Lecture: Pixels and Filters

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

• Color
• 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?

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

• What is color?
• Color spaces
• White balancing
What is color?

• The result of interaction between physical light in the environment and our visual system.

• A psychological property of our visual experiences when we look at objects and lights, not a physical property of those objects or lights.

Slide credit: Lana Lazebnik
Interaction of light and surfaces

What is the observed color of any surface under monochromatic light?
Inspired Drake in “Hotline Bling”
Overview of Color

• What is color?
• Color spaces
• Use cases
Linear color spaces

• Defined by a choice of three *primaries*
• The coordinates of a color are given by the weights of the primaries used to match it

mixing two lights produces colors that lie along a straight line in color space

mixing three lights produces colors that lie within the triangle they define in color space
RGB space

• Primaries are monochromatic lights (for monitors, they correspond to the three types of phosphors)
Nonlinear color spaces: HSV

- Perceptually meaningful dimensions: Hue, Saturation, Value (Intensity)
- RGB cube on its vertex
Overview of Color

• What is color?
• Color spaces
• Use cases
Uses of color in computer vision

Color histograms for indexing and retrieval

Uses of color in computer vision

Skin detection

M. Jones and J. Rehg, Statistical Color Models with Application to Skin Detection, IJCV 2002.
Uses of color in computer vision

Nude people detection

Uses of color in computer vision

Image segmentation and retrieval


Source: S. Lazebnik
Uses of color in computer vision

Robot soccer


Source: K. Grauman
Uses of color in computer vision

Building appearance models for tracking


Source: S. Lazebnik
What we will learn today?

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

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

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

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

• Color
• Image sampling and quantization
• Image histograms
• Images as functions
• Linear systems (filters)
• Convolution and correlation
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>n</th>
<th>62</th>
<th>79</th>
<th>33</th>
<th>119</th>
<th>120</th>
<th>05</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>3</td>
<td>62</td>
<td>12</td>
<td>78</td>
<td>34</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>58</td>
<td>197</td>
<td>46</td>
<td>46</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>135</td>
<td>5</td>
<td>103</td>
<td>191</td>
<td>68</td>
<td>0</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>29</td>
<td>26</td>
<td>37</td>
<td>0</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>69</td>
<td>144</td>
<td>147</td>
<td>187</td>
<td>102</td>
<td>62</td>
<td>208</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>252</td>
<td>0</td>
<td>166</td>
<td>123</td>
<td>62</td>
<td>0</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>166</td>
<td>63</td>
<td>127</td>
<td>17</td>
<td>1</td>
<td>0</td>
<td>99</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
Images as coordinates

Cartesian coordinates

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

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?

• Color
• Image sampling and quantization
• Image histograms
• Images as functions
• Linear systems (filters)
• Convolution and correlation
Applications of linear systems or filters

De-noising

Super-resolution

Salt and pepper noise

In-painting

Bertamio et al
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
Intuition behind linear systems

• We will view linear systems as a type of function that operates over images
• Translating an image or multiplying by a constant leaves the semantic content intact
  – but can reveal interesting patterns
Aside

- Neural networks and specifically convolutional neural networks are a type of system or non-linear system that contains multiple individual linear sub-systems.

- (we will learn more about this later in class)
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**.

\[
\begin{align*}
  f[n, m] & \rightarrow \text{System } S & \rightarrow g[n, m]
\end{align*}
\]
Images as coordinates

Cartesian coordinates

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

Notation for discrete functions
2D discrete 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 $f[n,m]$.

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

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

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

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

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

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

[Diagram of a $3 \times 3$ window with weights 1, 1, 1 in each cell]
Filter example #1: Moving Average

\[ f[n, m] \]

\[ g[n, m] \]

Courtesy of S. Seitz
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

\[ f[n, m] \]

\[ g[n, m] \]
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

In summary:

- This filter “Replaces” each 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:

\[ g[n, m] = \begin{cases} 255, & f[n, m] > 100 \\ 0, & \text{otherwise.} \end{cases} \]
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
    \[ S^{-1}[S[f_i[n,m]]] = f[n,m] \]
Properties of systems

• Spatial properties
  – Causality
    \[ f[n, m] = 0 \implies g[n, m] = 0 \]
    for \( n < n_0, m < m_0 \)
  – Shift invariance:
    \[ f[n - n_0, m - m_0] \xrightarrow{S} g[n - n_0, m - m_0] \]
What does shifting an image look like?

Cartesian coordinates

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

\[ g[n, m] = \frac{1}{9} \sum_{k=n-1}^{n+1} \sum_{l=m-1}^{m+1} f[k, l] \]
Is the moving average system is shift invariant?

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

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

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

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:

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

Yes!
Is the moving average system is casual?

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

\[
f[n, m] \quad g[n, m]
\]

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?
  - \( f1[n,m] + f2[n,m] > T \)
  - \( f1[n,m] < T \)
  - \( f2[n,m] < T \)
  No!
Linear shift invariant systems

• Satisfies two properties:

• Superposition property

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

• Shift invariance:

\[ f[n - n_0, m - m_0] \xrightarrow{S} g[n - n_0, m - m_0] \]
moving average system is \textit{shift invariant} and \textit{linear}

• We are going to use this as an example to dive into interesting properties about linear shift-invariant systems.

• Why are linear shift invariant systems important?

\textbf{Our visual system is a}

\textbf{linear shift invariant system}
2D impulse function

- Let’s look at a special function
- 1 at [0,0].
- 0 everywhere else
Impulse response to the moving average filter

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

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

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

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

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

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

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

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

• By passing an impulse response 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 \( f[n, m] \)

\[
f[n, m] \rightarrow \text{System } S \rightarrow g[n, m]
\]
Remember the Moving Average filter and how we already used it’s impulse response

\[
f[n, m]
\]

\[
g[n, m]
\]
Remember the Moving Average filter and how we already used it’s impulse response
Remember the Moving Average filter and how we already used it’s impulse response
Remember the Moving Average filter and how we already used it’s impulse response
Remember the Moving Average filter and how we already used it’s impulse response
Remember the Moving Average filter and how we already used it’s impulse response
General linear shift invariant system

Let’s say our input $f$ is a $3 \times 3$ image:

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

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

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

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 a 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.

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

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

Discrete convolution

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

- Color
- 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 \( h[\cdot, \cdot] \) that is 3x3. and an image \( f[\cdot, \cdot] \) 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.
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.

\[
\begin{align*}
\mathbf{f}[n, m] \ast \mathbf{h}[n, m] &= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \mathbf{f}[k, l] \mathbf{h}[n - k, m - l]
\end{align*}
\]

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

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

\[ f = \text{Image} \]
\[ h = \text{Kernel} \]

\[ f = \begin{array}{ccc}
    f_1 & f_2 & f_3 \\
    f_4 & f_5 & f_6 \\
    f_7 & f_8 & f_9
\end{array} \]

\[ h = \begin{array}{ccc}
    h_1 & h_2 & h_3 \\
    h_4 & h_5 & h_6 \\
    h_7 & h_8 & h_9
\end{array} \]

\[ f \ast h = f_1h_9 + f_2h_8 + f_3h_7 + f_4h_6 + f_5h_5 + f_6h_4 + f_7h_3 + f_8h_2 + f_9h_1 \]

X - flip
Y - flip
2D convolution example

Slide credit: Song Ho Ahn
2D convolution example

Slide credit: Song Ho Ahn
2D convolution example

\[
\begin{align*}
&= x[0,-1] \cdot H[1,1] + x[1,-1] \cdot H[0,1] + x[2,-1] \cdot H[-1,1] \\
&\quad + x[0,0] \cdot H[1,0] + x[1,0] \cdot H[0,0] + x[2,0] \cdot H[-1,0] \\
&\quad + x[0,1] \cdot H[1,-1] + x[1,1] \cdot H[0,-1] + x[2,1] \cdot H[-1,-1] \\
&= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot (-1) + 5 \cdot (-2) + 6 \cdot (-1) = -20
\end{align*}
\]

Slide credit: Song Ho Ahn
2D convolution example

\[
\begin{align*}
  y_{ij} &= x_{i-1,j-1} \cdot h_{1,1} + x_{i-1,j} \cdot h_{0,1} + x_{i-1,j+1} \cdot h_{-1,1} \\
  &\quad + x_{i,j-1} \cdot h_{1,0} + x_{i,j} \cdot h_{0,0} + x_{i,j+1} \cdot h_{-1,0} \\
  &\quad + x_{i+1,j-1} \cdot h_{1,-1} + x_{i+1,j} \cdot h_{0,-1} + x_{i+1,j+1} \cdot h_{-1,-1} \\
  &= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 2 \cdot 0 + 3 \cdot 0 + 0 \cdot 0 + 5 \cdot (-1) + 6 \cdot (-2) + 0 \cdot (-1) = -17
\end{align*}
\]

Slide credit: Song Ho Ahn
2D convolution example

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

Slide credit: Song Ho Ahn
2D convolution example

\[
\begin{align*}
  &= x[0,0] \cdot k[1,1] + x[1,0] \cdot k[0,1] + x[2,0] \cdot k[-1,1] \\
  &\quad + x[0,1] \cdot k[1,0] + x[1,1] \cdot k[0,0] + x[2,1] \cdot k[-1,0] \\
  &\quad + x[0,2] \cdot k[1,-1] + x[1,2] \cdot k[0,-1] + x[2,2] \cdot k[-1,-1] \\
  &= 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 1 + 4 \cdot 0 + 5 \cdot 0 + 6 \cdot 0 + 7 \cdot (-1) + 8 \cdot (-2) + 9 \cdot (-1) = -24
\end{align*}
\]

Slide credit: Song Ho Ahn
2D convolution example

\[
\begin{align*}
\text{Output} &= x[1,0] \cdot h[1,1] + x[2,0] \cdot h[0,1] + x[3,0] \cdot h[-1,1] \\
&+ x[1,1] \cdot h[1,0] + x[2,1] \cdot h[0,0] + x[3,1] \cdot h[-1,0] \\
&+ x[1,2] \cdot h[1,-1] + x[2,2] \cdot h[0,-1] + x[3,2] \cdot h[-1,-1] \\
&= 2 \cdot 1 + 3 \cdot 2 + 0 \cdot 1 + 5 \cdot 0 + 6 \cdot 0 + 0 \cdot 0 + 8 \cdot (-1) + 9 \cdot (-2) + 0 \cdot (-1) = -18
\end{align*}
\]
Convolution in 2D - examples

Original

\[ \ast \]

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

\[ = \]

\[ ? \]
Convolution in 2D - examples

Original

Convolution

Filtered
(no change)
Convolution in 2D - examples

Original

\[ \text{Original} \ast \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = ? \]
Convolution in 2D - examples

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

Shifted right
By 1 pixel
Convolution in 2D - examples

Original

\[ \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array} \times \frac{1}{9} \quad = \quad ? \]
Convolution 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”

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

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

\[ \begin{array}{c}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{array} \]

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

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

\[ \begin{array}{c}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{array} \]

\[ \begin{array}{c}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{array} \]

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

\[ \begin{array}{c}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{array} \]
What does blurring take away?

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

**Sharpening filter:** Accentuates differences with local average
Implementation detail: 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)
Slide credit: Wolfram Alpha
What we will learn today?

• Color
• 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:  \( ** \)

Cross correlation of two 2D signals \( f[n,m] \) and \( h[n,m] \)

\[
f[n, m] ** h[n, m] = \sum_k \sum_l f[k, l] h[n - k, m - l]
\]

- Equivalent to a convolution without the flip
- Use it to measure ‘similarity’ between \( f \) and \( h \).
(Cross) correlation – example
(Cross) correlation – example

\[ f \rightarrow g = f + \text{noise} \rightarrow g > 0.5 \]

Courtesy of J. Fessler
(Cross) correlation – example

\[ f \xrightarrow{S} g = f + \text{noise} \xrightarrow{g > 0.5} \]

numpys correlate

Courtesy of J. Fessler
(Cross) correlation – example
Convolution

f
g

\( f \ast g \)

Cross-correlation

f
g

\( f \ast * g \)
Cross Correlation Application: Vision system for TV remote control
- uses template matching

Properties

• 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:

\[ f[n, m] \ast \delta_{2}[n - n_0, m - m_0] = f[n - n_0, m - m_0] \]

• Shift-invariance:

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

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

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

• When is correlation equivalent to convolution?
• In other words, when is $f**g = f*g$?
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