Lecture: Pixels and Filters

Juan Carlos Niebles and Ranjay Krishna
Stanford Vision Lab
What we will learn today?

• Image sampling and quantization
• Image histograms
• Images as functions
• Linear systems (filters)
• Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7
What we will learn today?

• Image sampling and quantization
• Image histograms
• Images as functions
• Linear systems (filters)
• Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7
Types of Images

- Binary
- Gray Scale
- Color
Binary image representation

0: Black
1: White

Row 1
Row q

Slide credit: Ulas Bagci
Grayscale image representation

Slide credit: Ulas Bagci
Color Image - one channel
Color image representation
Images are sampled

What happens when we zoom into the images we capture?
Errors due Sampling

Slide credit: Ulas Bagci
Resolution

is a **sampling** parameter, defined in dots per inch (DPI) or equivalent measures of spatial pixel density, and its standard value for recent screen technologies is 72 dpi.

Slide credit: Ulas Bagci
Images are Sampled and Quantized

• An image contains discrete number of pixels
  – A simple example
  – Pixel value:
    • “grayscale”
      (or “intensity”): [0,255]
Images are Sampled and Quantized

• An image contains discrete number of pixels
  – A simple example
  – Pixel value:
    • “grayscale”
      (or “intensity”): [0, 255]
    • “color”
      – RGB: [R, G, B]
      – Lab: [L, a, b]
      – HSV: [H, S, V]
With this loss of information (from sampling and quantization),

Can we still use images for useful tasks?
What we will learn today?

• Image sampling and quantization
• Image histograms
• Images as functions
• Linear systems (filters)
• Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7
Histogram

- Histogram of an image provides the frequency of the brightness (intensity) value in the image.

def histogram(im):
    h = np.zeros(255)
    for row in im.shape[0]:
        for col in im.shape[1]:
            val = im[row, col]
            h[val] += 1
Histogram

- Histogram captures the distribution of gray levels in the image.
- How frequently each gray level occurs in the image
Histogram

Slide credit: Dr. Mubarak Shah
Histogram – use case
Histogram – another use case

Slide credit: Dr. Mubarak Shah
What we will learn today?

• Image sampling and quantization
• Image histograms
• Images as functions
• Linear systems (filters)
• Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7
Images as discrete functions

• Images are usually **digital** *(discrete)*:
  – **Sample** the 2D space on a regular grid

• Represented as a matrix of integer values

```
<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
</tbody>
</table>
```
Images as coordinates

Cartesian coordinates

\[
\begin{bmatrix}
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\end{bmatrix}
\begin{bmatrix}
f[-1, -1] & f[0, -1] & f[1, -1] \\
\hline
f[-1, 0] & f[0, 0] & f[1, 0] \\
\hline
f[-1, 1] & f[0, 1] & f[1, 1] \\
\hline
\vdots \\
\vdots \\
\vdots \\
\end{bmatrix}
\]

Notation for discrete functions
Images as coordinates

Cartesian coordinates

\[
\begin{bmatrix}
\vdots \\
\vdots \\
f[-1, -1] & f[0, -1] & f[1, -1] \\
\vdots \\
\vdots \\
\vdots \\
f[-1, 0] & f[0, 0] & f[1, 0] \\
\vdots \\
\vdots \\
\vdots \\
f[-1, 1] & f[0, 1] & f[1, 1] \\
\vdots \\
\vdots \\
\vdots \\
\end{bmatrix}
\]

Notation for discrete functions
Images as functions

- **An Image** as a function $f$ from $\mathbb{R}^2$ to $\mathbb{R}^M$:
  - $f(x, y)$ gives the **intensity** at position $(x, y)$
  - Defined over a rectangle, with a finite range:
    $$f: [a,b] \times [c,d] \rightarrow [0,255]$$
Images as functions

• **An Image** as a function $f$ from $\mathbb{R}^2$ to $\mathbb{R}^M$:
  - $f(x, y)$ gives the **intensity** at position $(x, y)$
  - Defined over a rectangle, with a finite range:
    $$f: [a, b] \times [c, d] \rightarrow [0, 255]$$

• A color image:
  $$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$
Histograms are a type of image function
What we will learn today?

- Image sampling and quantization
- Image histograms
- Images as functions
- Linear systems (filters)
- Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7
Systems and Filters

Filtering:

– Forming a new image whose pixel values are transformed from original pixel values

Goals:

• Goal is to extract useful information from images, or transform images into another domain where we can modify/enhance image properties
• Features (edges, corners, blobs...)
• super-resolution; in-painting; de-noising
De-noising
Salt and pepper noise

Super-resolution

In-painting

Bertamio et al.
System and Filters

- we define a system as a unit that converts an input function $f[n,m]$ into an output (or response) function $g[n,m]$, where $(n,m)$ are the independent variables.
  - In the case for images, $(n,m)$ represents the **spatial position in the image**.

$$f[n, m] \rightarrow \text{System } S \rightarrow g[n, m]$$
Images as coordinates

Cartesian coordinates

\[ f[n, m] = \begin{bmatrix}
  \ldots & f[-1, -1] & f[0, -1] & f[1, -1] & \ldots \\
  \ldots & f[-1, 0] & f[0, 0] & f[1, 0] & \ldots \\
  f[-1, 1] & f[0, 1] & f[1, 1] & \ldots \\
  \ldots & \ldots & \ldots & \ldots & \ldots 
\end{bmatrix} \]

Notation for discrete functions
2D discrete-space systems (filters)

$S$ is the **system operator**, defined as a mapping or assignment of a member of the set of possible outputs $g[n,m]$ to each member of the set of possible inputs $f[n,m]$.

$$ f[n,m] \rightarrow \boxed{\text{System } S} \rightarrow g[n,m] $$

$$ g = S[f], \quad g[n,m] = S\{f[n,m]\} $$

$$ f[n,m] \xrightarrow{S} g[n,m] $$
Filter example #1: Moving Average
Filter example #1: Moving Average

2D DS moving average over a $3 \times 3$ window of neighborhood

$$g[n, m] = \frac{1}{9} \sum_{k=n-1}^{n+1} \sum_{l=m-1}^{m+1} f[k, l]$$

$$= \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[n - k, m - l]$$
Filter example #1: Moving Average
Filter example #1: Moving Average

\[ f[n, m] \]

\[ g[n, m] \]
Filter example #1: Moving Average

\( f[n,m] \)

\( g[n,m] \)
Filter example #1: Moving Average

\[
\text{f}[n, m] = \sum_{k=0}^{m-1} g[n-k, m]
\]

\[
g[n, m] = \begin{cases}
0 & \text{if } n < 0 \\
10 & \text{if } n = 0 \\
20 & \text{if } n = 1 \\
30 & \text{if } n = 2 \\
0 & \text{otherwise}
\end{cases}
\]
Filter example #1: Moving Average
Filter example #1: Moving Average

\[ f[n, m] \]

\[ g[n, m] \]
Filter example #1: Moving Average

In summary:

- This filter creates a new pixel with an average of its neighborhood.
- Achieve smoothing effect (remove sharp features)
Filter example #1: Moving Average
Filter example #2: Image Segmentation

- Image segmentation based on a simple threshold:

\[ g[n, m] = \begin{cases} 
255, & f[n, m] > 100 \\ 
0, & \text{otherwise.} 
\end{cases} \]
Example Properties of systems

• Amplitude properties:
  
  — Additivity
  \[ S[f_i[n, m] + f_j[n, m]] = S[f_i[n, m]] + S[f_j[n, m]] \]

  — Homogeneity
  \[ S[\alpha f_i[n, m]] = \alpha S[f_i[n, m]] \]

  — Superposition
  \[ S[\alpha f_i[n, m] + \beta f_j[n, m]] = \alpha S[f_i[n, m]] + \beta S[f_j[n, m]] \]

  — Stability
  \[ |f[n, m]| \leq k \implies |g[n, m]| \leq ck \]

  — Invertibility
  \[ S^{-1}[S[f_i[n, m]]] = f[n, m] \]
Example Properties of linear systems

• Spatial properties
  — Causality
    \[
    \text{for } n < n_0, m < m_0, \text{ if } f[n, m] = 0 \implies g[n, m] = 0
    \]
  — Shift invariance:
    \[
    f[n - n_0, m - m_0] \xrightarrow{s} g[n - n_0, m - m_0]
    \]
These properties have to be true for **Linear Systems**

- **Amplitude properties:**
  - Additivity
    \[ S[f_i[n,m] + f_j[n,m]] = S[f_i[n,m]] + S[f_j[n,m]] \]
  - Homogeneity
    \[ S[\alpha f_i[n,m]] = \alpha S[f_i[n,m]] \]
  - Superposition
    \[ S[\alpha f_i[n,m] + \beta f_j[n,m]] = \alpha S[f_i[n,m]] + \beta S[f_j[n,m]] \]
These properties have to be true for Linear SHIFT INVARIANT Systems

• Amplitude properties:
  – Additivity

\[ S[f_i[n, m] + f_j[n, m]] = S[f_i[n, m]] + S[f_j[n, m]] \]

  – Homogeneity

\[ S[\alpha f_i[n, m]] = \alpha S[f_i[n, m]] \]

  – Superposition

\[ S[\alpha f_i[n, m] + \beta f_j[n, m]] = \alpha S[f_i[n, m]] + \beta S[f_j[n, m]] \]

  – Shift invariance:

\[ f[n - n_0, m - m_0] \overset{S}{\rightarrow} g[n - n_0, m - m_0] \]
What does shifting an image look like?

Cartesian coordinates

\[
f[n, m] = \begin{bmatrix}
\vdots & \vdots & \vdots \\
\ldots & f[-1, 1] & f[0, 1] & f[1, 1] \\
\ldots & f[-1, 0] & f[0, 0] & f[1, 0] \\
\ldots & f[-1, -1] & f[0, -1] & f[1, -1] \\
\vdots & \vdots & \vdots 
\end{bmatrix}
\]
Is the moving average system is shift invariant?

\[ g[n, m] = \frac{1}{9} \sum_{k=n-1}^{n+1} \sum_{l=m-1}^{m+1} f[k, l] \]

### f[n, m]

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<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>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
<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>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### g[n, m]

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>60</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>50</td>
<td>80</td>
<td>80</td>
<td>90</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>50</td>
<td>80</td>
<td>80</td>
<td>90</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>30</td>
<td>50</td>
<td>50</td>
<td>60</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Is the moving average system is **shift invariant**?

\[
f[n, m] \xrightarrow{s} g[n, m]
\]

\[
g[n, m] = \frac{1}{9} \sum_{k=n-1}^{n+1} \sum_{l=m-1}^{m+1} f[k, l] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[n-k, m-l]
\]

Let \( f[n - n_0, m - m_0] \) be a shifted input of \( f[n, m] \)

Now let’s pass \( f[n - n_0, m - m_0] \) through the system:

\[
f[n - n_0, m - m_0] \xrightarrow{s} \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[n - n_0 - k, m - m_0 - l]
\]

\[= g[n - n_0, m - m_0] \quad \text{Yes!} \]
Is the moving average system is casual?

\[ g[n, m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[n - k, m - l] \]

for \( n < n_0, m < m_0 \), if \( f[n, m] = 0 \) \( \implies g[n, m] = 0 \)
Linear Systems (filters)

\[ f[n, m] \rightarrow \text{System } S \rightarrow g[n, m] \]

- **Linear filtering:**
  - Form a new image whose pixels are a weighted sum of original pixel values
  - Use the same set of weights at each point

- **\( S \) is a linear system (function) iff it \( S \) satisfies**

\[
S[\alpha f_i[n, m] + \beta f_j[k, l]] = \alpha S[f_i[n, m]] + \beta S[f_j[k, l]]
\]

superposition property
Linear Systems (filters)

\[ f[n, m] \rightarrow \text{System } S \rightarrow g[n, m] \]

- Is the moving average a linear system?

- Is thresholding a linear system?
  - \( f_1[n,m] + f_2[n,m] > T \)
  - \( f_1[n,m] < T \)
  - \( f_2[n,m] < T \)

    No!
How do we characterize a linear shift invariant (LSI) system?

• Mathematically, how do you calculate \( g[n,m] \) from \( f[n,m] \)

\[
f[n, m] \rightarrow \text{System } S \rightarrow g[n, m]
\]
2D impulse function

• 1 at [0,0].
• 0 everywhere else
Impulse response to the moving filter

\[ \delta_2 \rightarrow h[n, m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \delta_2[n - k, m - l] \]
Impulse response to the moving filter

\[
i_2 \rightarrow h[n, m]
\]

\[
= \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \delta_2[n - k, m - l]
\]
Impulse response to the moving filter

\[
\delta_2 \rightarrow h[n, m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \delta_2[n - k, m - l]
\]
Impulse response to the moving filter

\[
\delta_2 \rightarrow h[n,m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \delta_2[n-k, m-l]
\]
Impulse response to the moving filter

\[
\delta_2 \rightarrow h[n, m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \delta_2[n-k, m-l]
\]
Impulse response to the moving filter

\[ \delta_2 \rightarrow h[n, m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \delta_2[n - k, m - l] \]
Impulse response to the moving filter

\[
\delta_2 \rightarrow h[n, m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \delta_2[n-k, m-l]
\]
Impulse response to the moving filter

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & 0 & \\
0 & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & 0 & \\
0 & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\delta_2 \xrightarrow{s} g[n,m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \delta_2[n-k, m-l]
\]
Impulse response of the 3 by 3 moving average filter

\[
h[n, m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \delta_2[n - k, m - l]
\]

\[
= \begin{bmatrix}
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
\end{bmatrix}
\]
Any linear shift invariant system

- By passing an impulse into a linear system, we get it’s impulse response.

- So, if we don’t know what the linear system is doing, we can pass an impulse into it to get a filter $h[n, m]$ that tells us what the system is actually doing.

$$\delta_2[n, m] \rightarrow \text{System } S \rightarrow h[n, m]$$

- But how do we use $h[n, m]$ to calculate $g[n, m]$ from $f[n, m]$?

$$f[n, m] \rightarrow \text{System } S \rightarrow g[n, m]$$
General linear shift invariant system

• Let’s say our input f is a 3x3 image:

<table>
<thead>
<tr>
<th>f[0,0]</th>
<th>f[0,1]</th>
<th>f[1,1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>f[1,0]</td>
<td>f[1,1]</td>
<td>f[1,2]</td>
</tr>
<tr>
<td>f[2,0]</td>
<td>f[2,1]</td>
<td>f[2,2]</td>
</tr>
</tbody>
</table>

We can rewrite $f[n,m]$ as a sum of delta functions:

$$f[n,m] = f[0,0] \times \delta_2[n - 0, m - 0]$$
$$+ f[0,1] \times \delta_2[n - 0, m - 1]$$
$$+ f[0,2] \times \delta_2[n - 0, m - 2]$$
$$+ \ldots$$

Or you can write it as:

$$f[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l] \times \delta_2[n - k, m - l]$$
3 properties we need:

Now, we know what happens when we send a delta function through an LSI system:

\[ \delta_2[n, m] \rightarrow \text{System } S \rightarrow h[n, m] \]

We also know that LSI systems shift the output if the input is shifted:

\[ \delta_2[n - k, m - l] \rightarrow \text{System } S \rightarrow h[n - k, m - l] \]

Finally, the superposition principle:

\[ S[ \alpha f_i[n, m] + \beta f_j[k, l] ] = \alpha S[ f_i[n, m] ] + \beta S[ f_j[k, l] ] \]
We can generalize this superposition principle...

\[ S[ \alpha f_i[n, m] + \beta f_j[k, l] ] = \alpha S[ f_i[n, m] ] + \beta S[f_j[k, l]] \]

... with our summation of deltas...

\[ f[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \times \delta_2[n - k, m - l] \]

... as follows:

\[
S \left[ \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \times \delta_2[n - k, m - l] \right] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \times S[\delta_2[n - k, m - l]]
\]
Linear shift invariant system

\[
S \left[ \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \times \delta_2[n - k, m - l] \right]

= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \times S[\delta_2[n - k, m - l]]

= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \times h[n - k, m - l]
\]
Linear Shift Invariant systems

An LSI system is completely specified by its impulse response.

\[ f[n, m] \rightarrow \mathcal{S} \text{ LSI} \rightarrow \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l] \]

Discrete convolution

\[ f[n, m] \ast h[n, m] \]
What we will learn today?

- Image sampling and quantization
- Image histograms
- Images as functions
- Linear systems (filters)
- **Convolution** and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7
2D convolution

2D convolution is very similar to 1D.
• The main difference is that we now have to iterate over 2 axis instead of 1.

\[ f[n, m] \ast h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l] \]

Assume we have a filter(h[]) that is 3x3. and an image (f[]) that is 7x7.
2D convolution

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

\[
f[n, m] \ast h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]
\]

Assume we have a filter \(h[],\) that is 3x3. and an image \(f[],\) that is 7x7.
2D convolution

2D convolution is very similar to 1D.
• The main difference is that we now have to iterate over 2 axis instead of 1.

\[ f[n, m] \ast h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l] \]

Assume we have a filter(h[,]) that is 3x3. and an image (f[,]) that is 7x7.
2D convolution

2D convolution is very similar to 1D.
• The main difference is that we now have to iterate over 2 axis instead of 1.

\[
f[n, m] \ast h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]
\]

Assume we have a filter(h[,]) that is 3x3. and an image (f[,]) that is 7x7.
2D convolution

2D convolution is very similar to 1D.
• The main difference is that we now have to iterate over 2 axis instead of 1.

\[ f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l] \]

Assume we have a filter(h[],) that is 3x3. and an image (f[],) that is 7x7.
2D convolution

2D convolution is very similar to 1D.
- The main difference is that we now have to iterate over 2 axis instead of 1.

\[ f[n, m] \ast h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \ h[n - k, m - l] \]

Assume we have a filter(h[,] that is 3x3. and an image (f[,] that is 7x7.
Convolution

\[ f \ast h = \sum_{k} \sum_{l} f(k, l)h(-k, -l) \]

- **f** = Image
- **h** = Kernel

**f**

\[
\begin{align*}
  f_1 & f_2 & f_3 \\
  f_4 & f_5 & f_6 \\
  f_7 & f_8 & f_9 \\
\end{align*}
\]

**h**

\[
\begin{align*}
  h_1 & h_2 & h_3 \\
  h_4 & h_5 & h_6 \\
  h_7 & h_8 & h_9 \\
\end{align*}
\]

**f \ast h**

\[
\begin{align*}
  f_1h_9 + f_2h_8 + f_3h_7 \\
  + f_4h_6 + f_5h_5 + f_6h_4 \\
  + f_7h_3 + f_8h_2 + f_9h_1 \\
\end{align*}
\]

**X - flip**

\[ h \]

**Y - flip**
2D convolution example

Input

 Kernel

 Output

Slide credit: Song Ho Ahn
2D convolution example

Slide credit: Song Ho Ahn
2D convolution example

Slide credit: Song Ho Ahn
2D convolution example

Slide credit: Song Ho Ahn
2D convolution example

\[
\begin{align*}
&= x[-1,0] \cdot h[1,1] + x[0,0] \cdot h[0,1] + x[1,0] \cdot h[-1,1] \\
&\quad + x[-1,1] \cdot h[1,0] + x[0,0] \cdot h[0,0] + x[1,0] \cdot h[-1,0] \\
&\quad + x[-1,2] \cdot h[1,-1] + x[0,2] \cdot h[0,-1] + x[1,2] \cdot h[-1,-1] \\
&= 0 \cdot 1 + 1 \cdot 3 + 2 \cdot 2 + 1 \cdot 0 + 0 \cdot 0 + 4 \cdot 0 + 5 \cdot 0 + 0 \cdot (-1) + 7 \cdot (-2) + 8 \cdot (-1) = -18
\end{align*}
\]

Slide credit: Song Ho Ahn
2D convolution example

Slide credit: Song Ho Ahn
2D convolution example

Slide credit: Song Ho Ahn
Convolution in 2D - examples

Original

*  

=?
Convolution in 2D - examples

Original * Filtered
(no change)
Convolution in 2D - examples
Convolution in 2D - examples

Original

Shifted right
By 1 pixel
Convolution in 2D - examples

Original

\[ \ast \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \ ? \]
Convolutions in 2D - examples

Original

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

Blur (with a box filter)
Convolution in 2D - examples

Original

(Note that filter sums to 1)

“details of the image”

\[ \begin{array}{ccc}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{array} \]  \quad - \quad \frac{1}{9}  \quad \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array} \quad = \quad ? \]
What does blurring take away?

- Let’s add it back:
Convolution in 2D – Sharpening filter

Sharpening filter: Accentuates differences with local average
Image support and edge effect

• A computer will only convolve **finite support signals**.
  • That is: images that are zero for n,m outside some rectangular region

• numpy’s convolution performs 2D DS convolution of finite-support signals.

\[
N_1 \times M_1 \ast N_2 \times M_2 = (N_1 + N_2 - 1) \times (M_1 + M_2 - 1)
\]
Image support and edge effect

• A computer will only convolve **finite support signals**.
• What happens at the edge?

- zero “padding”
- edge value replication
- mirror extension
- more (beyond the scope of this class)
What we will learn today?

- Image sampling and quantization
- Image histograms
- Images as functions
- Linear systems (filters)
  - Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7
(Cross) correlation (symbol: $\star \star$)

Cross correlation of two 2D signals $f[n,m]$ and $h[n,m]$:

$$f[n,m] \star \star h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[n,m] \times h[n+k, m+l]$$

Equivalent to a convolution without the flip
(Cross) correlation – example

![Image of a grid with labels 1 and 128 along one axis and an array of squares on the other axis.]

Courtesy of J. Fessler
(Cross) correlation – example

f
g = f + noise
g > 0.5

Courtesy of J. Fessler
(Cross) correlation – example

numpy’s correlate

g = f + noise

r > 0.5

g > 0.5

Courtesy of J. Fessler
(Cross) correlation – example

Norm. cross corr. score

Left

Right

scanline
Convolution

\[ f \ast h \]

Cross-correlation

\[ f \star h \]
Cross Correlation Application: Vision system for TV remote control
- uses template matching
Properties

• Commutative property:

\[ f ** h = h ** f \]

• Associative property:

\[ (f ** h_1) ** h_2 = f ** (h_1 ** h_2) \]

• Distributive property:

\[ f ** (h_1 + h_2) = (f ** h_1) + (f ** h_2) \]

The order doesn’t matter! \[ h_1 ** h_2 = h_2 ** h_1 \]
Properties

• Shift property:

\[ f[n, m] \ast \ast \delta_2[n - n_0, m - m_0] = f[n - n_0, m - m_0] \]

• Shift-invariance:

\[ g[n, m] = f[n, m] \ast \ast h[n, m] \]

\[ \implies f[n - l_1, m - l_1] \ast \ast h[n - l_2, m - l_2] \]

\[ = g[n - l_1 - l_2, m - l_1 - l_2] \]
Convolution vs. (Cross) Correlation

• A convolution is an integral that expresses the amount of overlap of one function as it is shifted over another function.
  — convolution is a filtering operation

• Correlation compares the similarity of two sets of data. Correlation computes a measure of similarity of two input signals as they are shifted by one another. The correlation result reaches a maximum at the time when the two signals match best.
  — correlation is a measure of relatedness of two signals
What we have learned today?

- Image sampling and quantization
- Image histograms
- Images as functions
- Linear systems (filters)
- Convolution and correlation