Note: I’ll ask the front row on the right to participate in a demo. All you have to do is say a number that I’ll give to you. If you don’t want to, it’s fine, but don’t sit in the front.
Let’s Take An Image
Let’s Fix Things

- We have noise in our image
- Let’s replace each pixel with a *weighted* average of its neighborhood
- Weights are *filter kernel*

\[
\begin{array}{ccc}
\text{Out} & \text{Out} & \text{Out} \\
\text{1/9} & \text{1/9} & \text{1/9} \\
\text{1/9} & \text{1/9} & \text{1/9} \\
\text{1/9} & \text{1/9} & \text{1/9} \\
\end{array}
\]
1D Case

**Signal/Front Row**

| 10 | 12 | 9  | 11 | 10 | 11 | 12 |

**Filter/David**

| 1/3 | 1/3 | 1/3 |

**Output**

| 10.33 | 10.66 | 10 | 10.66 | 11 |
Applying a Linear Filter

Input

<table>
<thead>
<tr>
<th>I11</th>
<th>I12</th>
<th>I13</th>
<th>I14</th>
<th>I15</th>
<th>I16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I21</td>
<td>I22</td>
<td>I23</td>
<td>I24</td>
<td>I25</td>
<td>I26</td>
</tr>
<tr>
<td>I31</td>
<td>I32</td>
<td>I33</td>
<td>I34</td>
<td>I35</td>
<td>I36</td>
</tr>
<tr>
<td>I41</td>
<td>I42</td>
<td>I43</td>
<td>I44</td>
<td>I45</td>
<td>I46</td>
</tr>
<tr>
<td>I51</td>
<td>I52</td>
<td>I53</td>
<td>I54</td>
<td>I55</td>
<td>I56</td>
</tr>
</tbody>
</table>

Filter

<table>
<thead>
<tr>
<th>F11</th>
<th>F12</th>
<th>F13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F21</td>
<td>F22</td>
<td>F23</td>
</tr>
<tr>
<td>F31</td>
<td>F32</td>
<td>F33</td>
</tr>
</tbody>
</table>

Output

<table>
<thead>
<tr>
<th>O11</th>
<th>O12</th>
<th>O13</th>
<th>O14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O21</td>
<td>O22</td>
<td>O23</td>
<td>O24</td>
</tr>
<tr>
<td>O31</td>
<td>O32</td>
<td>O33</td>
<td>O34</td>
</tr>
</tbody>
</table>
Applying a Linear Filter

Input & Filter

Output

\[ O_{11} = I_{11}F_{11} + I_{12}F_{12} + \ldots + I_{33}F_{33} \]
Applying a Linear Filter

\[
O_{12} = I_{12}F_{11} + I_{13}F_{12} + \ldots + I_{34}F_{33}
\]
**Applying a Linear Filter**

<table>
<thead>
<tr>
<th>Input</th>
<th>Filter</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>I11</td>
<td>F11</td>
<td>O11</td>
</tr>
<tr>
<td>I12</td>
<td>F12</td>
<td>O12</td>
</tr>
<tr>
<td>I13</td>
<td>F13</td>
<td>O13</td>
</tr>
<tr>
<td>I14</td>
<td>F21</td>
<td>O14</td>
</tr>
<tr>
<td>I15</td>
<td>F22</td>
<td>O15</td>
</tr>
<tr>
<td>I16</td>
<td>F23</td>
<td>O16</td>
</tr>
<tr>
<td>I21</td>
<td>F31</td>
<td>O21</td>
</tr>
<tr>
<td>I22</td>
<td>F32</td>
<td>O22</td>
</tr>
<tr>
<td>I23</td>
<td>F33</td>
<td>O23</td>
</tr>
<tr>
<td>I24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I56</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How many times can we apply a 3x3 filter to a 5x6 image?
Applying a Linear Filter

\[ O_{ij} = I_{ij}F_{11} + I_{i(j+1)}F_{12} + \ldots + I_{(i+2)(j+2)}F_{33} \]
Painful Details – Edge Cases
Convolution doesn’t keep the whole image. Suppose \( f \) is the image and \( g \) the filter.

**Full** – any part of \( g \) touches \( f \). **Same** – same size as \( f \); **Valid** – only when filter doesn’t fall off edge.

*f/g Diagram Credit: D. Lowe*
Painful Details – Edge Cases

What to about the “?” region?

Symm: fold sides over

Circular/Wrap: wrap around

pad/fill: add value, often 0
Painful Details – Does it Matter?

(I’ve applied the filter per-color channel)

Which padding did I use and why?

Note – this is a zoom of the filtered, not a filter of the zoomed
Painful Details – Does it Matter?

(I’ve applied the filter per-color channel)

Input Image

Box Filtered Symm Pad

Box Filtered Zero Pad

Note – this is a zoom of the filtered, not a filter of the zoomed
Practice with Linear Filters

Original

?
Practice with Linear Filters

Original

The Same!
Practice with Linear Filters

Original

Slide Credit: D. Lowe
Practice with Linear Filters

Original

Shifted **LEFT**
1 pixel
Practice with Linear Filters

Original

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

?
Practice with Linear Filters

Original

Shifted
Down
1 pixel
Practice with Linear Filters

Original

<table>
<thead>
<tr>
<th>1/9</th>
<th>1/9</th>
<th>1/9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
</tbody>
</table>

?
Practice with Linear Filters

Original

Blur (Box Filter)

Slide Credit: D. Lowe
Practice with Linear Filters

Original

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

\[
\begin{array}{ccc}
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\end{array}
\]
Practice with Linear Filters

Original

Sharpened
(Accentuates difference from local average)
Sharpening

before

after

Slide Credit: D. Lowe
Properties – Linear

Assume: I image f1, f2 filters

**Linear:** \( \text{apply}(I,f1+f2) = \text{apply}(I,f1) + \text{apply}(I,f2) \)

I is a box on black, and f1, f2 are rectangles

\[
A(\[\begin{array}{c}
\text{\[}\hspace{1cm}\text{\[}\hspace{1cm}\text{\[}
\end{array}\]
\end{array}\]
\] + \[\begin{array}{c}
\text{\[}\hspace{1cm}\text{\[}\hspace{1cm}\text{\[}
\end{array}\]
\end{array}\]
\]) = A(\[\begin{array}{c}
\text{\[}\hspace{1cm}\text{\[}\hspace{1cm}\text{\[}
\end{array}\]
\end{array}\]
\]) = \[\begin{array}{c}
\text{\[}\hspace{1cm}\text{\[}\hspace{1cm}\text{\[}
\end{array}\]
\end{array}\]
\]

\[
A(\[\begin{array}{c}
\text{\[}\hspace{1cm}\text{\[}\hspace{1cm}\text{\[}
\end{array}\]
\end{array}\]
\]) + A(\[\begin{array}{c}
\text{\[}\hspace{1cm}\text{\[}\hspace{1cm}\text{\[}
\end{array}\]
\end{array}\]
\]) = \[\begin{array}{c}
\text{\[}\hspace{1cm}\text{\[}\hspace{1cm}\text{\[}
\end{array}\]
\end{array}\]
\] + \[\begin{array}{c}
\text{\[}\hspace{1cm}\text{\[}\hspace{1cm}\text{\[}
\end{array}\]
\end{array}\]
\] = \[\begin{array}{c}
\text{\[}\hspace{1cm}\text{\[}\hspace{1cm}\text{\[}
\end{array}\]
\end{array}\]
\]

Note: I am showing filters un-normalized and blown up. They’re a smaller box filter (i.e., each entry is \(1/(\text{size}^2)\))
Properties – Shift-Invariant

Assume: \( I \) image, \( f \) filter

**Shift-invariant:** \( \text{shift}(\text{apply}(I,f)) = \text{apply}(\text{shift}(I,f)) \)

Intuitively: only depends on filter neighborhood

\[
A(\begin{array}{c}
\text{\includegraphics[width=1cm]{image1.png}}
\end{array}
, \begin{array}{c}
\text{\includegraphics[width=1cm]{image2.png}}
\end{array}) = \begin{array}{c}
\text{\includegraphics[width=1cm]{image3.png}}
\end{array}
\]

\[
A(\begin{array}{c}
\text{\includegraphics[width=1cm]{image4.png}}
\end{array}
, \begin{array}{c}
\text{\includegraphics[width=1cm]{image5.png}}
\end{array}) = \begin{array}{c}
\text{\includegraphics[width=1cm]{image6.png}}
\end{array}
\]
Painful Details – Signal Processing

Often called “convolution”. Actually cross-correlation.

Cross-Correlation
(Original Orientation)

Convolution
(Flipped in x and y)
Properties of Convolution

- Any shift-invariant, linear operation is a convolution
- Commutative: \( f \ast g = g \ast f \)
- Associative: \( (f \ast g) \ast h = f \ast (g \ast h) \)
- Distributes over +: \( f \ast (g + h) = f \ast g + f \ast h \)
- Scalars factor out: \( kf \ast g = f \ast kg = k (f \ast g) \)
- Identity (a single one with all zeros):
Questions?

• Nearly everything onwards is a convolution.
• This is important to get right.
Smoothing With A Box

Intuition: if filter touches it, it gets a contribution.
Solution – Weighted Combination

Intuition: weight contributions according to closeness to center.

\[ Filter_{ij} \propto 1 \]

\[ Filter_{ij} \propto \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]
Recognize the Filter?

It’s a Gaussian!

$$Filter_{ij} \propto \frac{1}{2\pi \sigma^2} \exp \left( - \frac{x^2 + y^2}{2\sigma^2} \right)$$
Smoothing With A Box & Gauss

Still have some speckles, but it’s not a big box

Input  
Box Filter  
Gauss. Filter
Gaussian Filters

$\sigma = 1$
filter = 21x21

$\sigma = 2$
filter = 21x21

$\sigma = 4$
filter = 21x21

$\sigma = 8$
filter = 21x21

Note: filter visualizations are independently normalized throughout the slides so you can see them better.
Applying Gaussian Filters
Applying Gaussian Filters

Input Image
(no filter)
Applying Gaussian Filters

$\sigma = 1$
Applying Gaussian Filters

$\sigma = 2$
Applying Gaussian Filters

\[ \sigma = 4 \]
Applying Gaussian Filters

$\sigma = 8$
Picking a Filter Size

Too small filter → bad approximation
Want size ≈ 6σ (99.7% of energy)
Left far too small; right slightly too small!

σ = 8, size = 21
σ = 8, size = 43
Runtime Complexity

Image size = \( N \times N = 6 \times 6 \)
Filter size = \( M \times M = 3 \times 3 \)

\[
\begin{align*}
&I_{11} & I_{12} & I_{13} & I_{14} & I_{15} & I_{16} \\
&F_{11} & F_{12} & F_{13} & \quad & I_{25} & I_{26} \\
&I_{21} & F_{21} & F_{22} & F_{23} & I_{35} & I_{36} \\
&I_{31} & F_{31} & F_{32} & F_{33} & I_{45} & I_{46} \\
&I_{41} & \quad & \quad & \quad & I_{55} & I_{56} \\
&I_{51} & \quad & \quad & \quad & \quad & \quad \\
&I_{61} & \quad & \quad & \quad & \quad & \quad
\end{align*}
\]

for ImageY in range(N):
  for ImageX in range(N):
    for FilterY in range(M):
      for FilterX in range(M):
        …

Time: \( O(N^2M^2) \)
Separability

Conv(vector, transposed vector) → outer product
Separability

\[ \text{Filter}_{ij} \propto \frac{1}{2\pi\sigma^2} \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]

\[ \rightarrow \]

\[ \text{Filter}_{ij} \propto \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{x^2}{2\sigma^2} \right) \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{y^2}{2\sigma^2} \right) \]
Separability

1D Gaussian $\ast$ 1D Gaussian = 2D Gaussian

Image $\ast$ 2D Gauss = Image $\ast$ (1D Gauss $\ast$ 1D Gauss)
= (Image $\ast$ 1D Gauss) $\ast$ 1D Gauss
Runtime Complexity

Image size = NxN = 6x6
Filter size = Mx1 = 3x1

for ImageY in range(N):
  for ImageX in range(N):
    for FilterY in range(M):
      ...
    for ImageX in range(N):
      for FilterX in range(M):
        ...

Time: O(N^2M)
Why Gaussian?

Gaussian filtering removes parts of the signal above a certain frequency. Often noise is high frequency and signal is low frequency.
Where Gaussian Fails
Applying Gaussian Filters

\[ \sigma = 1 \]
### Why Does This Fail?

Means can be arbitrarily distorted by outliers

#### Signal

| 10 | 12 | 9 | 8 | 1000 | 11 | 10 | 12 |

#### Filter

| 0.1 | 0.8 | 0.1 |

#### Output

| 11.5 | 9.2 | 107.3 | 801.9 | 109.8 | 10.3 |

What else is an “average” other than a mean?
Non-linear Filters (2D)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>81</td>
<td>13</td>
<td>22</td>
</tr>
<tr>
<td>125</td>
<td>830</td>
<td>76</td>
<td>80</td>
</tr>
<tr>
<td>144</td>
<td>92</td>
<td>108</td>
<td>95</td>
</tr>
<tr>
<td>132</td>
<td>102</td>
<td>106</td>
<td>87</td>
</tr>
</tbody>
</table>

[40, 081, 013, 125, 830, 076, 144, 092, 108]

Sort

[013, 040, 076, 081, 092, 108, 125, 144, 830]

92

[830, 076, 080, 092, 108, 095, 102, 106, 087]

Sort

[076, 080, 087, 092, 095, 102, 106, 108, 830]

95
Applying Median Filter

Median Filter (size=3)
Applying Median Filter

Median Filter
(size = 7)
Is Median Filtering Linear?

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 2 \\
2 & 2 & 2
\end{bmatrix}
+ \begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{bmatrix}
= \begin{bmatrix}
1 & 1 & 1 \\
1 & 2 & 2 \\
2 & 2 & 2
\end{bmatrix}
\]

Example from (I believe): Kristen Grauman
Some Examples of Filtering
Filtering – Sharpening

Image

Smoothed

Details
Filtering – Sharpening

Image

Details

$\alpha = 1$

"Sharpened" $\alpha = 1$

=
Filtering – Sharpening

Image + Details = \alpha \Rightarrow \text{“Sharpened” } \alpha=0
Filtering – Sharpening

Image

Details

+ $\alpha$

“Sharpened” $\alpha=2$
Filtering – Sharpening

Image + Details + \( \alpha \)

“Sharpened” \( \alpha = 0 \)
Filtering – Extreme Sharpening

Image + α

Details

“Sharpened” α=10
Filtering

What’s this Filter?

Dx

\[-1 \ 0 \ 1\]

Dy

\[-1 \ 0 \ 1\]
Filtering – Derivatives

$$(Dx^2 + Dy^2)^{1/2}$$
Filtering – Counting

How many “on” pixels have 10+ neighbors within 10 pixels?

Pixels

Disk

r=10

= ???
Filtering – Counting

How many “on” pixels have 10+ neighbors within 10 pixels?

Pixels \times \text{Density} = \text{Answer}
Filtering – Missing Data

Oh no! Missing data!
(and we know where)

Common with many non-normal cameras (e.g., depth cameras)
Filtering – Missing Data

Image

Binary Mask

Per-element /
Filtering – Missing Data

Image

Binary Mask

Per-element /
Filtering – Missing Data

Before
Filtering – Missing Data

After
Filtering – Missing Data

After (without missing data)