Lecture 3: Image Filtering

( unless otherwise stated slides are taken or adopted from Bob Woodham, Jim Little and Fred Tung )
Menu for Today (January 14, 2020)

Topics: Image Filtering (also topic for next week)

- Image as a function
- Linear filters
- Correlation / Convolution
- Filter examples: Box, Gaussian

Readings:

- Today’s Lecture: Forsyth & Ponce (2nd ed.) 4.1, 4.5
- Next Lecture: none

Reminders:

- Complete Assignment 0 (optional, ungraded) due today
- Assignment 1: Image Filtering and Hybrid Images is out, due January 28th
- Likely moving of midterm from 25th to 27th (midterm will cover subset)
- Office hours posted, will start tomorrow
Today’s “fun” Example:

Developed by the French company Varioptic, the lenses consist of an oil-based and a water-based fluid sandwiched between glass discs. Electric charge causes the boundary between oil and water to change shape, altering the lens geometry and therefore the lens focal length.

The intended applications are: **auto-focus** and **image stabilization**. No moving parts. Fast response. Minimal power consumption.

**Video Source:** [https://www.youtube.com/watch?v=2c6lCdDFOY8](https://www.youtube.com/watch?v=2c6lCdDFOY8)
Today’s “fun” Example:

Developed by the French company **Varioptic**, the lenses consist of an oil-based and a water-based fluid sandwiched between glass discs. Electric charge causes the boundary between oil and water to change shape, altering the lens geometry and therefore the lens focal length.

The intended applications are: **auto-focus** and **image stabilization**. No moving parts. Fast response. Minimal power consumption.

**Video Source:** [https://www.youtube.com/watch?v=2c6lCdDFOY8](https://www.youtube.com/watch?v=2c6lCdDFOY8)
Today’s “fun” Example:

Electrostatic field between the column of water and the electron (other side of power supply attached to the pipe) — see full video for complete explanation

Video Source: https://www.youtube.com/watch?v=NjLJ77luBdM
Today’s “fun” Example:

**Electrostatic** field between the column of water and the electron (other side of power supply attached to the pipe) — see full video for complete explanation

*Video Source: [https://www.youtube.com/watch?v=NjLJ77luBdM](https://www.youtube.com/watch?v=NjLJ77luBdM)*
Today’s “fun” Example:

As one example, in 2010, Cognex signed a licence agreement with Varioptic to add auto-focus capability to its DataMan line of industrial ID readers (press release May 29, 2012)

Video Source: https://www.youtube.com/watch?v=EU8LXxip1NM
Today’s “fun” Example:

As one example, in 2010, **Cognex** signed a licence agreement with Varioptic to add auto-focus capability to its DataMan line of industrial ID readers (press release May 29, 2012)

**Video Source:** https://www.youtube.com/watch?v=EU8LXxip1NM
Lecture 2: Re-cap

Surface reflection depends on both the viewing \((\theta_v, \phi_v)\) and illumination \((\theta_i, \phi_i)\) direction, with Bidirectional Reflection Distribution Function: \(\text{BRDF}(\theta_i, \phi_i, \theta_v, \phi_v)\)

**Lambertian** surface:

\[
\text{BRDF}(\theta_i, \phi_i, \theta_v, \phi_v) = \frac{\rho d}{\pi}
\]

**Mirror** surface: all incident light reflected in one directions \((\theta_v, \phi_v) = (\theta_r, \phi_r)\)

Slide adopted from: Ioannis (Yannis) Gkioulekas (CMU)
At a **microscopic** level, the process is **stochastic** (e.g., photon bouncing/being emitted in a random direction for a Lambertian surface), which (in part) causes **noise** in images under very low light scenarios; other sources of noise:

- electronic circuits
- variation in the number of photons sensed (quantum efficiency)
- quantization noise
Lecture 2: Re-cap

We take a “physics-based” approach to image formation
— Treat camera as an instrument that takes measurements of the 3D world

Basic abstraction is the **pinhole camera**

**Lenses** overcome limitations of the pinhole model while trying to preserve it as a useful abstraction

When **maximum accuracy** required, it is necessary to model additional details of each particular camera (and camera setting)
— Aside: This is called camera calibration
Lecture 2: Re-cap Pinhole Camera Abstraction

Pinhole Camera Abstraction

Diagram showing the pinhole camera model with image plane, pinhole, and object, with distances labeled as $f'$ and $z$.
Lecture 2: Re-cap Projection

3D object point $P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ projects to 2D image point $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$ where

- **Perspective**
  
  $x' = f' \frac{x}{z}$
  $y' = f' \frac{y}{z}$

- **Weak Perspective**
  
  $x' = m x$
  $y' = m y$
  
  $m = \frac{f'}{z_0}$

- **Orthographic**
  
  $x' = x$
  $y' = y$
Lecture 2: Re-cap Projection

**Camera Matrix**

\[ C = \begin{bmatrix} f' & 0 & 0 & 0 \\ 0 & f' & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \]

\[ P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ projects to 2D image point } \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \text{ where } P' = CP \]

\[ x' = f' \frac{x}{z} \]
\[ y' = f' \frac{y}{z} \]
Lecture 2: Re-cap Projection

**Camera Matrix**

\[
P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ projects to 2D image point } \quad P' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \text{ where } \quad P' = CP
\]

\[
C = \begin{bmatrix} f' & 0 & 0 & 0 \\ 0 & f' & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\]

\[
\begin{bmatrix} f' & 0 & 0 & 0 \\ 0 & f' & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} f'x \\ f'y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f'x}{z'} \\ \frac{f'y}{z'} \end{bmatrix}
\]

\[
x' = f' \frac{x}{z} \quad y' = f' \frac{y}{z}
\]
Lecture 2: Re-cap Projection

**Camera Matrix**

\[
C = \begin{bmatrix}
  f' & 0 & 0 & 0 \\
  0 & f' & 0 & 0 \\
  0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

Pixels are squared / lens is perfectly symmetric

Sensor and pinhole perfectly aligned

Coordinate system centered at the pinhole

\[
P = \begin{bmatrix}
  x \\
  y \\
  z \\
  1 \\
\end{bmatrix}
\]
projects to 2D image point \( P' = \begin{bmatrix}
  x' \\
  y' \\
  1 \\
\end{bmatrix} \)

where \( P' = CP \)
Lecture 2: Re-cap Projection

**Camera Matrix**

\[
C = \begin{bmatrix}
    f_x' & 0 & 0 & 0 \\
    0 & f_y' & 0 & 0 \\
    0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

- Pixels are squared / lens is perfectly symmetric
- Sensor and pinhole perfectly aligned
- Coordinate system centered at the pinhole

\[
P = \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix}
\]

projects to 2D image point \( P' = \begin{bmatrix}
x' \\
y' \\
1 \\
\end{bmatrix} \)

where \( P' = CP \)
Lecture 2: Re-cap Projection

Camera Matrix

\[
C = \begin{bmatrix}
    f' & 0 & 0 & c_x \\
    0 & f' & 0 & c_y \\
    0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

- Pixels are squared / lens is perfectly symmetric
- Sensor and pinhole perfectly aligned
- Coordinate system centered at the pinhole

\[
P = \begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix}
\]

projects to 2D image point

\[
P' = \begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix}
\]

where

\[
P' = CP
\]
Camera Matrix

\[
C = \begin{bmatrix}
    f_x' & 0 & 0 & c_x \\
    0 & f_y' & 0 & c_y \\
    0 & 0 & 1 & 0 \\
\end{bmatrix} \in \mathbb{R}^{4 \times 4}
\]

Pixels are squared / lens is perfectly symmetric
Sensor and pinhole perfectly aligned
Coordinate system centered at the pinhole

\[
P = \begin{bmatrix}
    x \\
    y \\
    z \\
    1 \\
\end{bmatrix}
\]
projects to 2D image point \( P' = \begin{bmatrix}
    x' \\
    y' \\
    1 \\
\end{bmatrix} \)
where \( P' = CP \)
— If pinhole is **too big** then many directions are averaged, blurring the image

— If pinhole is **too small** then diffraction becomes a factor, also blurring the image

— Generally, pinhole cameras are **dark**, because only a very small set of rays from a particular scene point hits the image plane

— Pinhole cameras are **slow**, because only a very small amount of light from a particular scene point hits the image plane per unit time

The role of a lens is to **capture more light** while preserving, as much as possible, the abstraction of an ideal pinhole camera.
Lecture 2: Re-cap Lenses
Lecture 2: Re-cap Snell’s Law

\[ n_1 \sin \alpha_1 = n_2 \sin \alpha_2 \]
Lecture 2: Re-cap Thin Lens Equation

\[ \frac{1}{z'} - \frac{1}{z} = \frac{1}{f} \]
Lecture 2: Re-cap

* image credit: https://catlikecoding.com/unity/tutorials/advanced-rendering/depth-of-field/circle-of-confusion/lens-camera.png
Lecture 2: Re-cap Thin Lens Equation

\[ \frac{1}{z'} - \frac{1}{z} = \frac{1}{f} \]
Another way of looking at the **focal length** of a lens. The incoming rays, parallel to the optical axis, **converge to a single point a distance f behind the lens**. This is where we want to place the image plane.
Lecture 2: Re-cap

Chromatic **aberration**

- Index of refraction depends on wavelength, $\lambda$, of light
- Light of different colours follows different paths
- Therefore, not all colours can be in equal focus

**Scattering** at the lens surface

- Some light is reflected at each lens surface

There are other **geometric phenomena/distortions**

- pincushion distortion
- barrel distortion
- etc
Human Eye

— The eye has an **iris** (like a camera)

— **Focusing** is done by changing shape of lens

— When the eye is properly focused, light from an object outside the eye is imaged on the **retina**

— The retina contains light receptors called **rods** and **cones**

Slide adopted from: Steve Seitz
Human Eye

— The eye has an iris (like a camera)

— **Focusing** is done by changing shape of lens

— When the eye is properly focused, light from an object outside the eye is imaged on the retina

— The retina contains light receptors called rods and cones

\[ \text{pupil} = \text{pinhole / aperture} \]

\[ \text{retina} = \text{film / digital sensor} \]

*Slide adopted from: Steve Seitz*
Two-types of **Light Sensitive Receptors**

**Rods**
- 75-150 million rod-shaped receptors
- not involved in color vision, gray-scale vision only
- operate at night
- highly sensitive, can responding to a single photon
- yield relatively poor spatial detail

**Cones**
- 6-7 million cone-shaped receptors
- color vision
- operate in high light
- less sensitive
- yield higher resolution

*Slide adopted from: James Hays*
Human Eye

Density of rods and cones

Slide adopted from: James Hays
Lecture 2: **Summary**

— We discussed a “physics-based” approach to image formation. Basic abstraction is the **pinhole camera**.

— **Lenses overcome limitations** of the pinhole model while trying to preserve it as a useful abstraction

— Projection equations: **perspective**, weak perspective, orthographic

— Thin lens equation

— Some “aberrations and **distortions**” persist (e.g. spherical aberration, vignetting)

— The **human eye** functions much like a camera
What is **Computer Vision**?

Compute vision, broadly speaking, is a research field aimed to enable computers to **process and interpret visual data**, as sighted humans can.

![Diagram](https://www.flickr.com/photos/flamephoenix1991/8376271918)

**Image (or video)**

**Sensing Device**

**Interpreting Device**

**Interpretation**

- blue sky,
- trees,
- fountains,
- UBC, ...

What is **Computer Vision**?

Compute vision, broadly speaking, is a research field aimed to enable computers to *process and interpret visual data*, as sighted humans can.

![Diagram of computer vision process](https://www.flickr.com/photos/flamephoenix1991/8376271918)

- **Image (or video)**
- **Sensing Device**
- **Interpreting Device**
- **Interpretation**
  - blue sky, trees, fountains, UBC, …
Image as a **2D Function**

A (grayscale) image is a 2D function

I(X,Y)

grayscale image
Image as a 2D Function

A (grayscale) image is a 2D function

\[ I(X, Y) \]

\textbf{domain:} \((X, Y) \in ([1, \text{width}], [1, \text{height}])\)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Image as a **2D Function**

A (grayscale) image is a 2D function

What is the **range** of the image function?

**Domain:** \( (X, Y) \in ([1, \text{width}], [1, \text{height}]) \)
Image as a **2D Function**

A (grayscale) image is a 2D function

What is the **range** of the image function?

\[ I(X, Y) \in [0, 255] \in \mathbb{Z} \]

**domain**: \((X, Y) \in ([1, \text{width}], [1, \text{height}])\)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Adding two Images

Since images are functions, we can perform operations on them, e.g., average

\[ I(X, Y) + G(X, Y) \]
Adding two Images

\[
a = \frac{I(X, Y)}{2} + \frac{G(X, Y)}{2}
\]

\[
b = \frac{I(X, Y) + G(X, Y)}{2}
\]
Adding two Images

$$a = \frac{I(X, Y)}{2} + \frac{G(X, Y)}{2}$$

$$b = \frac{I(X, Y) + G(X, Y)}{2}$$

Question:

- $a = b$
- $a > b$
- $a < b$
Adding two Images

Red pixel in camera man image = 98
Red pixel in moon image = 200

$$\frac{98}{2} + \frac{200}{2} = 49 + 100 = 149$$

$$\frac{98 + 200}{2} = \left\lfloor \frac{298}{2} \right\rfloor = \frac{255}{2} = 127$$

Question:

- $a = b$
- $a > b$
- $a < b$
Adding two Images

It is often convenient to convert images to doubles when doing processing.

In Python

```python
from PIL import Image
img = Image.open('cameraman.png')
import numpy as np
imgArr = np.asarray(img)

# Or do this
import matplotlib.pyplot as plt
camera = plt.imread('cameraman.png');```
What types of **transformations** can we do?

- **Filtering**
  - $I(X, Y)$
  - $I'(X, Y)$
  - Changes range of image function

- **Warping**
  - $I(X, Y)$
  - $I'(X, Y)$
  - Changes domain of image function
What types of **filtering** can we do?

**Point** Operation

Point processing

**Neighborhood** Operation

“filtering”
Examples of **Point Processing**

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

\[ I(X, Y) \]

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

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Examples of **Point Processing**

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

\[ I(X, Y) \]

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

\[ I(X, Y) - 128 \]

**Slide Credit**: Ioannis (Yannis) Gkioullekas (CMU)
Examples of **Point Processing**

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

\[ I(X, Y) \]

**invert**

\[ I(X, Y) - 128 \]

**lighten**

\[ \frac{I(X, Y)}{2} \]

**raise contrast**

**non-linear raise contrast**

---

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
Examples of **Point Processing**

- **Original**: $I(X, Y)$
- **Darken**: $I(X, Y) - 128$
- **Lower Contrast**: $\frac{I(X, Y)}{2}$
- **Non-linear Lower Contrast**: $\left(\frac{I(X, Y)}{255}\right)^{1/3} \times 255$
- **Invert**: $I(X, Y)$
- **Lighten**: $I(X, Y)$
- **Raise Contrast**: $\frac{I(X, Y)}{2}$
- **Non-linear Raise Contrast**: $\left(\frac{I(X, Y)}{255}\right)^{1/3} \times 255$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Examples of **Point Processing**

- **Original**
- **Darken**
  
- **Lower Contrast**
  
- **Non-linear Lower Contrast**

- **Invert**

- **Lighten**

- **Raise Contrast**

- **Non-linear Raise Contrast**

\[
I(X, Y) - 128
\]

\[
\frac{I(X, Y)}{2}
\]

\[
\left( \frac{I(X, Y)}{255} \right)^{1/3} \times 255
\]

\[
255 - I(X, Y)
\]

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
Examples of **Point Processing**

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

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

\[ I(X, Y) \]
\[ I(X, Y) - 128 \]
\[ \frac{I(X, Y)}{2} \]
\[ \left( \frac{I(X, Y)}{255} \right)^{1/3} \times 255 \]

- **255 - I(X, Y)**
- **I(X, Y) + 128**

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)

40
Examples of **Point Processing**

- **original**
- **darken**
  \[ I(X, Y) \]
  \[ I(X, Y) - 128 \]
- **lower contrast**
  \[ \frac{I(X, Y)}{2} \]
- **non-linear lower contrast**
  \[ \left( \frac{I(X, Y)}{255} \right)^{1/3} \times 255 \]

- **invert**
  \[ 255 - I(X, Y) \]
- **lighten**
  \[ I(X, Y) + 128 \]
- **raise contrast**
  \[ I(X, Y) \times 2 \]
- **non-linear raise contrast**

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Examples of Point Processing

- **original**
- **darken**
  \[ I(X, Y) - 128 \]
- **lower contrast**
  \[ \frac{I(X, Y)}{2} \]
- **non-linear lower contrast**
  \[ \left( \frac{I(X, Y)}{255} \right)^{1/3} \times 255 \]
- **invert**
  \[ 255 - I(X, Y) \]
- **lighten**
  \[ I(X, Y) + 128 \]
- **raise contrast**
  \[ I(X, Y) \times 2 \]
- **non-linear raise contrast**
  \[ \left( \frac{I(X, Y)}{255} \right)^2 \times 255 \]

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Examples of **Point Processing**

- **original**
- **darken**
  \[ I(X, Y) \]
  \[ I(X, Y) - 128 \]
  \[ \frac{I(X, Y)}{2} \]
  \[ \left( \frac{I(X, Y)}{255} \right)^{1/3} \times 255 \]

- **invert**
  \[ 255 - I(X, Y) \]

- **lighten**
  \[ I(X, Y) + 128 \]

- **lower contrast**
  \[ I(X, Y) \times 2 \]

- **non-linear lower contrast**
  \[ \left( \frac{I(X, Y)}{255} \right)^2 \times 255 \]

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
What types of **transformations** can we do?

- **Filtering**
  - Changes range of image function

- **Warping**
  - Changes domain of image function

---

**Slide Credit**: Ioannis (Yannis) Gkioulekas (CMU)
What types of **filtering** can we do?

**Point** Operation

Point processing

**Neighborhood** Operation

“filtering”
Linear Filters

Let $I(X, Y)$ be an $n \times n$ digital image (for convenience we let width = height).

Let $F(X, Y)$ be another $m \times m$ digital image (our “filter” or “kernel”).

For convenience we will assume $m$ is odd. (Here, $m = 5$)
**Linear Filters**

Let \( k = \left\lfloor \frac{m}{2} \right\rfloor \)

Compute a new image, \( I'(X, Y) \), as follows

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j)
\]

**Intuition:** each pixel in the output image is a linear combination of the same pixel and its neighboring pixels in the original image
Linear Filters

For a given $X$ and $Y$, superimpose the filter on the image centered at $(X, Y)$
Linear Filters

For a given $X$ and $Y$, superimpose the filter on the image centered at $(X, Y)$.

Compute the new pixel value, $I'(X, Y)$, as the sum of $m \times m$ values, where each value is the product of the original pixel value in $I(X, Y)$ and the corresponding values in the filter.
The computation is repeated for each 

\((X, Y)\)
Linear Filter Example

\[ I(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]

Image \( I(X, Y) \)

Output \( I'(X, Y) \)

Filter \( F(X, Y) \)

Output Image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[ I'(X, Y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} F(I, J) I(X + i, Y + j) \]
Linear Filter Example

The linear filter example involves a convolution operation between an input image $I(X, Y)$ and a filter $F(X, Y)$. The output image $I'(X, Y)$ is calculated as:

$$I'(X, Y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} F(I, J) I(X + i, Y + j)$$

where $F(X, Y)$ is a 3x3 filter with pixel values:

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

The image $I(X, Y)$ has pixel values:

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

The output image $I'(X, Y)$ is:

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[ I'(X, Y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} F(I, J) I(X + i, Y + j) \]
Linear Filter Example

$I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X+i, Y+j)$

Slide Credit: Ioannis (Yannis) Gkioulkekas (CMU)
Linear Filter Example

The linear filter example is shown in the diagram. The filter $F(X, Y)$ is applied to the image $I(X, Y)$ to produce the output image $I'(X, Y)$. The filter is defined as a $5 \times 5$ matrix, and the convolution is calculated as:

$$I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j)$$

The output image $I'(X, Y)$ is shown on the right, with the filter applied to a specific region highlighted. The slide credit is given to Ioannis (Yannis) Gkioulekas (CMU).
Linear Filter Example

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]
Linear Filter Example

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[ I' (X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I,J) I(X+i,Y+j) \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter **Example**

$$I(X,Y)$$

$$F(X,Y)$$

$$1 \quad 1 \quad 1 \quad 1 \quad 1$$

$$\frac{1}{9}$$

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I,J) I(X+i,Y+j)$$

**Slide Credit**: Ioannis (Yannis) Gkioulakes (CMU)
Linear Filter Example

\[ F(X, Y) \]

\[ I(X, Y) \]

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]
Linear Filter Example

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[
F(X, Y) = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
I(X, Y) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 90 & 90 & 90 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 90 & 90 \\
\end{bmatrix}
\]

\[
I'(X, Y) = \frac{1}{9} \left( \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) \cdot I(X+i, Y+j) \right)
\]

Output

\[
I'(X, Y) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
**Linear Filter Example**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]

*Output* \( I'(X, Y) \)

*Filter* \( F(X, Y) \)

*Input* \( I(X, Y) \)

*Image* (signal)
Linear Filter Example

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X+i, Y+j)
\]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[ I'(X, Y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} F(I, J) I(X + i, Y + j) \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
**Linear Filter Example**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]

*image (signal)*

*filter*

*output*
Linear Filter \textbf{Example}

\[
F(X, Y)
\]

\[
\frac{1}{9}
\]

\[
I(X, Y)
\]

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j)
\]

\[
\begin{array}{ccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 30 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{ccccccccccc}
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 & 0 \\
0 & 20 & 40 & 60 & 60 & 60 & 40 & 20 & 0 \\
0 & 30 & 50 & 80 & 80 & 90 & 60 & 30 & 0 \\
0 & 30 & 50 & 80 & 80 & 90 & 60 & 30 & 0 \\
0 & 20 & 30 & 50 & 50 & 60 & 40 & 20 & 0 \\
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 & 0 \\
10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 & 0 \\
10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\textbf{Slide Credit:} Ioannis (Yannis) Gkioulekas (CMU)
Linear Filters

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j)
\]

For a given \(X\) and \(Y\), superimpose the filter on the image centered at \((X, Y)\)

Compute the new pixel value, \(I'(X, Y)\), as the sum of \(m \times m\) values, where each value is the product of the original pixel value in \(I(X, Y)\) and the corresponding values in the filter.
Linear Filters

Let’s do some accounting …

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j)
\]
Linear Filters

Let’s do some accounting …

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j) \]

At each pixel, \((X, Y)\), there are \(m \times m\) multiplications.
Linear Filters

Let’s do some accounting …

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j)
\]

At each pixel, \((X, Y)\), there are \(m \times m\) multiplications.

There are \(n \times n\) pixels in \((X, Y)\).
Linear Filters

Let’s do some accounting …

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X+i, Y+j) \]

At each pixel, \((X, Y)\), there are \(m \times m\) multiplications.

There are \(n \times n\) pixels in \((X, Y)\).

**Total:** \(m^2 \times n^2\) multiplications
Linear Filters

Let’s do some accounting …

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j)
\]

At each pixel, \((X, Y)\), there are \(m \times m\) multiplications.

There are \(n \times n\) pixels in \((X, Y)\).

**Total:** \(m^2 \times n^2\) multiplications

When \(m\) is fixed, small constant, this is \(O(n^2)\). But when \(m \approx n\) this is \(O(m^4)\).
Linear Filters: **Boundary Effects**
Three standard ways to deal with boundaries:

1. **Ignore these locations:** Make the computation undefined for the top and bottom $k$ rows and the leftmost and rightmost $k$ columns
Linear Filters: **Boundary Effects**

Three standard ways to deal with boundaries:

1. **Ignore these locations:** Make the computation undefined for the top and bottom $k$ rows and the leftmost and rightmost $k$ columns.

2. **Pad the image with zeros:** Return zero whenever a value of $I$ is required at some position outside the defined limits of $X$ and $Y$. 
Linear Filters: **Boundary Effects**
Linear Filters: **Boundary Effects**

Three standard ways to deal with boundaries:

1. **Ignore these locations:** Make the computation undefined for the top and bottom \( k \) rows and the leftmost and rightmost \( k \) columns.

2. **Pad the image with zeros:** Return zero whenever a value of \( I \) is required at some position outside the defined limits of \( X \) and \( Y \).

3. **Assume periodicity:** The top row wraps around to the bottom row; the leftmost column wraps around to the rightmost column.
Linear Filters: **Boundary** Effects
Linear Filters: **Boundary Effects**
A short exercise …
Example 1: Warm up

Original

Filter

Result

0 0 0
0 1 0
0 0 0
Example 1: Warm up

Original

Filter

Result
(no change)
Example 2:

Original

Filter

Result
Example 2:

Original

Filter

Result
(sift left by 1 pixel)
Example 3:

Original

Filter

Result

(filter sums to 1)
Example 3:

Original

Filter
(filter sums to 1)

Result
(blur with a box filter)
Example 4:

Original

Filter

Result

(-filter sums to 1)
Example 4:

<table>
<thead>
<tr>
<th>Original</th>
<th>Filter</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Original Image" /></td>
<td><img src="image2.png" alt="Filter Image" /></td>
<td><img src="image3.png" alt="Result Image" /></td>
</tr>
</tbody>
</table>

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

- Filter sums to 1

Original

Filter

Result

(sharpening)
Example 4: Sharpening

Before

After
Example 4: Sharpening

Before

After

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
**Linear Filters**: Correlation vs. Convolution

**Definition: Correlation**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j) \]
Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j)
\]

Definition: **Convolution**

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X - i, Y - j)
\]
Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j) \]
Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j) \]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>g</td>
<td>h</td>
<td>i</td>
</tr>
</tbody>
</table>

Filter

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Image

Output

\[ = 1a + 2b + 3c + 4d + 5e + 6f + 7g + 8h + 9i \]
Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j)I(X + i, Y + j) \]

Definition: **Convolution**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j)I(X - i, Y - j) \]
Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j)
\]

Definition: **Convolution**

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X - i, Y - j)
\]

\[
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
  g & h & i \\
\end{array}
\]

\[
\begin{array}{ccc}
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{array}
\]

= 9a + 8b + 7c
+ 6d + 5e + 4f
+ 3g + 2h + 1i
**Linear Filters:** Correlation vs. Convolution

**Definition: Correlation**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j) \]

**Definition: Convolution**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X - i, Y - j) \]

![Filter](rotated by 180)

\[
\begin{array}{ccc}
\text{Filter} & \text{Image} & \text{Output} \\
\begin{array}{ccc}
a & b & c \\
d & e & f \\
g & h & i \\
\end{array} & \begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array} & = 9a + 8b + 7c + 6d + 5e + 4f + 3g + 2h + 1i
\]

93
Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j)I(X + i, Y + j) \]

Definition: **Convolution**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j)I(X - i, Y - j) \]

\[ = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(-i, -j)I(X + i, Y + j) \]

**Note:** if \( F(X, Y) = F(-X, -Y) \) then correlation = convolution.
Preview: Why convolutions are important?

Who has heard of **Convolutional Neural Networks** (CNNs)?
Preview: Why convolutions are important?

Who has heard of Convolutional Neural Networks (CNNs)?

What about Deep Learning?
Why convolutions are important?

Who has heard of **Convolutional Neural Networks** (CNNs)?

What about **Deep Learning**?

Basic operations in CNNs are convolutions (with learned linear filters) followed by non-linear functions.

**Note**: This results in non-linear filters.
Linear Filters: **Properties**

Let $\otimes$ denote convolution. Let $I(X, Y)$ be a digital image.

**Superposition:** Let $F_1$ and $F_2$ be digital filters

$$
(F_1 + F_2) \otimes I(X, Y) = F_1 \otimes I(X, Y) + F_2 \otimes I(X, Y)
$$
Linear Filters: **Properties**

Let $\otimes$ denote convolution. Let $I(X, Y)$ be a digital image.

**Superposition:** Let $F_1$ and $F_2$ be digital filters

$$(F_1 + F_2) \otimes I(X, Y) = F_1 \otimes I(X, Y) + F_2 \otimes I(X, Y)$$

**Scaling:** Let $F$ be digital filter and let $k$ be a scalar

$$(kF) \otimes I(X, Y) = F \otimes (kI(X, Y)) = k(F \otimes I(X, Y))$$
Linear Filters: Properties

Let $\otimes$ denote convolution. Let $I(X, Y)$ be a digital image

**Superposition:** Let $F_1$ and $F_2$ be digital filters

$$(F_1 + F_2) \otimes I(X, Y) = F_1 \otimes I(X, Y) + F_2 \otimes I(X, Y)$$

**Scaling:** Let $F$ be digital filter and let $k$ be a scalar

$$(kF) \otimes I(X, Y) = F \otimes (kI(X, Y)) = k(F \otimes I(X, Y))$$

**Shift Invariance:** Output is local (i.e., no dependence on absolute position)
Linear Filters: Shift Invariance

Output does not depend on absolute position
Linear Filters: **Properties**

Let $\otimes$ denote convolution. Let $I(X,Y)$ be a digital image

**Superposition:** Let $F_1$ and $F_2$ be digital filters

$$(F_1 + F_2) \otimes I(X,Y) = F_1 \otimes I(X,Y) + F_2 \otimes I(X,Y)$$

**Scaling:** Let $F$ be digital filter and let $k$ be a scalar

$$(kF) \otimes I(X,Y) = F \otimes (kI(X,Y)) = k(F \otimes I(X,Y))$$

**Shift Invariance:** Output is local (i.e., no dependence on absolute position)

An operation is **linear** if it satisfies both **superposition** and **scaling**
**Linear Systems**: Characterization Theorem

*Any* linear, shift invariant operation can be expressed as convolution.
Example 5: Smoothing with a Box Filter

Filter has equal positive values that sum up to 1

Replaces each pixel with the average of itself and its local neighborhood

— Box filter is also referred to as average filter or mean filter

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)
Example 5: Smoothing with a Box Filter

Forsyth & Ponce (2nd ed.) Figure 4.1 (left and middle)
Example 5: Smoothing with a Box Filter

What happens if we increase the width (size) of the box filter?
Example 5: Smoothing with a Box Filter

Gonzales & Woods (3rd ed.) Figure 3.3
Menu for Today (January 14, 2020)

Topics: Image Filtering (also topic for next week)

- Image as a function
- Linear filters
- Correlation / Convolution
- Filter examples: Box, Gaussian

Readings:

- Today’s Lecture: Forsyth & Ponce (2nd ed.) 4.1, 4.5
- Next Lecture: none

Reminders:

- Assignment 0 (ungraded) due today, January 14
- Assignment 1: Image Filtering and Hybrid Images (is out January 14)