Image filtering
Course announcements

• Office hours for rest of semester:
  Gaurav      Tuesday noon-2 pm.
  Shashank    Thursday 3-5pm.
  Yannis      Friday 3-5 pm.

• Homework 1 will be available on Wednesday and due two weeks from then.

• Make sure you are on Piazza (sign up on your own using the link on the course website).
  - How many of you aren’t already on Piazza?

• Make sure to take the start-of-semester survey (link posted on Piazza).
Overview of today’s lecture

• Types of image transformations.
• Point image processing.
• Linear shift-invariant image filtering.
• Convolution.
• Image gradients.
Most of these slides were adapted directly from:

- Kris Kitani (15-463, Fall 2016).

Inspiration and some examples also came from:

- Fredo Durand (Digital and Computational Photography, MIT).
- Kayvon Fatahalian (15-769, Fall 2016).
Types of image transformations
What is an image?
What is an image?

A (color) image is a 3D tensor of numbers.
What is an image?

- red
- green
- blue
colorized for visualization

- color image patch
- How many bits are the intensity values?

actual intensity values per channel

Each channel is a 2D array of numbers.
What is an image?

A (grayscale) image is a 2D function.

What is the range of the image function $f$?

A (grayscale) image is a 2D function.
What types of image transformations can we do?

Filtering
changes pixel values

Warping
changes pixel locations
What types of image transformations can we do?

Filtering:

\[ G(x) = h\{F(x)\} \]

changes range of image function

Warping:

\[ G(x) = F(h\{x\}) \]

changes domain of image function
What types of image filtering can we do?

Point Operation

Neighborhood Operation

point processing

“filtering”
Point processing
Examples of point processing

- original
- darken
- lower contrast
- non-linear lower contrast
- invert
- lighten
- raise contrast
- non-linear raise contrast
Examples of point processing

- original
- darken
- lower contrast
- non-linear lower contrast
- invert
- lighten
- raise contrast
- non-linear raise contrast

How would you implement these?
Examples of point processing

- original
- darken
- lower contrast
- non-linear lower contrast

How would you implement these?

$x$
$x - 128$

- invert
- lighten
- raise contrast
- non-linear raise contrast
Examples of point processing

original

darken

lower contrast

non-linear lower contrast

$x$

$x - 128$

$\frac{x}{2}$

invert

lighten

raise contrast

non-linear raise contrast

How would you implement these?
How would you implement these?

Examples of point processing

original  darken  lower contrast  non-linear lower contrast

\[ x \]
\[ x - 128 \]
\[ \frac{x}{2} \]
\[ \left( \frac{x}{255} \right)^{1/3} \times 255 \]

invert  lighten  raise contrast  non-linear raise contrast
Examples of point processing

- **original**
- **darken**
- **lower contrast**
- **non-linear lower contrast**

- $x$
- $x - 128$
- $\frac{x}{2}$
- $\left( \frac{x}{255} \right)^{1/3} \times 255$

- **invert**
- **lighten**
- **raise contrast**
- **non-linear raise contrast**

- $255 - x$
Examples of point processing

- **original**
- **darken**: $x - 128$
- **lower contrast**: $\frac{x}{2}$
- **non-linear lower contrast**: $\left(\frac{x}{255}\right)^{1/3} \times 255$
- **invert**: $255 - x$
- **lighten**: $x + 128$
- **raise contrast**: $\frac{x}{255}$

How would you implement these?
Examples of point processing

- Original: $x$
- Darken: $x - 128$
- Lower contrast: $\frac{x}{2}$
- Non-linear lower contrast: $\left(\frac{x}{255}\right)^{1/3} \times 255$
- Invert: $255 - x$
- Lighten: $x + 128$
- Raise contrast: $x \times 2$

How would you implement these?
Examples of point processing

- **Original**: $x$
- **Darken**: $x - 128$
- **Lower contrast**: $\frac{x}{2}$
- **Non-linear lower contrast**: $\left(\frac{x}{255}\right)^{1/3} \times 255$
- **Invert**: $255 - x$
- **Lighten**: $x + 128$
- **Raise contrast**: $x \times 2$
- **Non-linear raise contrast**: $\left(\frac{x}{255}\right)^2 \times 255$

How would you implement these?
Many other types of point processing

camera output

image after stylistic tonemapping

[Bae et al., SIGGRAPH 2006]
Many other types of point processing
Linear shift-invariant image filtering
Linear shift-invariant image filtering

- Replace each pixel by a linear combination of its neighbors (and possibly itself).
- The combination is determined by the filter’s kernel.
- The same kernel is shifted to all pixel locations so that all pixels use the same linear combination of their neighbors.
Example: the box filter

• also known as the 2D rect (not rekt) filter
• also known as the square mean filter

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

• replaces pixel with local average
• has smoothing (blurring) effect
Let’s run the box filter

\[
\frac{1}{9} \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Note that we assume that the kernel coordinates are centered.

\[
h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]
\]
Let’s run the box filter

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

output

filter

image (signal)
Let’s run the box filter

$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$
Let’s run the box filter

The box filter is a simple averaging filter. Its kernel is a $3 \times 3$ matrix:

$$
g[k, l] = \begin{cases} 1/9 & \text{if } (k, l) = (0, 0) \\ 0 & \text{otherwise} \end{cases}
$$

The output of the filter at position $(m, n)$ is the average of the input image in the corresponding window:

$$
h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]
$$
Let’s run the box filter

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

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

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

kernel \[ g[\cdot, \cdot] \]
Let’s run the box filter

\[
\begin{align*}
\text{image} & \quad f[\cdot, \cdot] \\
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 0 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\end{align*}
\]

\[
\text{output} & \quad h[\cdot, \cdot] \\
\begin{array}{cccccccc}
0 & 10 & 20 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]
\]
Let’s run the box filter

\[
h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]\]
Let’s run the box filter

\[ h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l] \]
Let’s run the box filter

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

output \hspace{1cm} filter \hspace{1cm} image (signal)
Let’s run the box filter

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

output \quad h[\cdot, \cdot]

image (signal)
Let’s run the box filter

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

output  \hspace{2cm} filter  \hspace{2cm} image (signal)
Let’s run the box filter

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

output \quad filter \quad image (signal)
Let’s run the box filter

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

**Output**

**Image (Signal)**
Let’s run the box filter

Let's run the box filter

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

output  kernel  image (signal)
Let’s run the box filter

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

Output: $h[\cdot, \cdot]$

Filter: $f[\cdot, \cdot]$

Image (signal): $g[\cdot, \cdot]$
Let’s run the box filter

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

output filter image (signal)
Let’s run the box filter

\[
h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]\]
Let's run the box filter

![Box Filter Diagram]

Let's run the box filter

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

output filter image (signal)
Let’s run the box filter

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

input \hspace{1cm} filter \hspace{1cm} output

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 0 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 0 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccccccc}
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 \\
0 & 20 & 40 & 60 & 60 & 60 & 40 & 20 \\
0 & 30 & 50 & 80 & 80 & 90 & 60 & 30 \\
0 & 30 & 50 & 80 & 80 & 90 & 60 & 30 \\
0 & 20 & 30 & 50 & 50 & 60 & 40 & 20 \\
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 \\
10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\
10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\
\end{array}
\]
... and the result is

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

output \hspace{1cm} filter \hspace{1cm} image (signal)
Some more realistic examples
Some more realistic examples
Some more realistic examples
Convolution
Convolution for 1D continuous signals

Definition of filtering as convolution:

\[(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x - y)dy\]

filtered signal \quad \text{notice the flip} \quad \text{filter} \quad \text{input signal}
Convolution for 1D continuous signals

Definition of filtering as convolution:

\[(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x - y)dy\]

Consider the box filter example:

1D continuous box filter

\[f(x) = \begin{cases} 
1 & |x| \leq 0.5 \\
0 & \text{otherwise}
\end{cases}\]

Filtering output is a blurred version of \(g\)

\[(f * g)(x) = \int_{-0.5}^{0.5} g(x - y)dy\]
Convolution for 2D discrete signals

Definition of filtering as convolution:

\[(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x - i, y - j)\]

- \(f \) is the filter
- \(I \) is the input image
- \((f * g)\) is the filtered image

notice the flip
Convolution for 2D discrete signals

Definition of filtering as convolution:

\[(f \ast g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x-i, y-j)\]

If the filter \(f(i, j)\) is non-zero only within \(-1 \leq i, j \leq 1\), then

\[(f \ast g)(x, y) = \sum_{i,j=-1}^{1} f(i, j)I(x-i, y-j)\]

The kernel we saw earlier is the 3x3 matrix representation of \(f(i, j)\).
Convolution vs correlation

Definition of discrete 2D convolution:

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x - i, y - j)$$

Definition of discrete 2D correlation:

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x + i, y + j)$$

- Most of the time won’t matter, because our kernels will be symmetric.
- Will be important when we discuss frequency-domain filtering (lectures 5-6).
A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example: box filter

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

column
row

What is the rank of this filter matrix?
A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example: box filter

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\quad = \quad
\begin{bmatrix}
1 \\
1 \\
1 \\
\end{bmatrix}
\quad \ast \quad
\begin{bmatrix}
1 & 1 & 1 \\
\end{bmatrix}
\]

Why is this important?
Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example: box filter

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

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

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

column

row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).
Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

If the image has $M \times M$ pixels and the filter kernel has size $N \times N$:

• What is the cost of convolution with a non-separable filter?
Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example: box filter

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

= \[
\begin{array}{c}
1 \\
1 \\
1 \\
\end{array}
\]

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

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

If the image has M x M pixels and the filter kernel has size N x N:
• What is the cost of convolution with a non-separable filter? \( M^2 \times N^2 \)
• What is the cost of convolution with a separable filter?
Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

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

example: box filter

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

If the image has \(M \times M\) pixels and the filter kernel has size \(N \times N\):

- What is the cost of convolution with a non-separable filter? \(\rightarrow M^2 \times N^2\)
- What is the cost of convolution with a separable filter? \(\rightarrow 2 \times N \times M^2\)
A few more filters

original

3x3 box filter

do you see any problems in this image?
The Gaussian filter

• named (like many other things) after Carl Friedrich Gauss

• kernel values sampled from the 2D Gaussian function:

\[ f(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \]

• weight falls off with distance from center pixel

• theoretically infinite, in practice truncated to some maximum distance

Any heuristics for selecting where to truncate?
The Gaussian filter

- named (like many other things) after Carl Friedrich Gauss
- kernel values sampled from the 2D Gaussian function:
  \[ f(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \]
- weight falls off with distance from center pixel
- theoretically infinite, in practice truncated to some maximum distance

Any heuristics for selecting where to truncate?
- usually at 2-3\(\sigma\)
The Gaussian filter

- named (like many other things) after Carl Friedrich Gauss
- kernel values sampled from the 2D Gaussian function:
  \[ f(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \]
- weight falls off with distance from center pixel
- theoretically infinite, in practice truncated to some maximum distance

Any heuristics for selecting where to truncate?
- usually at 2-3σ

Is this a separable filter? Yes!

<table>
<thead>
<tr>
<th>kernel</th>
<th>( \frac{1}{16} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 1</td>
</tr>
<tr>
<td></td>
<td>2 4 2</td>
</tr>
<tr>
<td></td>
<td>1 2 1</td>
</tr>
</tbody>
</table>
Gaussian filtering example
Gaussian vs box filtering

original

Which blur do you like better?

7x7 Gaussian

7x7 box
How would you create a soft shadow effect?
How would you create a soft shadow effect?

Gaussian blur

overlay
Other filters

input

filter

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

output

?
Other filters

input

0 0 0
0 1 0
0 0 0

unchanged
Other filters

input

filter

output

unchanged

input

filter

output

?
Other filters

input | filter | output
--- | --- | ---
input | 0 0 0 | unchanged
input | 0 1 0 | shifted to left by one
Other filters

input

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

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

output

?
Other filters

- do nothing for flat areas
- stress intensity peaks

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{array} \quad - \quad \frac{1}{9} \quad \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
Sharpening examples
Sharpening examples
Sharpening examples
Sharpening examples
do you see any problems in this image?
Do not overdo it with sharpening

What is wrong in this image?
Image gradients
What are image edges?

gray-scale image

Very sharp discontinuities in intensity.

domain $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$
Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?
Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

✓ You take derivatives: derivatives are large at discontinuities.

How do you differentiate a discrete image (or any other discrete signal)?
Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

✓ You take derivatives: derivatives are large at discontinuities.

How do you differentiate a discrete image (or any other discrete signal)?

✓ You use finite differences.
Finite differences

High-school reminder: definition of a derivative using forward difference

\[ f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \]
Finite differences

High-school reminder: definition of a derivative using forward difference

\[ f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \]

Alternative: use central difference

\[ f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h} \]

For discrete signals: Remove limit and set \( h = 2 \)

\[ f'(x) = \frac{f(x + 1) - f(x - 1)}{2} \]

What convolution kernel does this correspond to?
Finite differences

High-school reminder: definition of a derivative using forward difference

\[ f'(x) = \lim_{{h \to 0}} \frac{{f(x + h) - f(x)}}{{h}} \]

Alternative: use central difference

\[ f'(x) = \lim_{{h \to 0}} \frac{{f(x + 0.5h) - f(x - 0.5h)}}{{h}} \]

For discrete signals: Remove limit and set \( h = 2 \)

\[ f'(x) = \frac{{f(x + 1) - f(x - 1)}}{{2}} \]

\[
\begin{array}{c|c|c|c}
-1 & 0 & 1 & ? \\
1 & 0 & -1 & ? \\
\end{array}
\]
Finite differences

High-school reminder: definition of a derivative using forward difference

\[ f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \]

Alternative: use central difference

\[ f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h} \]

For discrete signals: Remove limit and set \( h = 2 \)

\[ f'(x) = \frac{f(x + 1) - f(x - 1)}{2} \quad \text{1D derivative filter} \]

\[
\begin{bmatrix}
1 & 0 & -1
\end{bmatrix}
\]
The Sobel filter

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

Sobel filter

\[
\begin{array}{ccc}
1 & 2 & 1 \\
\end{array}
\]

What filter is this?

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

1D derivative filter
The Sobel filter

\[
\begin{pmatrix}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 \\
2 \\
1 \\
\end{pmatrix}
\]

\[
1 \quad 0 \quad -1
\]

1D derivative filter

In a 2D image, does this filter responses along horizontal or vertical lines?
The Sobel filter

\[
\begin{pmatrix}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1
\end{pmatrix}
= \begin{pmatrix}
1 \\
2 \\
1
\end{pmatrix}
\times
\begin{pmatrix}
1 & 0 & -1
\end{pmatrix}
\]

1D derivative filter

Does this filter return large responses on vertical or horizontal lines?
The Sobel filter

Horizontal Sobel filter:

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

What does the vertical Sobel filter look like?
The Sobel filter

Horizontal Sobel filter:

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

\[
\begin{array}{c}
1 \\
2 \\
1
\end{array}
\]

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

Vertical Sobel filter:

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

\[
\begin{array}{c}
1 \\
0 \\
-1
\end{array}
\]

\[
\begin{array}{ccc}
1 & 2 & 1 \\
1 & 2 & 1
\end{array}
\]
Sobel filter example

original

which Sobel filter?

which Sobel filter?
Sobel filter example

original

horizontal Sobel filter

vertical Sobel filter
Sobel filter example

- Original
- Horizontal Sobel filter
- Vertical Sobel filter
Several derivative filters

- Sobel:
  - 1 0 -1
  - 2 0 -2
  - 1 0 -1

- Prewitt:
  - 1 0 -1
  - 1 0 -1
  - 1 0 -1

- Scharr:
  - 3 0 -3
  - 10 0 -10
  - 0 0 0

- Roberts:
  - 0 1
  - -1 0
  - 1 0

• How are the other filters derived and how do they relate to the Sobel filter?
• How would you derive a derivative filter that is larger than 3x3?
Computing image gradients

1. Select your favorite derivative filters.

\[ S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]
Computing image gradients

1. Select your favorite derivative filters.

\[ S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]

2. Convolve with the image to compute derivatives.

\[ \frac{\partial f}{\partial x} = S_x \otimes f \quad \frac{\partial f}{\partial y} = S_y \otimes f \]
Computing image gradients

1. Select your favorite derivative filters.

\[
S_x = \begin{bmatrix}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1 \\
\end{bmatrix}
\quad
S_y = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{bmatrix}
\]

2. Convolve with the image to compute derivatives.

\[
\frac{\partial f}{\partial x} = S_x \otimes f \\
\frac{\partial f}{\partial y} = S_y \otimes f
\]

3. Form the image gradient, and compute its direction and amplitude.

\[
\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \\
\theta = \tan^{-1} \left( \frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}} \right) \\
||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}
\]

\begin{array}{l}
\text{gradient} \\
\text{direction} \\
\text{amplitude}
\end{array}
How does the gradient direction relate to these edges?
How do you find the edge of this signal?

intensity plot
How do you find the edge of this signal?

Using a derivative filter:

What’s the problem here?
Differentiation is very sensitive to noise

When using derivative filters, it is critical to blur first!

Input → Gaussian Blur → Blurred Signal → Convolution → Derivative of Blurred Signal

How much should we blur?
Derivative of Gaussian (DoG) filter

Derivative theorem of convolution:

$$\frac{\partial}{\partial x}(h \ast f) = \left(\frac{\partial}{\partial x} h\right) \ast f$$

- How many operations did we save?
- Any other advantages beyond efficiency?
Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

\[
\frac{f''(x)}{h^2} = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}
\]

1D derivative filter

\[1 \ 0 \ -1\]

Laplace filter
Basically a second derivative filter.
• We can use finite differences to derive it, as with first derivative filter.

\[
f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}
\]

1D derivative filter
[1 0 -1]

\[
f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}
\]

Laplace filter
[1 -2 1]
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering.
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

“zero crossings” at edges
Laplace and LoG filtering examples

Laplacian of Gaussian filtering

Laplace filtering
Laplacian of Gaussian vs Derivative of Gaussian

Laplacian of Gaussian filtering

Derivative of Gaussian filtering
Laplacian of Gaussian vs Derivative of Gaussian

Zero crossings are more accurate at localizing edges (but not very convenient).
2D Gaussian filters

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

Gaussian

Derivative of Gaussian

Laplacian of Gaussian

how does this relate to this lecture’s cover picture?
References

Basic reading:
• Szeliski textbook, Section 3.2