REAL2016


https://my.matterport.com/show/?m=HNdsE2YsCv5
Outline

What we have learned so far:
• Image formation
• Camera Calibration and Single View Metrology

Introduction to Pixels and Image Filtering
• Review of lighting
• Reflection and absorption
• What is image filtering and how do we do it?
• Color models (if time allows)

Reading: [FP] Chapters 7,8
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

- **Matching / Indexing / Detection**

*database of local descriptors*
Reflection models

- **Albedo**: fraction of light that is reflected
  - Determines color (amount reflected at each wavelength)

Very low albedo (hard to see shape)

Higher albedo

Credit: D. Hoiem
Reflection models

- **Specular reflection**: mirror-like
  - Light reflects at incident angle
  - Reflection color = incoming light color
Reflection models

- **Diffuse reflection**
  - Light scatters in all directions (proportional to cosine with surface normal)
  - Observed intensity is independent of viewing direction
  - Reflection color depends on light color and albedo
Surface orientation and light intensity

- Amount of light that hits surface from distant point source depends on angle between surface normal and source

\[ I(x) = \rho(x)(S \cdot N(x)) \]

prop to cosine of relative angle
Reflection models

Lambertian: reflection all diffuse

Mirrored: reflection all specular

Glossy: reflection mostly diffuse, some specular

Credit: D. Hoiem
The plight of the poor pixel

- A pixel’s brightness is determined by
  - Light source (strength, direction, color)
  - Surface orientation
  - Surface material and albedo
  - Reflected light and shadows from surrounding surfaces
  - Gain on the sensor

- A pixel’s brightness tells us nothing by itself
Basis for interpreting intensity images

- **Key idea**: for nearby scene points, most factors do not change much
- The information is mainly contained in *local differences* of brightness
Darkness = Large Difference in Neighboring Pixels
Next few classes: different 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
The raster image (pixel matrix)
Image filtering

- Image filtering: compute function of local neighborhood at each position
- Linear filtering: function is a weighted sum/difference of pixel values
- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

Credit: D. Hoiem
Convolution

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

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

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

- MATLAB: conv2 vs. filter2 (also imfilter)
Example: box filter

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

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

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

\[ f[\ldots] \quad h[\ldots] \]

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

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

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

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

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

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

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

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

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

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

\[ f[\ldots] \]

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

\[ h[\ldots] \]

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

\[ f[\ldots] \]

\[ h[\ldots] \]

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

\[ f[\ldots] \]

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

\[ h[\ldots] \]

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

\[ f[\ldots] \qquad h[\ldots] \]

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

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

$g[\cdot, \cdot]$
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

0 0 0
0 1 0
0 0 0

Source: D. Lowe
Practice with linear filters

Original

Shifted left
By 1 pixel

0 0 0
0 0 1
0 0 0

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

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)

Source: D. Hoiem
Other filters

Source: D. Hoiem

Horizontal Edge (absolute value)

Sobel

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

Horizontal Gradient

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

or

| -1 | 0 | 1 |

Vertical Gradient

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

or

| -1 | 0 | 1 |
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 \( e = [0, 0, 1, 0, 0] \),
  \( a * e = a \)
Important filter: Gaussian

- Spatially-weighted average

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

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

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 filtering (center location only)

\[
\begin{array}{ccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{array}
\times
\begin{array}{ccc}
2 & 3 & 3 \\
3 & 5 & 5 \\
4 & 4 & 6 \\
\end{array}
\]

The filter factors into a product of 1D filters:

\[
\begin{array}{ccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{array}
= \begin{array}{c}
1 \\
2 \\
1 \\
\end{array} \times \begin{array}{c}
1 \\
2 \\
1 \\
\end{array}
\]

Perform filtering along rows:

\[
\begin{array}{ccc}
1 & 2 & 1 \\
\end{array}
\times
\begin{array}{ccc}
2 & 3 & 3 \\
3 & 5 & 5 \\
4 & 4 & 6 \\
\end{array}
= \begin{array}{c}
11 \\
18 \\
18 \\
\end{array}
\]

Followed by filtering along the remaining column:

Source: K. Grauman
Practical matters

How big should the filter be?

- Values at edges should be near zero ← important!
- 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
Practical matters

• methods (MATLAB):
  - clip filter (black): `imfilter(f, g, 0)`
  - wrap around: `imfilter(f, g, 'circular')`
  - copy edge: `imfilter(f, g, 'replicate')`
  - reflect across edge: `imfilter(f, g, 'symmetric')`

Source: S. Marschner
Practical matters

- What is the size of the output?
- MATLAB: `filter2(g, f, shape)`
  - `shape` = ‘full’: output size is sum of sizes of f and g
  - `shape` = ‘same’: output size is same as f
  - `shape` = ‘valid’: output size is difference of sizes of f and g

Source: S. Lazebnik
A little more about color…
Digital Color Images

![Image of Digital Color Images]

**Figure 2.17**

(a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Source: D. Hoiem

---

CMOS sensor

Bayer Filter

- Incoming Light
- Filter Layer
- Sensor Array
- Resulting Pattern
Color Image

Source: D. Hoiem
Images in Matlab

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

- How can we represent color?

Source: D. Hoiem
Color spaces: RGB

Default color space

Some drawbacks
- Strongly correlated channels
- Non-perceptual

Color spaces: HSV

Intuitive color space

H (S=1,V=1)
S (H=1,V=1)
V (H=1,S=0)
Color spaces: YCbCr

Fast to compute, good for compression, used by TV

\[
\begin{align*}
Y' & = 16 + \frac{65.738 \cdot R'_D}{256} + \frac{129.057 \cdot G'_D}{256} + \frac{25.064 \cdot B'_D}{256} \\
C_B & = 128 + \frac{-37.945 \cdot R'_D}{256} - \frac{74.494 \cdot G'_D}{256} + \frac{112.439 \cdot B'_D}{256} \\
C_R & = 128 + \frac{112.439 \cdot R'_D}{256} - \frac{94.154 \cdot G'_D}{256} - \frac{18.285 \cdot B'_D}{256}
\end{align*}
\]
Color spaces: CIE L*a*b*

“Perceptually uniform” color space

Luminance = brightness
Chrominance = color

L
(a=0,b=0)

a
(L=65,b=0)

b
(L=65,a=0)
Which contains more information? (a) intensity (1 channel) (b) chrominance (2 channels)
Most information in intensity

Only color shown – constant intensity
Most information in intensity

Only intensity shown – constant color

Source: D. Hoiem
Most information in intensity

Original image

Source: D. Hoiem
Important Take Home Messages

- Image is a matrix of numbers (light intensities at different orientations)
  - Interpreted mainly through local comparisons

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

- Attend to details: filter size, extrapolation, cropping

- Color spaces beyond RGB sometimes useful