Point Processing & Filtering
$f(x,y) = \text{reflectance}(x,y) \ast \text{illumination}(x,y)$

Reflectance in $[0,1]$, illumination in $[0,\infty]$
Problem: Dynamic Range

The real world is High dynamic range
Long Exposure

Real world

Picture

High dynamic range

$10^{-6}$ to $10^6$

$0$ to $255$
Short Exposure

Real world

10^{-6} \quad \text{High dynamic range} \quad 10^6

Picture

10^{-6} \quad 0 \text{ to } 255 \quad 10^6
Image Acquisition Pipeline

Lens
- Scene radiance (W/sr/m²)
- Sensor irradiance

Shutter
- Sensor exposure
- \( \Delta t \)

CCD
- Analog voltages

ADC
- Digital values

Remapping
- Pixel values
Simple Point Processing: Enhancement

**FIGURE 3.9**
(a) Aerial image.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0,$ and $5.0$, respectively. (Original image for this example courtesy of NASA.)
Power-law transformations

$$s = cr^\gamma$$

**FIGURE 3.6** Plots of the equation $s = cr^\gamma$ for various values of $\gamma$ ($c = 1$ in all cases).
Basic Point Processing

**FIGURE 3.3** Some basic gray-level transformation functions used for image enhancement.
Negative

**FIGURE 3.4**
(a) Original digital mammogram.
(b) Negative image obtained using the negative transformation in Eq. (3.2-1).
(Courtesy of G.E. Medical Systems.)
Log

**FIGURE 3.5**
(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$. 
Contrast Stretching

**FIGURE 3.10**
Contrast stretching.
(a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)
Image Histograms

FIGURE 3.15 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)
Histogram Equalization

FIGURE 3.17 (a) Images from Fig. 3.15. (b) Results of histogram equalization. (c) Corresponding histograms.
Color Transfer [Reinhard, et al, 2001]

Limitations of Point Processing

Q: What happens if I reshuffle all pixels within the image?

A: It’s histogram won’t change. No point processing will be affected...
What is an image?

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

- $f(x, y)$ gives the intensity at position $(x, y)$
- Realistically, we expect the image only to be defined over a rectangle, with a finite range:
  - $f: [a,b] \times [c,d] \rightarrow [0,1]$

A color image is just three functions pasted together. We can write this as a “vector-valued” function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$
Images as functions
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
1D Example: Audio

[Graph showing a waveform with low and high frequencies]
Sampling in digital audio

- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
  - how can we be sure we are filling in the gaps correctly?
Sampling and Reconstruction

- Simple example: a sign wave
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
Undersampling

• What if we “missed” things between the samples?
• Simple example: undersampling a sine wave
  – unsurprising result: information is lost
  – surprising result: indistinguishable from lower frequency

© 2006 Steve Marschner • 26
Undersampling

• What if we “missed” things between the samples?

• Simple example: undersampling a sine wave
  – unsurprising result: information is lost
  – surprising result: indistinguishable from lower frequency
  – also, was always indistinguishable from higher frequencies
  – **aliasing**: signals “traveling in disguise” as other frequencies
Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what’s happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):

Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)
Aliasing in images

Disintegrating textures
What’s happening?

Input signal:

\[ x = 0:.05:5; \text{ imagesc} \left( \sin \left( (2^x) \cdot x \right) \right) \]

Plot as image:

Alias!

Not enough samples
Antialiasing

What can we do about aliasing?

Sample more often
  • Join the Mega-Pixel craze of the photo industry
  • But this can’t go on forever

Make the signal less “wiggly”
  • Get rid of some high frequencies
  • Will loose information
  • But it’s better than aliasing
Preventing aliasing

- Introduce lowpass filters:
  - remove high frequencies leaving only safe, low frequencies
  - choose lowest frequency in reconstruction (disambiguate)
Linear filtering: a key idea

• Transformations on signals; e.g.:
  – bass/treble controls on stereo
  – blurring/sharpening operations in image editing
  – smoothing/noise reduction in tracking

• Key properties
  – linearity: $\text{filter}(f + g) = \text{filter}(f) + \text{filter}(g)$
  – shift invariance: behavior invariant to shifting the input
    • delaying an audio signal
    • sliding an image around

• Can be modeled mathematically by convolution
Moving Average

• basic idea: define a new function by averaging over a sliding window

• a simple example to start off: smoothing
Moving Average

- Can add weights to our moving average
- \textit{Weights} \([\ldots, 0, 1, 1, 1, 1, 1, 0, \ldots] / 5\)
Cross-correlation

Let $F$ be the image, $H$ be the kernel (of size $2k+1 \times 2k+1$), and $G$ be the output image:

$$G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]$$

This is called a **cross-correlation** operation:

$$G = H \otimes F$$

- Can think of as a “dot product” between local neighborhood and kernel for each pixel
In 2D: box filter

$h[\cdot, \cdot]$
Image filtering

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

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

\[
\begin{align*}
g[m, n] &= \sum_{k,l} h[k, l] \ f[m + k, n + l]
\end{align*}
\]
Image filtering

\[ f[\ldots] \]

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

\[ g[\ldots] \]

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

Credit: S. Seitz
Image filtering

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

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

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

Credit: S. Seitz
Image filtering

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

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

Credit: S. Seitz
Image filtering

\[ f[\ldots] \]

\[ g[\ldots] \]

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

Credit: S. Seitz
Image filtering

\[ f[\ldots] \]

\[ g[\ldots] \]

\[ h[\ldots] \] \( \frac{1}{9} \)

Credit: S. Seitz
Image filtering

\[ f[\ldots] \]

\[ g[\ldots] \]

\[ h[\ldots] \]

Credit: S. Seitz
### Image filtering

Consider an image $f[\cdot, \cdot]$ and a convolution kernel $h[\cdot, \cdot]$.

The convolution operation is given by:

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

Credit: S. Seitz
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)
Linear filters: examples

Original

Blur (with a mean filter)

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

Filtered (no change)

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{array}{ccc}
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
\end{array}
\]
Practice with linear filters

Original

Shifted left By 1 pixel

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)
Back to the box filter
Moving Average

- Can add weights to our moving average
- **Weights** [..., 0, 1, 1, 1, 1, 1, 0, ...] / 5
Weighted Moving Average

- bell curve (gaussian-like) weights […, 1, 4, 6, 4, 1, …]
Moving Average In 2D

What are the weights $H$?

$$F[x, y]$$

$$H[u, v]$$
Gaussian filtering

A Gaussian kernel gives less weight to pixels further from the center of the window.

\[ h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}} \]

This kernel is an approximation of a Gaussian function:
Mean vs. Gaussian filtering
Important filter: Gaussian

Weight contributions of neighboring pixels by nearness

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

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

Slide credit: Christopher Rasmussen
Gaussian Kernel

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

- Standard deviation \( \sigma \): determines extent of smoothing

Source: K. Grauman
Gaussian filters

\[ \sigma = 1 \text{ pixel} \quad \sigma = 5 \text{ pixels} \quad \sigma = 10 \text{ pixels} \quad \sigma = 30 \text{ pixels} \]
Choosing kernel width

• The Gaussian function has infinite support, but discrete filters use finite kernels

Source: K. Grauman
Practical matters

How big should the filter be?

Values at edges should be near zero

Rule of thumb for Gaussian: set filter half-width to about $3 \sigma$
Cross-correlation vs. Convolution

cross-correlation: \[ G = H \otimes F \]

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i + u, j + v] \]

A convolution operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i - u, j - v] \]

It is written:

\[ G = H \ast F \]

Convolution is \textbf{commutative} and \textbf{associative}
Convolution

Adapted from F. Durand
Convolution is nice!

• Notation: \( b = c \ast a \)

• Convolution is a multiplication-like operation
  – commutative \( a \ast b = b \ast a \)
  – associative \( a \ast (b \ast c) = (a \ast b) \ast c \)
  – distributes over addition \( a \ast (b + c) = a \ast b + a \ast c \)
  – scalars factor out \( \alpha a \ast b = a \ast \alpha b = \alpha(a \ast b) \)
  – identity: unit impulse \( e = [..., 0, 0, 1, 0, 0, ...] \)
    \[ a \ast e = a \]

• Conceptually no distinction between filter and signal

• Usefulness of associativity
  – often apply several filters one after another: \(((a \ast b_1) \ast b_2) \ast b_3)\)
  – this is equivalent to applying one filter: \(a \ast (b_1 \ast b_2 \ast b_3)\)
Gaussian and convolution

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

• Convolution with self is another Gaussian

\[ \ast \]

– Convolving twice with Gaussian kernel of width \( \sigma \)

\[ = \]

= convolving once with kernel of width \( \sigma \sqrt{2} \)

Source: K. Grauman
This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?
Throw away every other row and column to create a 1/2 size image - called *image sub-sampling*.
Image sub-sampling

1/2  1/4  (2x zoom)  1/8  (4x zoom)

Aliasing! What do we do?
Sampling an image

Examples of GOOD sampling
Undersampling

Examples of BAD sampling -> Aliasing
Gaussian (lowpass) pre-filtering

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?
Subsampling with Gaussian pre-filtering

Gaussian 1/2  G 1/4  G 1/8
Compare with...

1/2

1/4 (2x zoom)

1/8 (4x zoom)
Gaussian (lowpass) pre-filtering

Solution: filter the image, \textit{then} subsample

- Filter size should double for each $\frac{1}{2}$ size reduction. Why?
- How can we speed this up?
Image Pyramids

Idea: Represent \( N \times N \) image as a “pyramid” of \( 1 \times 1, 2 \times 2, 4 \times 4, \ldots, 2^k \times 2^k \) images (assuming \( N = 2^k \))

Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*
A bar in the big images is a hair on the zebra’s nose; in smaller images, a stripe; in the smallest, the animal’s nose.
Gaussian pyramid construction

Repeat
- Filter
- Subsample

Until minimum resolution reached
- can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!
What are they good for?

Improve Search

• Search over translations
  – Classic coarse-to-fine strategy
• Search over scale
  – Template matching
  – E.g. find a face at different scales
Taking derivative by convolution
Partial derivatives with convolution

For 2D function \( f(x,y) \), the partial derivative is:

\[
\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon}
\]

For discrete data, we can approximate using finite differences:

\[
\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x, y)}{1}
\]

To implement above as convolution, what would be the associated filter?

Source: K. Grauman
Partial derivatives of an image

\[ \frac{\partial f(x, y)}{\partial x} \quad \text{or} \quad \frac{\partial f(x, y)}{\partial y} \]

Which shows changes with respect to \( x \)?

\[
\begin{pmatrix}
-1 & 1 \\
-1 & 1
\end{pmatrix}
\]
Finite difference filters

Other approximations of derivative filters exist:

Prewitt:  \[ M_x = \begin{array}{ccc} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{array} \ ; \quad M_y = \begin{array}{ccc} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{array} \]

Sobel:  \[ M_x = \begin{array}{ccc} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{array} \ ; \quad M_y = \begin{array}{ccc} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{array} \]

Roberts:  \[ M_x = \begin{array}{cc} 0 & 1 \\ -1 & 0 \end{array} \ ; \quad M_y = \begin{array}{cc} 1 & 0 \\ 0 & -1 \end{array} \]

Source: K. Grauman
The gradient points in the direction of most rapid increase in intensity.

- How does this direction relate to the direction of the edge?

The gradient direction is given by \( \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \).

The edge strength is given by the gradient magnitude

\[
\| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2}
\]
Image Gradient

\[ \nabla f = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]
Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

\[ f(x) \]

\[ \frac{d}{dx} f(x) \]

Where is the edge?

Source: S. Seitz
Solution: smooth first

- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz
Derivative theorem of convolution

\[
\frac{\partial}{\partial x} (h \ast f) = (\frac{\partial}{\partial x} h) \ast f
\]

This saves us one operation:

\[
\frac{\partial}{\partial x} h
\]

\[
(\frac{\partial}{\partial x} h) \ast f
\]
Derivative of Gaussian filter

\[
* \begin{bmatrix} 1 & -1 \end{bmatrix} =
\]
Derivative of Gaussian filter

Which one finds horizontal/vertical edges?
Example

input image ("Lena")
Compute Gradients (DoG)

X-Derivative of Gaussian

Y-Derivative of Gaussian

Gradient Magnitude
Get Orientation at Each Pixel

Threshold at minimum level
Get orientation

\[ \theta = \text{atan2}(-gy, gx) \]
im = im2double(imread(filename));
g = fspecial('gaussian',15,2);
imagesc(g);
surfl(g);
gim = conv2(im,g,'same');
imagesc(conv2(im,[-1 1],'same'));
imagesc(conv2(gim,[-1 1],'same'));
dx = conv2(g,[-1 1],'same');
Surfl(dx);
imagesc(conv2(im,dx,'same'));
Practical matters

What is the size of the output?

MATLAB: filter2(g, f, shape) or conv2(g,f,shape)

- \textit{shape} = ‘full’: output size is sum of sizes of f and g
- \textit{shape} = ‘same’: output size is same as f
- \textit{shape} = ‘valid’: output size is difference of sizes of f and g

Source: S. Lazebnik
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): \texttt{imfilter(f, g, 0)}
  - wrap around: \texttt{imfilter(f, g, \texttt{‘circular’})}
  - copy edge: \texttt{imfilter(f, g, \texttt{‘replicate’})}
  - reflect across edge: \texttt{imfilter(f, g, \texttt{‘symmetric’})}

Source: S. Marschner
Review: Smoothing vs. derivative filters

Smoothing filters
- Gaussian: remove “high-frequency” components; “low-pass” filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
  - **One:** constant regions are not affected by the filter

Derivative filters
- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
  - **Zero:** no response in constant regions
- High absolute value at points of high contrast
Template matching

Goal: find an object in image

Main challenge: What is a good similarity or distance measure between two patches?

- Correlation
- Zero-mean correlation
- Sum Square Difference
- Normalized Cross Correlation
Matching with filters

Goal: find 🍀 in image

Method 0: filter the image with eye patch

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

What went wrong?

Input

Filtered Image

Side by Derek Hoiem
Matching with filters

Goal: find 🕵️‍♂️ in image

Method 1: filter the image with zero-mean eye

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

True detections
False detections
Matching with filters

Goal: find 🧠 in image

Method 2: SSD

\[ h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2 \]
Can SSD be implemented with linear filters?

\[ h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2 \]
Matching with filters

Goal: find in image

Method 2: SSD

\[ h[m, n] = \sum_{k,l} (g[k, l] - f[m + k, n + l])^2 \]

What's the potential downside of SSD?

Side by Derek Hoiem
Matching with filters

Goal: find in image

Method 3: Normalized cross-correlation

\[
h[m, n] = \frac{\sum_{k,l} (g[k, l] - \bar{g})(f[m + k, n + l] - \bar{f}_{m,n})}{\left(\sum_{k,l} (g[k, l] - \bar{g})^2 \sum_{k,l} (f[m + k, n + l] - \bar{f}_{m,n})^2\right)^{0.5}}
\]

mean template

mean image patch

Side by Derek Hoiem
Matching with filters

Goal: find 🕍 in image

Method 3: Normalized cross-correlation

Input

Normalized X-Correlation

Thresholded Image

True detections
Matching with filters

Goal: find 🕳️ in image

Method 3: Normalized cross-correlation

Input Normalized X-Correlation Thresholded Image

True detections
Q: What is the best method to use?

A: Depends

Zero-mean filter: fastest but not a great matcher

SSD: next fastest, sensitive to overall intensity

Normalized cross-correlation: slowest, invariant to local average intensity and contrast