Pixels and Image Filtering

Computational Photography
Derek Hoiem
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 replaces film with a sensor array

- Each cell in the array is a light-sensitive diode that converts photons to electrons
Sensor Array

CCD sensor
The raster image (pixel matrix)
The raster image (pixel matrix)
Perception of Intensity

from Ted Adelson
Perception of Intensity

from Ted Adelson
Digital Color Images

https://commons.wikimedia.org/wiki/File:BayerPatternFiltration.png
Color Image

R

G

B
Images in Python

```python
im = cv2.imread(filename) # read image
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) # order channels as RGB
im = im / 255 # values range from 0 to 1
```

- **RGB image** `im` is a `H x W x 3` matrix (numpy.ndarray)
- `im[0,0,0]` = top-left pixel value in R-channel
- `im[y, x, c]` = `y+1` pixels down, `x+1` pixels to right in the `c`th channel
- `im[H-1, W-1, 2]` = bottom-right pixel in B-channel

![Pixel Values Matrix](attachment:image_matrix.png)
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

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

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

Slide credit: David Lowe (UBC)
Image filtering

\[ f[.,.] \]

\[ h[.,.] \]

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

Credit: S. Seitz
Image filtering

\[ f[\cdot,\cdot] \quad h[\cdot,\cdot] \]

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

Credit: S. Seitz
Image filtering

\[ f[\ldots] \]

\[ g[\ldots] \]

\[ h[\ldots] \]

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

Credit: S. Seitz
Image filtering

\[ f[\cdot,\cdot] \quad h[\cdot,\cdot] \]

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

Credit: S. Seitz
Image filtering

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

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

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

Credit: S. Seitz
Image filtering

\[ f[\cdots] \]

\[ h[\cdots] \]

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

Credit: S. Seitz
Image filtering

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

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

Credit: S. Seitz
Image filtering

\[ f[\cdot, \cdot] \quad h[\cdot, \cdot] \]

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

Credit: S. Seitz
Informally, what does the filter do?

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

\[
\begin{array}{cccc}
0 & 1 & 1 & 0 \\
1 & 2 & 2 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1 & 1 & 2 \\
\end{array}
\]

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

\[
\begin{array}{cccc}
0 & 1 & 1 & 0 \\
1 & 2 & 2 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1 & 1 & 2 \\
\end{array}
\]
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

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0
\end{array}
\quad - \quad \frac{1}{9}
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{array}
\]

(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

Vertical Edge
(absolute value)
Other filters

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>

Horizontal Edge (absolute value)
How could we synthesize motion blur?

```python
theta = 30
len = 21
mid = (len-1)/2

fil = np.zeros((len,len))
fil[:,int(mid)] = 1/len
R = cv2.getRotationMatrix2D((mid,mid),theta,1)
fil = cv2.warpAffine(fil,R,(len,len))

im_fil = cv2.filter2D(im, -1, fil)
```
Correlation vs. Convolution

• 2d correlation

\[
im\_fil[m, n] = \sum_{k, l} \text{fil}[k, l] \text{im}[m + k, n + l]
\]

\[
im\_fil = \text{cv2.filter2d}(\text{im}, -1, \text{fil})
\]

• 2d convolution

\[
im\_fil[m, n] = \sum_{k, l} \text{fil}[k, l] \text{im}[m - k, n - l]
\]

\[
im\_fil = \text{scipy.signal.convolve2d}(\text{im}, \text{fil}, [\text{opts}])
\]

• “convolve” mirrors the kernel, while “filter” doesn’t

\[
\text{cv2.filter2D}(\text{im}, -1, \text{cv2.flip(}\text{fil}, -1)) \text{ same as }
\]

\[
\text{signal.convolve2d}(\text{im}, \text{fil}, \text{mode='same'}, \text{boundary='symm'})
\]
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 (image)

• 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: \( k a * b = a * k b = k (a * b) \)

• 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

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

2D filtering (center location only)

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

The filter factors into a product of 1D filters:

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

Perform filtering along rows:

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

Followed by filtering along the remaining column:

\[
\begin{array}{ccc}
11 \\
18 \\
18 \\
\end{array}
\]

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 kernel half-width to \( \geq 3 \sigma \)
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 (Python):
  • clip filter (black): \texttt{convolve2d}(f, g, boundary=\textquoteleft fill\textquoteright,0)
  • wrap around: \texttt{convolve2d}(f, g, boundary=\textquoteleft wrap\textquoteright)
  • reflect across edge: \texttt{convolve2d}(f, g, boundary=\textquoteleft symm\textquoteright)
Practical matters

- What is the size of the output?

Python: \texttt{convolve2d}(g, f, mode)
  - \textit{mode} = ‘full’: output size is sum of sizes of \( f \) and \( g \)
  - \textit{mode} = ‘same’: output size is same as \( f \)
  - \textit{mode} = ‘valid’: output size is difference of sizes of \( f \) and \( g \)
Application: Representing Texture

Source: Forsyth
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

Mean abs responses

A

B

C
Representing texture by mean abs response
Project 1: Hybrid Images


Gaussian Filter!

Laplacian Filter!

Project Instructions:
https://courses.engr.illinois.edu/cs445/fa2022/projects/hybrid/ComputationalPhotography_ProjectHybrid.html
Take-home messages

• Image is a matrix of numbers

• Linear filtering is a 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, set up notebook)
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:

   $\text{grad}_x(y,x) = \text{im}(y,x+1) - \text{im}(y,x)$ 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