Image Filtering

Faisal Qureshi
faisal.qureshi@uoi.ca
What is an image?

- Image can be thought of as a function

  \[ f : \mathbb{R}^2 \rightarrow \mathbb{R} \]

  \[ f(x, y) \] gives the intensity at position \((x, y)\)

- In practice images are defined over rectangular regions with a finite range of intensities

  \[ f : [a, b] \times [c, d] \rightarrow [0, 1] \]
Images as Surfaces
Color Images

Projection of primary colour lights (picture from Wikipedia)

sRGB color rectangular with D65 white point (picture from Wikipedia)
A photograph of Mohammed Alim Khan (1880–1944), Emir of Bukhara, taken in 1911 by Sergei Mikhailovich Prokudin-Gorskii using three exposures with blue, green, and red filters. (From Wikipedia)
Color Images

- We can write a color image (a multi-channel image) as a vector-valued function

\[ f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix} \]

- It is easy to add extra channels to an image, such as an alpha channel typically used for encoding transparency

\[ f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \\ \alpha(x, y) \end{bmatrix} \]
Color (RGB) Images

Lena original

Red

Green

Blue
Digital Image

- We work with digital (or discrete) image
  - Sample on regular grid
  - Quantize each sample

\[ f[i, j] = \text{Quantize}\{f(\Delta i, \Delta j)\} \]
Digital Images

- A single channel discrete image can be represented as a matrix:

```
array([154, 166, 166, ..., 149, 157, 157],
      [157, 175, 175, ..., 157, 149, 149],
      [166, 171, 181, ..., 157, 157, 157],
      ..., 
      [126, 128, 124, ..., 104, 113, 101],
      [128, 137, 137, ..., 102, 102, 102],
      [137, 128, 124, ..., 102, 102, 113]], dtype=uint8)
```
Digital Images

- A multi-channel discrete image can be represented as a set of matrices (or as a tensor)

\[
\begin{array}{cccccc}
210 & 218 & 218 & \ldots & 226 & 222 \\
222 & 230 & 226 & \ldots & 222 & 226 \\
218 & 226 & 218 & \ldots & 222 & 222 \\
\vdots \\
218 & 214 & 222 & \ldots & 214 & 214 \\
214 & 218 & 218 & \ldots & 210 & 210 \\
218 & 214 & 222 & \ldots & 210 & 214 \\
\end{array}
\]  
\text{dtype=uint8)
Image Filtering
Image Filtering

• How can we remove the noise from an image?

More noisy
Image Filtering

• How can we remove the noise from an image?
More noisy

Less noisy
More noisy

Less noisy
More noisy

Less noisy
More noisy  

Less noisy
More noisy

Less noisy
Data Smoothing in 1D
Data Smoothing in 1D

Averaging
Data Smoothing in 1D

[Graph showing data smoothing by averaging a vector [1/3, 1/3, 1/3]]
Data Smoothing in 1D

Averaging

\([1/3, 1/3, 1/3]\)
Data Smoothing in 1D

Averaging

[1/3, 1/3, 1/3]

Averaging

Averaging
Data Smoothing in 1D
Data Smoothing in 1D

Corruption
Data Smoothing in 1D

Corruption

Smoothing
Data Smoothing in 1D

- Plot of Sin(x)
- Plot of noisy Sin(x)
- Plot of smoothed Sin(x)

Corruption → Smoothing
Averaging to get rid of noise
Averaging in 2D

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>90</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>

Averaged
Averaging in 2D

\[
\begin{array}{cccc}
1 & 2 & 1 & 4 & 5 \\
1 & 3 & 90 & 4 & 5 \\
30 & 1 & 1 & 3 & 1 \\
1 & 2 & 3 & 1 & 4 \\
1 & 3 & 2 & 60 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
\hline
\\hline
\end{array}
\]

Averaged
Averaging in 2D

\[
\frac{1+2+1+1+3+90+30+1+1}{9} = 14.4
\]

Averaged
Averaging in 2D

$$\frac{1+2+1+1+3+90+30+1+1}{9} = 14.4$$
Averaging in 2D

\[
\begin{array}{cccc}
1 & 2 & 1 & 4 \\
1 & 3 & 90 & 4 \\
30 & 1 & 1 & 3 \\
1 & 2 & 3 & 1 \\
1 & 3 & 2 & 60 \\
\end{array}
\]

\[
\frac{2+1+4+3+90+4+1+1+3}{9} = 12.1
\]

Averaged
\[
\frac{1+3+1+3+1+4+2+60+1}{9} = 8.4
\]
What happens to the missing values?

We will come back to this later

Averaged
Averaging Filter

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>1</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>90</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>

Linear filtering (often referred to as convolution)

Removes sharp features or smoothes the image
Averaging Filter for Image Smoothing (Noise Removal)

Original image

3x3 average filter

5x5 average filter

7x7 average filter
Image filtering

• Image enhancement:
  Denoise, resize, increase contrast

• Analysis:
  Texture, edges, feature points

• Pattern matching and recognition:
  Template matching
Image Filtering

Original image

Blurred

Sharpened

Smoothed with edge-preserving filter

Courtesy: Computer Vision: Algorithms and Applications by R. Szelsiki
Examples of some kernels

3-by-3 averaging

<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>

5-by-5 summing

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

3-by-3 Gaussian

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

3-by-3 Gaussian + Sobel x

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Sobel y

<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>

Sobel x

<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>

3-by-3 Gaussian + Sobel x

<table>
<thead>
<tr>
<th>2</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Linear Filters

Source: D. Lowe
Linear Filters
Linear Filters

Source: D. Lowe
Linear Filters

Source: D. Lowe
Linear Filters

Sobel filter:

\[
\begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1
\end{bmatrix}
\]
2D Linear Filtering

- A neighbourhood operator where the output pixel value is equal to a weighted sum of input pixels

\[
g(i, j) = \sum_{k,l} f(i + k, j + l) h(k, l)
\]

Eg. the following is an 3-by-3 averaging kernel

<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>
Linear Filtering in 1D
Linear Filtering in 1D

Signal

\[ f = \]
Linear Filtering in 1D

Signal
\[ f = \]

Mask, kernel or filter
\[ h = \]

-\( w \)  -\( w+1 \)  0  w-1  w
Linear Filtering in 1D

Signal
\[ f = \text{Mask, kernel or filter} \]

Mask, kernel or filter
\[ h = \text{width} = 2 \times w + 1 \]
Linear Filtering in 1D

Signal
\[ f = \]

Mask, kernel or filter
\[ h = \]
width = \( 2 \times w + 1 \)

Cross-correlation:
\[ CC(i) = \sum_{k \in [-w, w]} f(i + k)h(k) \]
Linear Filtering in 1D

Signal
\[ f = \]

Mask, kernel or filter
\[ h = \]
width = 2 \* w + 1

Cross-correlation:
\[ CC(i) = \sum_{k \in [-w,w]} f(i + k)h(k) \]

Convolution:
\[ (f \ast h)(i) = \sum_{k \in [-w,w]} f(i - k)h(k) \]
Linear Filtering 1D Example

\[ f = \begin{bmatrix} 1 & 3 & 4 & 1 & 10 & 3 & 0 & 1 \end{bmatrix} \]

\[ h = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \]

Cross-correlation: \[ CC(i) = \sum_{k \in [-w, w]} f(i + k)h(k) \]

Convolution: \[ (f \ast h)(i) = \sum_{k \in [-w, w]} f(i - k)h(k) \]
Linear Filtering 1D Example

\[ f = \begin{bmatrix} 1 & 3 & 4 & 1 & 10 & 3 & 0 & 1 \end{bmatrix} \]

\[ h = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \]

Cross-correlation: \[ CC(i) = \sum_{k \in [-w, w]} f(i + k)h(k) \]

\[ \begin{bmatrix} -3 & 2 & -6 & -2 & 10 & 2 \end{bmatrix} \]

Convolution: \[ (f * h)(i) = \sum_{k \in [-w, w]} f(i - k)h(k) \]
Linear Filtering 1D Example

\[ f = \begin{bmatrix} 1 & 3 & 4 & 1 & 10 & 3 & 0 & 1 \end{bmatrix} \]

\[ h = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \]

Cross-correlation:

\[ CC(i) = \sum_{k \in [-w, w]} f(i + k)h(k) \]

\[ \begin{bmatrix} -3 & 2 & -6 & -2 & 10 & 2 \end{bmatrix} \]

Convolution:

\[ (f \ast h)(i) = \sum_{k \in [-w, w]} f(i - k)h(k) \]

\[ \begin{bmatrix} 3 & -2 & 6 & 2 & -10 & -2 \end{bmatrix} \]
Linear Filtering 1D Example

\[ f = \begin{bmatrix} 1 & 3 & 4 & 1 & 10 & 3 & 0 & 1 \end{bmatrix} \]

\[ h = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \]

Cross-correlation:

\[ CC(i) = \sum_{k \in [-w, w]} f(i + k)h(k) \]

Convolution:

\[ (f \ast h)(i) = \sum_{k \in [-w, w]} f(i - k)h(k) \]
Linear filtering in 2D

Cross-correlation:

\[ CC(i, j) = \sum_{k \in [-w, w]} \sum_{l \in [-h, h]} f(i + k, j + l) h(k, l) \]

Convolution:

\[ (f \ast h)(i, j) = \sum_{k \in [-w, w]} \sum_{l \in [-h, h]} f(i - k, j - l) h(k, l) \]
Convolution

- Equivalent to flipping the filter in both directions
- Convolution and cross-correlation are the same for symmetric filters

[Source: S. Fidler]
Optical Convolution

- Camera Shake

\[ \text{Camera Shake} \quad = \quad \ast \]

*Figure: Fergus, et al., SIGGRAPH 2006*

- Blur in out-of-focus regions of an image.

*Figure: Bokeh: [http://lullaby.homepage.dk/diy-camera/bokeh.html](http://lullaby.homepage.dk/diy-camera/bokeh.html)*

Click for more info

[Source: N. Snavely]
Linear Filtering

• Linearity

\[ \text{filter}(f_1, f_2) = \text{filter}(f_1) + \text{filter}(f_2) \]

• Shift invariance

\[ \text{filter} (\text{shift}(f)) = \text{shift}(\text{filter}(f)) \]

Any linear, shift-invariant filter can be represented as a convolution
Properties of Convolution

• Commutative: \( a \ast b = b \ast a \)

• Associative: \( a \ast (b \ast c) = (a \ast b) \ast c \)

  • E.g., \(((a \ast b_1) \ast b_2) \ast b_3) = a \ast (b_1 \ast b_2 \ast b_3)\)

• Distributes over addition: \( a \ast (b + c) = a \ast b + a \ast c \)

• Scalar factors out: \( k a \ast b = a \ast kb + k(a \ast b) \)

• Identity: \( a \ast e = a \) where \( e \) is unit impulse \( e = [0, 1, 0] \)
Properties of Convolution

• The Fourier transform of two convolved images is the produce of their individual Fourier transforms

\[ \mathcal{F}(f \ast h) = \mathcal{F}(f) \cdot \mathcal{F}(h) \]

• This is a very important result. Why?

• Consider the relative complexity of convolution and Fourier transform

\[ f \ast h = \mathcal{F}^{-1}(\mathcal{F}(f) \cdot \mathcal{F}(h)) \]
Gaussian Filter

- This kernel is an approximation of the Gaussian function

\[ G(k, l; 0, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{k^2 + l^2}{2\sigma^2}} \]

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

\[
\begin{array}{cccc}
0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\
0.013 & 0.059 & 0.097 & 0.059 & 0.013 \\
0.022 & 0.097 & 0.159 & 0.097 & 0.022 \\
0.013 & 0.059 & 0.097 & 0.059 & 0.013 \\
0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\
\end{array}
\]

5 x 5, \( \sigma = 1 \)
Gaussian Filter: Parameters

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

Source: K. Grauman
Gaussian filter: Variance

Source: K. Grauman
Gaussian Filters

- Blurs the image, i.e., removes “high-frequency” components from the image (acts as low-pass filter)

- Convolution with itself is a Gaussian

- Convolving twice with Gaussian kernel of width $\sigma$ is the same as convolving once with kernel of width $\sigma\sqrt{2}$

- Apply a Gaussian filter with $\sigma_1$ and then apply a Gaussian filter with $\sigma_2$ is the same as applying once with Gaussian filter with $\sqrt{\sigma_1^2 + \sigma_2^2}$
Gaussian Filters

• Separable kernel — factors into outer product of two 1D Gaussians

• All values are positive

• Values sum to 1

• The size of the mask (plus the variance) determines the extent of smoothing
Gaussian Filters

\[
\begin{align*}
\sigma &= 0.1 \\
\sigma &= 2 \\
\sigma &= 5 \\
\sigma &= 12 \\
\sigma &= 25
\end{align*}
\]
Average

Gaussian
Separable Filters

- A n-dimensional filter that can be expressed as an outer product of n 1-dimensional filters
Separable Filters

2D convolution

Two 1D convolutions

(1) (2)
Separable Filters

• For separable filters

\[ O(w_kwh) + O(h_kwh) \]

• For non-separable filters

\[ O(w_kh_kwh) \]

\( w_k \) and \( h_k \) denote kernel width and height, respectively. \( w \) and \( h \) represent image width and height, respectively.
Separable Filters
Separable Filters
Separable Filters
Separable Filter
Separable Filters

• Use *Singular Value Decomposition* (SVD) to determine if a filter is separable

• If only one singular value is non-zero then it is separable

\[ F = U \Sigma V^T = \sum_i \sigma_i u_i v_i^T \]

with \( \Sigma = \text{diag}(\sigma_i) \)

• Vertical and horizontal filters are: \( \sqrt{\sigma_1} u_1 \) and \( \sqrt{\sigma_1} v_1^T \)
Show that 2D Gaussian Filter is Separable

\[
G(x, y) = \frac{1}{2\pi\sigma^2} \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right)
\]

\[
= \frac{1}{2\pi\sigma^2} \exp \left( -\frac{x^2}{2\sigma^2} \right) \exp \left( -\frac{y^2}{2\sigma^2} \right)
\]

\[
= \left[ \frac{1}{\sigma\sqrt{2\pi}} \exp \left( -\frac{x^2}{2\sigma^2} \right) \right] \left[ \frac{1}{\sigma\sqrt{2\pi}} \exp \left( -\frac{y^2}{2\sigma^2} \right) \right]
\]

\[
= G_1(x)G_1(y)
\]
What happens to the missing values?

We will come back to this later.

Averaged
## Missing values?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>3</td>
<td>90</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>3</td>
<td>2</td>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>90</td>
<td>4</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>60</td>
</tr>
</tbody>
</table>

Set to a particular value (say 0)
### Missing values?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>90</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>90</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>60</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>1</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>90</td>
<td>4</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>60</td>
</tr>
</tbody>
</table>

**Copy edge**
### Missing values?

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>90</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>90</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Wrap around (reflect)
Missing values?

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>90</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>90</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>

Typically not done in practice, since the output size is not the same as the input size, which leads to host of practical headaches.

Do nothing

*Output size is not the same as input*
Linear Filtering: Rules of Thumbs

• For Gaussian filters, set filter half-width to about $3\sigma$

• Typically filter dimensions are odd, so we can express filter width and height as follows:
  
  \[
  \text{Filter width} = 2w_k + 1 \\
  \text{Filter height} = 2h_k + 1
  \]

  Here, $w_k$ and $h_k$ are filter half-width and half-height, respectively.

• Where possible, prefer separable filters (significant speed up for large images)
Integral Image

\[ S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j] \]

\[
\begin{array}{c|c|c|c|c|c}
  i & 1 & 2 & 3 & -1 & 3 \\
  \hline
  2 & 34 & 5 & 3 & 2 \\
  \hline
  3 & 2 & 3 & 42 & 5 \\
  \hline
  -3 & 1 & 4 & 98 & 3 \\
  \hline
  1 & 2 & 3 & 2 & 5 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c}
  j & 1 & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} \\
  \hline
  2 & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} \\
  \hline
  3 & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} \\
  \hline
  -3 & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} \\
  \hline
  1 & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} & \hspace{1cm} \\
\end{array}
\]
Integral Image

$$X[i, j]$$

$$S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j]$$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>42</td>
<td>5</td>
</tr>
<tr>
<td>-3</td>
<td>1</td>
<td>4</td>
<td>98</td>
<td>3</td>
</tr>
<tr>
<td>-3</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

\[ S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j] \]
### Integral Image

\[ S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j] \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>-1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>42</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>-3</td>
<td>1</td>
<td>4</td>
<td>98</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
Integral Image

\[ S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j] \]

\[
\begin{array}{cccc}
1 & 2 & 3 & -1 \\
2 & 34 & 5 & 3 \\
3 & 2 & 3 & 42 \\
-3 & 1 & 4 & 98 \\
1 & 2 & 3 & 2 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 3 & 6 & 5 \\
3 &  &  & \\
 &  &  & \\
 &  &  & \\
1 & 2 & 3 & 5 \\
\end{array}
\]
Integral Image

\[ S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j] \]

\[
\begin{array}{cccccc}
1 & 2 & 3 & -1 & 3 & \\
2 & 34 & 5 & 3 & 2 & \\
3 & 2 & 3 & 42 & 5 & \\
-3 & 1 & 4 & 98 & 3 & \\
1 & 2 & 3 & 2 & 5 & \\
\end{array}
\]

\[
\begin{array}{cccccc}
1 & 3 & 6 & 5 & 8 & \\
3 & 39 & 47 & 49 & 54 & \\
\end{array}
\]
Integral Image

\[
X[i, j] \quad S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j]
\]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>-1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>34</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>42</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>98</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

|   | 1 | 3 | 6 | 5  | 8 |
|---|---|---|----|----|
| 1 | 3 | 39| 47 | 49 | 54|
| 2 | 3 | 44| 55 | 99 |109|
| 3 | 3 | 42| 57 |199 |212|
| 4 | 45| 63|207 |225 |   |
Integral Image

\[ S[m, n] = \sum_{i=m}^{n} \sum_{j=n}^{n} X[i, j] \]

<table>
<thead>
<tr>
<th>X[i, j]</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>-1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>34</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>42</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>1</td>
<td>4</td>
<td>98</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S[m, n]</th>
<th>1</th>
<th>3</th>
<th>6</th>
<th>5</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>39</td>
<td>47</td>
<td>49</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>44</td>
<td>55</td>
<td>99</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>42</td>
<td>57</td>
<td>199</td>
<td>212</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>45</td>
<td>63</td>
<td>207</td>
<td>225</td>
<td></td>
</tr>
</tbody>
</table>

192
Integral Image

\[ S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j] \]

\[
\begin{array}{|c|c|c|c|c|}
\hline
1 & 2 & 3 & -1 & 3 \\
\hline
2 & 34 & 5 & 3 & 2 \\
\hline
3 & 2 & 3 & 42 & 5 \\
\hline
-3 & 1 & 4 & 98 & 3 \\
\hline
1 & 2 & 3 & 2 & 5 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|}
\hline
1 & 3 & 6 & 5 & 8 \\
\hline
3 & 39 & 47 & 49 & 54 \\
\hline
6 & 44 & 55 & 99 & 109 \\
\hline
3 & 42 & 57 & 199 & 212 \\
\hline
4 & 45 & 63 & 207 & 225 \\
\hline
\end{array}
\]

192

199
Integral Image

\[ S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j] \]

\[ X[i, j] \]

\[
\begin{array}{cccc}
1 & 2 & 3 & -1 \\
2 & 34 & 5 & 3 \\
3 & 2 & 3 & 42 \\
-3 & 1 & 4 & 98 \\
1 & 2 & 3 & 2 \\
\end{array}
\]

192

\[
\begin{array}{cccccc}
1 & 3 & 6 & 5 & 8 \\
3 & 39 & 47 & 49 & 54 \\
6 & 44 & 55 & 99 & 109 \\
3 & 42 & 57 & 199 & 212 \\
4 & 45 & 63 & 207 & 225 \\
\end{array}
\]

199-3
Integral Image

\[ X[i, j] \]

\[ S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j] \]
Integral Image

\[ S[m, n] = \sum_{i \leq m} \sum_{j \leq n} X[i, j] \]

\[
\begin{array}{cccc}
1 & 2 & 3 & -1 \\
2 & 34 & 5 & 3 \\
3 & 2 & 3 & 42 \\
-3 & 1 & 4 & 98 \\
1 & 2 & 3 & 2 \\
\end{array}
\]

\[ 192 \]

\[
\begin{array}{cccc}
1 & 3 & 6 & 5 \\
3 & 39 & 47 & 49 \\
6 & 44 & 55 & 99 \\
-3 & 42 & 57 & 199 \\
4 & 45 & 63 & 207 \\
\end{array}
\]

\[ 199 - 3 - 5 + 1 = 192 \]
Integral Images

• No matter the size of summing window, we can compute sum using 3 arithmetic operations
Integral Images

• No matter the size of summing window, we can compute sum using 3 arithmetic operations
Integral Images

- No matter the size of summing window, we can compute sum using 3 arithmetic operations
Integral Images

- No matter the size of summing window, we can compute sum using 3 arithmetic operations
Integral Images

- No matter the size of summing window, we can compute sum using 3 arithmetic operations.
Integral Images

- No matter the size of summing window, we can compute sum using 3 arithmetic operations
Integral Images

• No matter the size of summing window, we can compute sum using 3 arithmetic operations

\[ A-B-C+D \]
Image Denoising

Gaussian Filter

Slide credit: D. Hoiem
Gaussian Filtering

- Smoothing with larger standard deviations suppresses noise, but also blurs the image

Slide credit: S. Lazebnik
Salt-and-Pepper Noise

3 x 3  5 x 5  7 x 7
Median Filtering

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>90</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>31</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

Replace with the median value
Median Filtering

Replace with the median value

Finding median
# Median Filtering

<table>
<thead>
<tr>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>90</td>
<td>27</td>
</tr>
<tr>
<td>33</td>
<td>31</td>
<td>30</td>
</tr>
</tbody>
</table>

Replace with the median value

## Finding median

| 10 | 15 | 20 | 23 | 90 | 27 | 33 | 31 | 30 |
Median Filtering

Replace with the median value

Finding median

Sort
Median Filtering

Replace with the median value

Finding median

10 15 20 23 90 27 33 31 30

Sort

10 15 20 23 27 30 31 33 90
Median Filtering

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>90</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>31</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

Replace with the median value

**Finding median**

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>23</th>
<th>90</th>
<th>27</th>
<th>33</th>
<th>31</th>
<th>30</th>
</tr>
</thead>
</table>

Sort

|   | 10 | 15 | 20 | 23 | 27 | 30 | 31 | 33 | 90 |
Median Filtering

Finding median

10 | 15 | 20 | 23 | 90 | 27 | 33 | 31 | 30

Sort

10 | 15 | 20 | 23 | 27 | 30 | 31 | 33 | 90
Median Filtering

- Non-linear filtering
- Robustness to outliers

Filter width = 5
Non-linear Filtering

- Weighted median filtering
  - Pixels that are further afar from centre count less

- Bilateral filtering
  - Pixel contributions are weighed by spatial distances and intensity differences
  - Edge preserving
The slides on Bilateral Filtering are taken from https://people.csail.mit.edu/sparis/bf_course/
Blur Comes from Averaging across Edges

Same Gaussian kernel everywhere.
Bilateral Filter [Aurich 95, Smith 97, Tomasi 98]

No Averaging across Edges

The kernel shape depends on the image content.
Bilateral Filter Definition: an Additional Edge Term

Same idea: *weighted average of pixels.*

\[
BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s} (\| p - q \|) G_{\sigma_r} (\| I_p - I_q \|) I_q
\]

- **Normalization factor**: \( \frac{1}{W_p} \)
- **Space weight**: \( G_{\sigma_s} (\| p - q \|) \)
- **Range weight**: \( G_{\sigma_r} (\| I_p - I_q \|) \)
Illustration a 1D Image

- 1D image = line of pixels

- Better visualized as a plot
Gaussian Blur and Bilateral Filter

**Gaussian blur**

\[ GB[I]_p = \sum_{q \in S} G_{\sigma}(\|p - q\|) I_q \]

**Bilateral filter**

[Aurich 95, Smith 97, Tomasi 98]

\[ BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_b}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) I_q \]
Bilateral Filter on a Height Field

\[
BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) \cdot G_{\sigma_r}(\| I_p - I_q \|) \cdot I_q
\]

reproduced from [Durand 02]
Space and Range Parameters

\[
BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) I_q
\]

- **space** \(\sigma_s\): spatial extent of the kernel, size of the considered neighborhood.

- **range** \(\sigma_r\): “minimum” amplitude of an edge
Influence of Pixels

Only pixels close in space and in range are considered.
Exploring the Parameter Space

\[ \sigma_r = 0.1 \]  \hspace{2cm} \[ \sigma_r = 0.25 \]  \hspace{2cm} \[ \sigma_r = \infty \] (Gaussian blur)

\[ \sigma_s = 2 \]

\[ \sigma_s = 6 \]

\[ \sigma_s = 18 \]
Varying the Range Parameter

\[ \sigma_s = 2 \]

\[ \sigma_s = 6 \]

\[ \sigma_s = 18 \]

\[ \sigma_r = 0.1 \]

\[ \sigma_r = 0.25 \]

\[ \sigma_r = \infty \] (Gaussian blur)
Iterating the Bilateral Filter

\[ I_{(n+1)} = BF [I_{(n)}] \]

- Generate more piecewise-flat images
- Often not needed in computational photo.
Bilateral Filtering Color Images

For gray-level images

\[ BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(I_p - I_q) I_q \]

For color images

\[ BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(C_p - C_q) C_q \]

The bilateral filter is extremely easy to adapt to your need.
Hard to Compute

- Nonlinear
  \[ BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(\| I_p - I_q \|) I_q \]

- Complex, spatially varying kernels
  - Cannot be precomputed, no FFT...

- Brute-force implementation is slow > 10min
Summary

• Image filtering (linear filters)
  • Correlation
  • Convolution

• Image smoothing
  • Median Filtering (non-linear filters)
  • Bilateral Filtering (non-linear filters)

• Gaussian kernel

• Separability