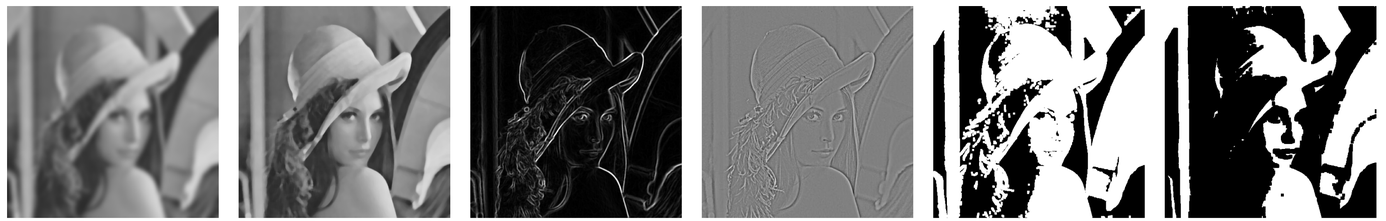
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 3: Image Filtering

(unless otherwise stated slides are taken or adopted from **Bob Woodham, Jim Little** and **Fred Tung**)

Menu for Today

Topics: Image Filtering

- **Image** as a **function**
- **Linear** filters
- **Correlation** / **Convolution**

Readings:

- **Today's** Lecture: Szeliski 3.1-3.3, Forsyth & Ponce (2nd ed.) 4.1, 4.5

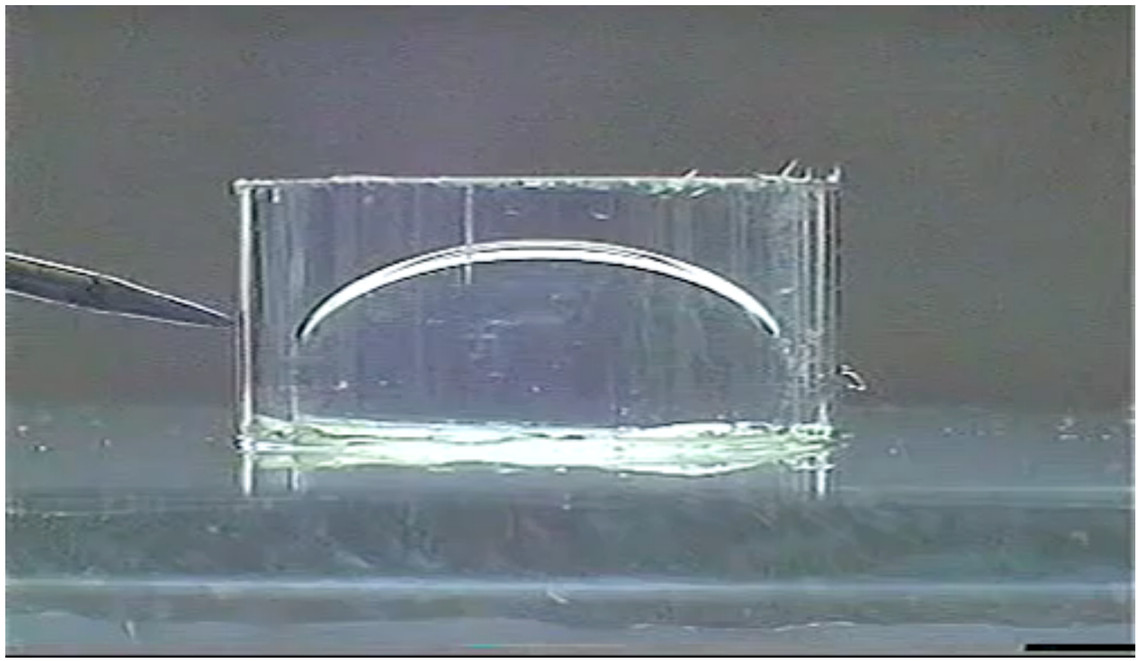
Reminders:

- Complete **Assignment 0** (optional, ungraded) due **Thursday**

Today's "**fun**" Example:

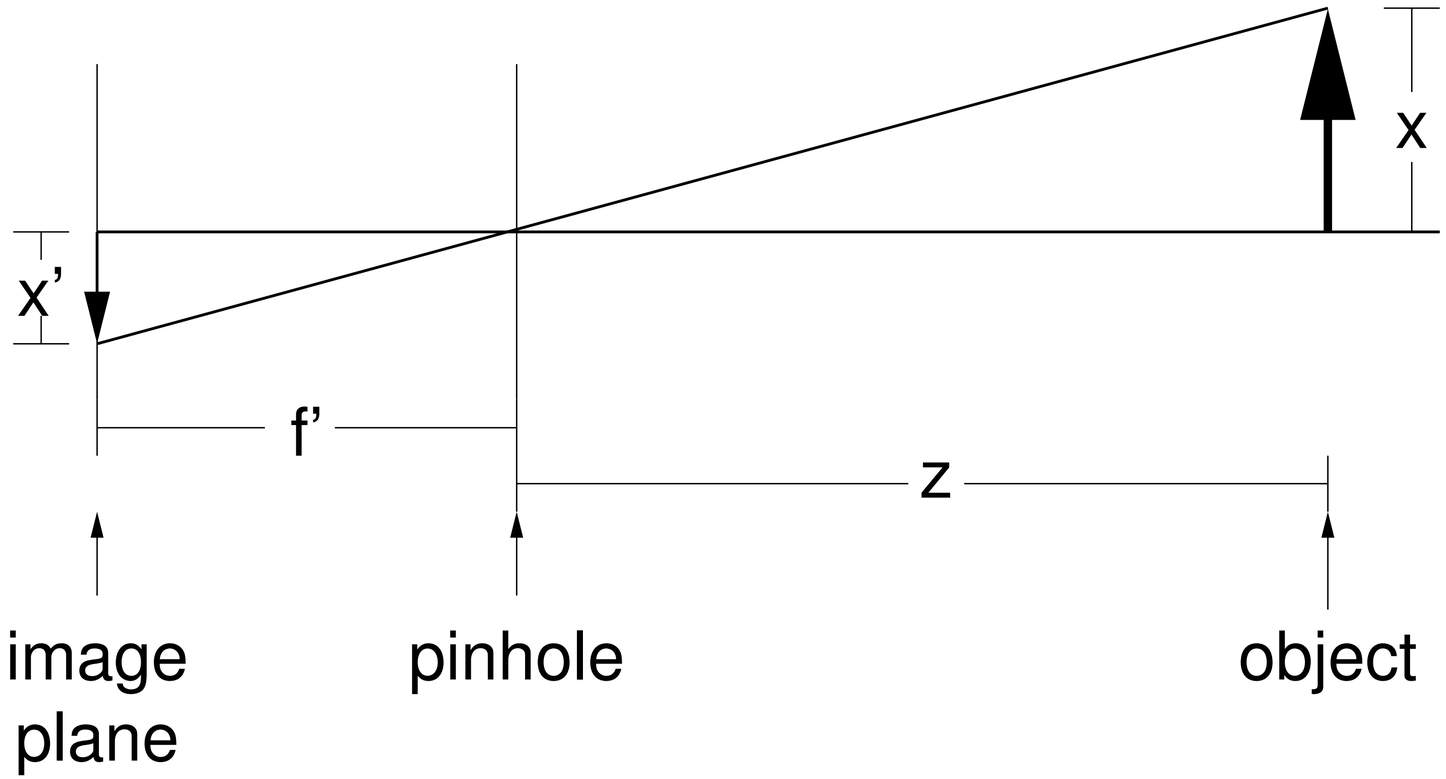
Developed by the French company **Varioptic**, the lenses consist of an oil-based and a water-based fluid sandwiched between glass discs. Electric charge causes the boundary between oil and water to change shape, altering the lens geometry and therefore the lens focal length

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



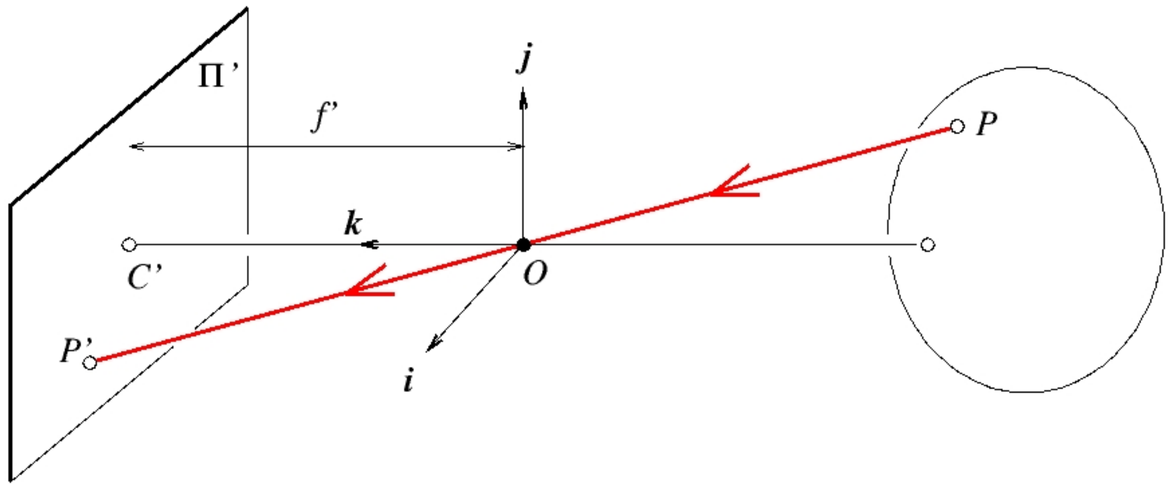
Video Source: https://www.youtube.com/watch?v=2c6lCdDFOY8

Recap: **Pinhole** Camera



$$x' = f' \frac{x}{z}$$

Perspective Projection



Forsyth & Ponce (1st ed.) Figure 1.4

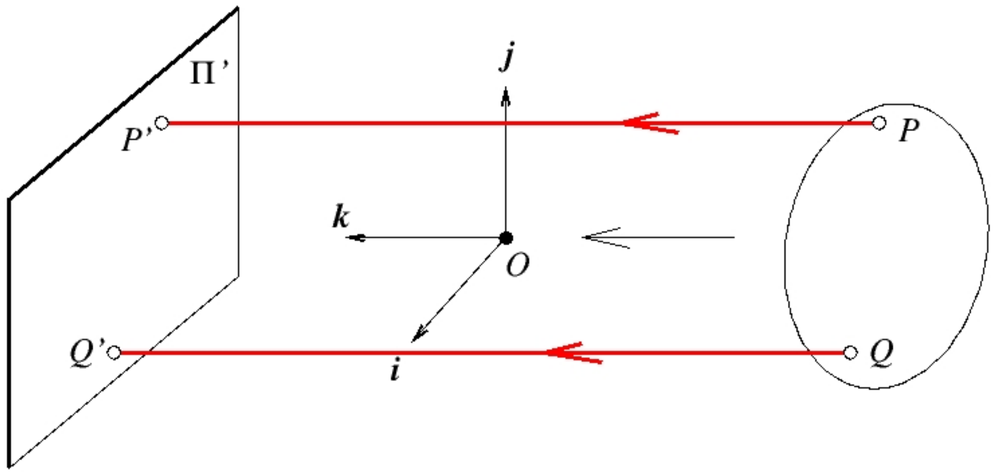
3D object point

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

$$x' = f' \frac{x}{z}$$

$$y' = f' \frac{y}{z}$$

Orthographic Projection



$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \end{bmatrix}$$

Forsyth & Ponce (1st ed.) Figure 1.6

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

where $\begin{aligned} x' &= x \\ y' &= y \end{aligned}$

Weak Perspective

3.1

3D object point $P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ in Π_0 projects to 2D image point $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

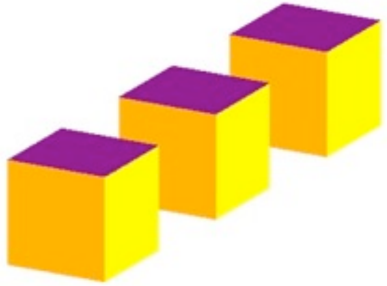
where $\boxed{\begin{aligned} x' &= mx \\ y' &= my \end{aligned}}$ and $m = \frac{f'}{z_0}$

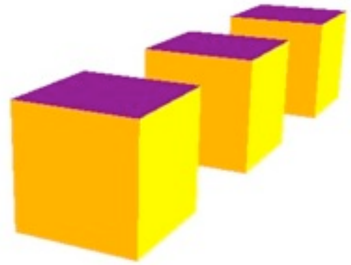
Summary of **Projection Equations**

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	$x' = x$ $y' = y$

Orthographic/Weak vs Perspective Projection





Parallel lines are preserved if depth variation in scene << depth of scene

→ orthographic projection model is appropriate

Parallel Lines

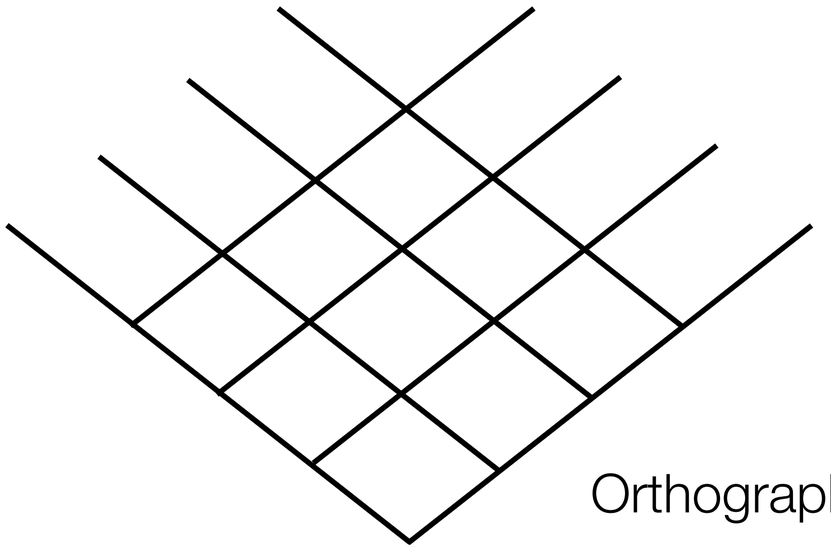
In a pinhole/perspective projection, parallel lines meet at a point

— the point is called **vanishing point**

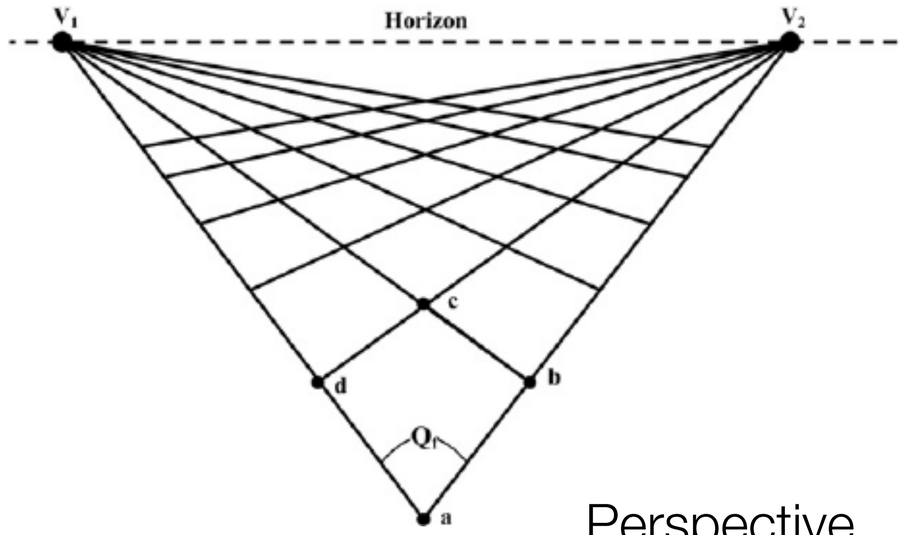
Sets of parallel lines one the same plane lead to **collinear** vanishing points

— the line is called a **horizon line** for that plane

In orthographic or weak projection, parallel lines stay parallel, e.g. a chessboard:

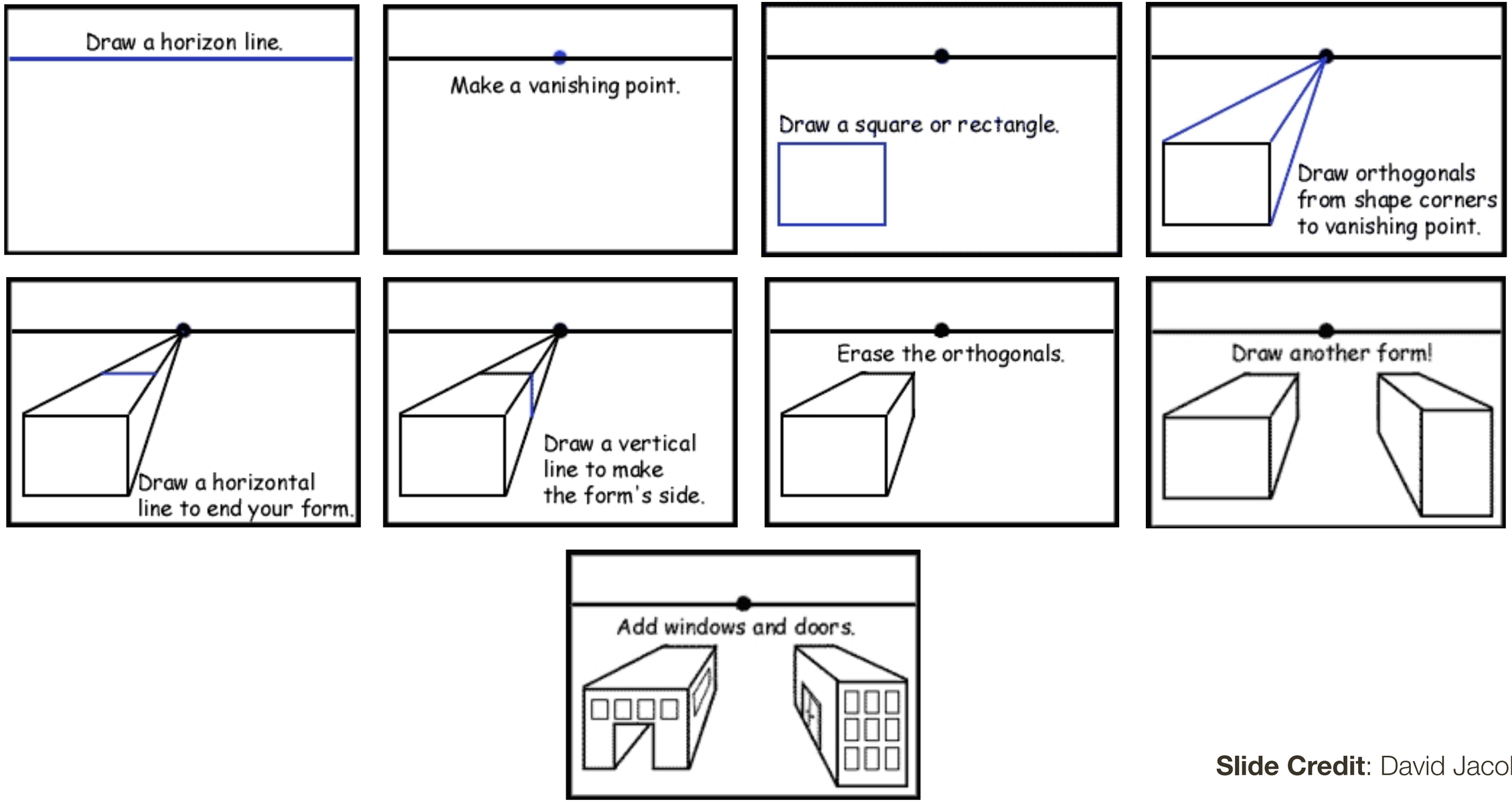


Orthographic



Perspective

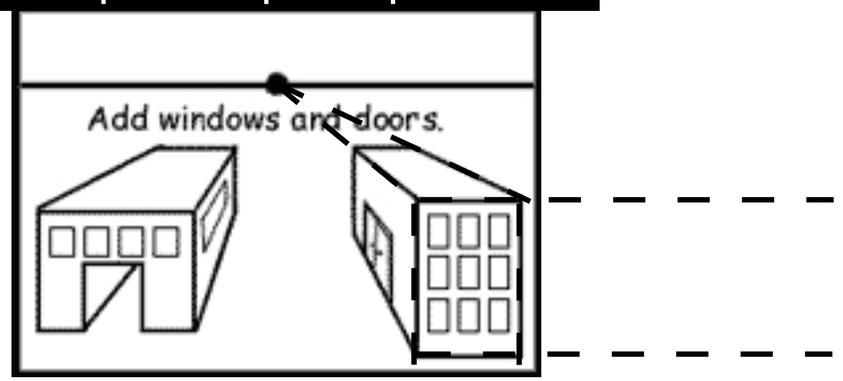
Vanishing Points



Slide Credit: David Jacobs

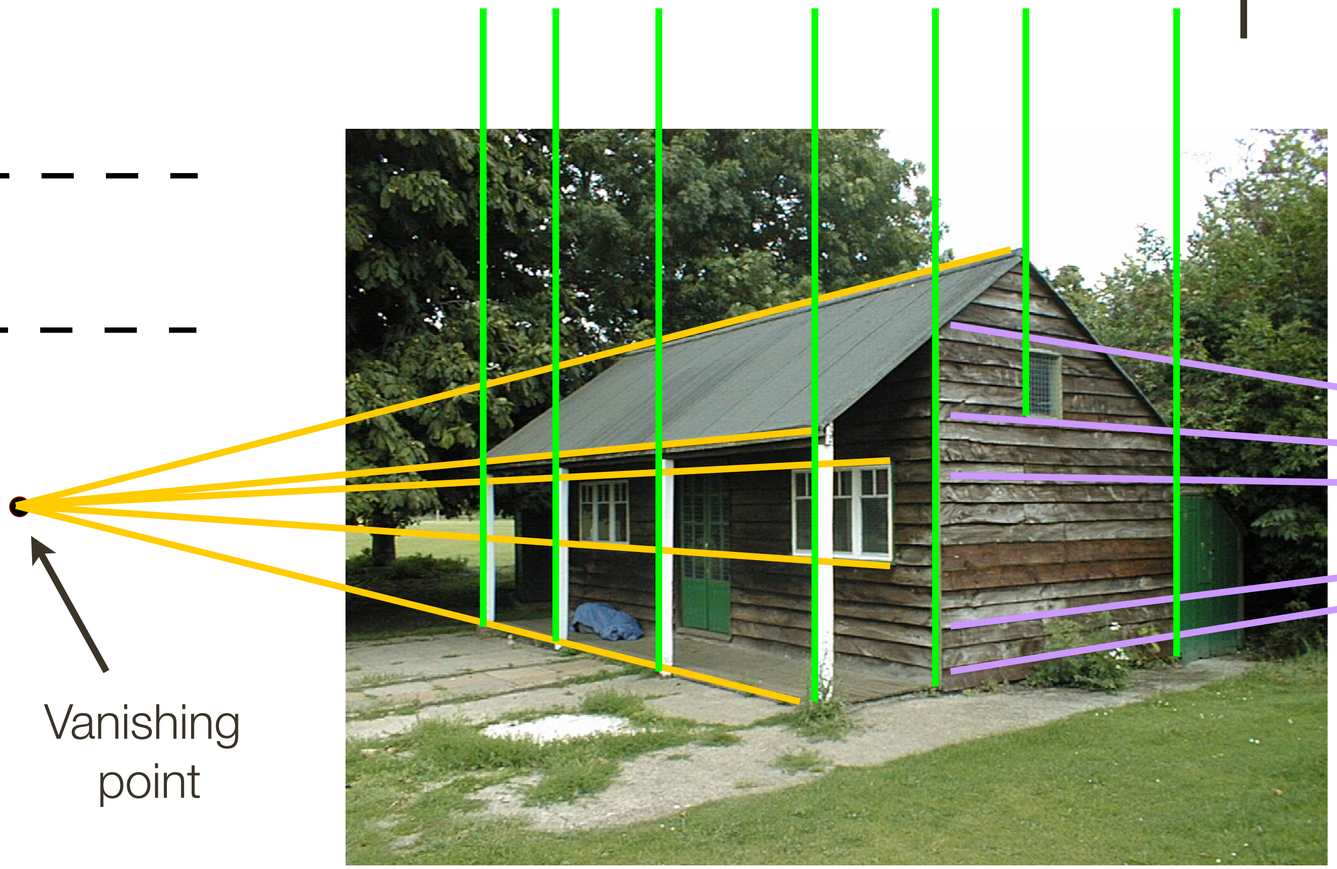
Vanishing Points

One point perspective



Built environment typically has 3 parallel line directions → 3 vanishing points

Two/three point perspective



Vertical vanishing point (at infinity)

Vanishing point

Vanishing point

Slide Credit: Efros (Berkeley), photo from Criminisi

Perspective Aside

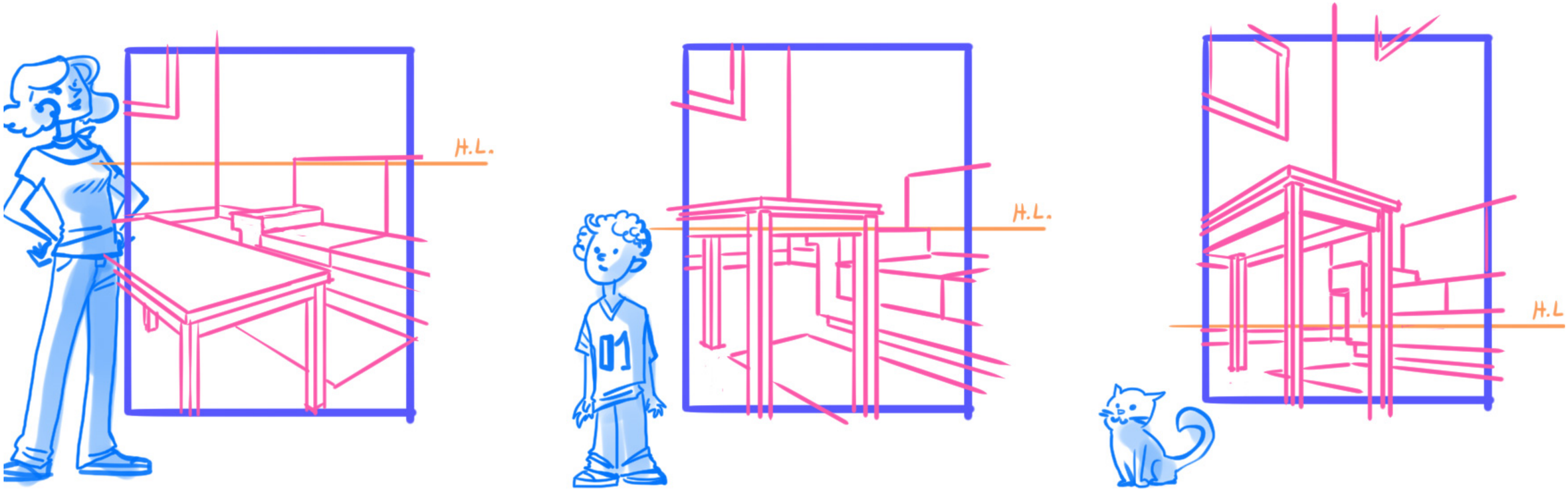
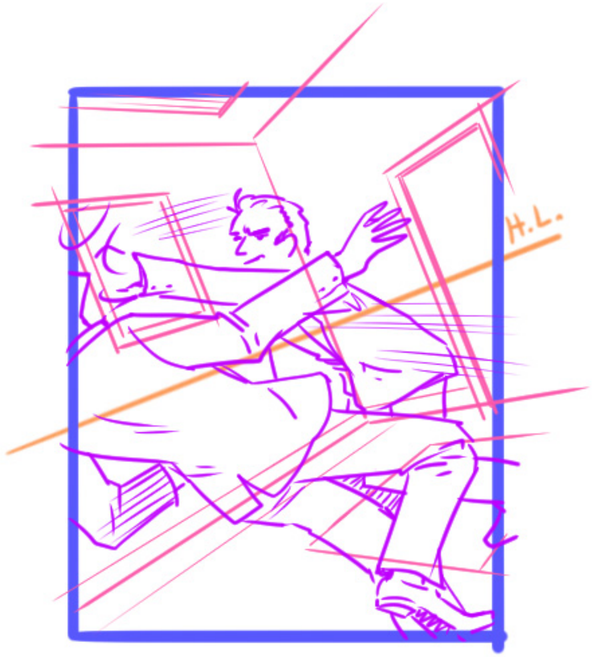


Image credit: http://www.martinacecilia.com/place-vanishing-points/

Perspective Aside



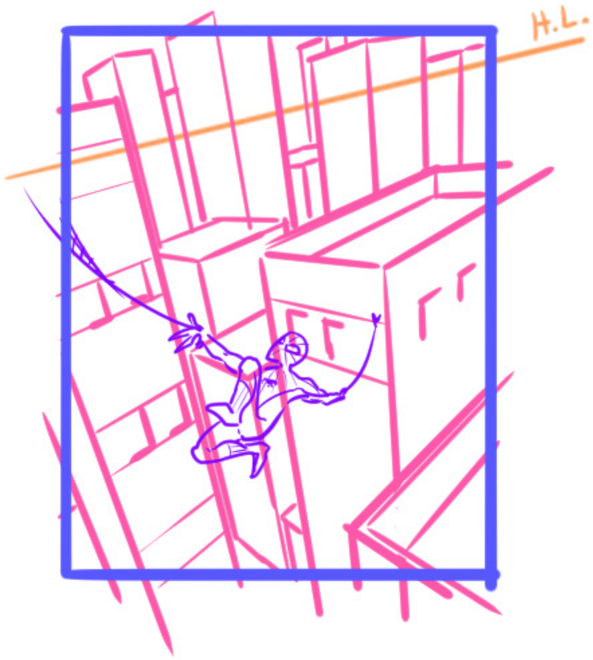


Image credit: http://www.martinacecilia.com/place-vanishing-points/

Perspective Projection: Matrix Form

Camera Matrix

$$\mathbf{C} = \begin{bmatrix} f' & 0 & 0 & 0 \\ 0 & f' & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

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 $s\,P' = \mathbf{C}P$

Perspective Projection: Matrix Form

Camera Matrix

$$\mathbf{C} = \begin{bmatrix} f'_x & 0 & 0 & 0 \\ 0 & f'_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

~~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 $s\,P' = \mathbf{C}P$

Perspective Projection: Matrix Form

Camera Matrix

$$\mathbf{C} = \begin{bmatrix} f'_x & 0 & 0 & c_x \\ 0 & f'_y & 0 & c_y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

~~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 $s\,P' = \mathbf{C}P$

Perspective Projection: Matrix Form

Camera Matrix

$$\mathbf{C} = \begin{bmatrix} f'_x & 0 & 0 & c_x \\ 0 & f'_y & 0 & c_y \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbb{R}_{4\times 4}$$

~~Pixels are squared / lens is perfectly symmetric~~

~~Sensor and pinhole perfectly aligned~~

~~Coordinate system centered at the pinhole~~

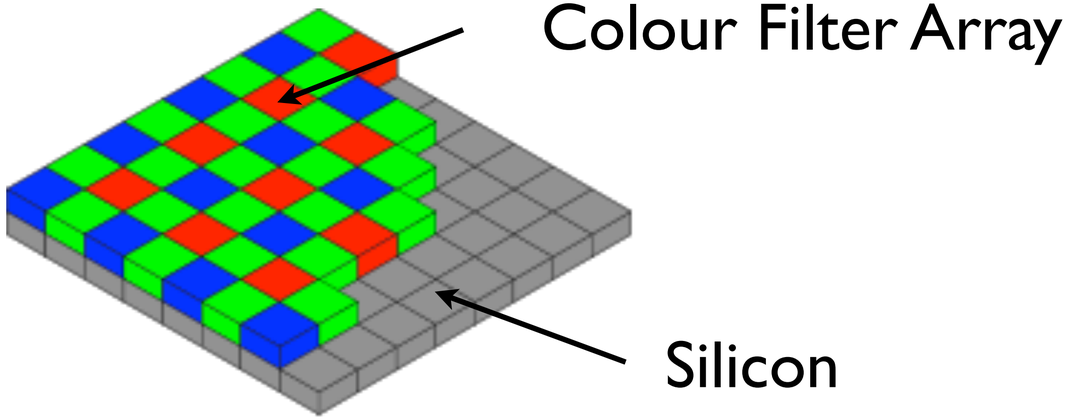
$P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ projects to 2D image point $P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$ where $s\,P' = \mathbf{C}P$

Projection Illusion

Next: Image **Filtering**

- Image as a function
- Image filtering, linear and nonlinear
- Convolution and correlation

Digital Sensor



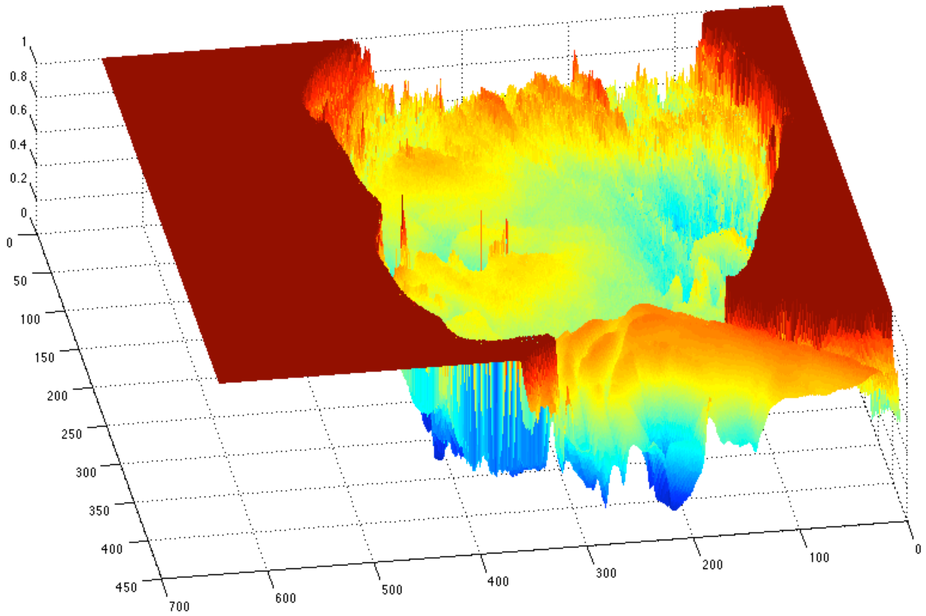
- Analogue image is sampled by a CMOS (or CCD) sensor
- RGB colour filters arranged in a “Bayer” pattern
- Counts from this sensor are camera RAW
- For viewing we need an RGB value per pixel

Image as a **2D Function**

A (grayscale) image is a 2D function

grayscale image

$I(X, Y)$



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

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

domain: $(X, Y) \in ([1, width], [1, hight])$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Adding two Images

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

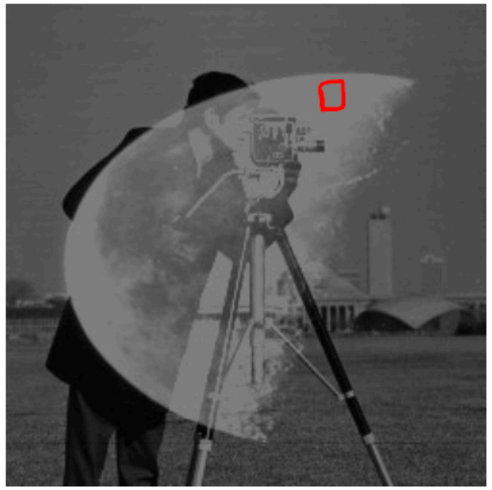
$I(X,Y)$

$G(X,Y)$

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

Adding two Images

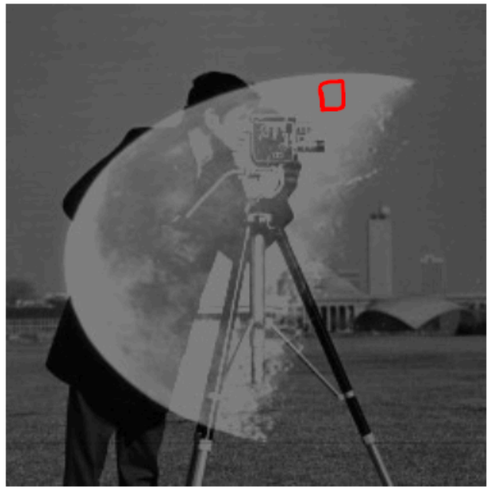
$$a = \frac{I(X,Y)}{2} + \frac{G(X,Y)}{2}$$



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

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

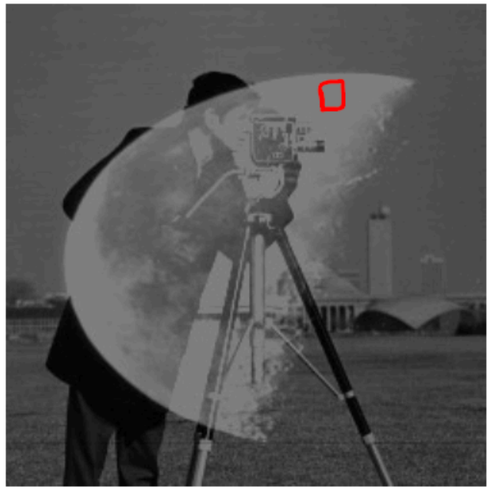
$$\frac{98}{2} + \frac{200}{2} = 49 + 100 = 149$$

Question:

$a = b$

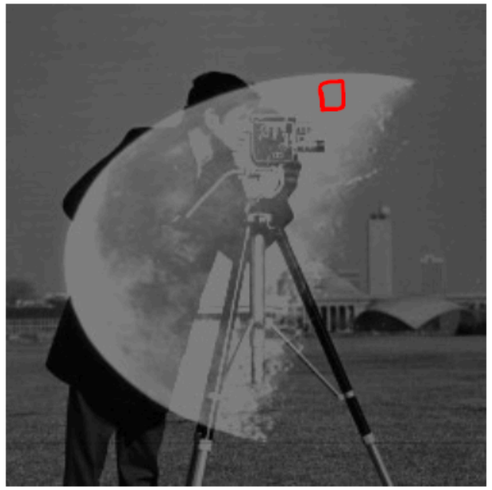
$a > b$

$a < b$



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

Adding two Images



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

In Python

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

# Or do this
import matplotlib.pyplot as plt
camera = plt.imread('cameraman.png');
```

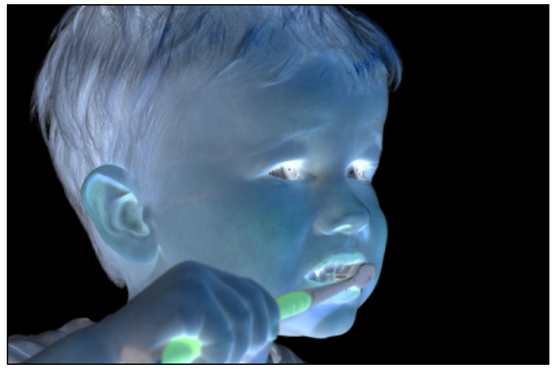
or “imgArr=np.array(img).astype(np.float32)/255.0”

What types of **transformations** can we do?

$I(X,Y)$

Filtering

$I'(X,Y)$

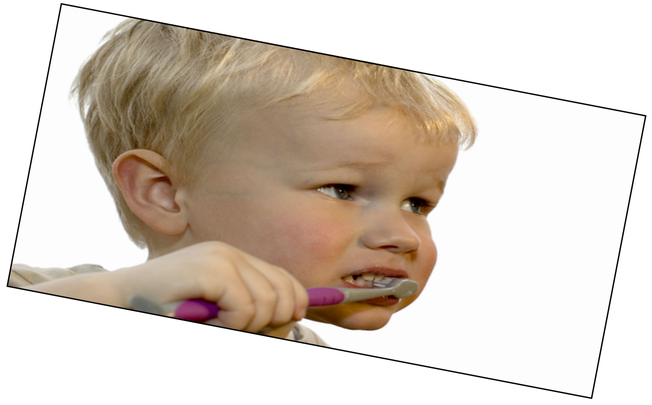


changes range of image function

$I(X,Y)$

Warping

$I'(X,Y)$

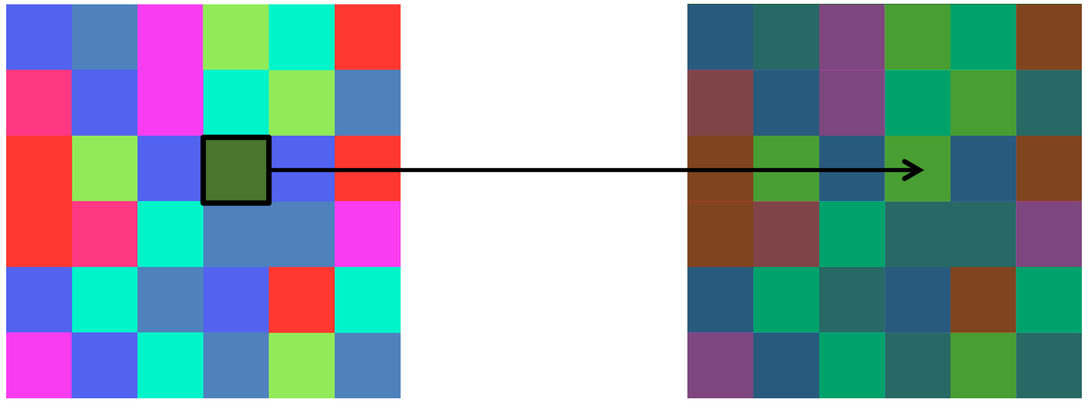


changes domain of image function

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

What types of **filtering** can we do?

Point Operation



point processing

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

$255 - I(X,Y)$

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

$I(X,Y)$

darken

$I(X,Y) - 128$

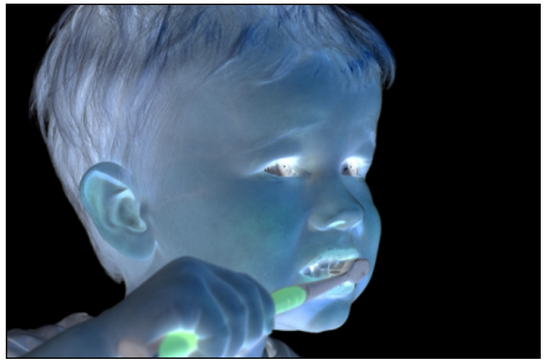
lower contrast

$\frac{I(X,Y)}{2}$

non-linear lower contrast

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

invert



$255 - I(X,Y)$

lighten

$I(X,Y) + 128$

raise contrast

$I(X,Y) \times 2$

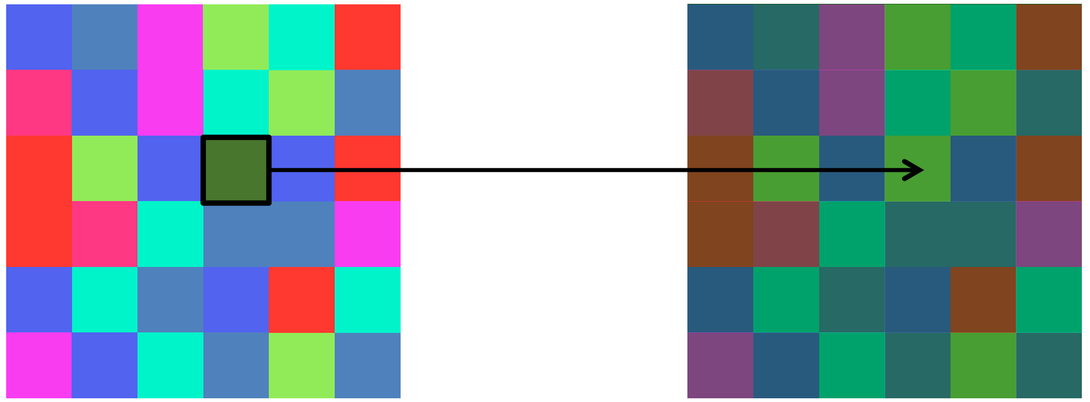
non-linear raise contrast

$\left(\frac{I(X,Y)}{255}\right)^{2} \times 255$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

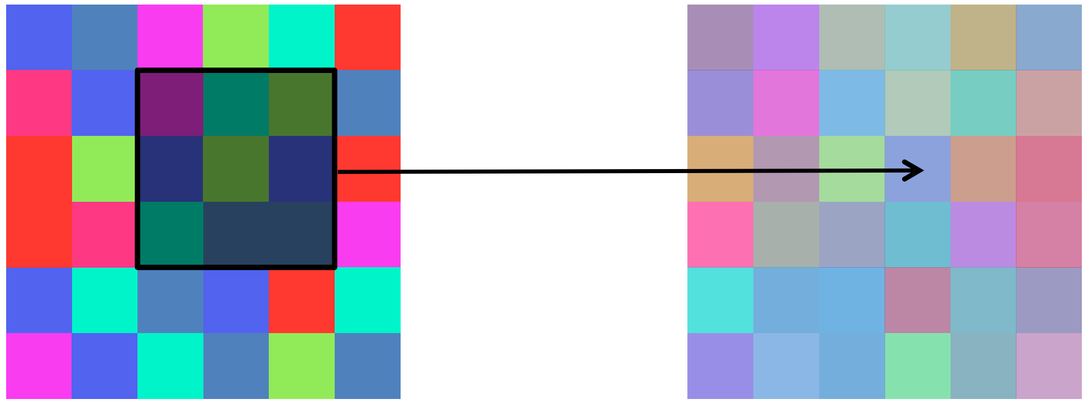
What types of **filtering** can we do?

Point Operation



point processing

Neighborhood Operation



“filtering”

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Operators

- How are photo filters implemented?

original image

blur

sharpen

edge filter

Non-Linear Operators

- How are photo filters implemented?

original image

edge preserve smooth

median

canny edges

Linear Filtering: Correlation

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

element wise
(dot) product

65	98	123
65	96	115
63	91	107

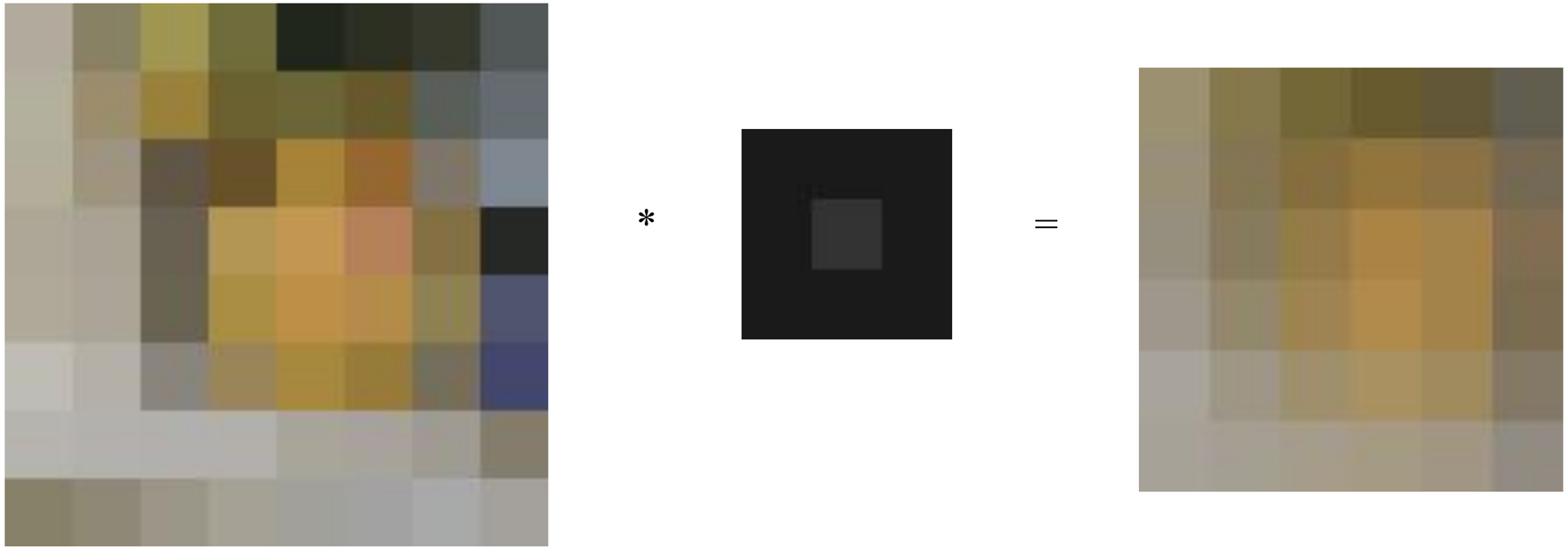
.*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$$= \begin{aligned} &0.1 * 65 + 0.1 * 98 + 0.1 * 123 + \\ &0.1 * 65 + 0.2 * 96 + 0.1 * 115 + \\ &0.1 * 63 + 0.1 * 91 + 0.1 * 107 \end{aligned}$$

$$= 92$$

Correlation Example



- With colour images, perform the dot products over each band

Correlation

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$I(x, y)$

$*$

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$f(x, y)$

$=$

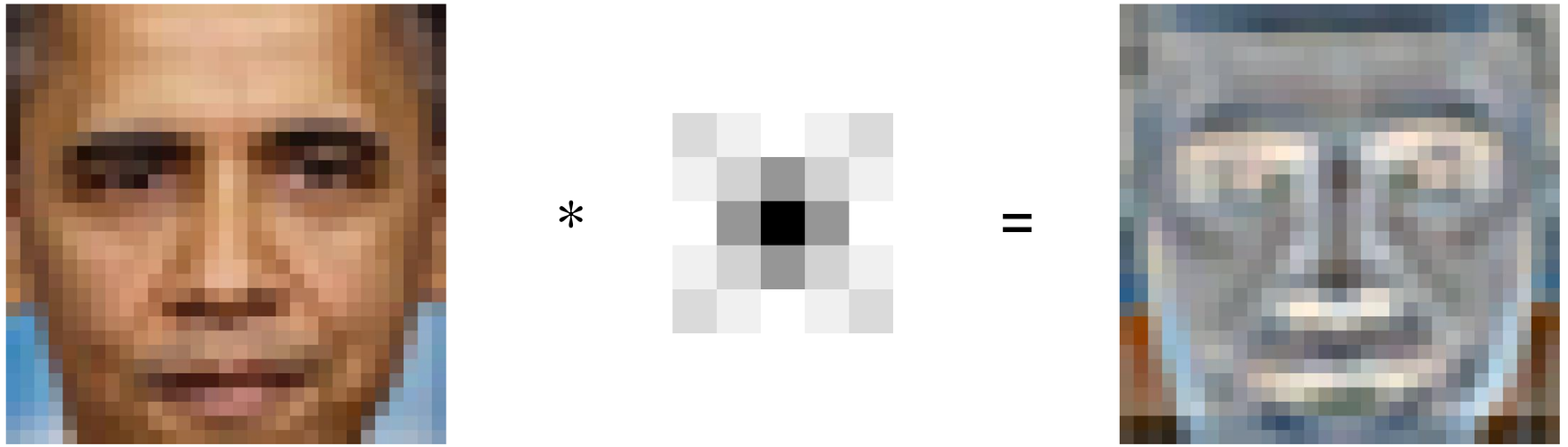
69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$I_{cr}(x, y)$

3.2

Correlation Example

- Centre-surround filter

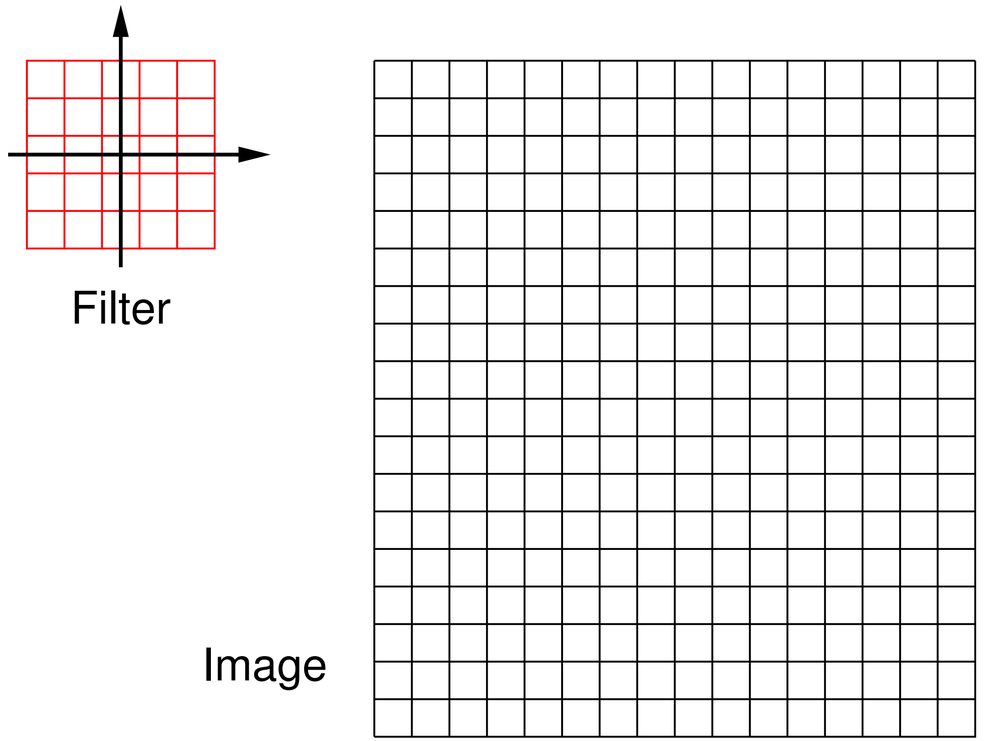


59	81	82	104	139	…	*	0	2	3	2	0	=	−2	−3	−2	−3	−5	…
52	77	93	112	133			2	0	−4	0	2		−1	−3	−2	−3	−4	
69	96	100	110	124			3	−4	−14	−4	3		−1	−2	0	1	1	
89	115	100	118	124			2	0	−4	0	2		−3	−4	0	1	1	
96	118	118	132	141			0	2	3	2	0		−3	−4	0	0	0	
75	105	112	136	154									−1	−2	0	−1	−1	
63	99	130	147	145									1	−1	−1	−1	0	
59	114	140	151	142									1	−3	−3	−1	0	
58	132	145	149	142									1	−4	−3	−1	−1	
58	131	146	140	131									1	−4	−4	−2	0	
		…													…			

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**")

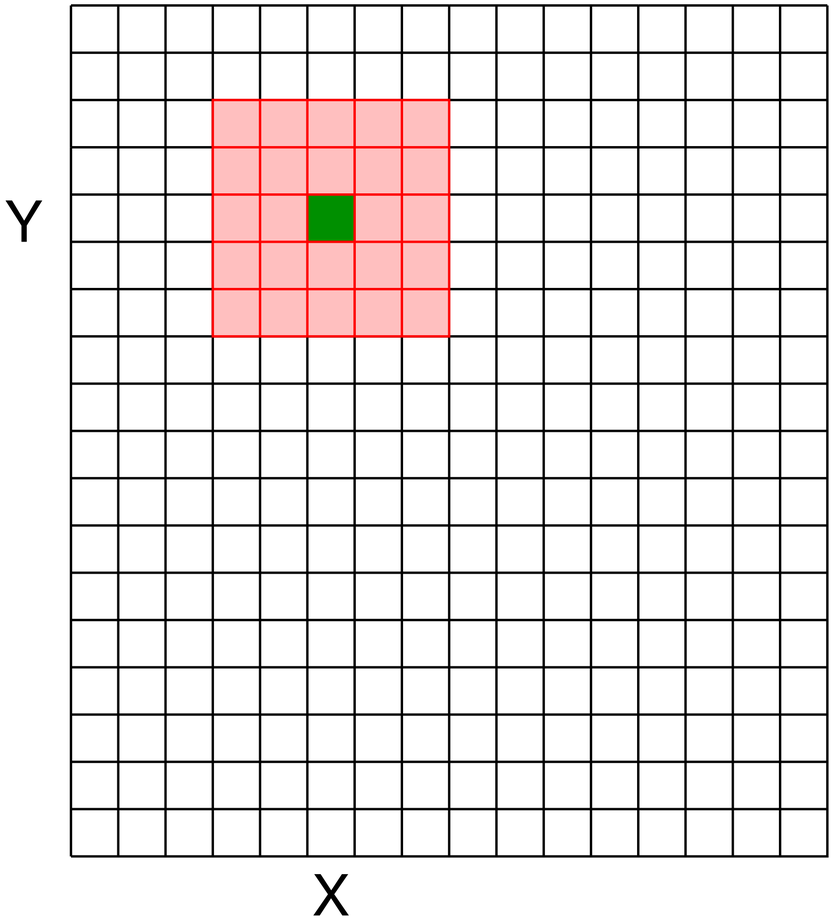


For convenience we will assume m is odd. (Here, $m = 5$)

Linear **Filters**

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

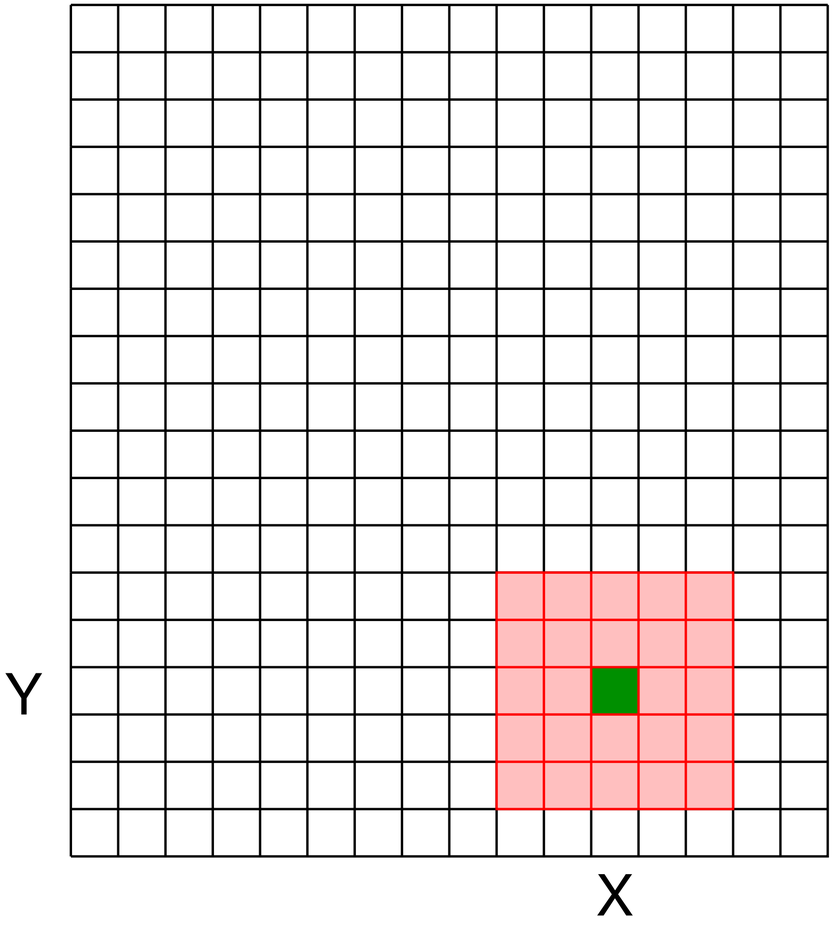
Compute the new pixel value, $I'(X, Y)$, as the sum of $m \times m$ values, where each value is the product of the original pixel value in $I(X, Y)$ and the corresponding values in the filter



Linear **Filters**

The computation is repeated for each

$$(X, Y)$$



Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

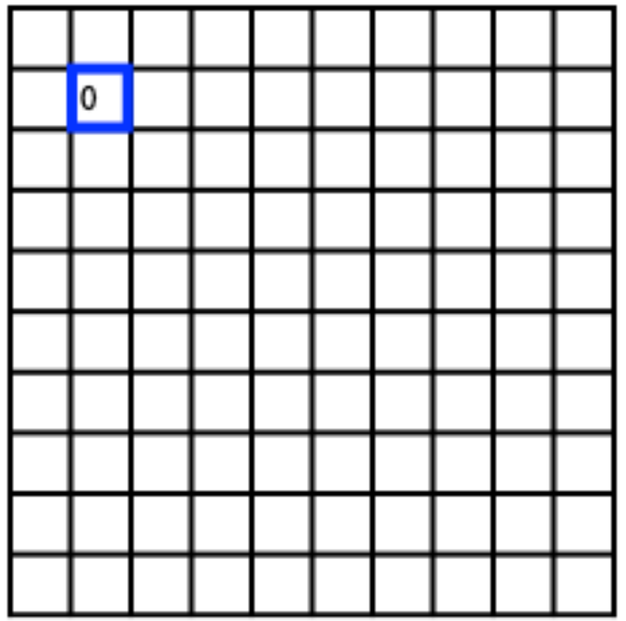
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$



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

output | filter | image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0								

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10							

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10							

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20						

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20						

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30					

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30				

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30			

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

output | filter | image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20		

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0								

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20							

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40						

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0								

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30							

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10								

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

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

$I'(X,Y)$: output

$F(i,j)$: filter

$I(X+i, Y+j)$: image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	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	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	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	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

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

output | filter | image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear **Filters**

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

output | filter | image (signal)

For a give 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 …

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

output | filter | image (signal)

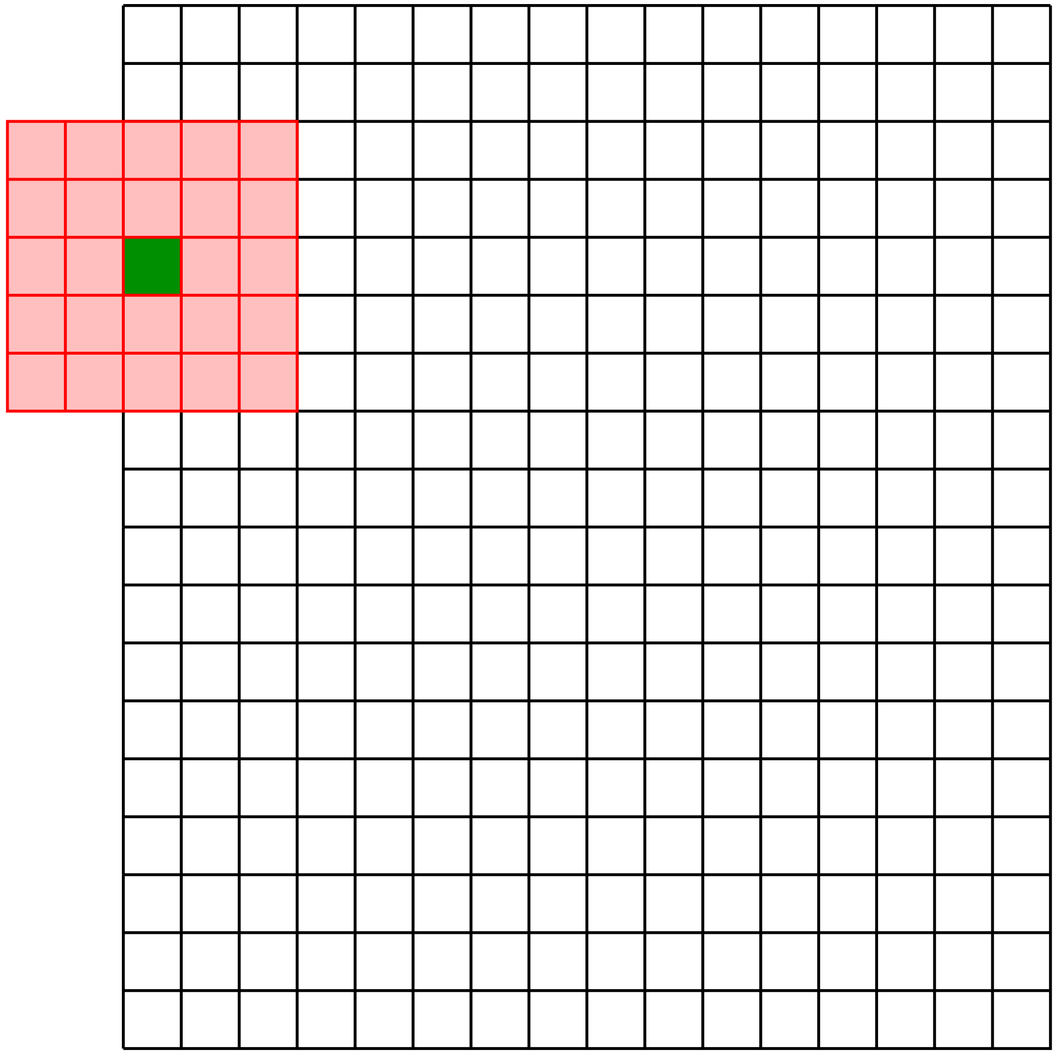
At each pixel, (X, Y), there are $m \times m$ multiplications

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

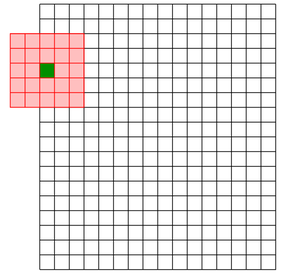
Total: $m^2 \times n^2$ multiplications

When m is fixed, small constant, this is $\mathcal{O}(n^2)$. But when $m \approx n$ this is $\mathcal{O}(m^4)$.

Linear Filters: **Boundary** Effects



Linear Filters: **Boundary** Effects

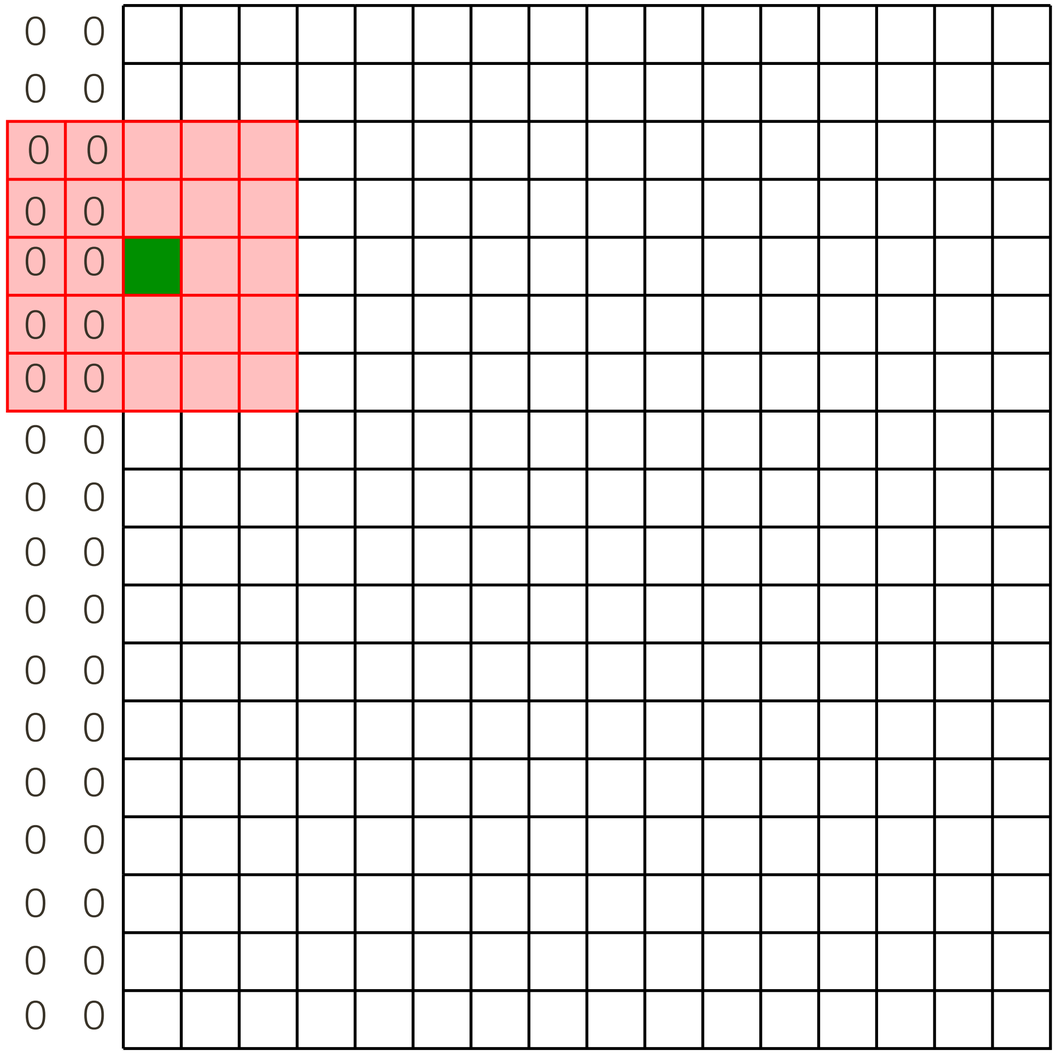


Three standard ways to deal with boundaries:

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

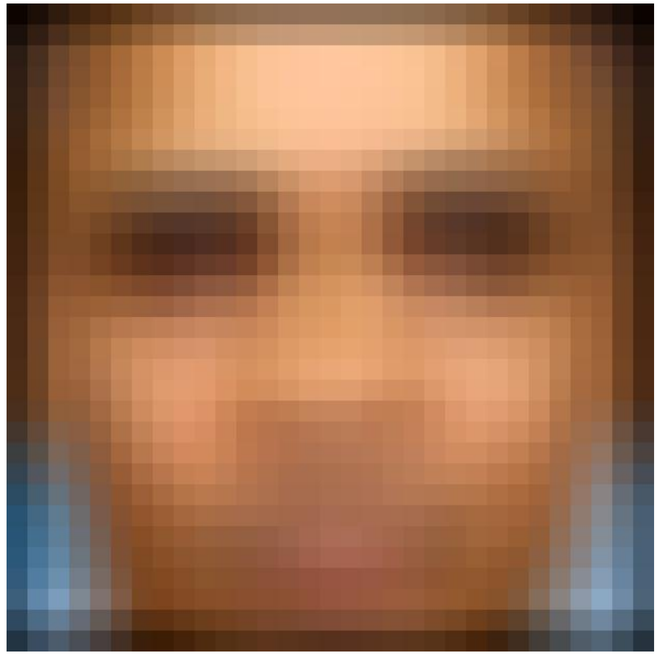
Linear Filters: **Boundary** Effects



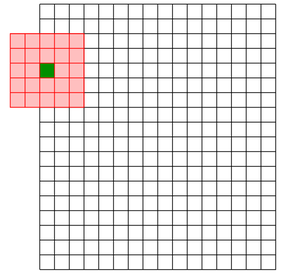
Zero Padding

- Notice decrease in brightness at edge

$$* \begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix} =$$



Linear Filters: **Boundary** Effects



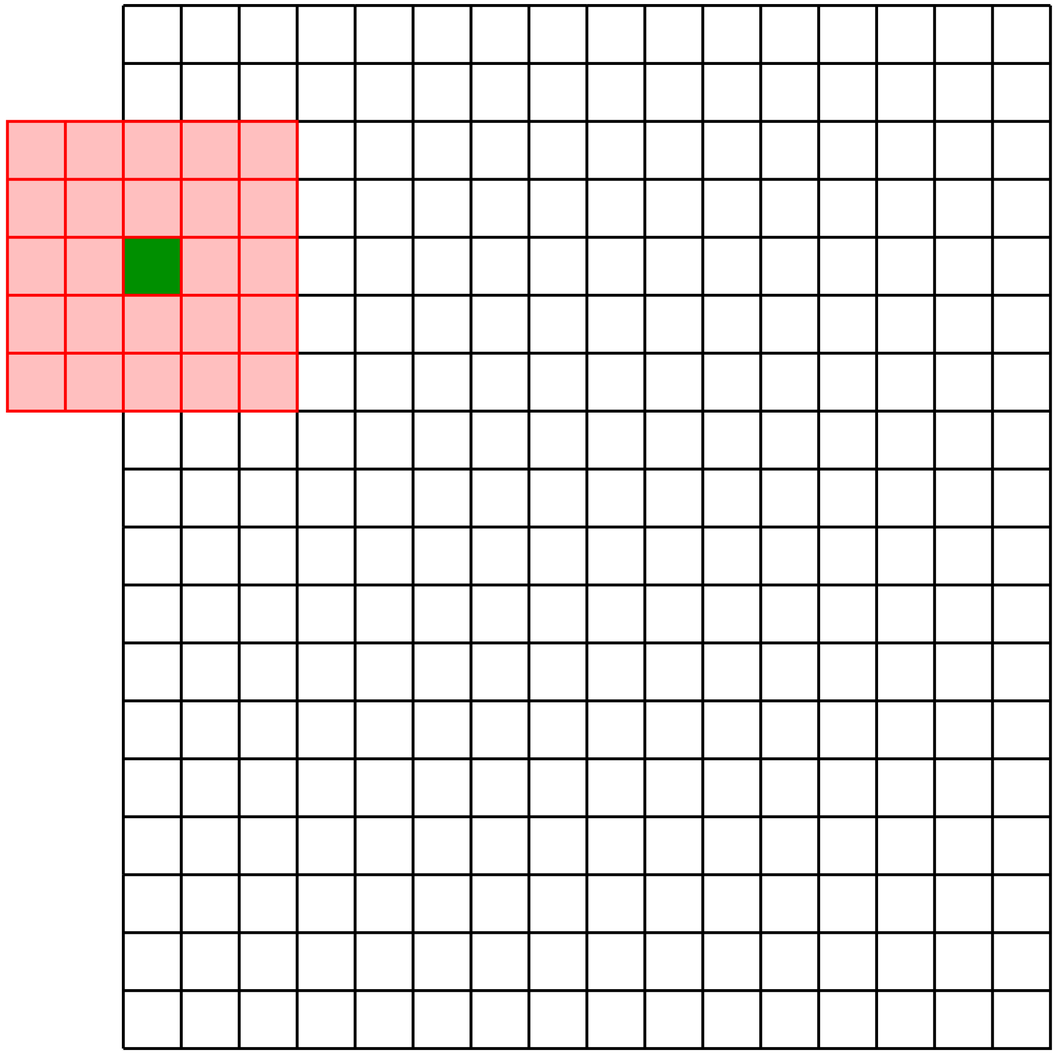
Three standard ways to deal with boundaries:

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

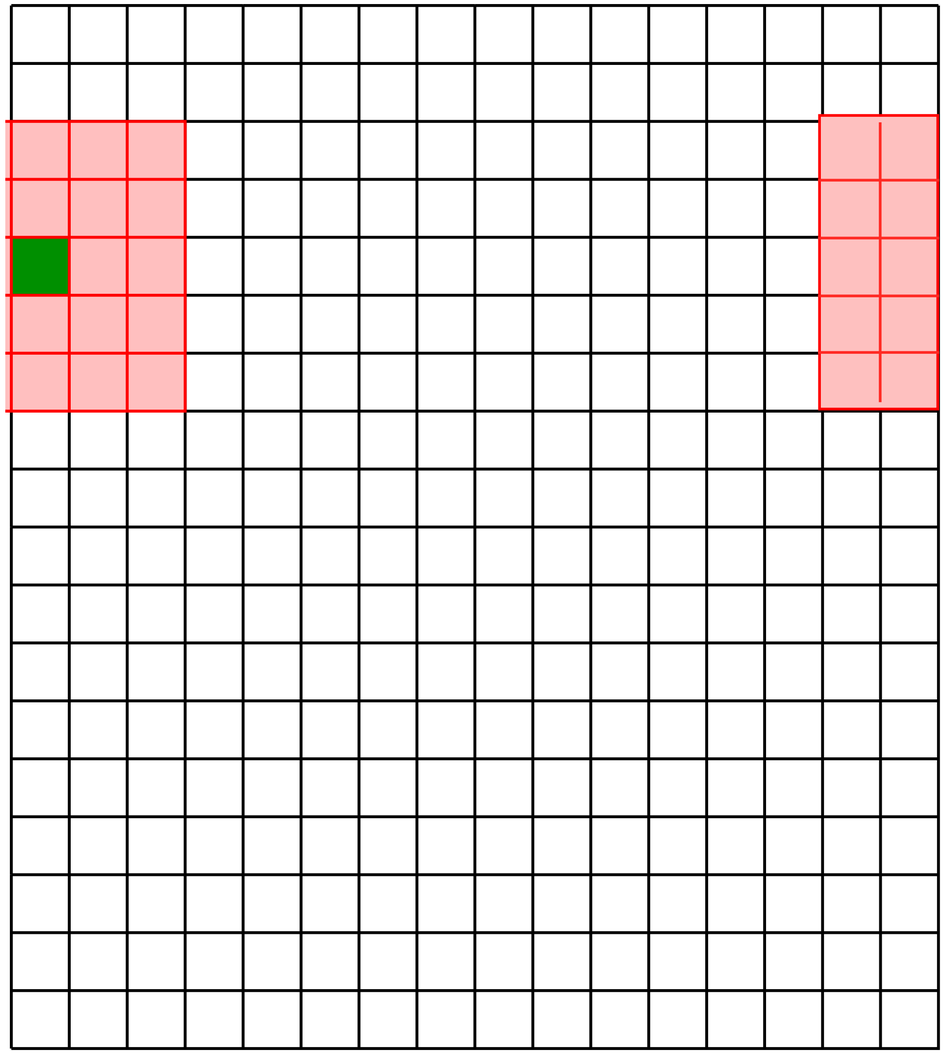
2. **Pad the image with zeros**: Return zero whenever a value of I is required at some position outside the defined limits of X and Y

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

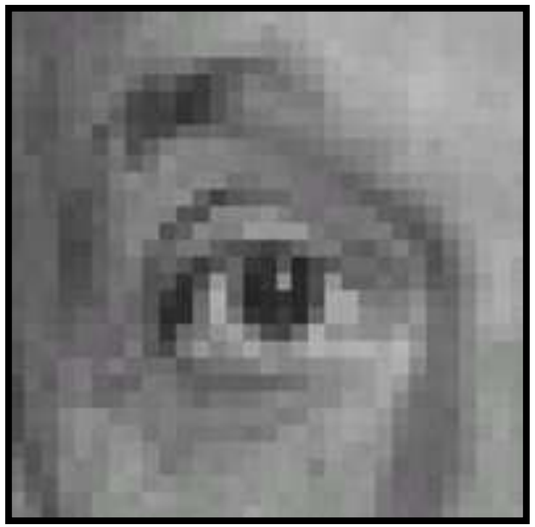


Linear Filters: **Boundary** Effects



A short exercise …

Example 1: Warm up



0	0	0
0	1	0
0	0	0

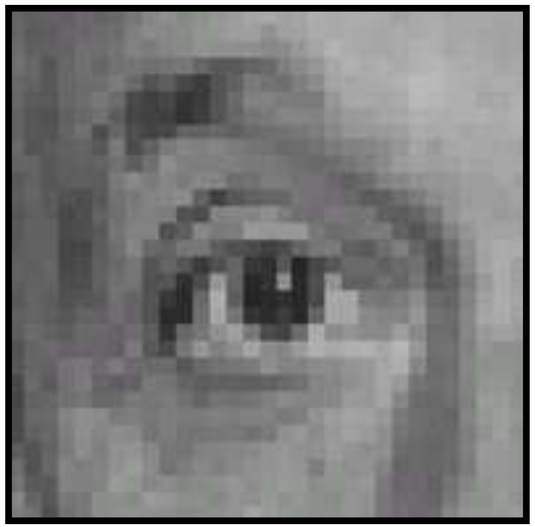
?

Original

Filter

Result

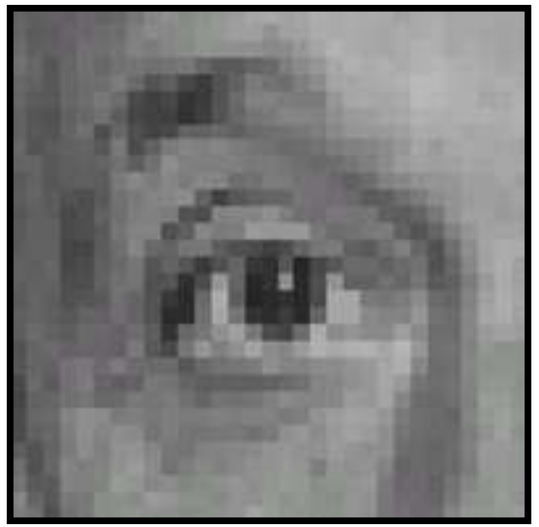
Example 1: Warm up



Original

0	0	0
0	1	0
0	0	0

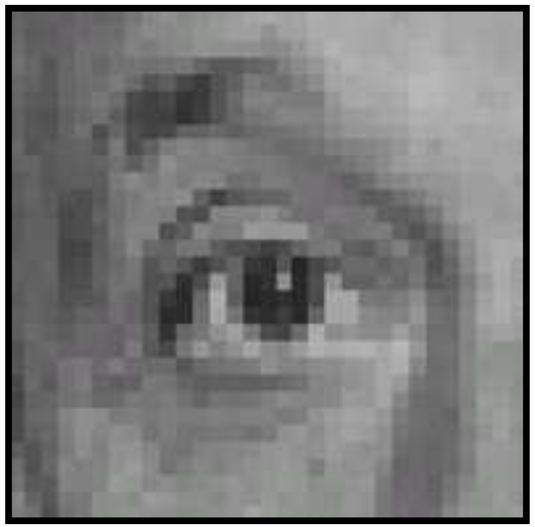
Filter



Result

(no change)

Example 2:

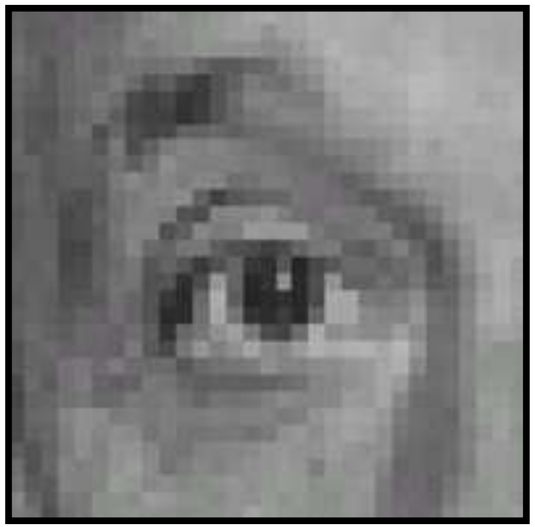


0	0	0
0	0	1
0	0	0

?

Original **Filter** **Result**

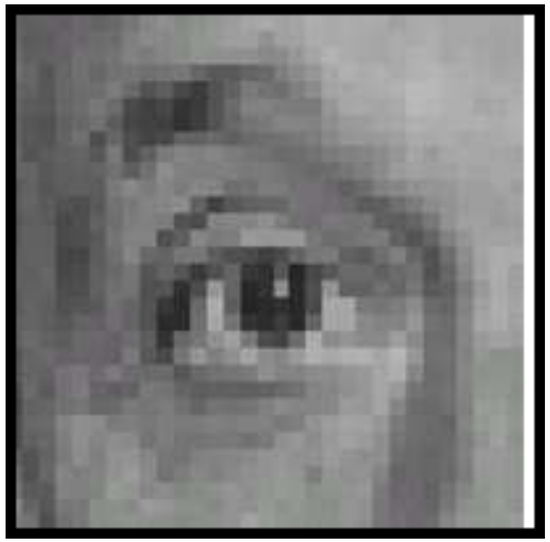
Example 2:



Original

0	0	0
0	0	1
0	0	0

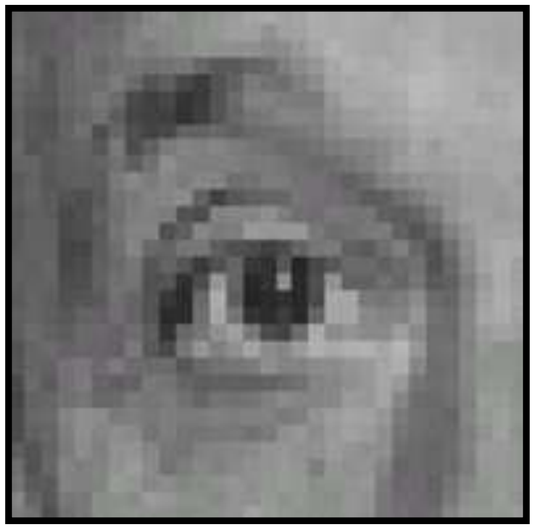
Filter



Result

(sift left by 1 pixel)

Example 3:



Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

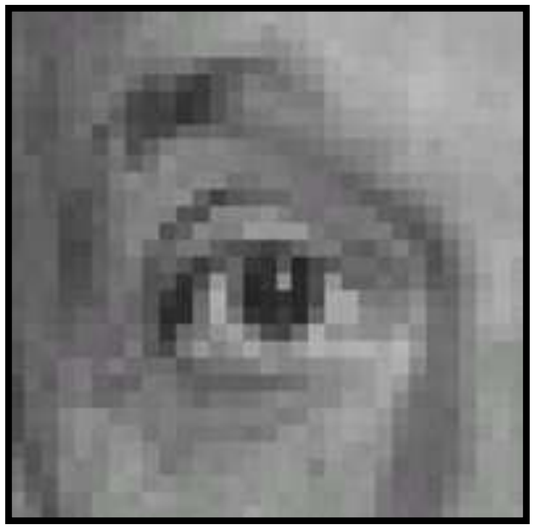
Filter

(filter sums to 1)

?

Result

Example 3:

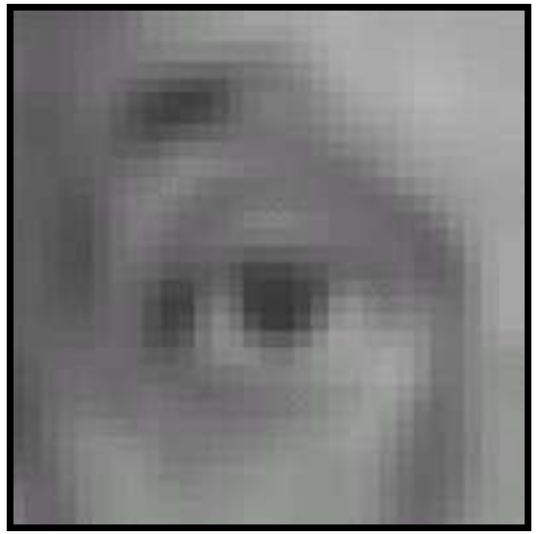


Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Filter

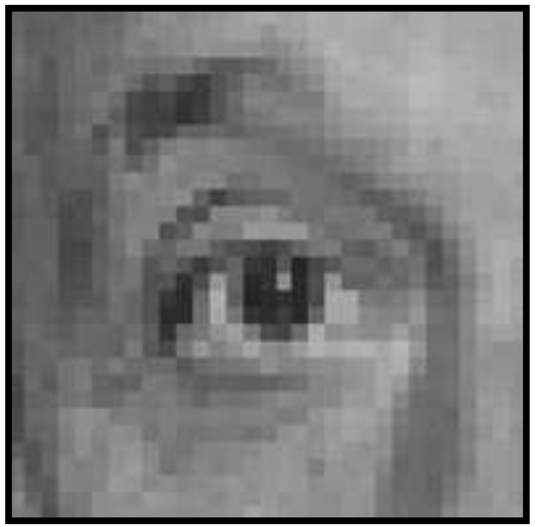
(filter sums to 1)



Result

(blur with a box filter)

Example 4:



Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

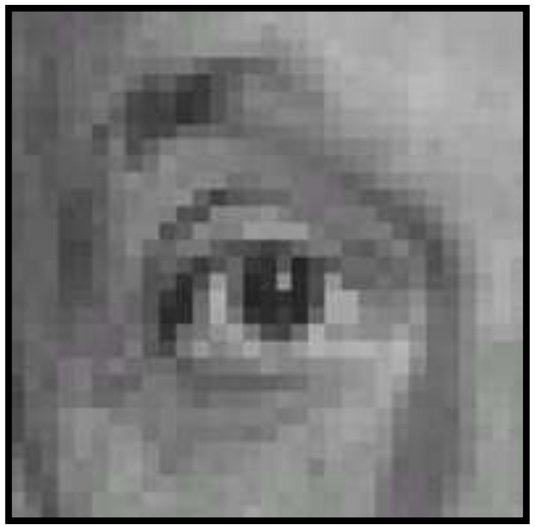
Filter

(filter sums to 1)

?

Result

Example 4:



Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

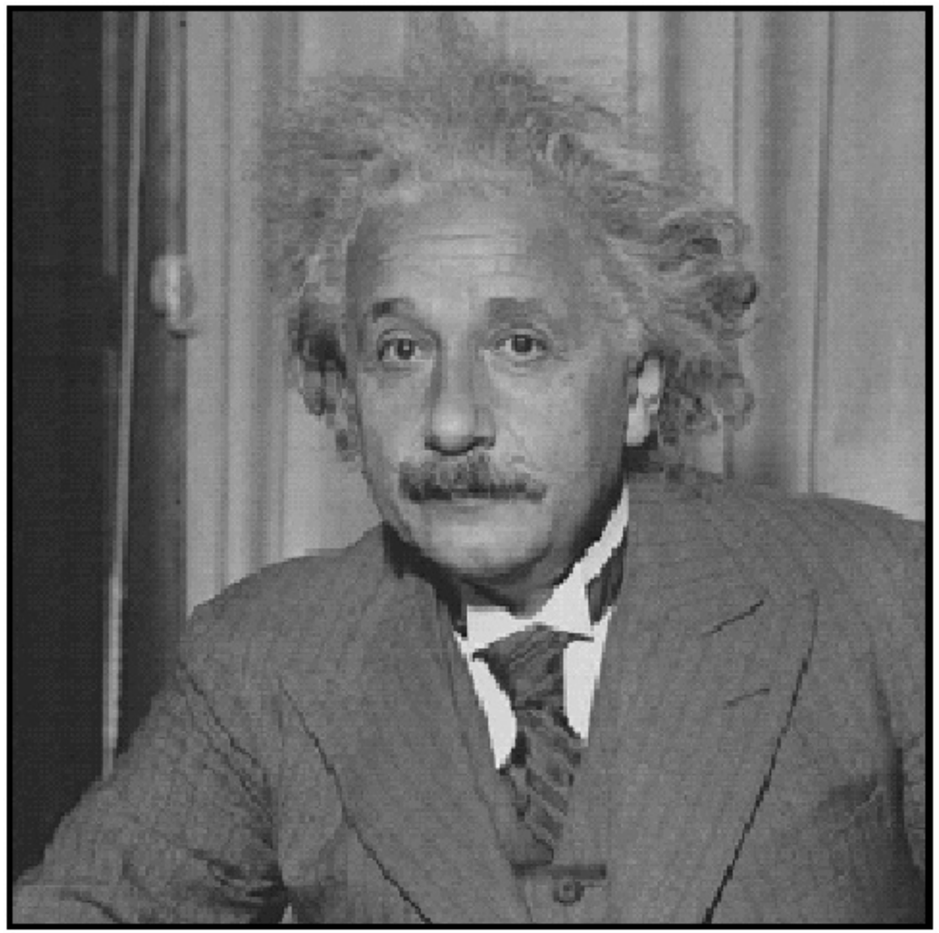
Filter

(filter sums to 1)

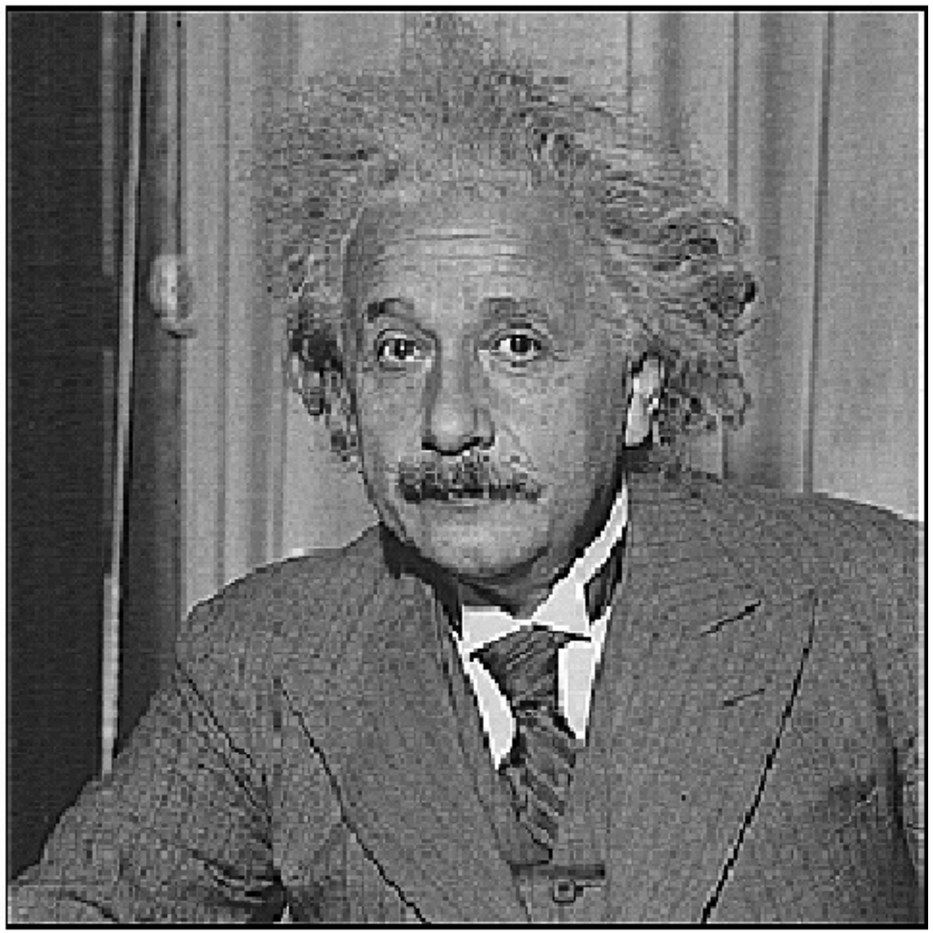
Result

(sharpening)

Example 4: Sharpening



Before



After

Example 4: Sharpening

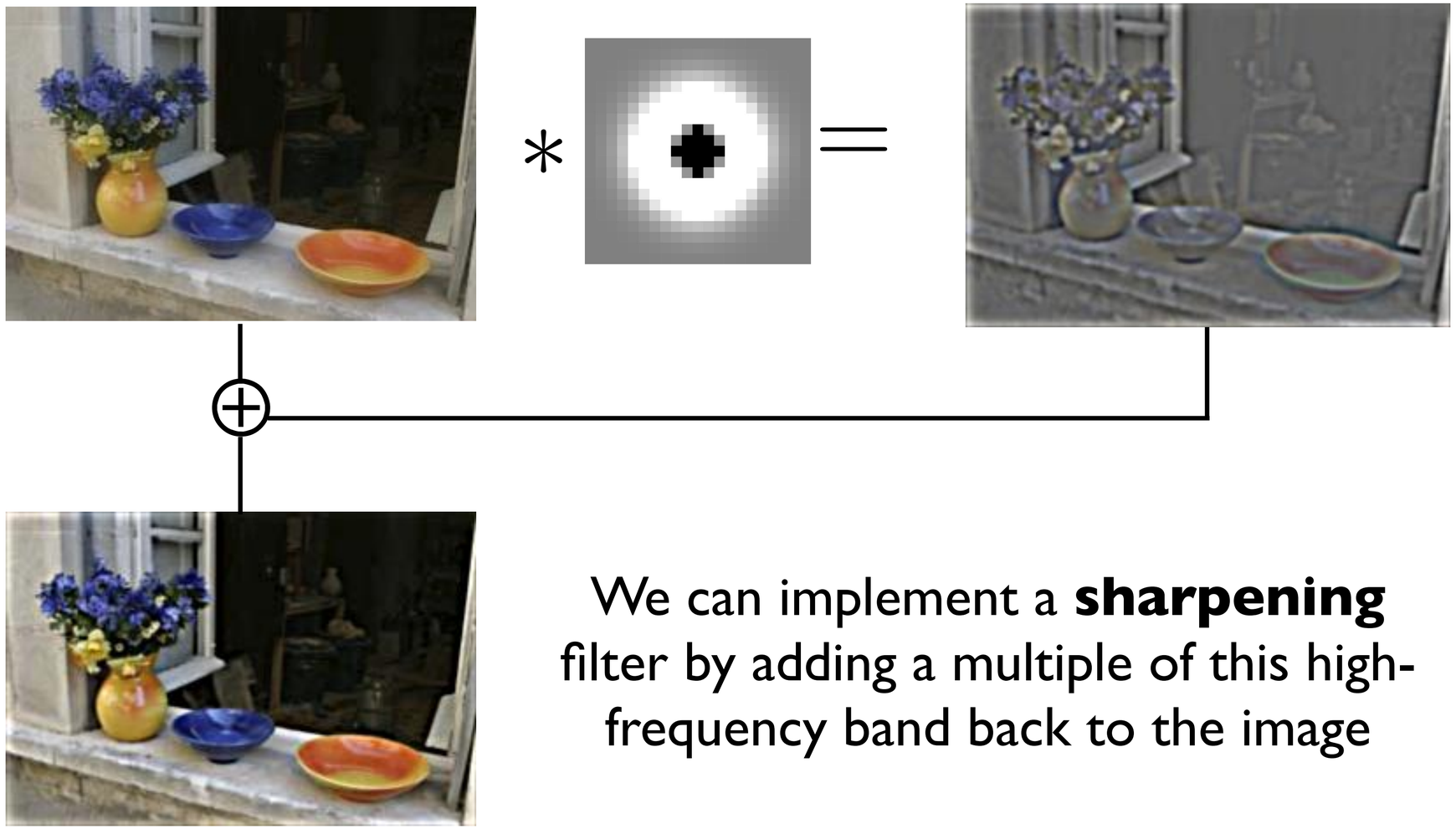
Before

After

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Centre Surround Filter

- Useful for extracting features at a certain **scale**



We can implement a **sharpening** filter by adding a multiple of this high-frequency band back to the image

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**

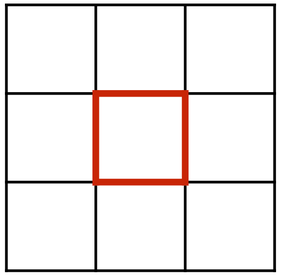
$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

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**

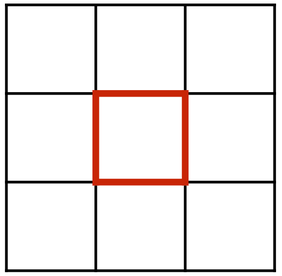
$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X-i, Y-j)$$

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image



Output

= 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)

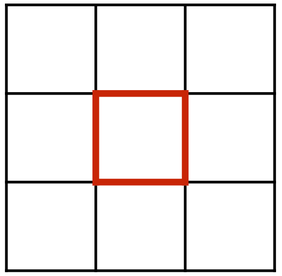
i	h	g
f	e	d
c	b	a

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image



Output

= 9a + 8b + 7c
+ 6d + 5e + 4f
+ 3g + 2h + 1i

Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

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

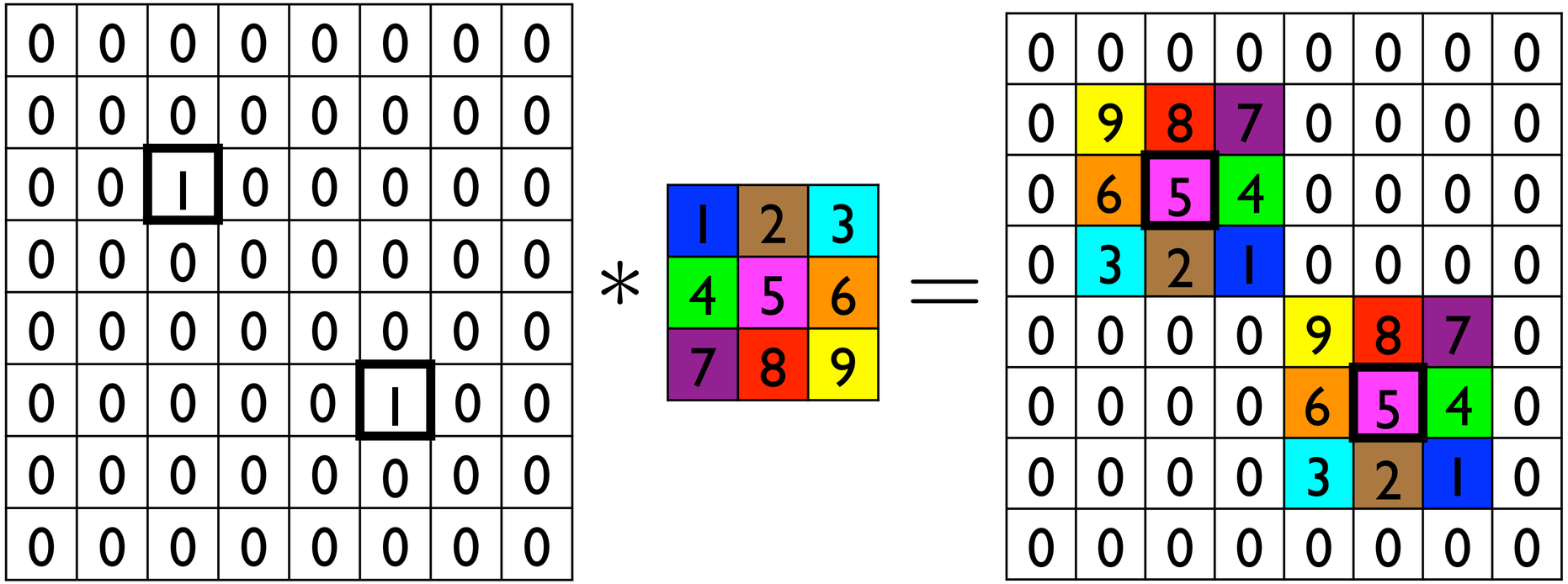
Definition: **Convolution**

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

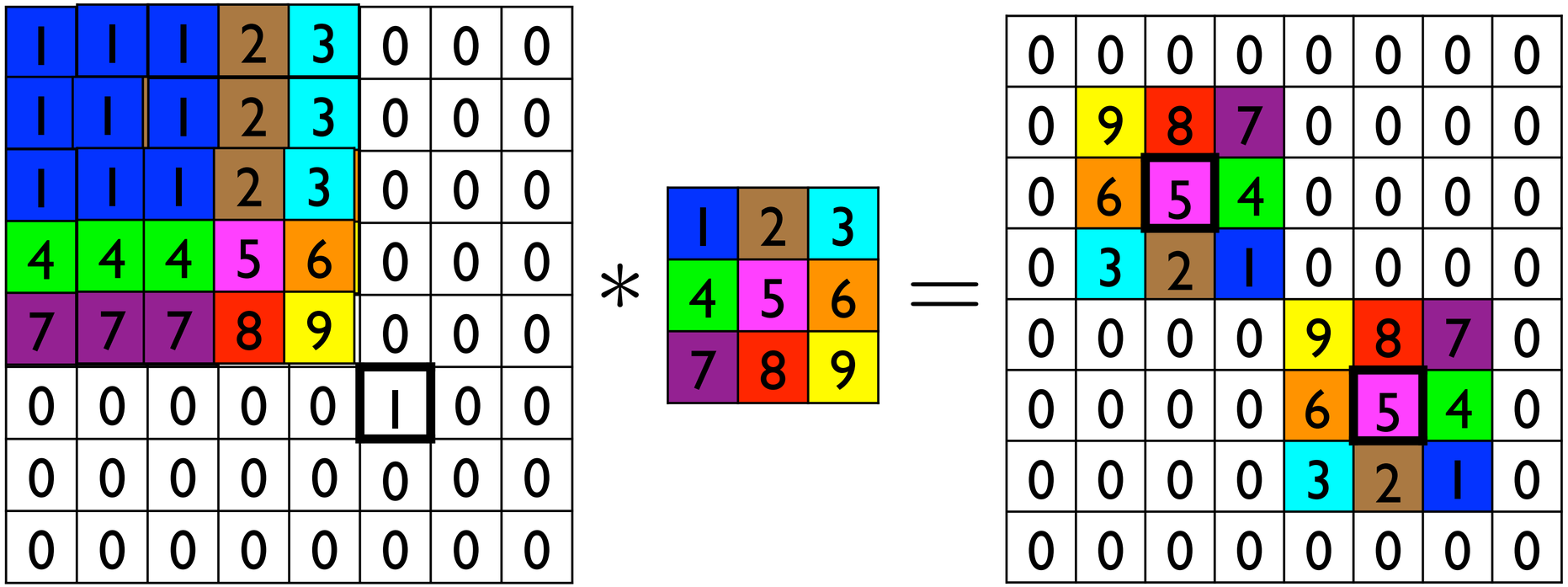
$$= \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(-i,-j) I(X+i, Y+j)$$

Note: if $F(X,Y) = F(-X,-Y)$ then correlation = convolution.

Point Spread Function



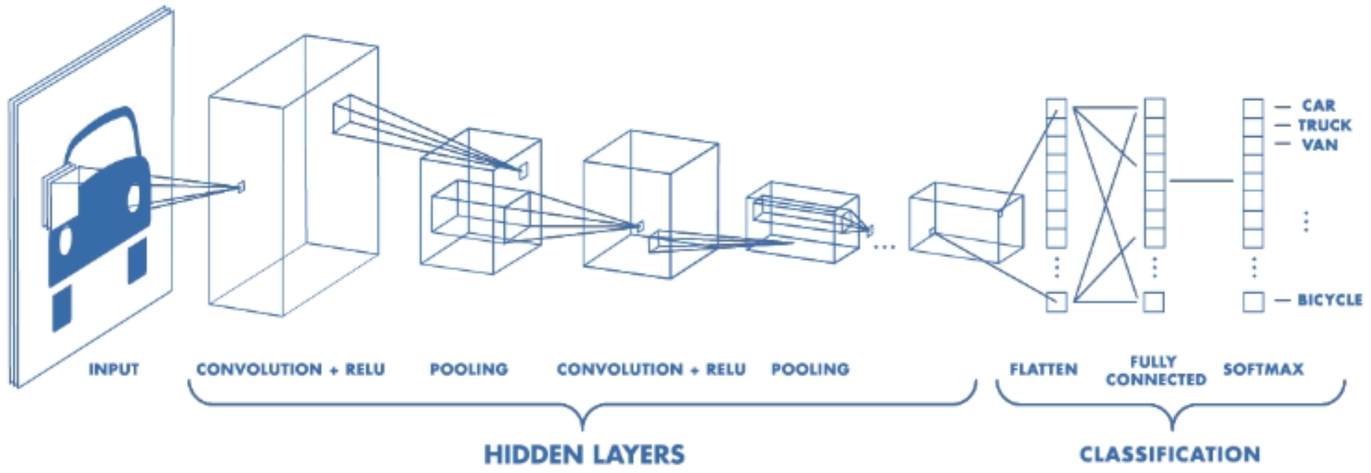
Point Spread Function



- The point spread function is the correlation kernel rotated by 180° (= the convolution kernel)

Preview: Why convolutions are important?

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



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**

3.3 Convolution as matrix multiplication

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

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

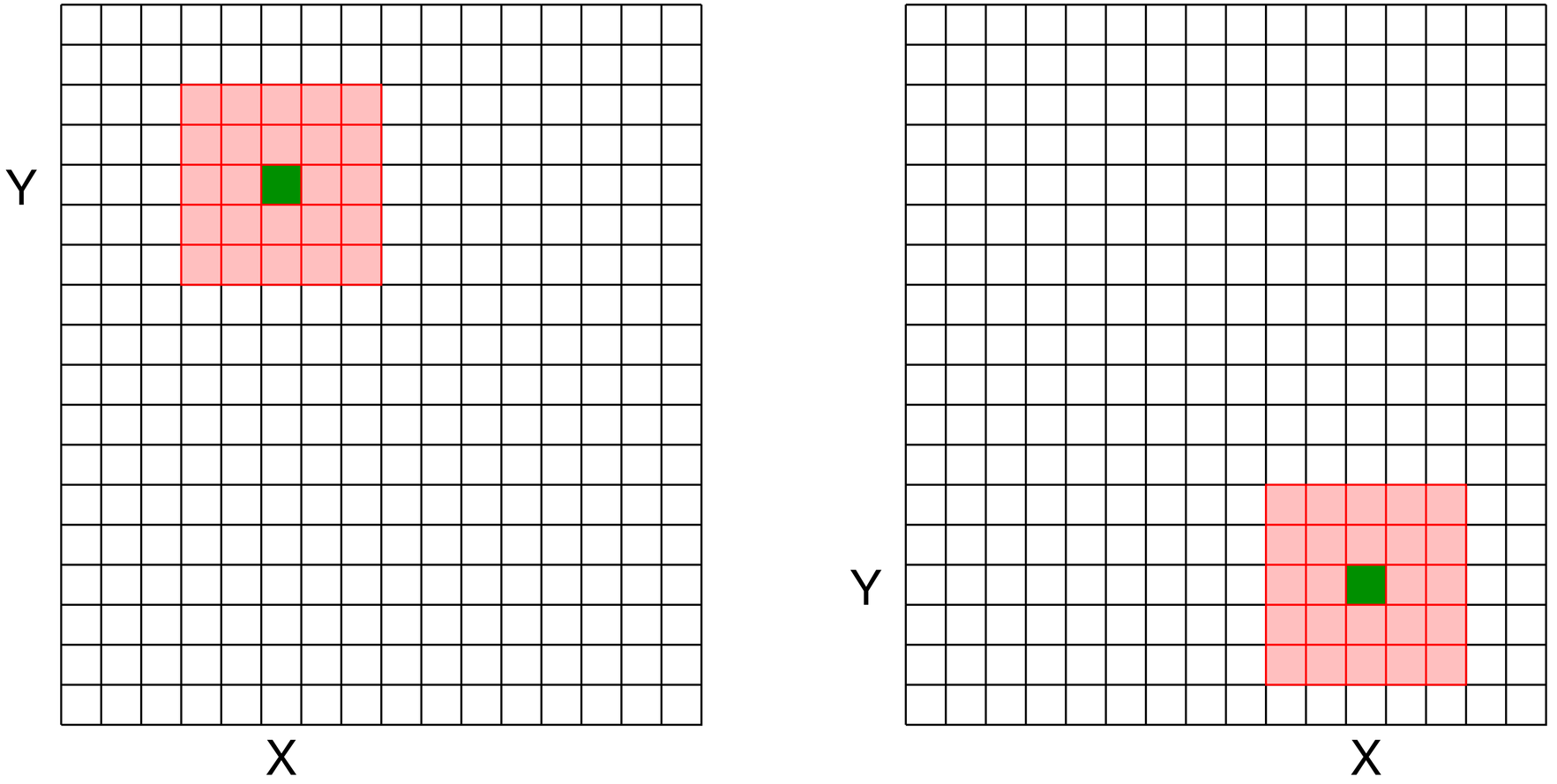
Scaling: Let F be digital filter and let k be a scalar

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

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

Linear Filters: Shift Invariance

Same linear operation is applied everywhere, no dependence on absolute position



Linear Systems: Characterization Theorem

Any linear, shift invariant operation can be expressed as convolution

Menu for Today

Topics: Image Filtering

- **Image** as a **function**
- **Linear** filters
- **Correlation** / **Convolution**

Readings:

- **Today's** Lecture: Szeliski 2 3.1-3.3, Forsyth & Ponce (2nd ed.) 4.1, 4.5

Reminders:

- Complete **Assignment 0** (optional, ungraded) due **Thursday**

iClicker test next lecture

Please sign up for the iClicker course **CPSC 425 101 2023W1 Computer Vision**

See also the UBC iClicker Student Guide, contact LT Hub if stuck