THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision

Lecture 12: Midterm Review

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

Midterm Details

Closed book, no calculators

Format similar to posted practice problems

- Part A: Multiple-part true/false
- Part B: Short answer

No coding questions

No complex math questions (see no calculators above)

Midterm Review: Readings

Lecture 1–11 **slides**

Assigned **readings** from Forsyth & Ponce (2nd ed.)
— Paper “Texture Synthesis by Non-parametric Sampling”

Assignments 1–2

iClicker questions

Lecture exercises / examples

Practice problems (with solutions)

Paths to **Understanding**

Five distinct “paths” to a deeper understanding of CPSC 425 course material:

1. mathematics (i.e., theory)

2. “visualize” computation(s)

3. experiment
- — on simple (test) cases
- — on real images

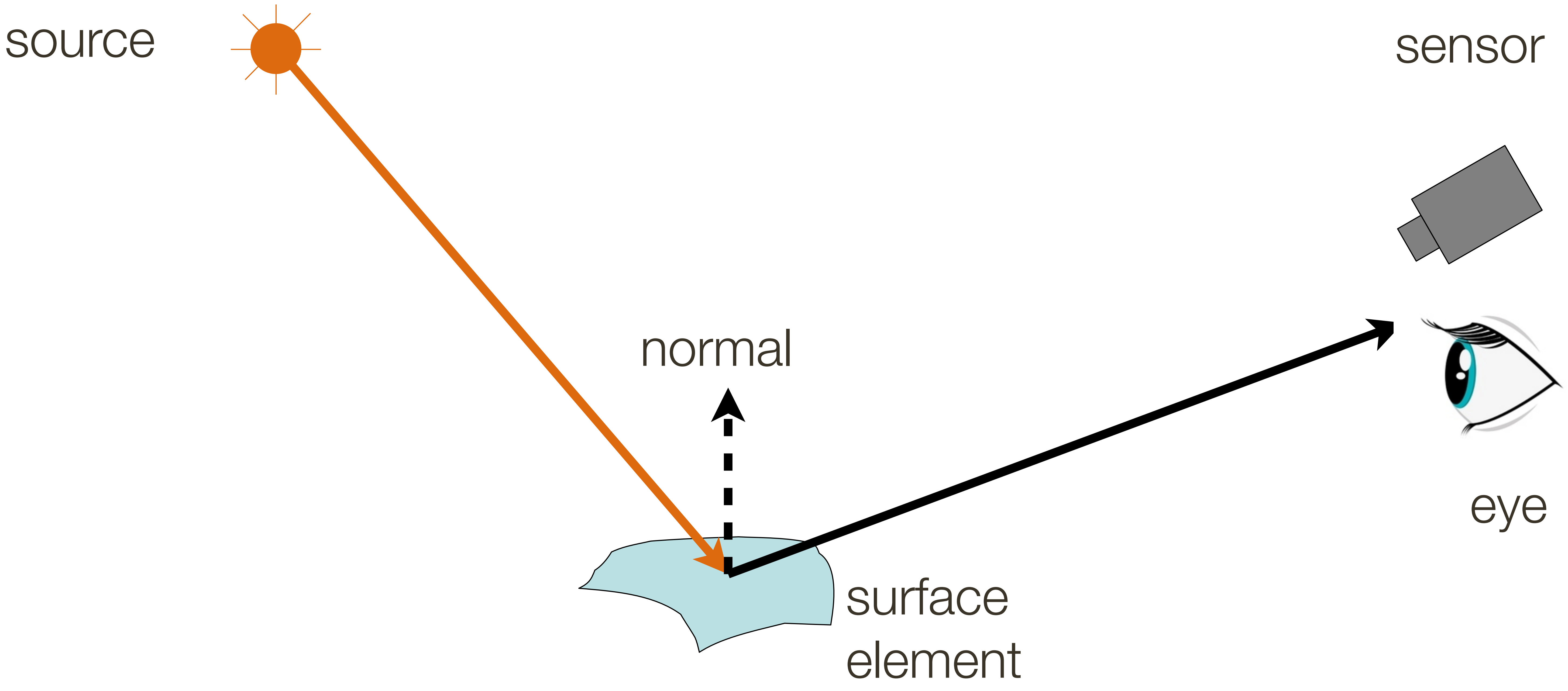
4. read code

5. write code

Overview: Image Formation, Cameras and Lenses

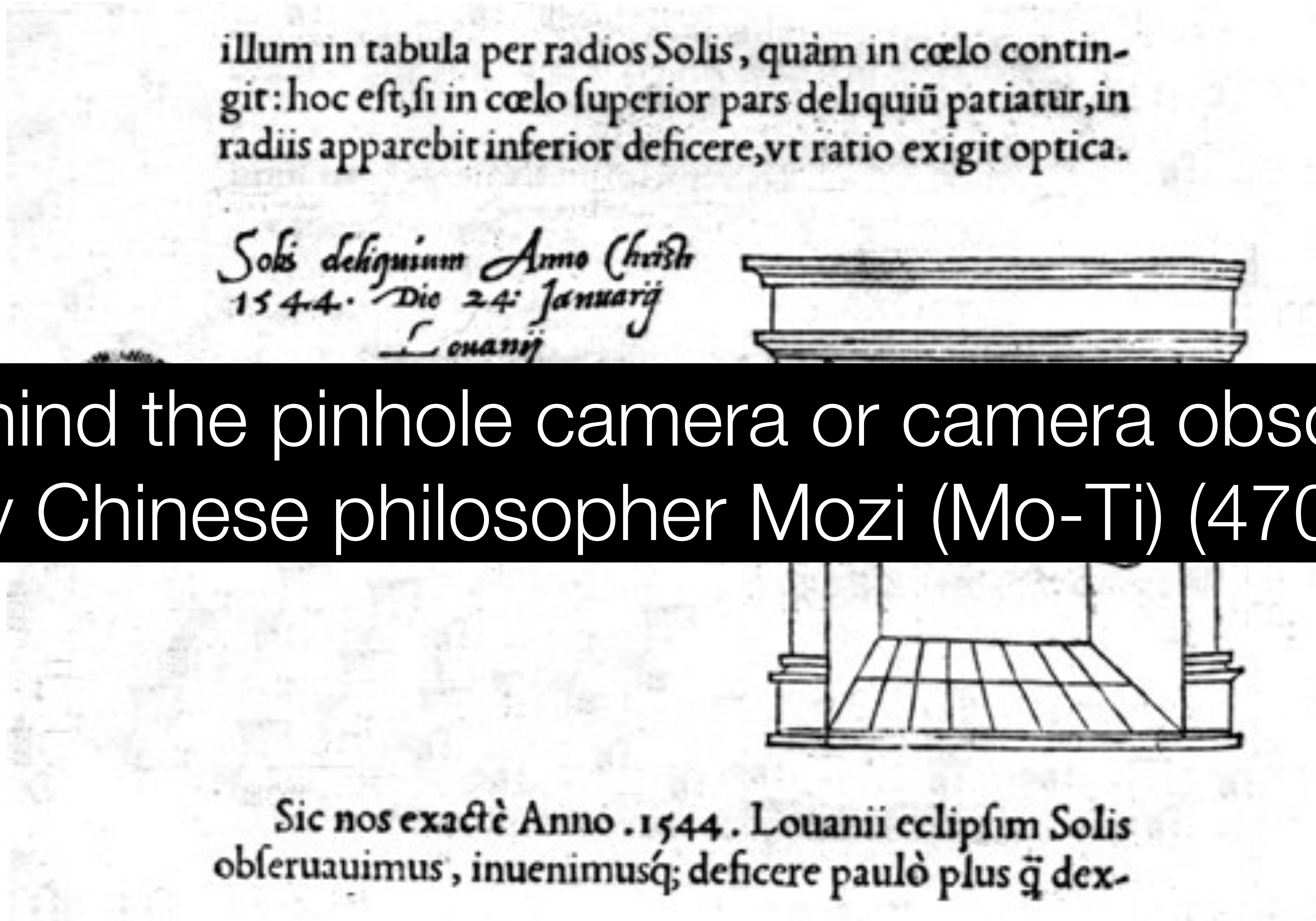
The **image formation process** that produces a particular image depends on

- **Lightening** condition
- Scene **geometry**
- **Surface** properties
- Camera **optics**



Sensor (or eye) **captures amount of light** reflected from the object

Camera Obscura (latin for “dark chamber”)



illum in tabula per radios Solis, quàm in cœlo contingit: hoc eſt, ſi in cœlo ſuperior pars deliquiũ patiatur, in radiis apparebit inferior deficere, vt ratio exigit optica.

Solis deliquium Anno Christi
1544. Die 24 Januarij
Louanij

Sic nos exactè Anno .1544. Louanii eclipſim Solis obſeruauimus, inuenimusq; deficere paulò plus q̃ dex-

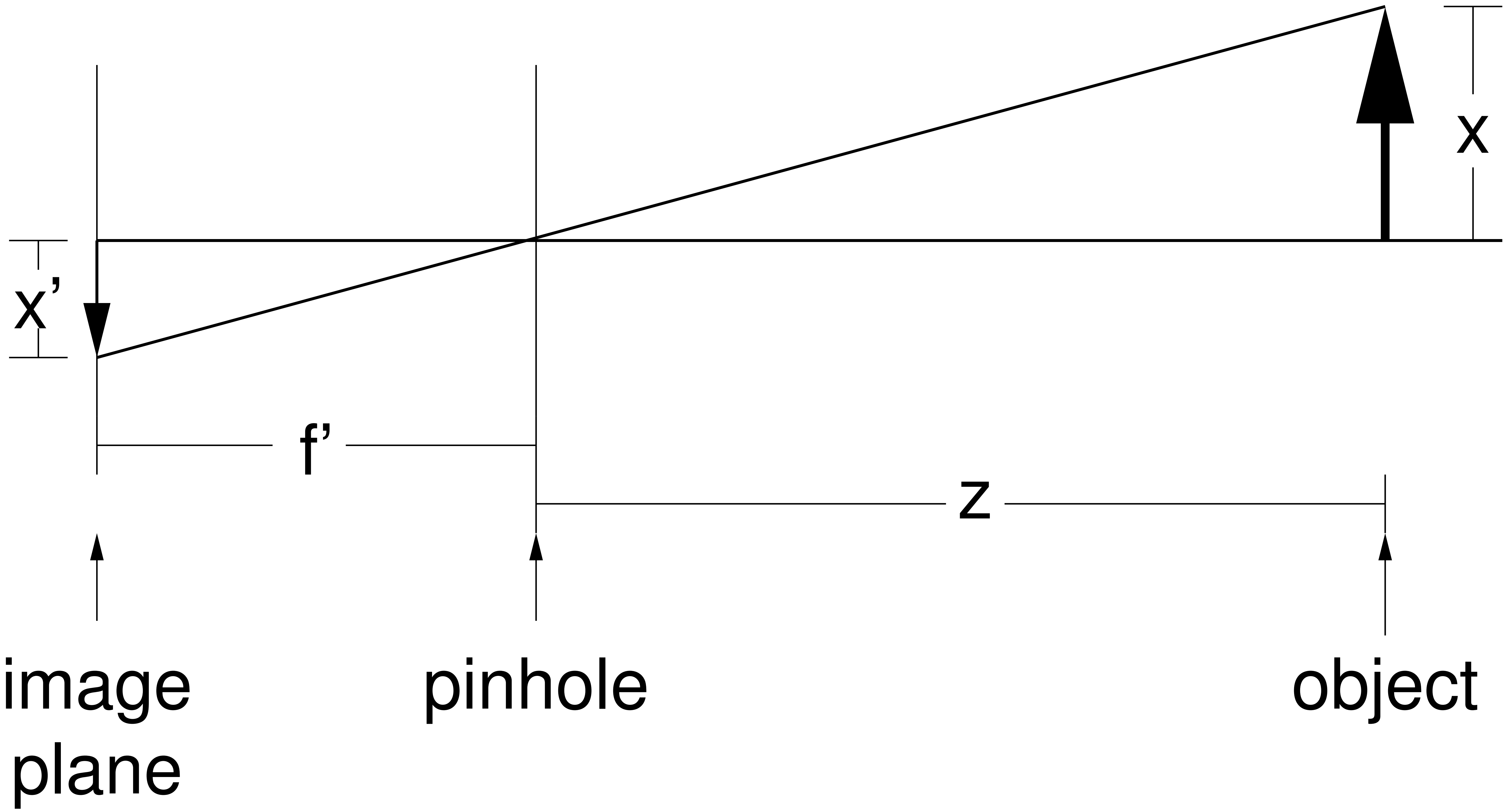
principles behind the pinhole camera or camera obscura were first mentioned by Chinese philosopher Mozi (Mo-Ti) (470 to 390 BCE)

Reinerus Gemma-Frisius observed an eclipse of the sun at Louvain on January 24, 1544. He used this illustration in his book, “De Radio Astronomica et Geometrica,” 1545. It is thought to be the first published illustration of a camera obscura.

Credit: John H., Hammond, “Th Camera Obscure, A Chronicle”

Pinhole Camera (Simplified)

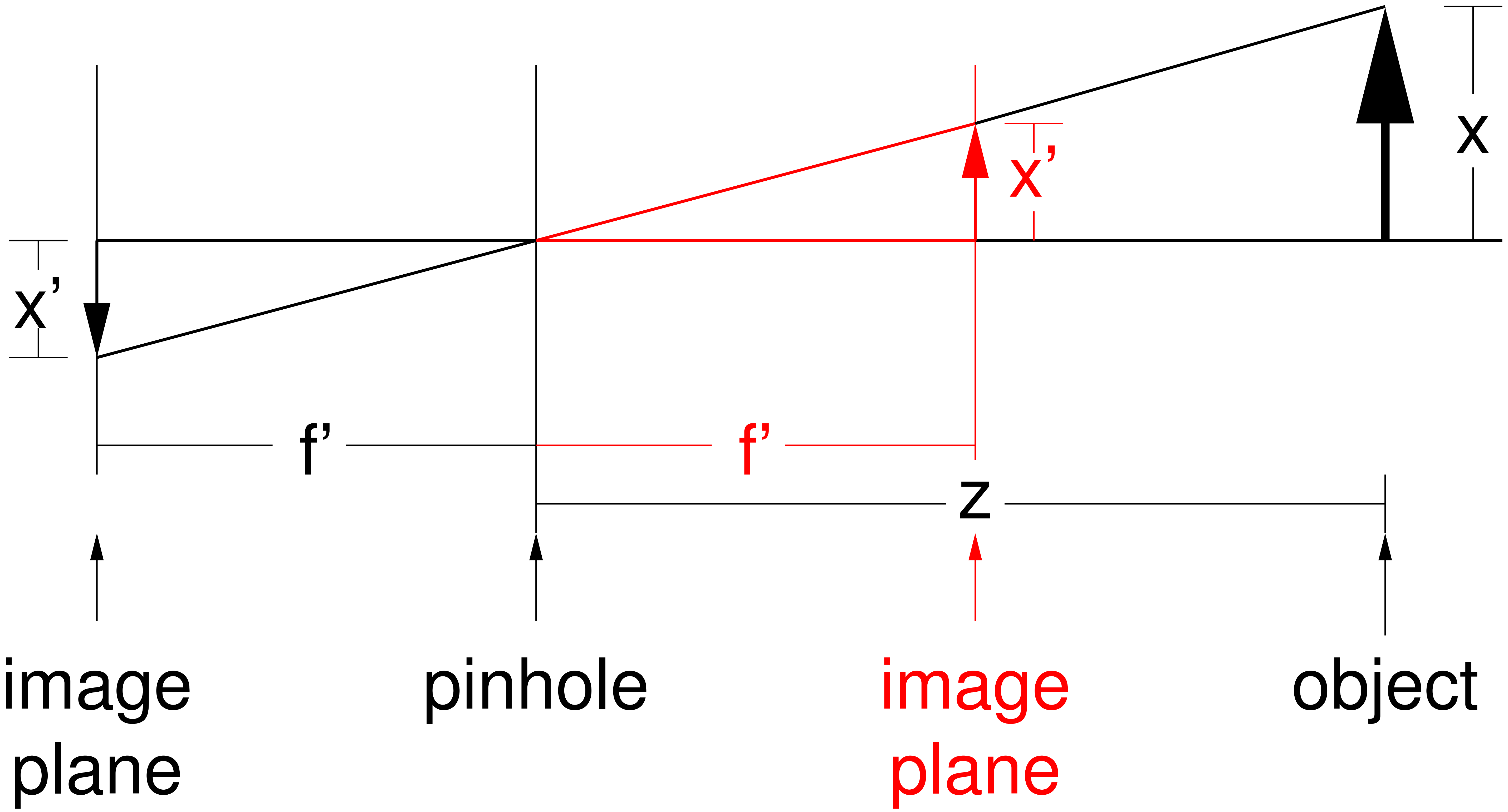
f’ is the **focal length** of the camera



Note: In a pinhole camera we can adjust the focal length, all this will do is change the **size** of the resulting image

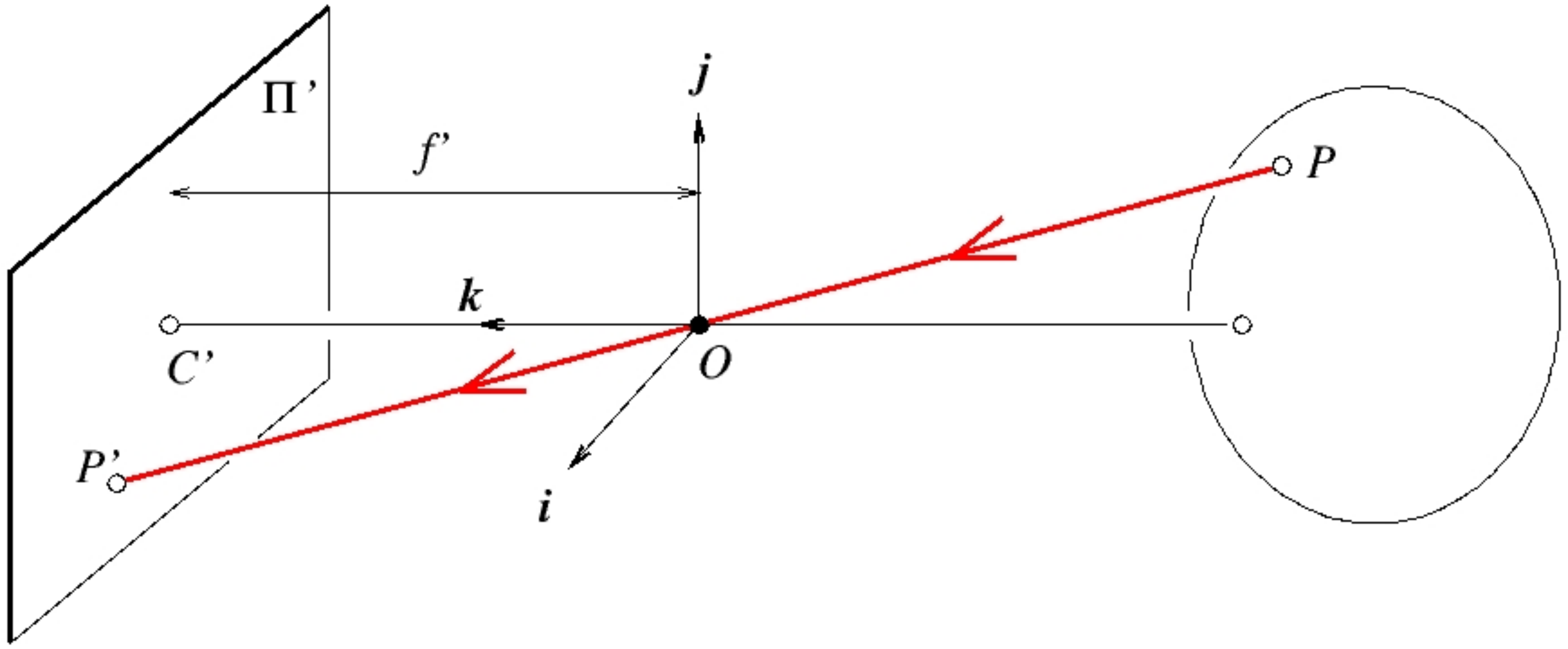
Pinhole Camera (Simplified)

It is convenient to think of the **image plane** which is in from of the pinhole



What happens if object moves towards the camera? Away from the camera?

Perspective Projection



3D object point

Forsyth & Ponce (1st ed.) Figure 1.4

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

$$\boxed{\begin{aligned} x' &= f' \frac{x}{z} \\ y' &= f' \frac{y}{z} \end{aligned}}$$

Note: this assumes world coordinate frame at the optical center (pinhole) and aligned with the image plane, image coordinate frame aligned with the camera coordinate frame

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

Sample Question: Image Formation

True of **false**: A pinhole camera uses an orthographic projection.

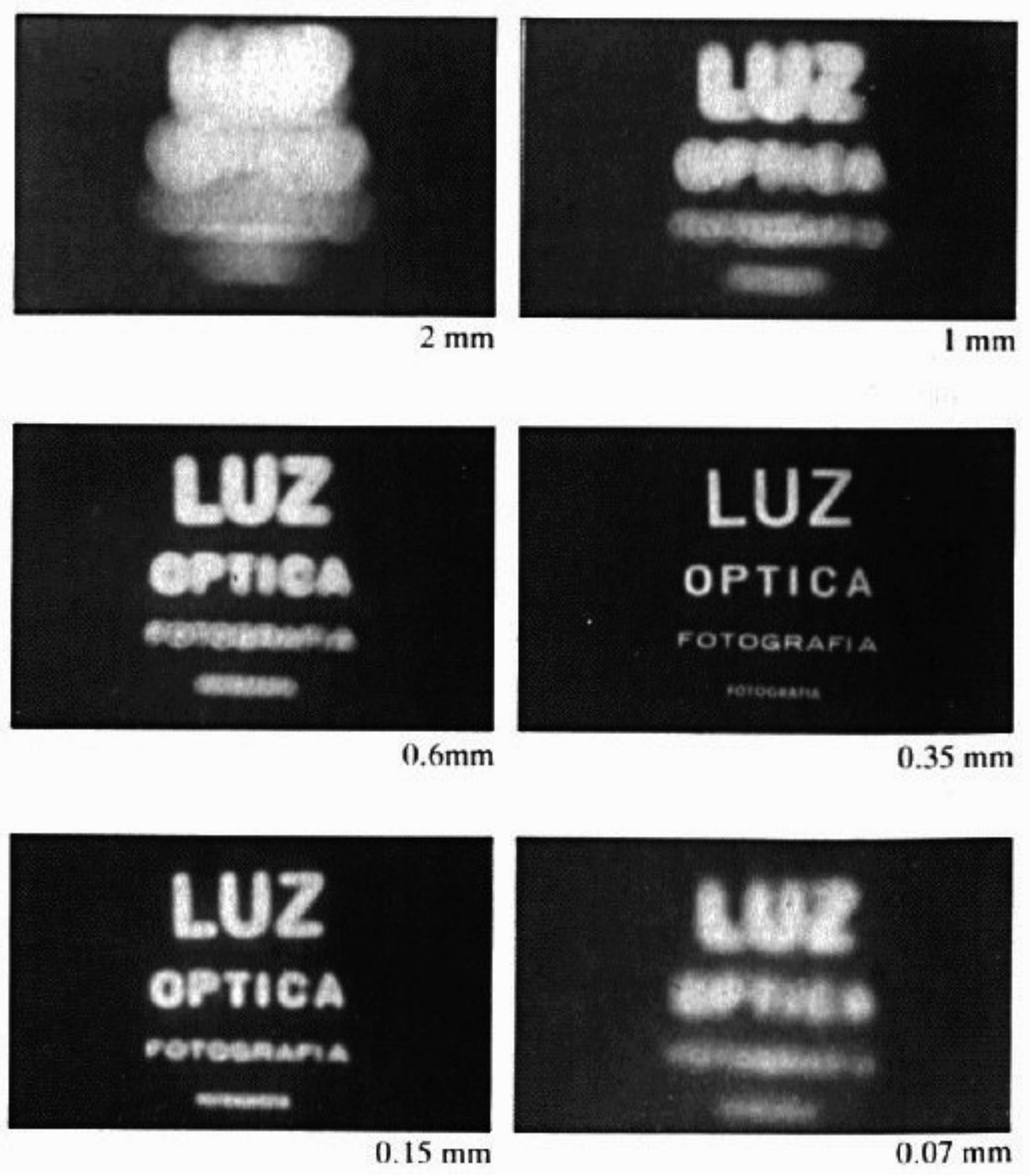
Why **Not** a Pinhole Camera?

— 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



2 mm

1 mm

0.6mm

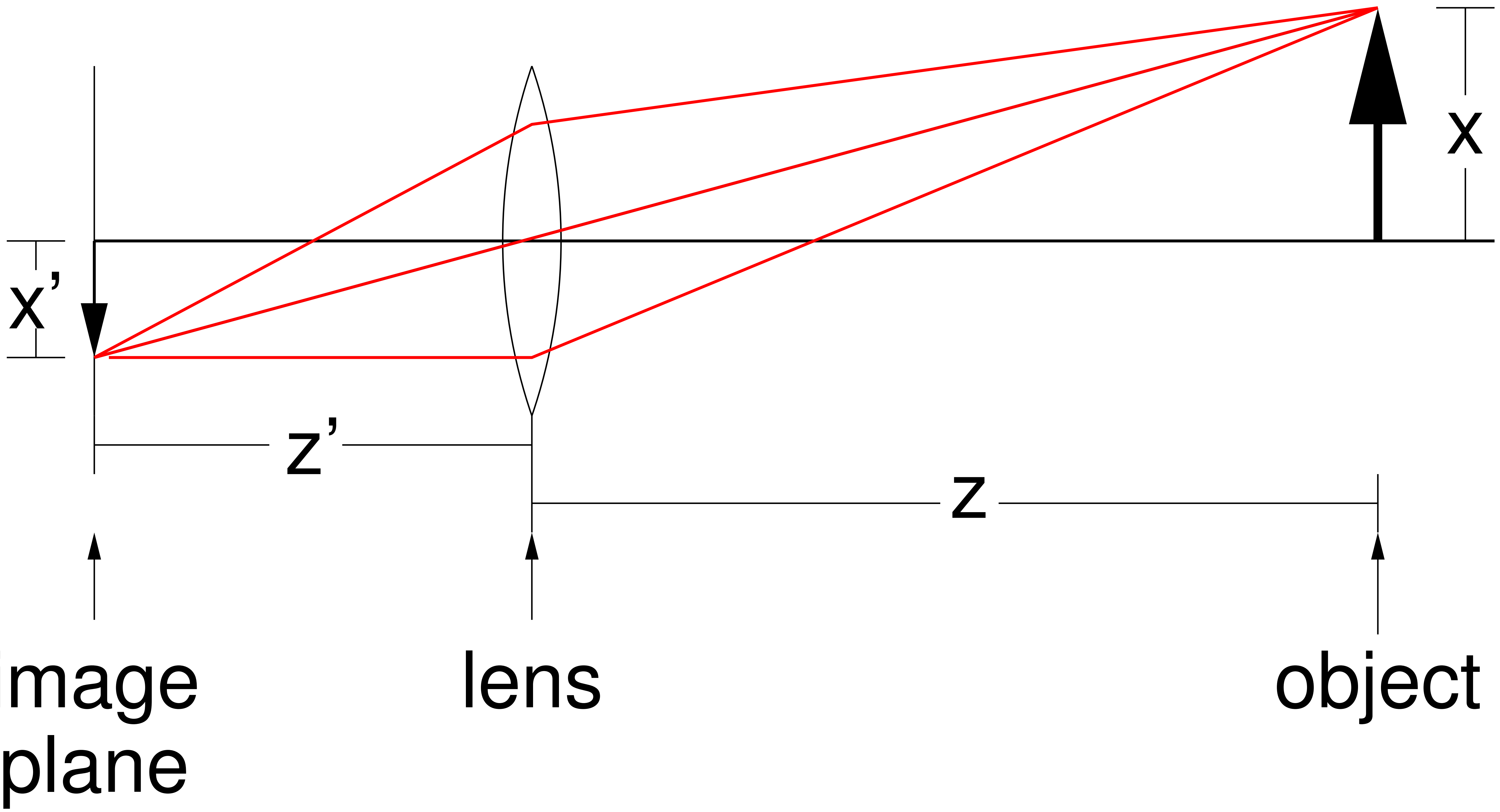
0.35 mm

0.15 mm

0.07 mm

Image Credit: Credit: E. Hecht. “Optics,” Addison-Wesley, 1987

Pinhole Model (Simplified) **with Lens**



Vignetting

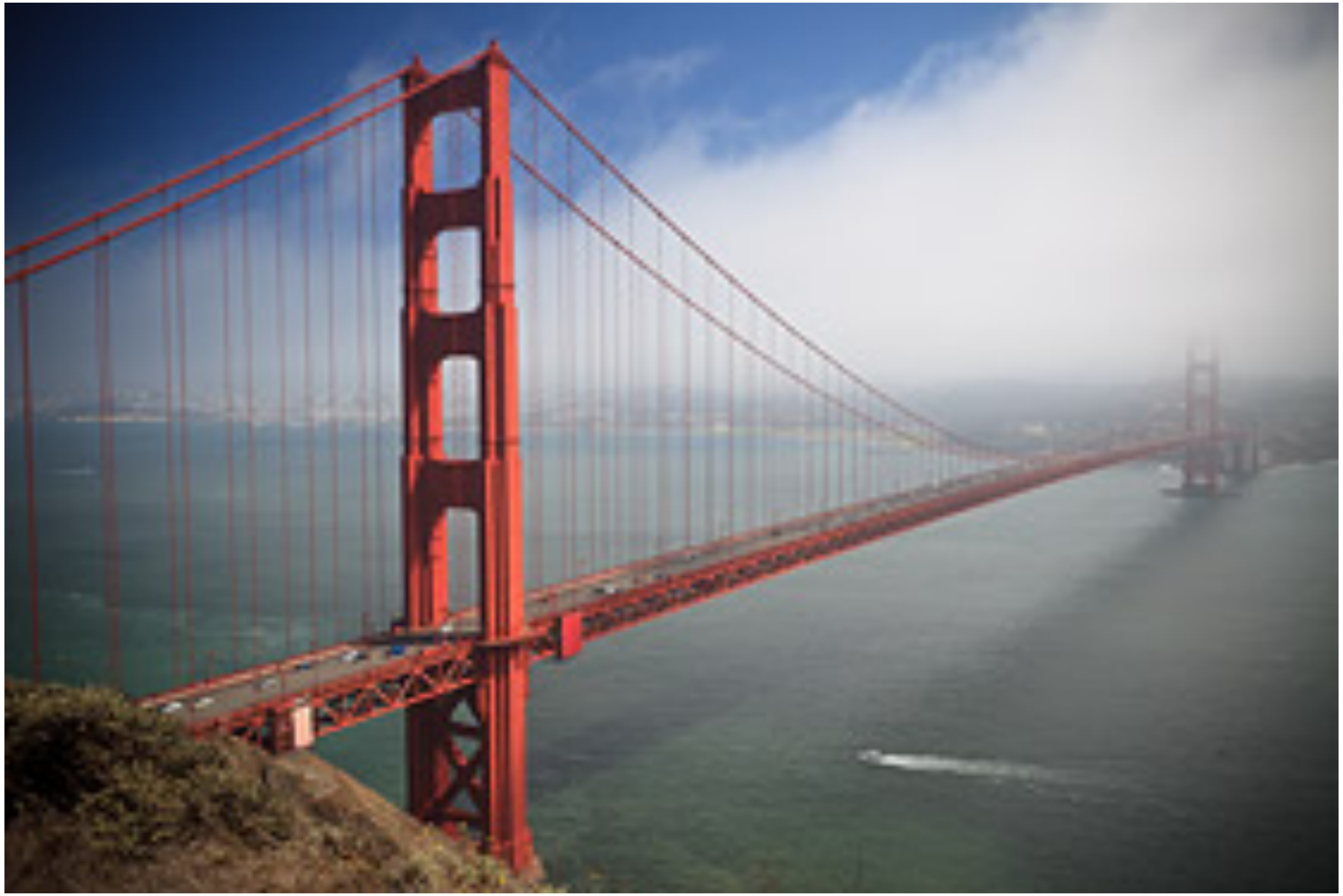


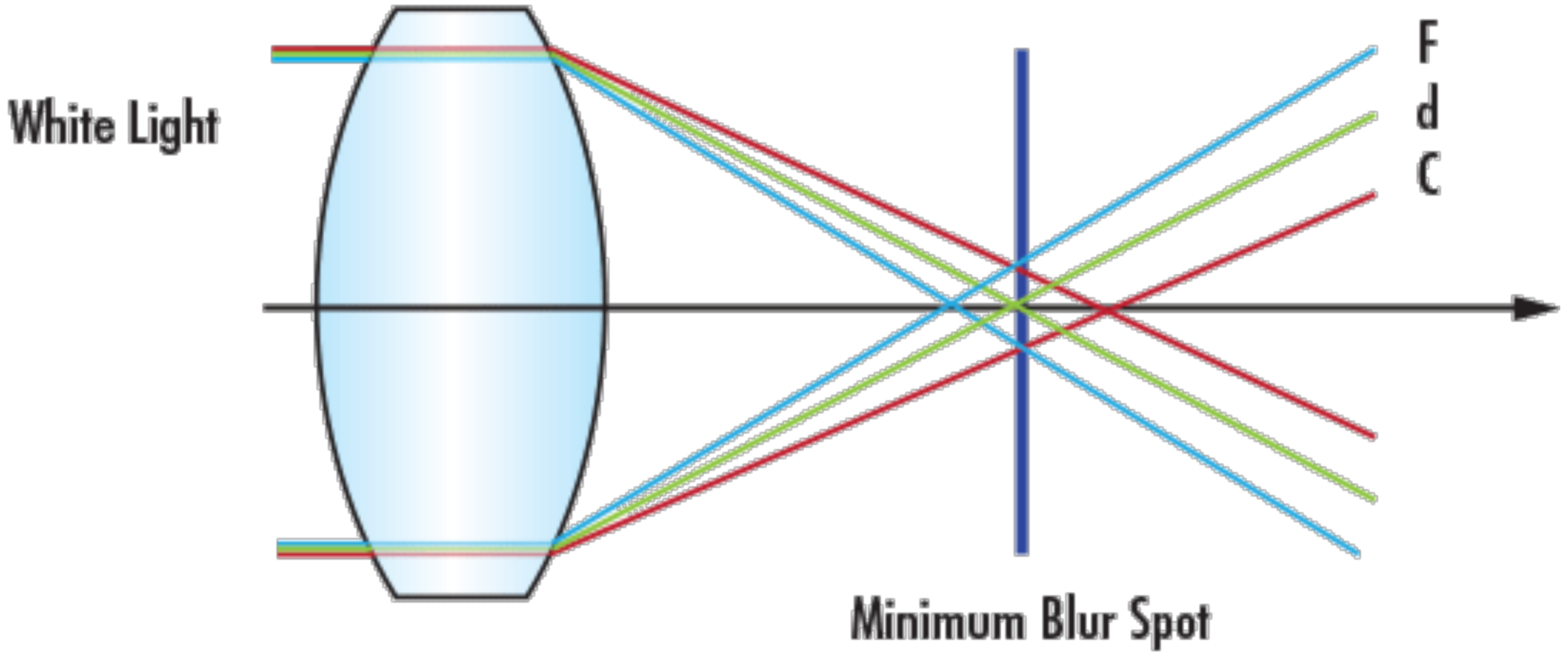
Image Credit: Cambridge in Colour

Chromatic **Aberration**

— Index of **refraction depends on wavelength**, λ, of light

— Light of different colours follows different paths

— Therefore, not all colours can be in equal focus



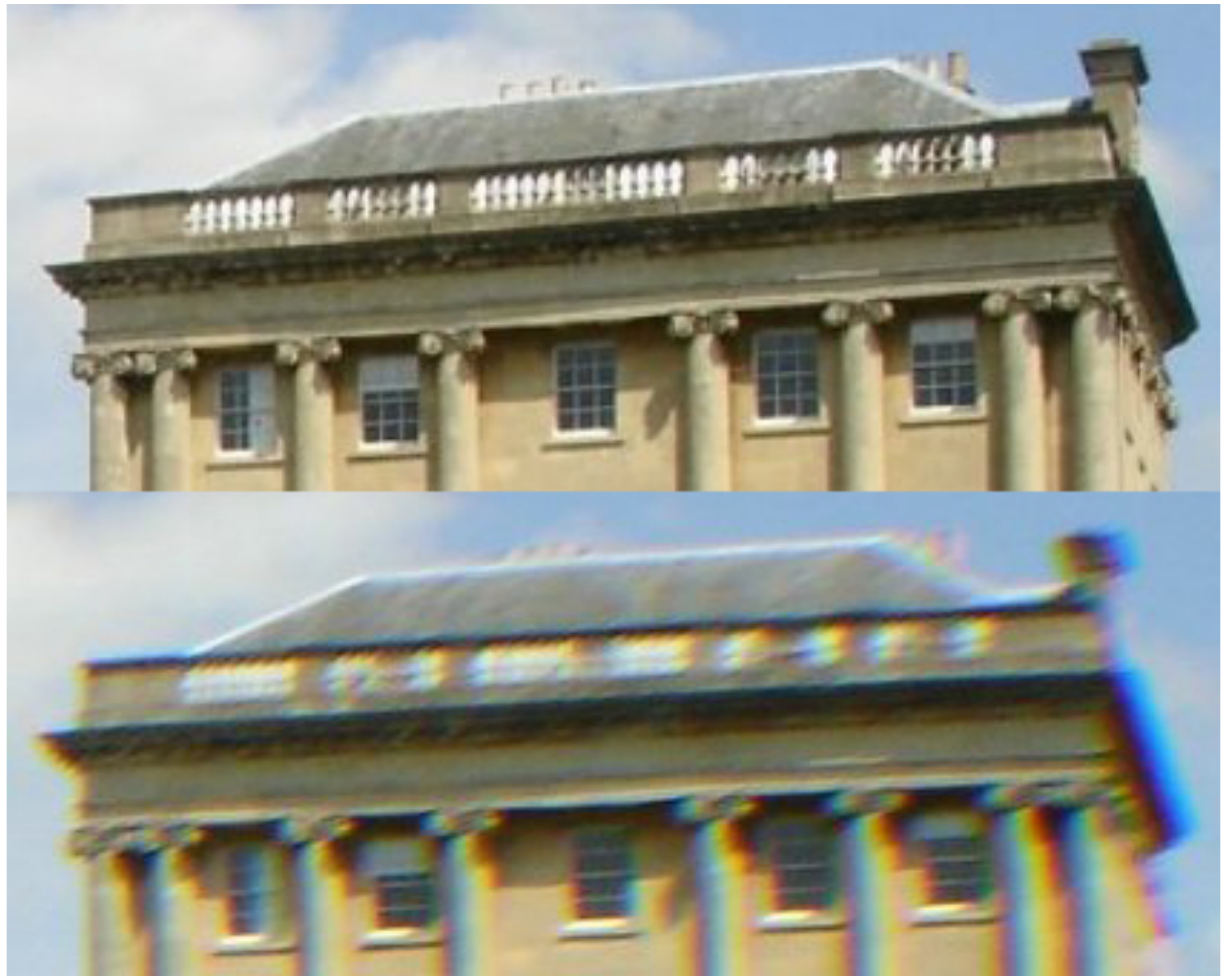
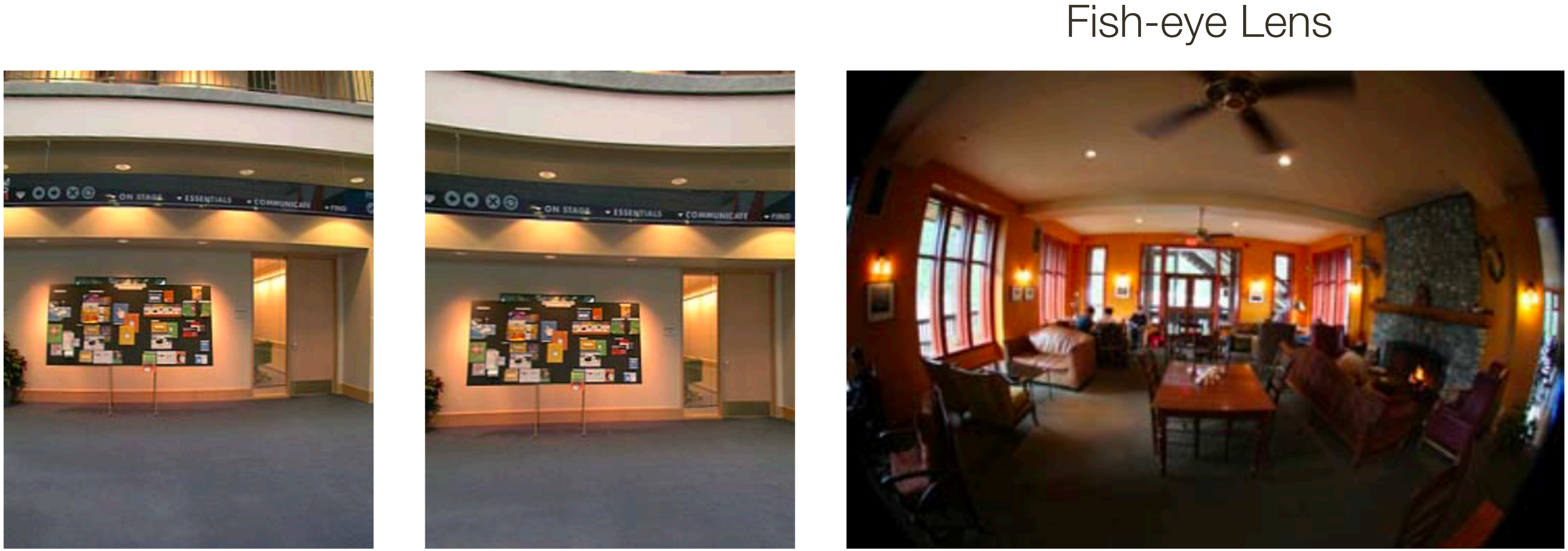


Image Credit: Trevor Darrell

Lens **Distortion**


Fish-eye Lens

Szeliski (1st ed.) Figure 2.13

Lines in the world are no longer lines on the image, they are curves!

Sample Question: Cameras and Lenses

True of **false**: Snell’s Law describes how much light is reflected and how much passes through the boundary between two materials.

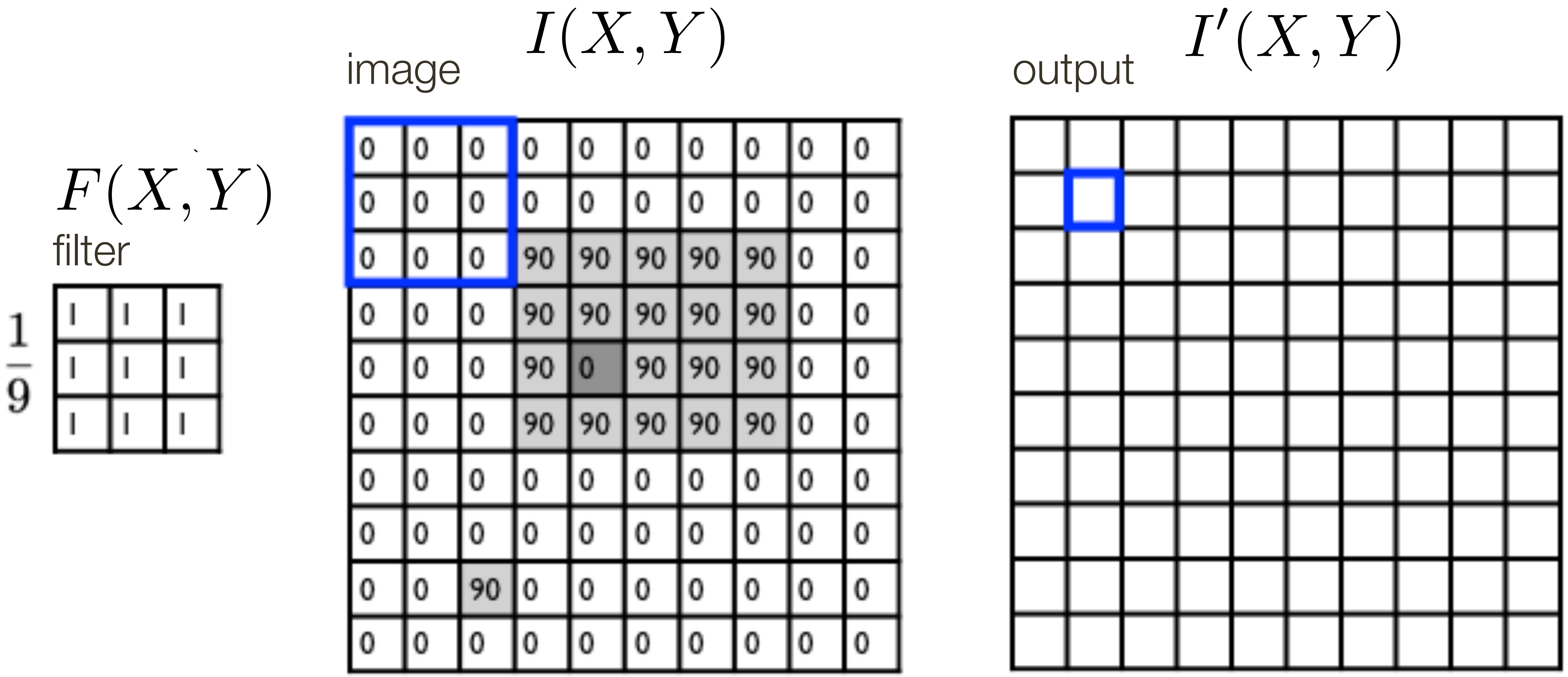
Linear **Filters**

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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 Filter **Example**



$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

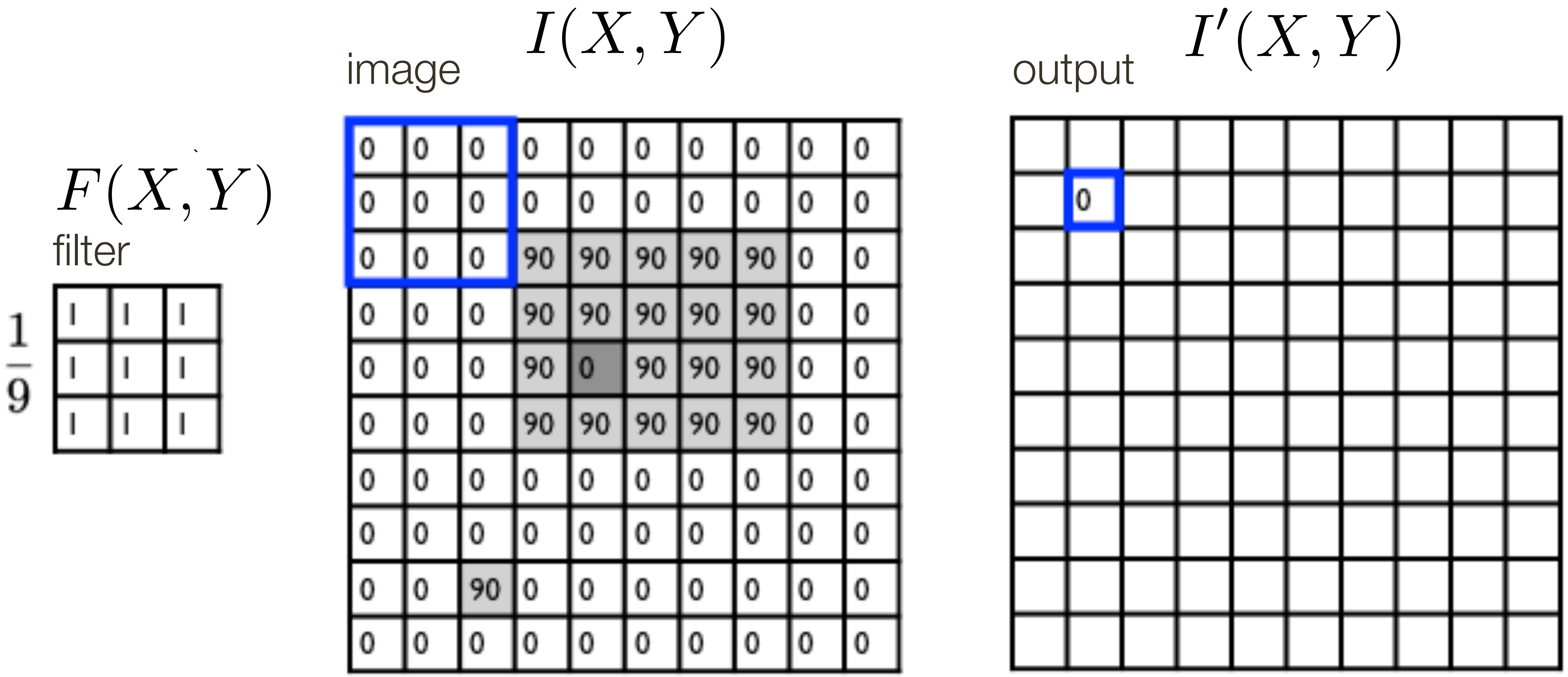
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

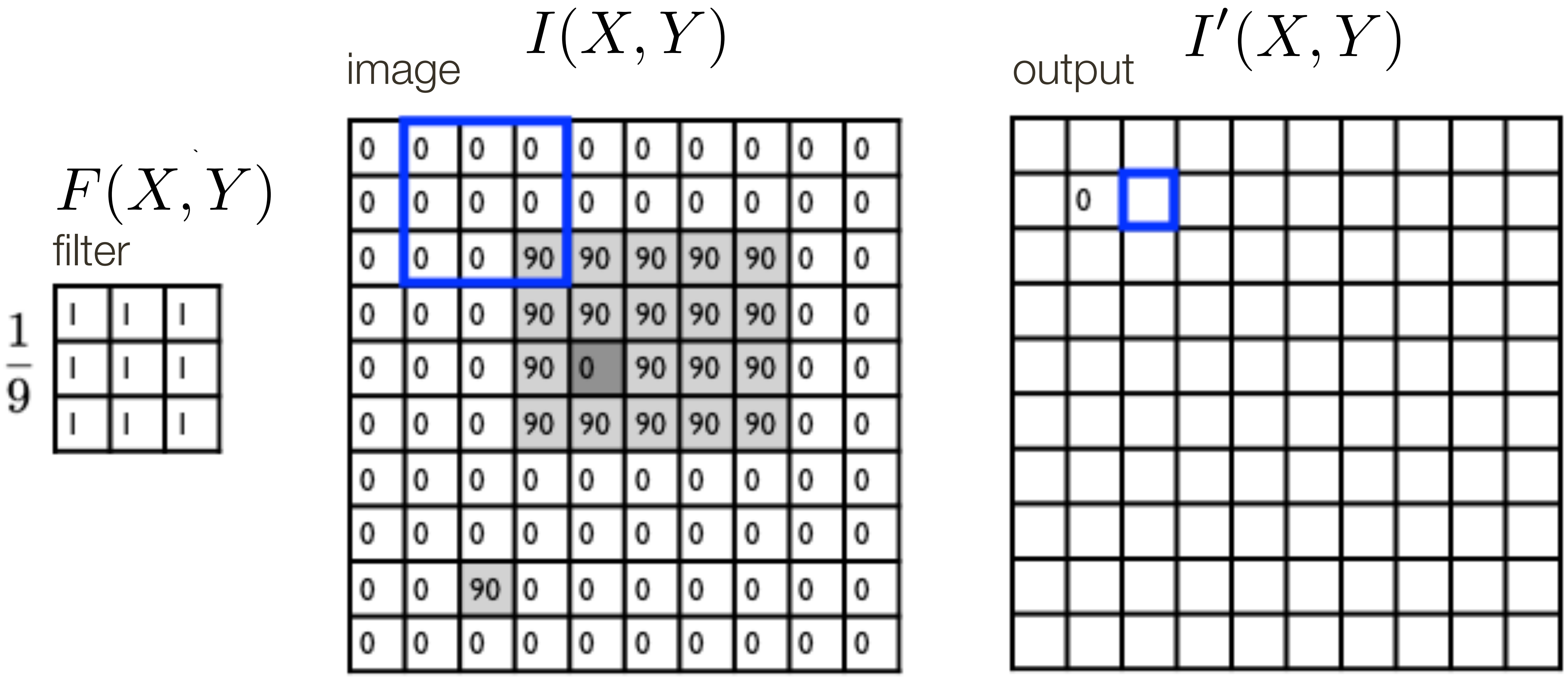
output $I'(X,Y)$

	0								

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

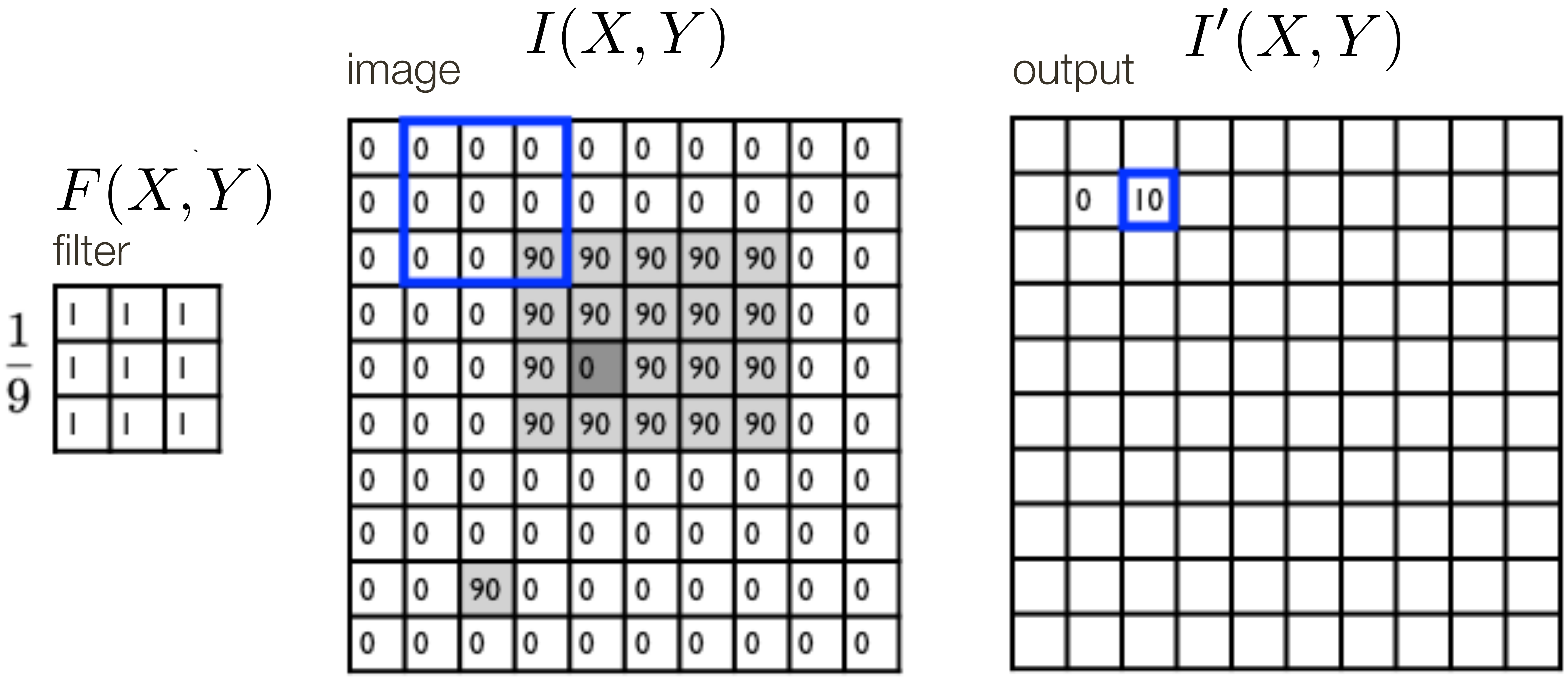
output $I'(X,Y)$

	0								

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

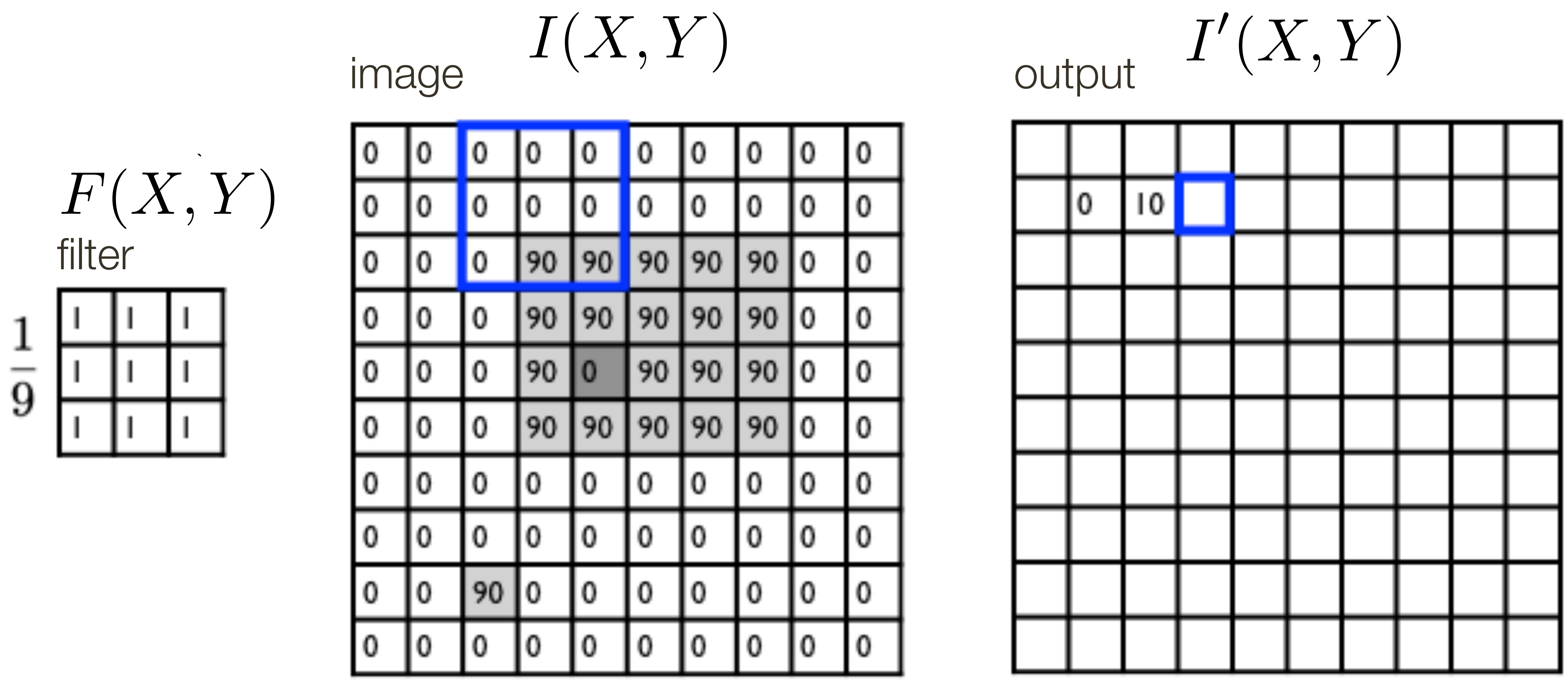
output $I'(X,Y)$

	0	10							

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

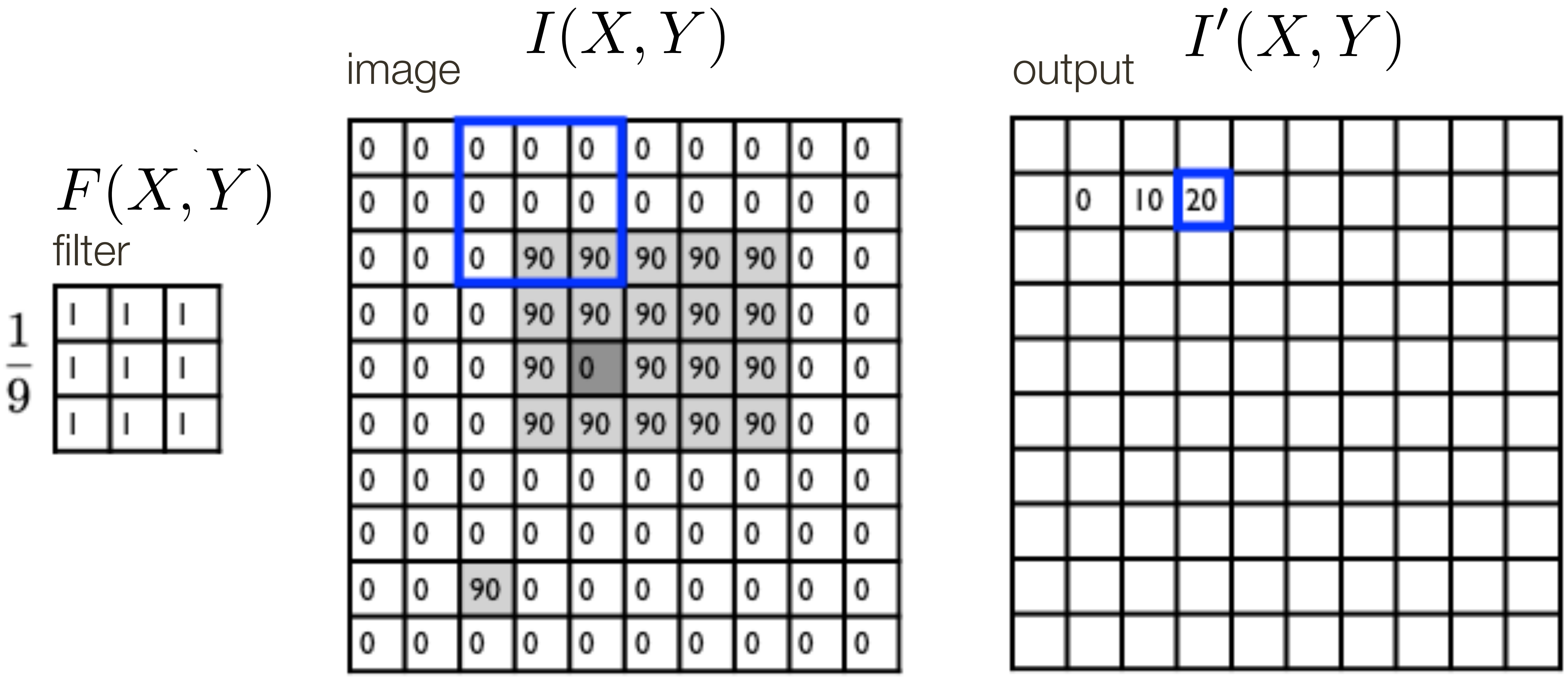
output $I'(X,Y)$

	0	10							

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**



$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

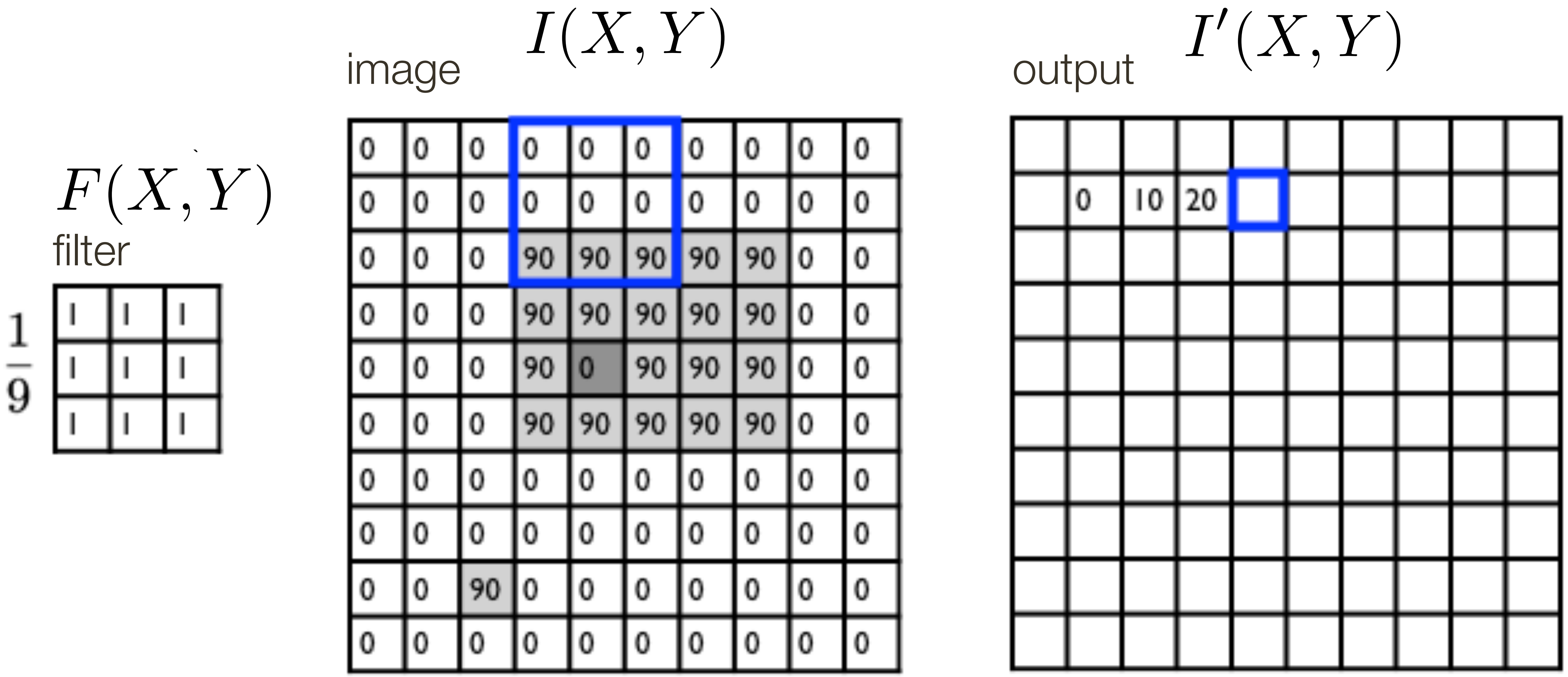
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20						

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

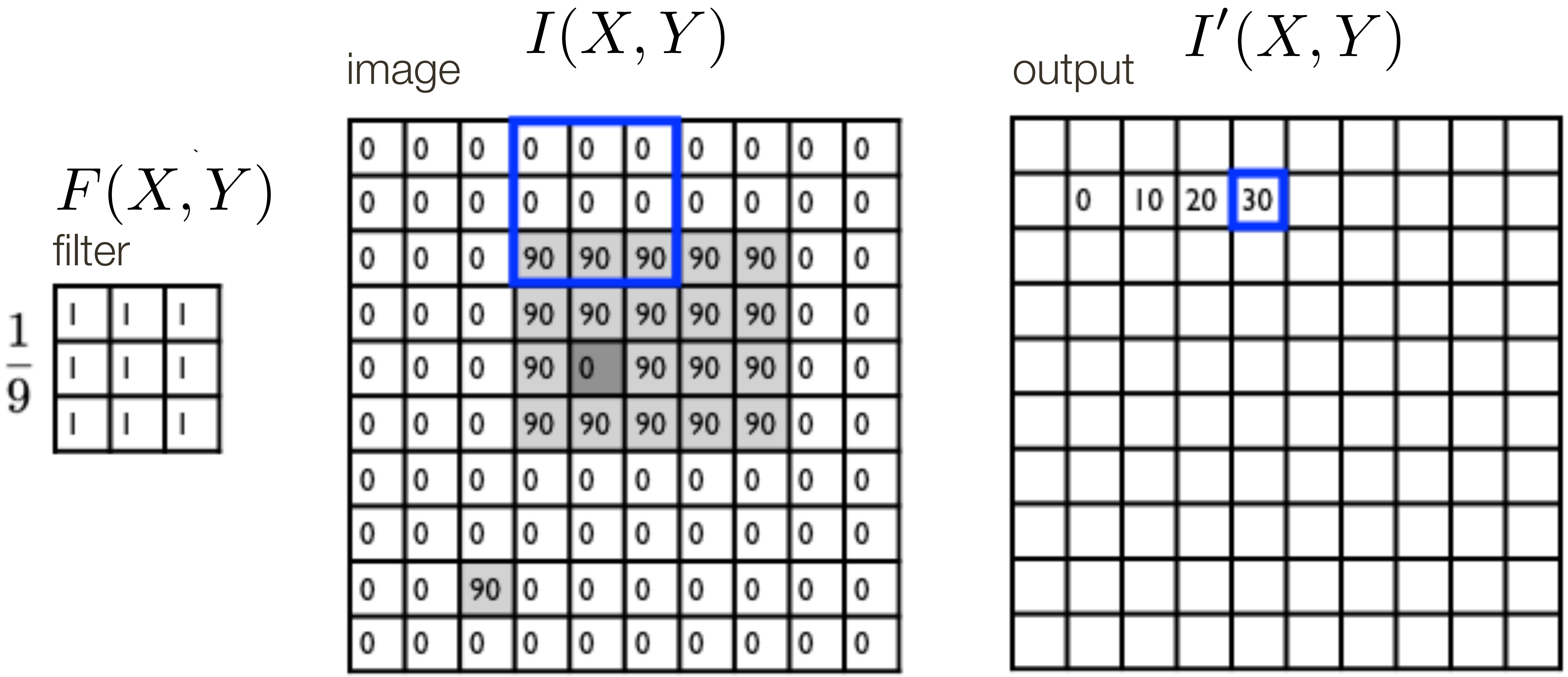
output $I'(X,Y)$

	0	10	20						

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

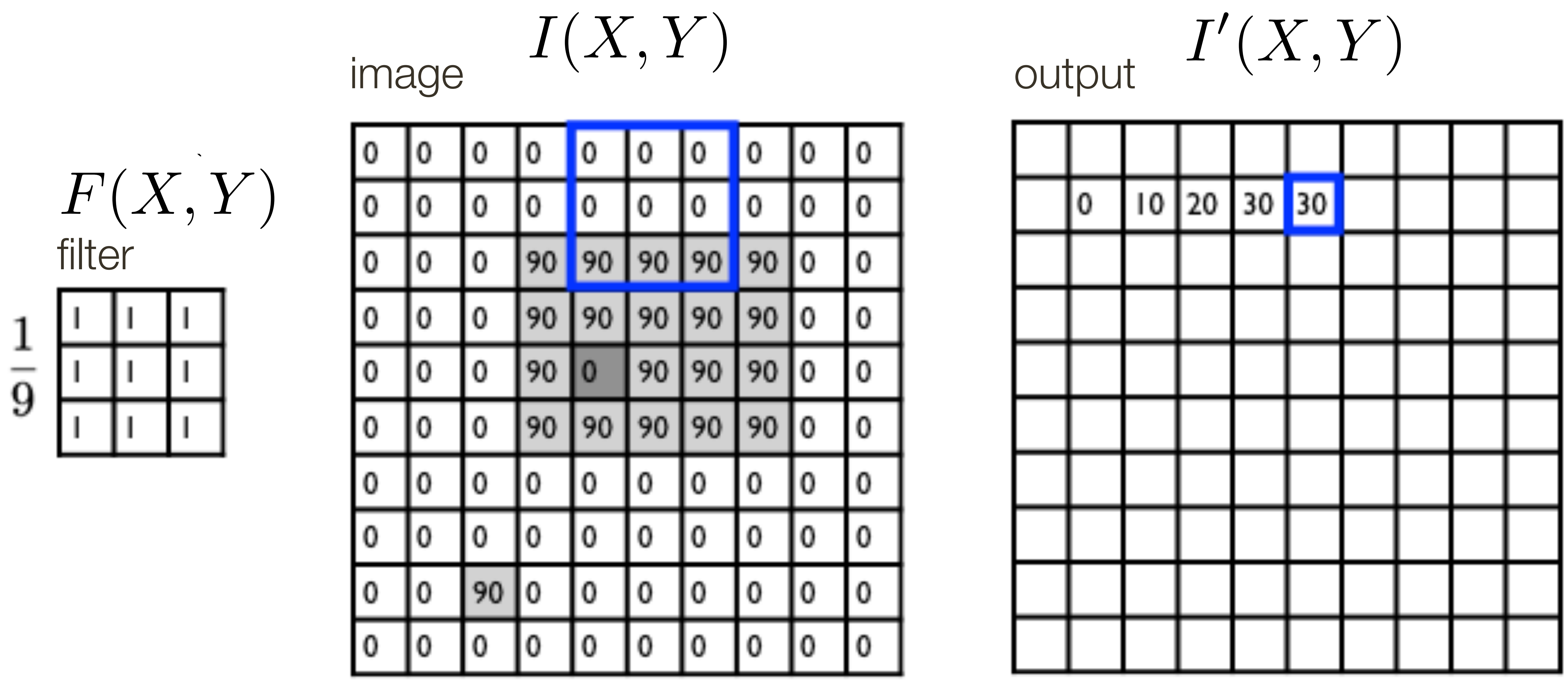
output $I'(X,Y)$

	0	10	20	30					

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**



$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

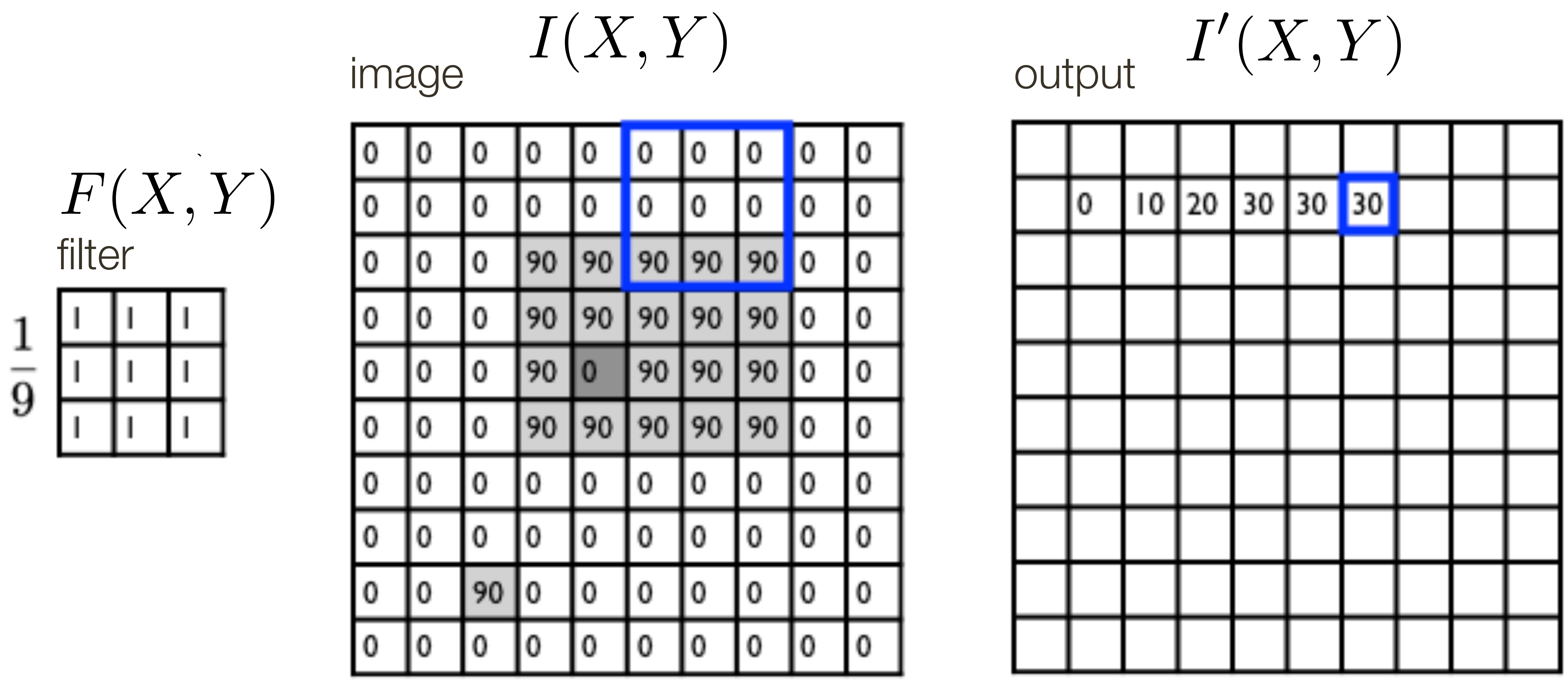
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20	30	30				

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k}\underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**



$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

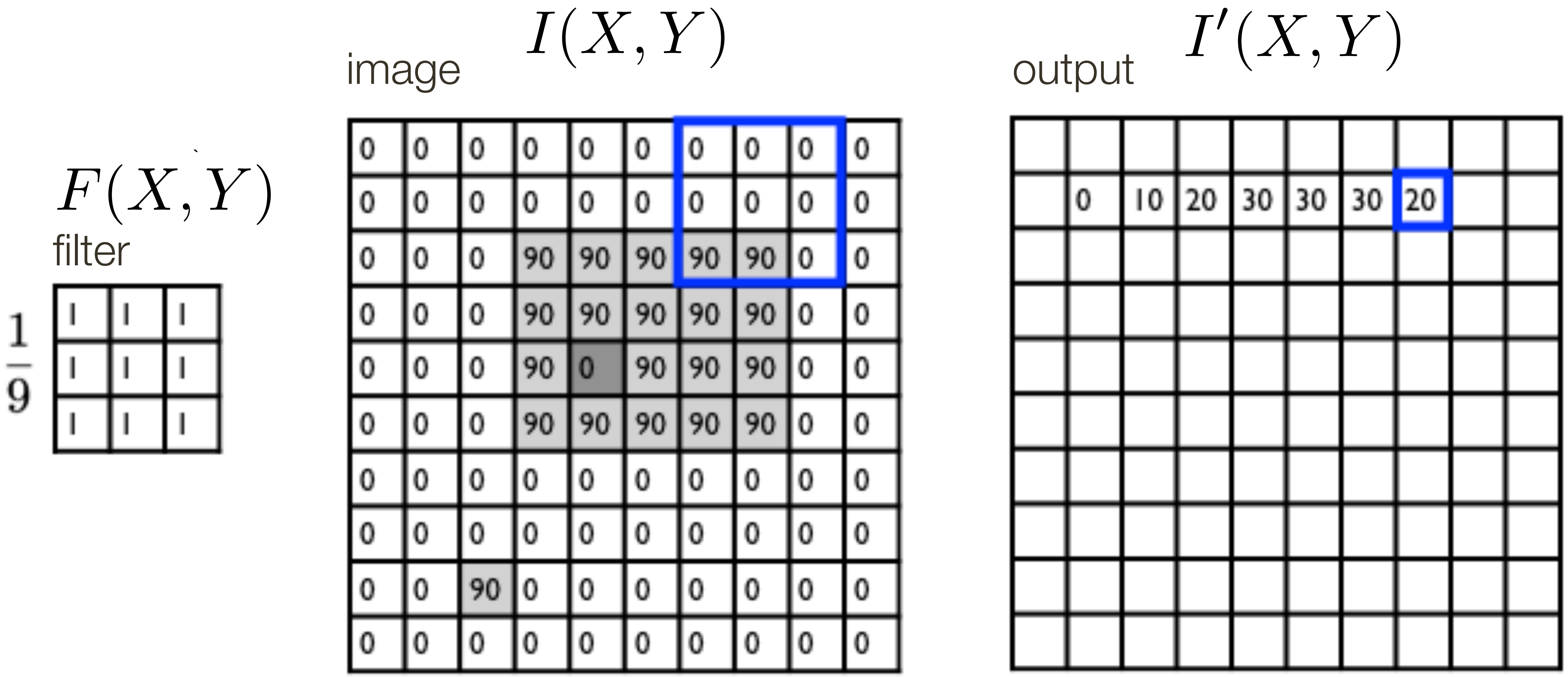
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20	30	30	30			

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

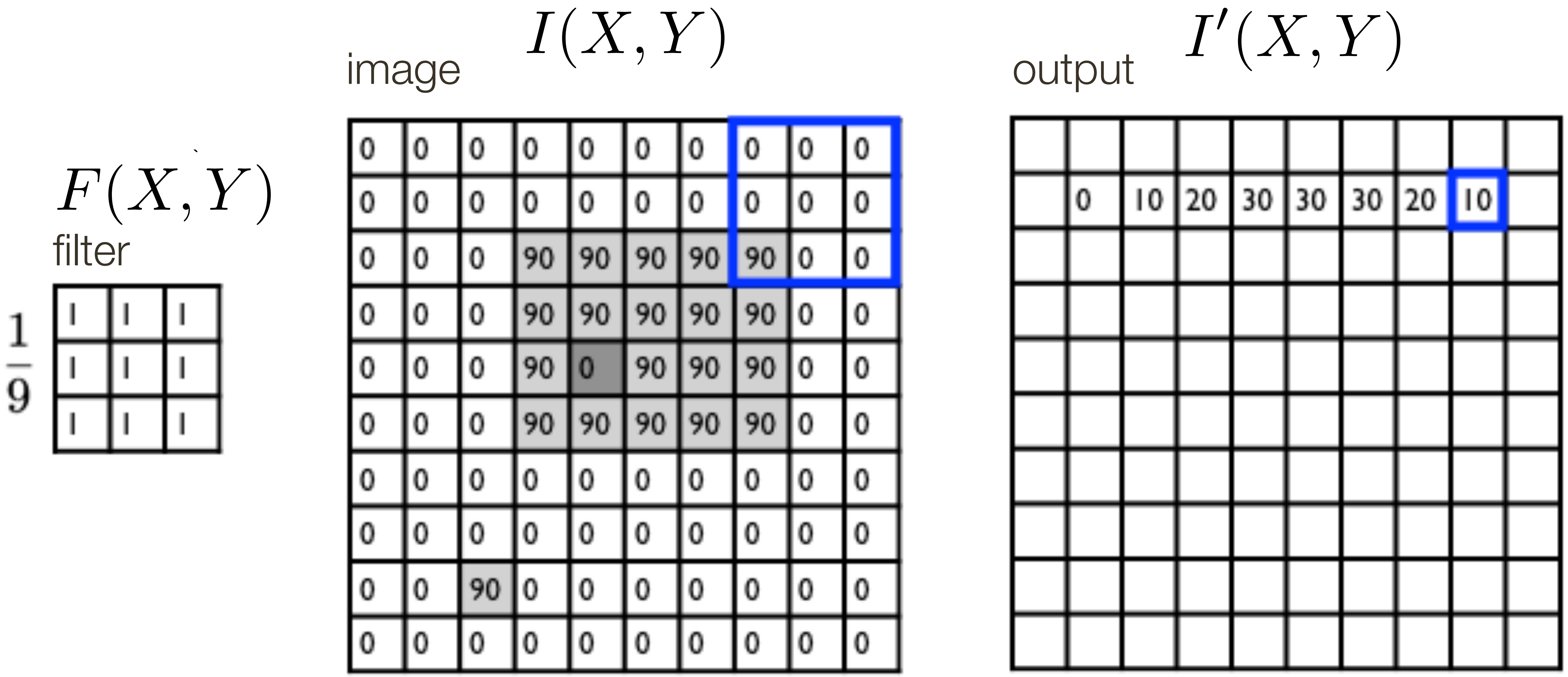
output $I'(X,Y)$

	0	10	20	30	30	30	20		

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**



$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

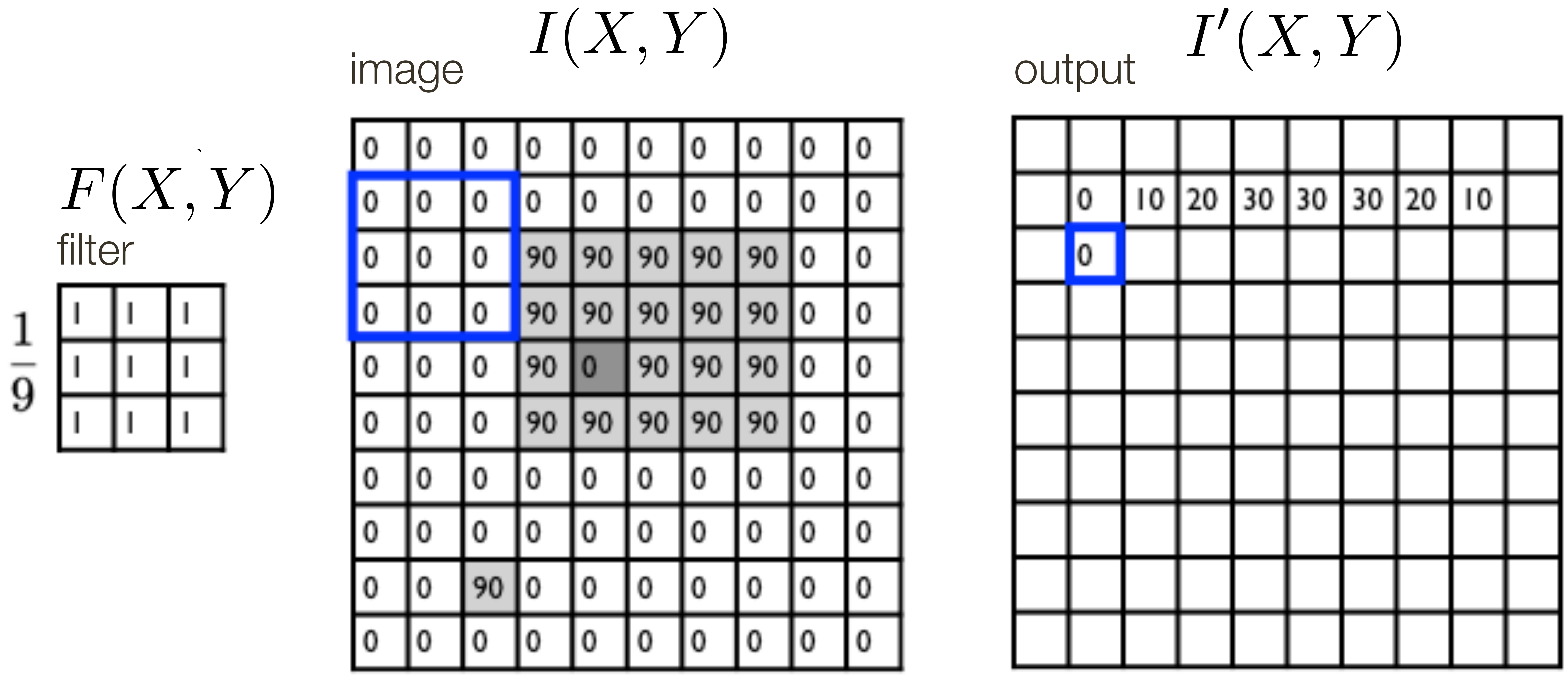
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20	30	30	30	20	10	

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**



1	1	1
1	1	1
1	1	1

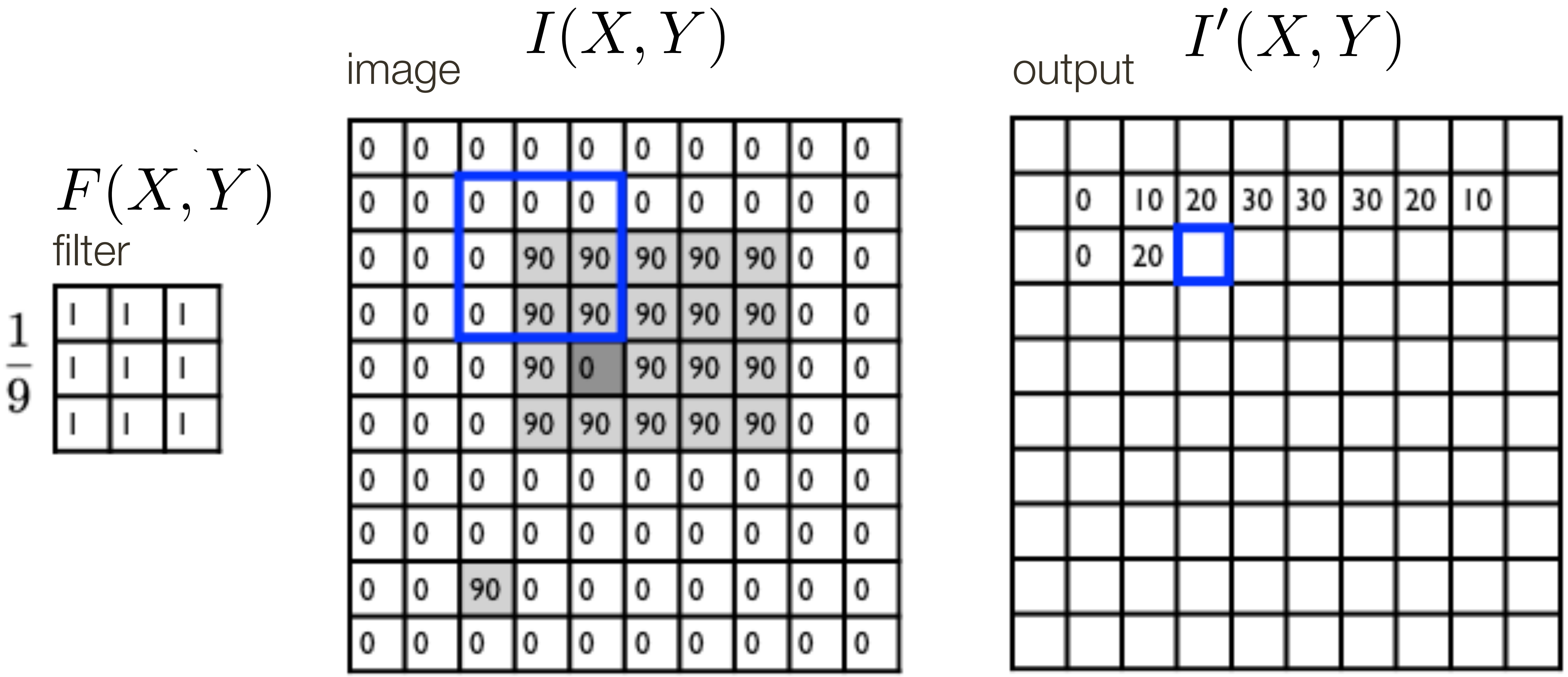
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20	30	30	30	20	10	
	0								

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

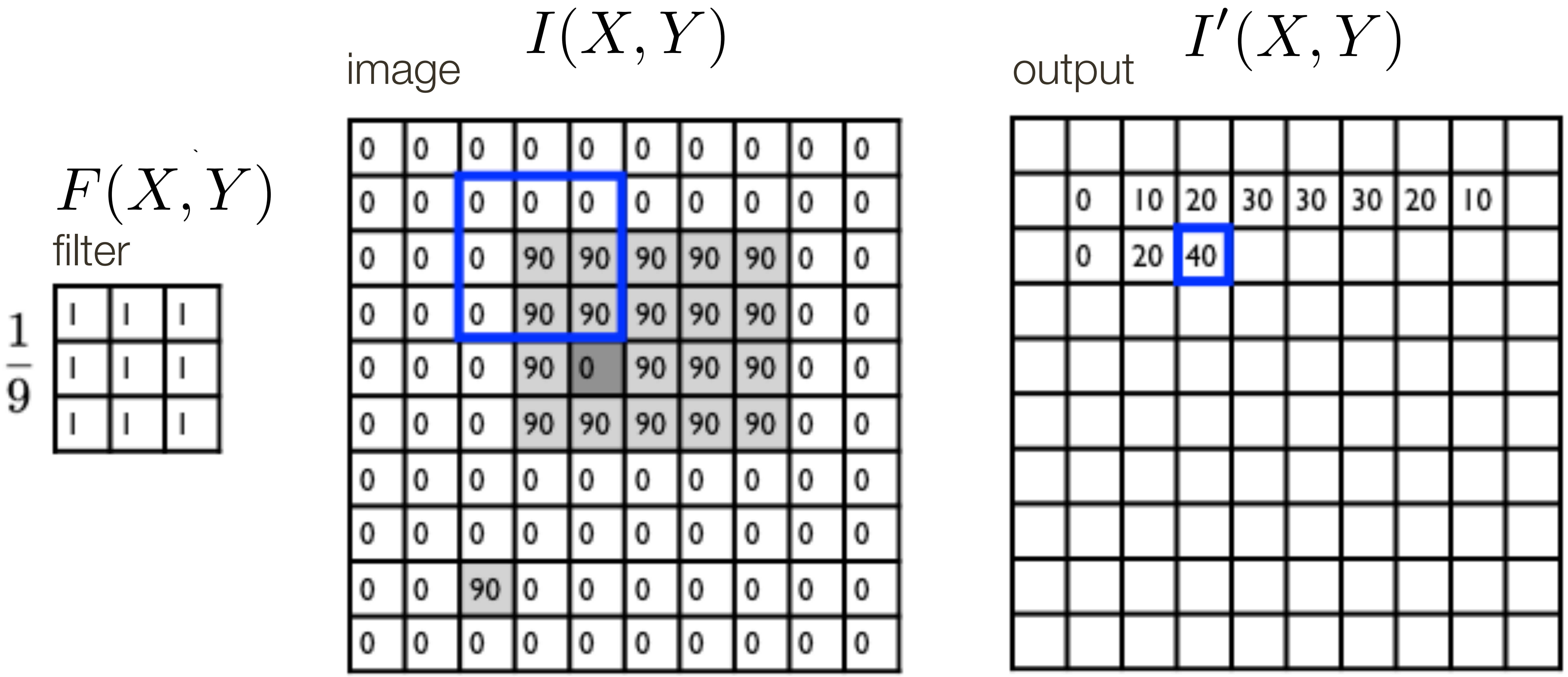
output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20							

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

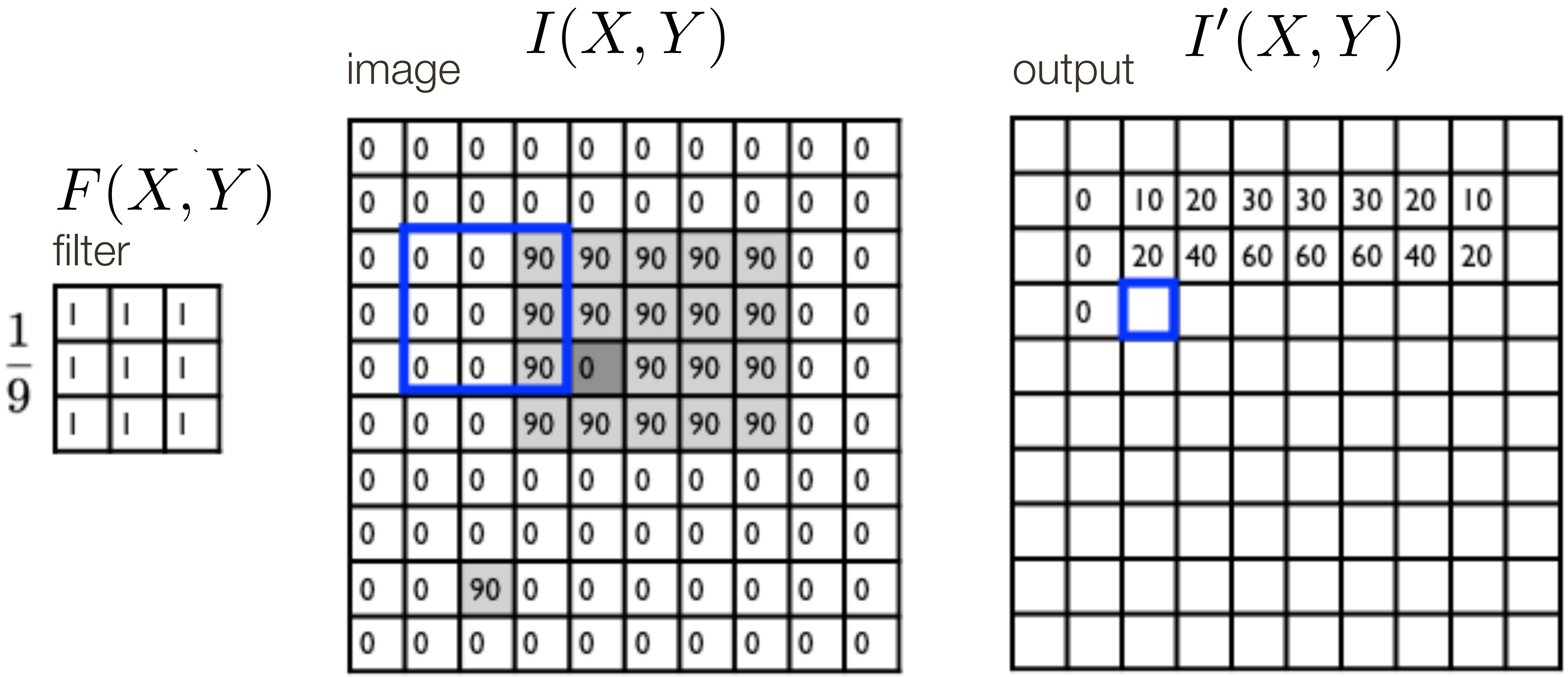
output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40						

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

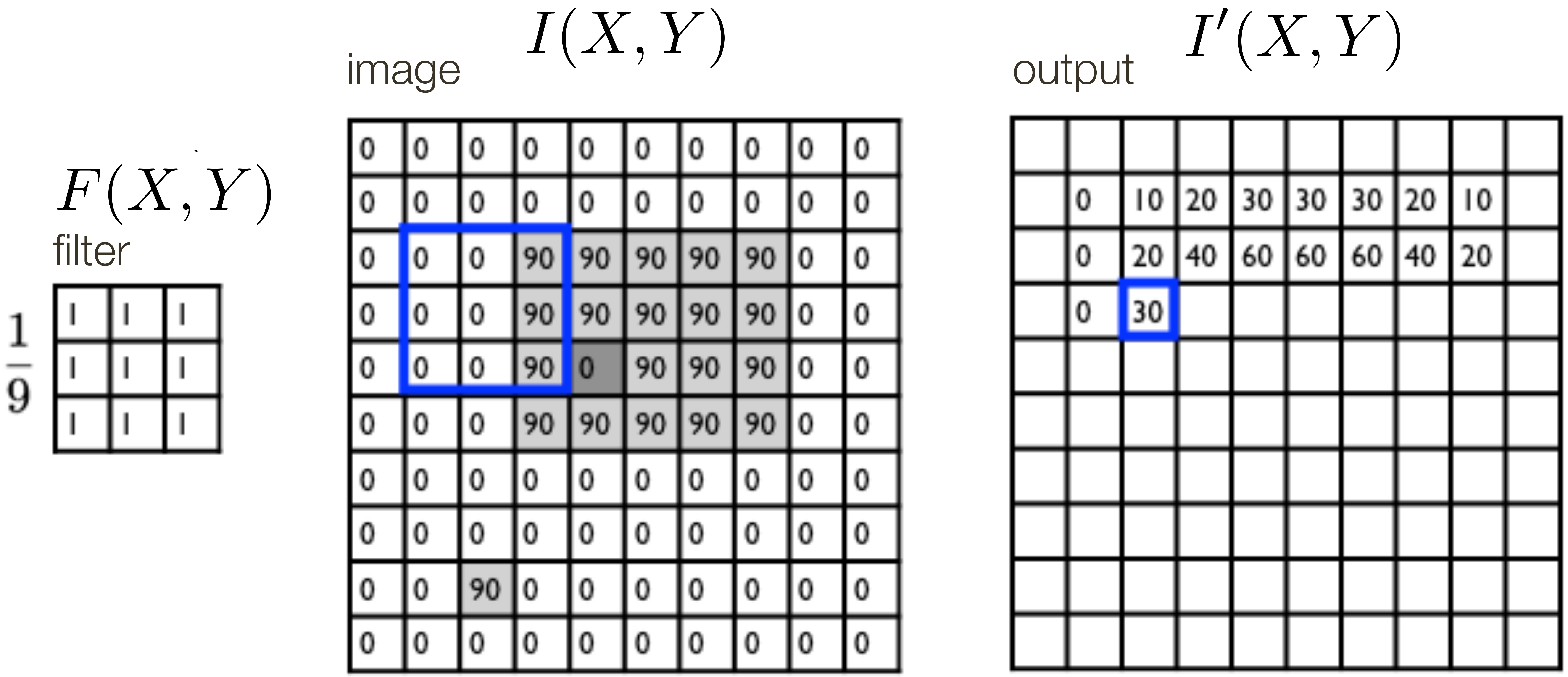
output $I'(X,Y)$

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

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

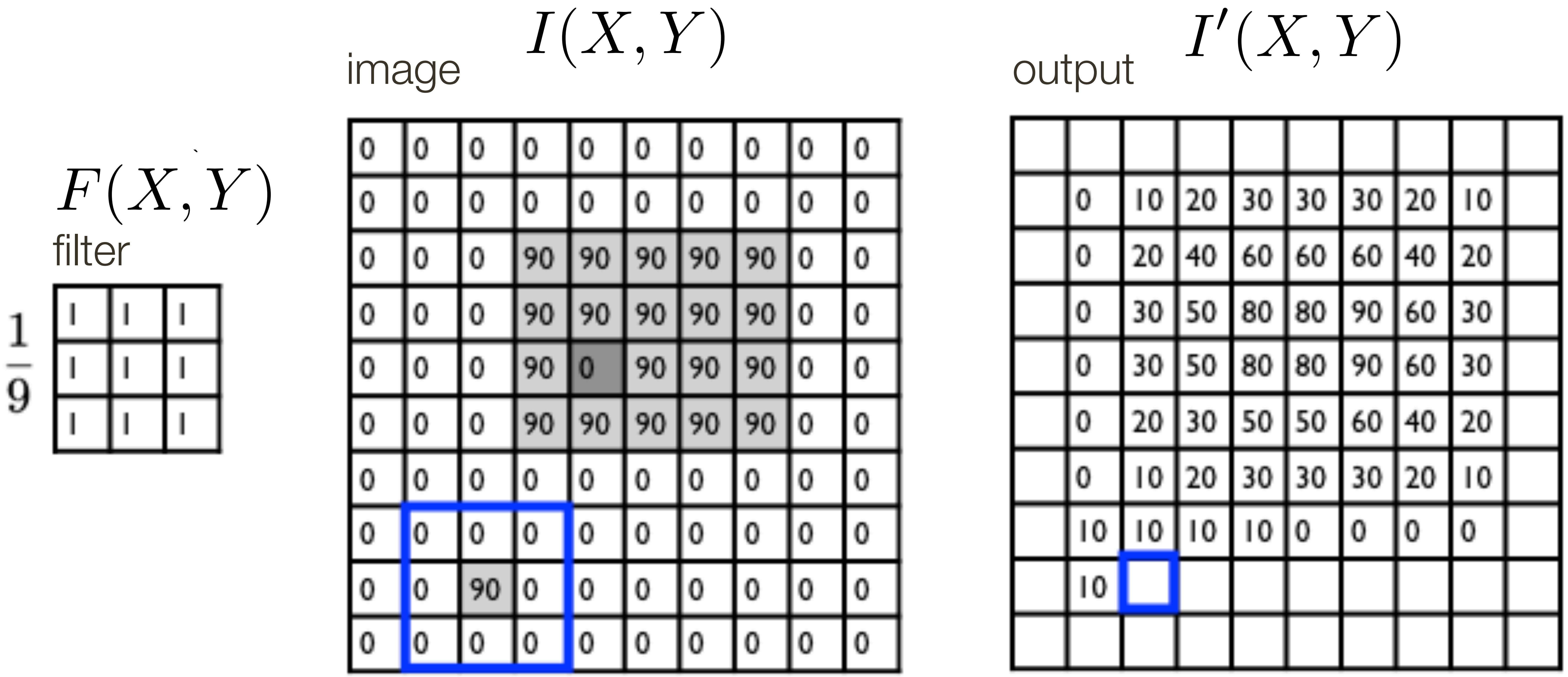
output $I'(X,Y)$

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

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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

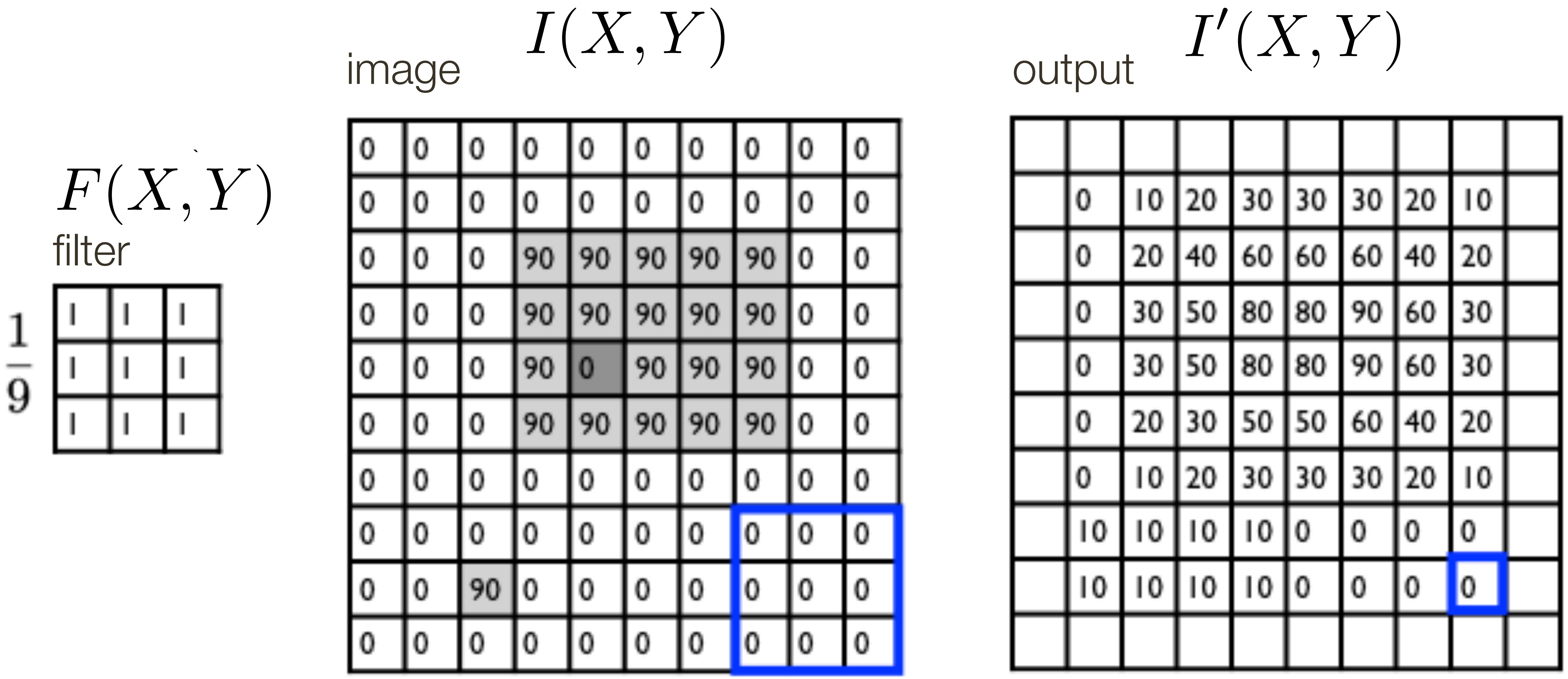
output $I'(X,Y)$

	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								

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

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	

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}} \underbrace{I(X+i, Y+j)}_{\text{image (signal)}}$$

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

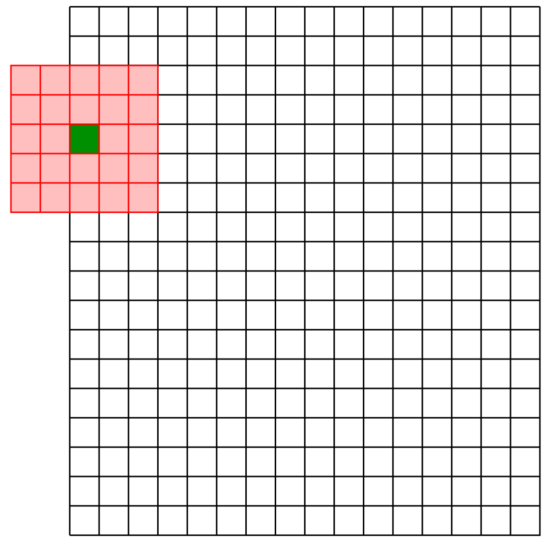
output $I'(X,Y)$

	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	

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(I,J)}_{\text{filter}}\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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**

— The **correlation** of $F(X,Y)$ and $I(X,Y)$ is:

$$\underbrace{\boxed{I'(X,Y)}}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{\boxed{F(I,J)}}_{\text{filter}}\underbrace{\boxed{I(X+i,Y+j)}}_{\text{image (signal)}}$$

— **Visual interpretation**: Superimpose the filter F on the image I at (X,Y), perform an element-wise multiply, and sum up the values

— **Convolution** is like **correlation** except filter “flipped”

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

Linear System: **Characterization** Theorem

Any linear, shift invariant operation can be expressed as a convolution

Linear System: **Characterization** Theorem

Any linear, shift invariant operation can be expressed as a convolution

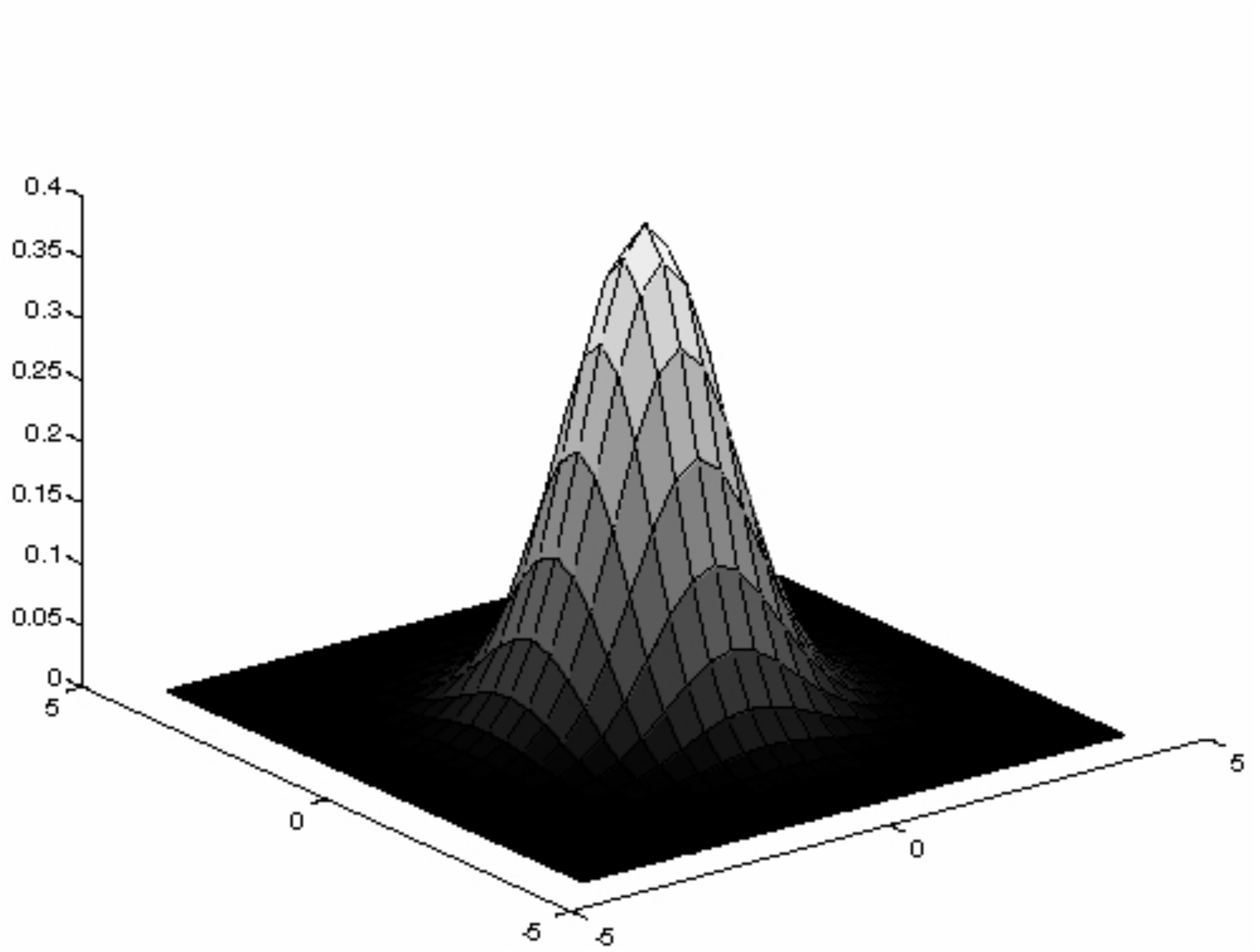
(`if and only if’ result)

Smoothing with a Gaussian

Idea: Weight contributions of pixels by spatial proximity (nearness)

2D **Gaussian** (continuous case):

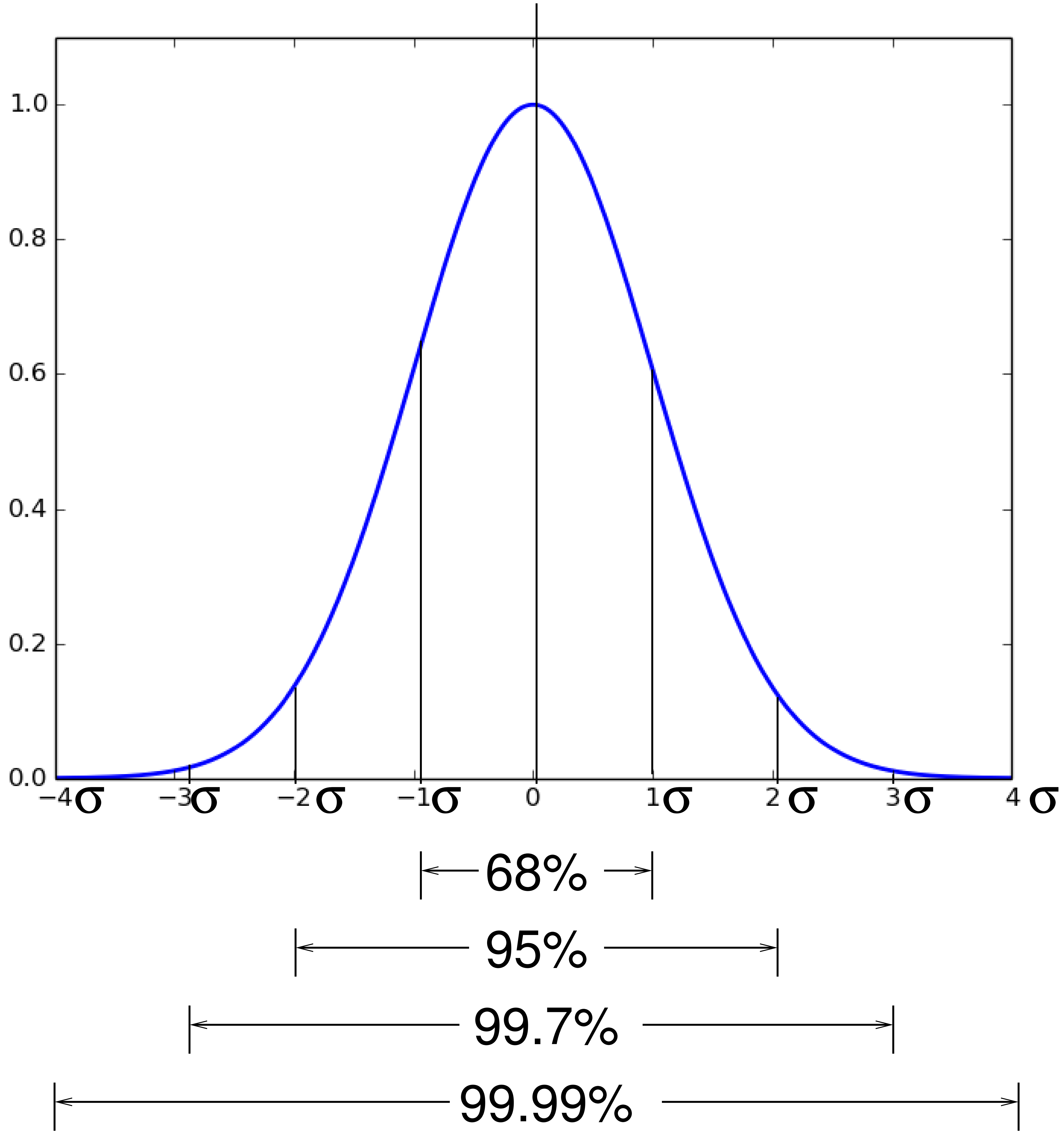
$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$



Forsyth & Ponce (2nd ed.)

Figure 4.2

Gaussian: Area Under the Curve



Efficient Implementation: **Separability**

A 2D function of x and y is **separable** if it can be written as the product of two functions, one a function only of x and the other a function only of y

Both the 2D box filter and the 2D Gaussian filter are separable

Both can be implemented as two 1D convolutions:

— First, convolve each row with a 1D filter

— Then, convolve each column with a 1D filter

— Aside: or vice versa

The **2D Gaussian** is the only (non trivial) 2D function that is both separable and rotationally invariant.

Linear Filters: Additional Properties

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

— Convolution is **associative**. That is,

$$G \otimes (F \otimes I(X,Y)) = (G \otimes F) \otimes I(X,Y)$$

— Convolution is **symmetric**. That is,

$$(G \otimes F) \otimes I(X,Y) = (G \otimes F) \otimes I(X,Y)$$

Convolving $I(X,Y)$ with filter F and then convolving the result with filter G can be achieved in single step, namely convolving $I(X,Y)$ with filter $G \otimes F = F \otimes G$

Bilateral Filter

An edge-preserving non-linear filter

Like a Gaussian filter:

— The filter weights depend on spatial distance from the center pixel
— Pixels nearby (in space) should have greater influence than pixels far away

Unlike a Gaussian filter:

— The filter weights also depend on range distance from the center pixel
— Pixels with similar brightness value should have greater influence than pixels with dissimilar brightness value

Bilateral Filter

Gaussian filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by:

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

(with appropriate normalization)

Bilateral filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by a product:

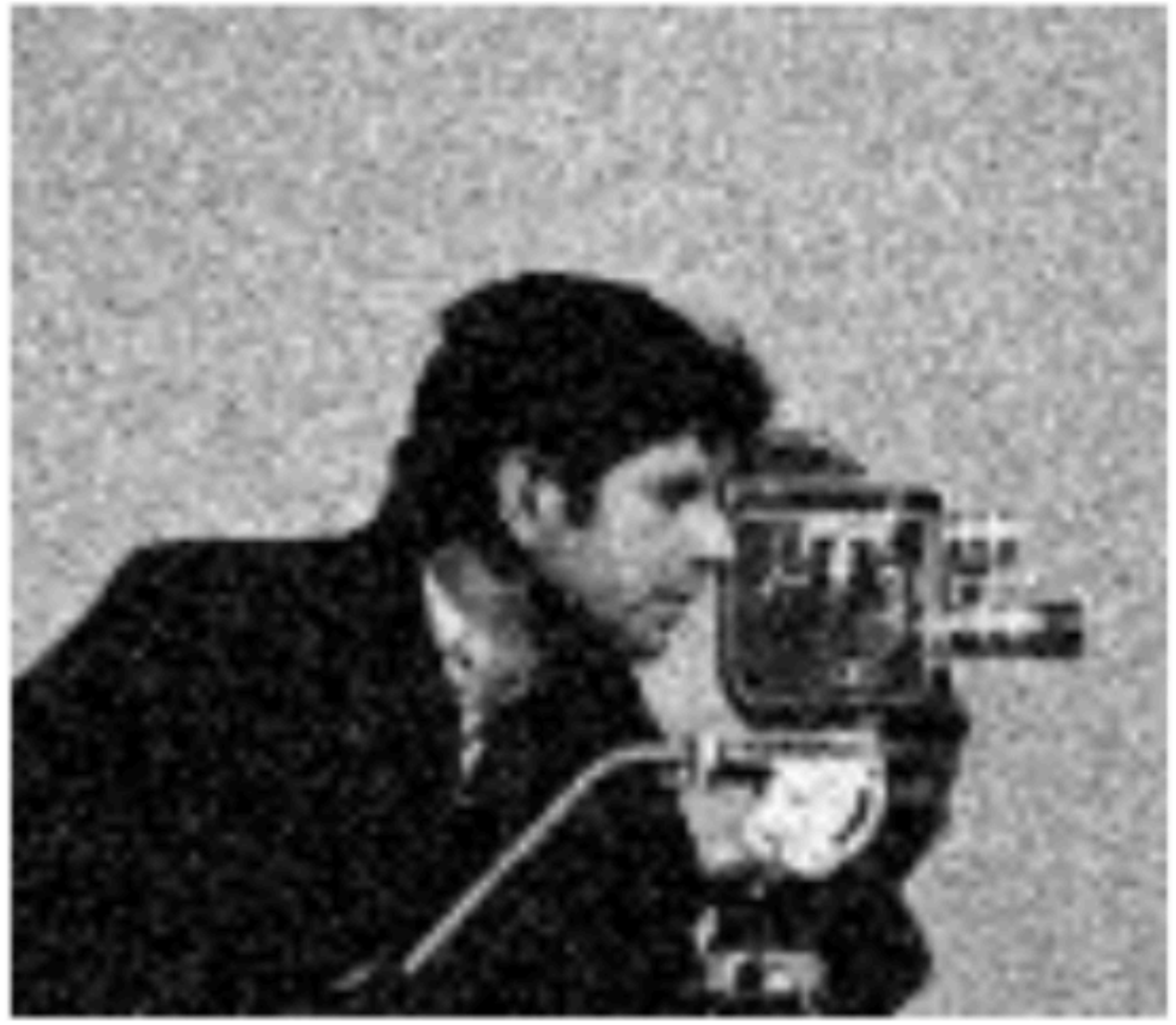
domain kernel

$$\exp^{-\frac{x^2+y^2}{2\sigma_d^2}} \exp^{-\frac{(I(X+x,Y+y)-I(X,Y))^2}{2\sigma_r^2}}$$

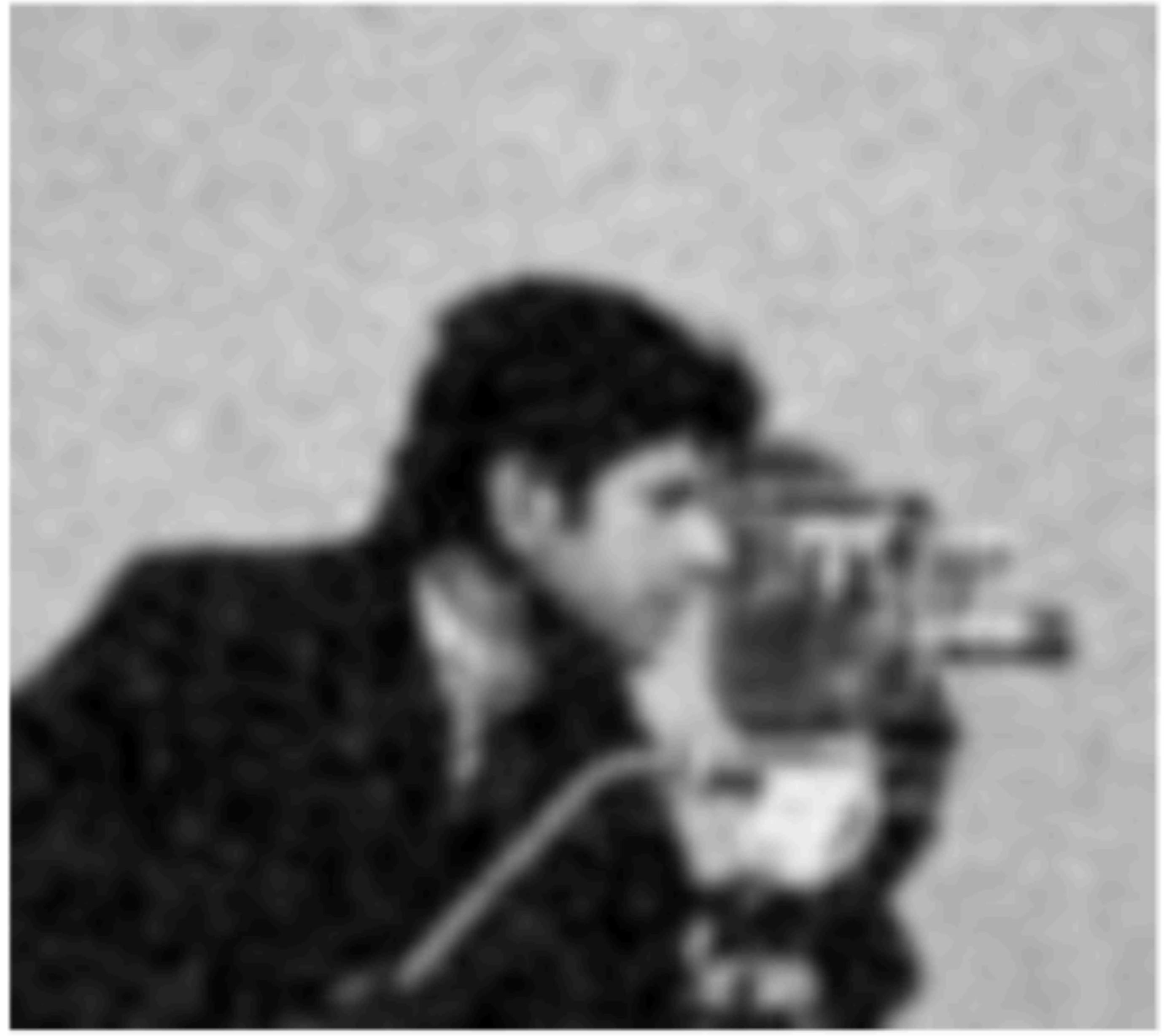
range kernel

(with appropriate normalization)

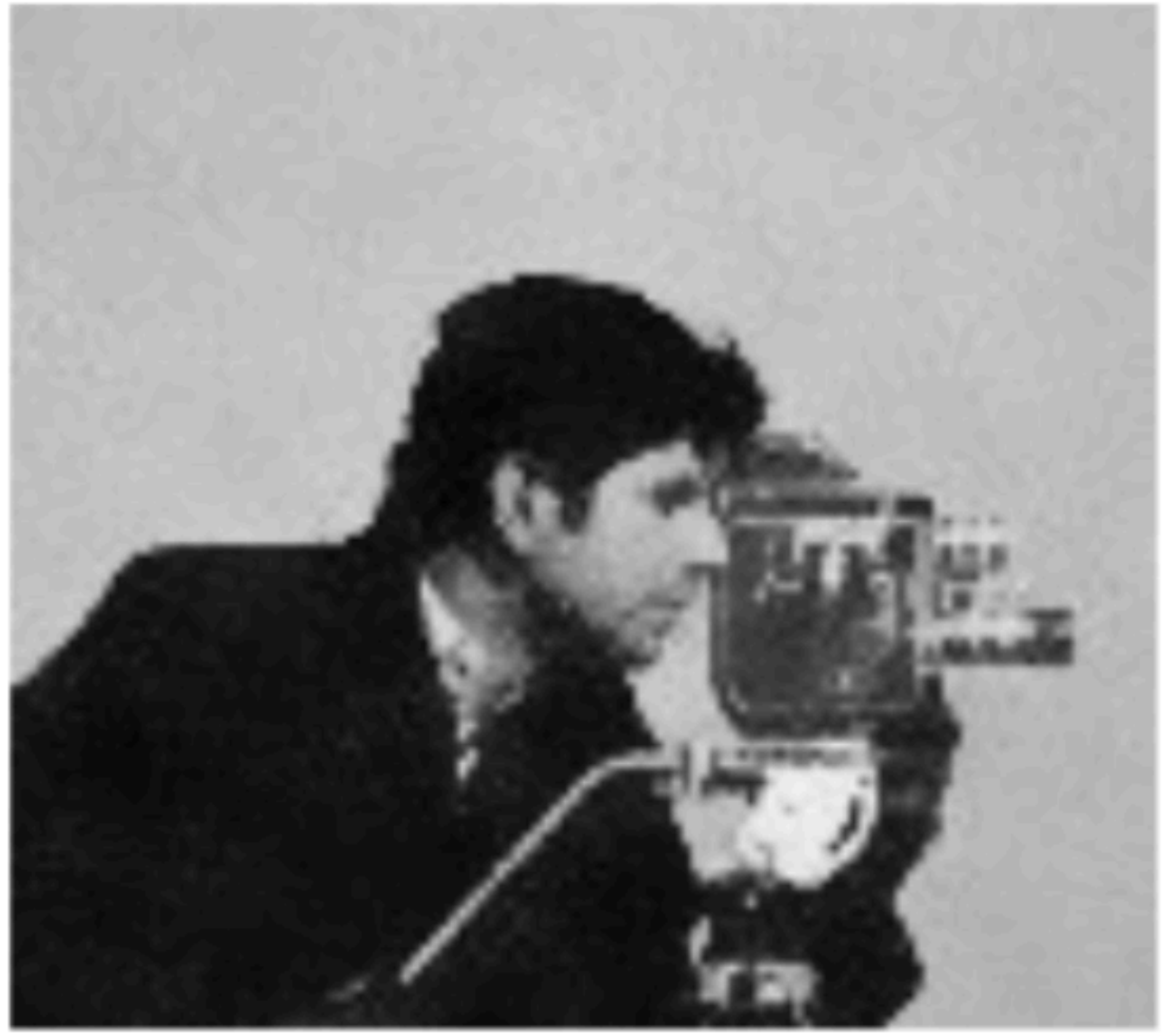
Bilateral Filter Application: Denoising



Noisy Image



Gaussian Filter



Bilateral Filter

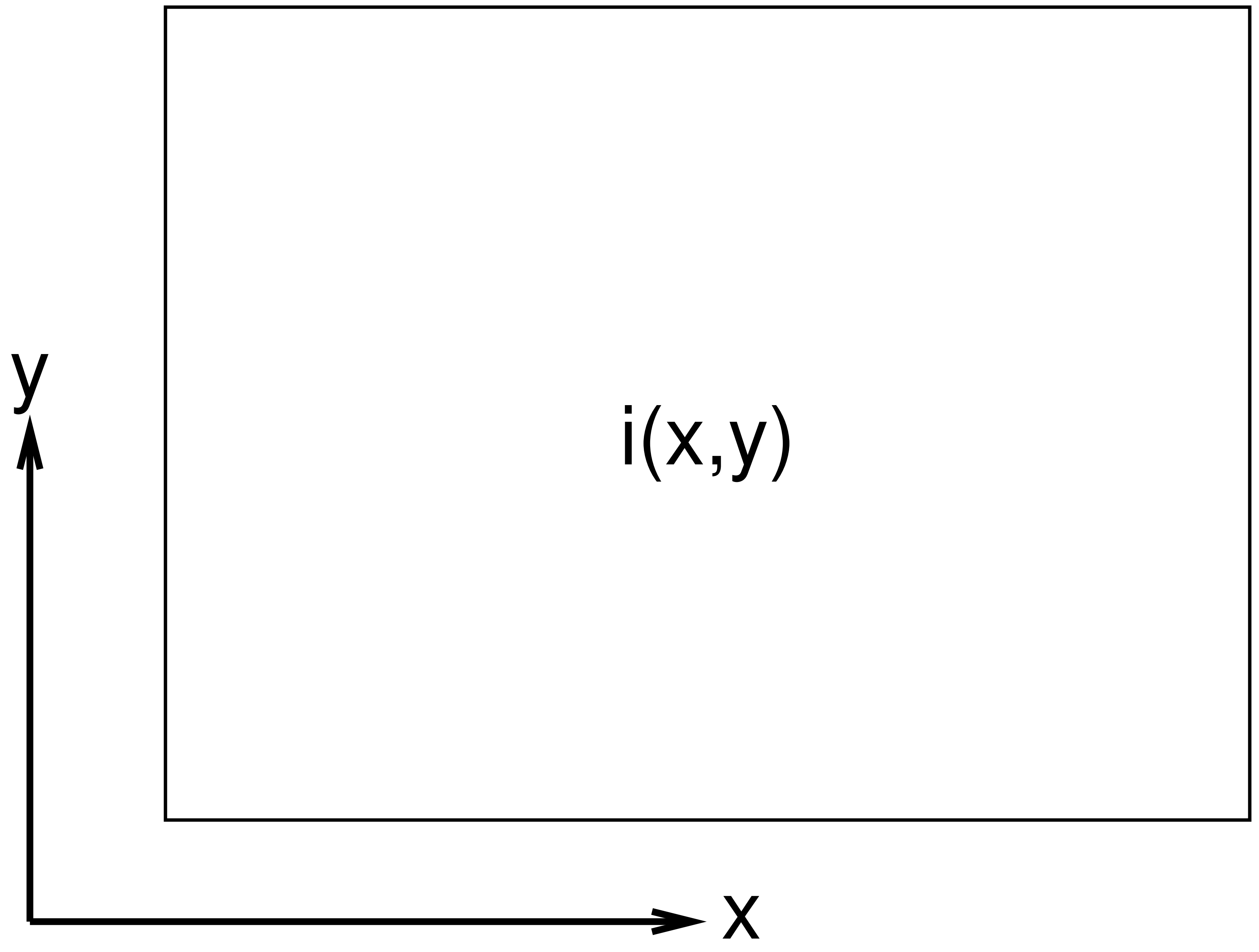
Slide Credit: Alexander Wong

Sample Question: Filters

What does the following 3 × 3 linear, shift invariant filter compute when applied to an image?

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Continuous Case

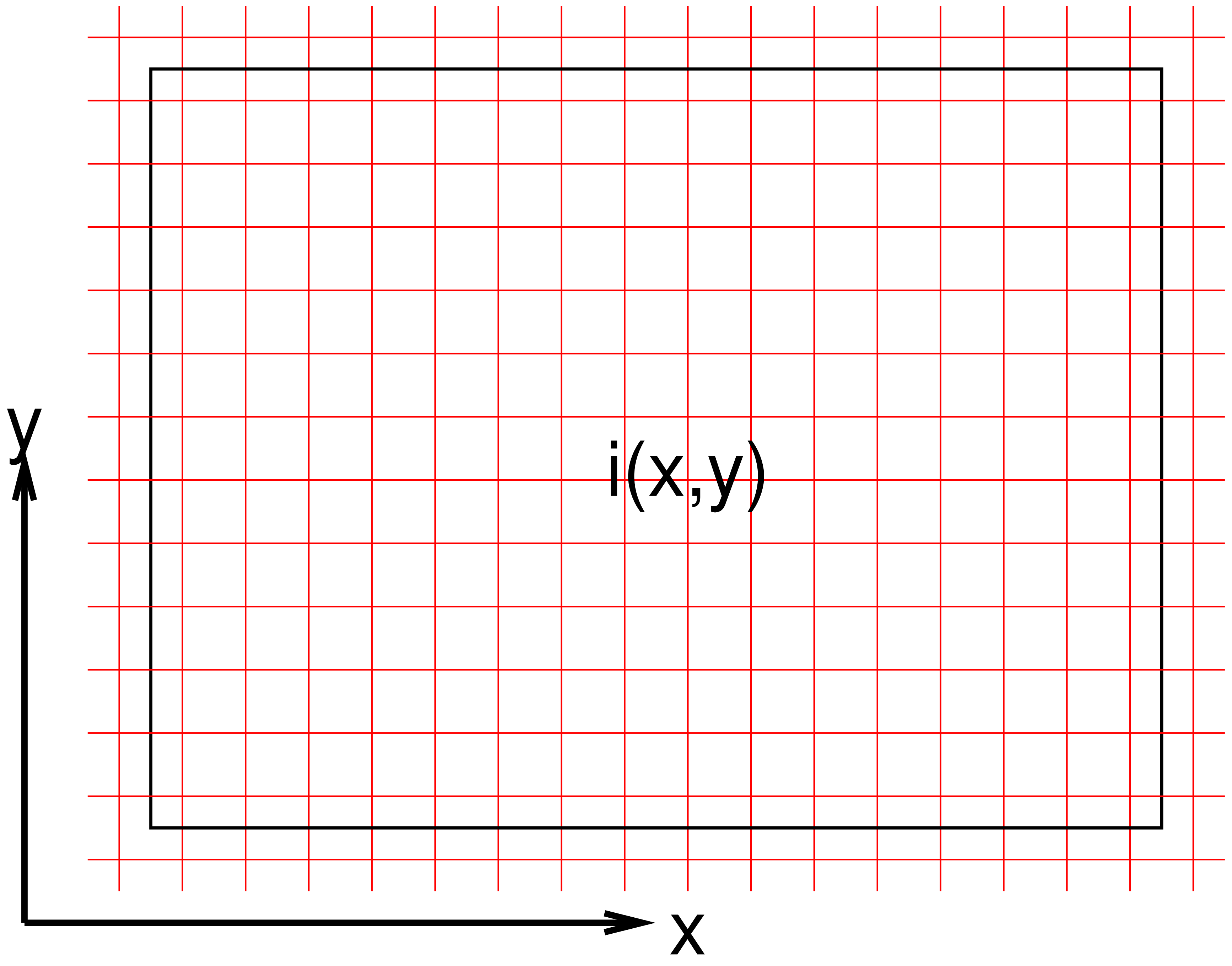


Denote the image as a function, $i(x, y)$, where x and y are spatial variables

Aside: The convention for this section is to use lower case letters for the continuous case and upper case letters for the discrete case

Discrete Case

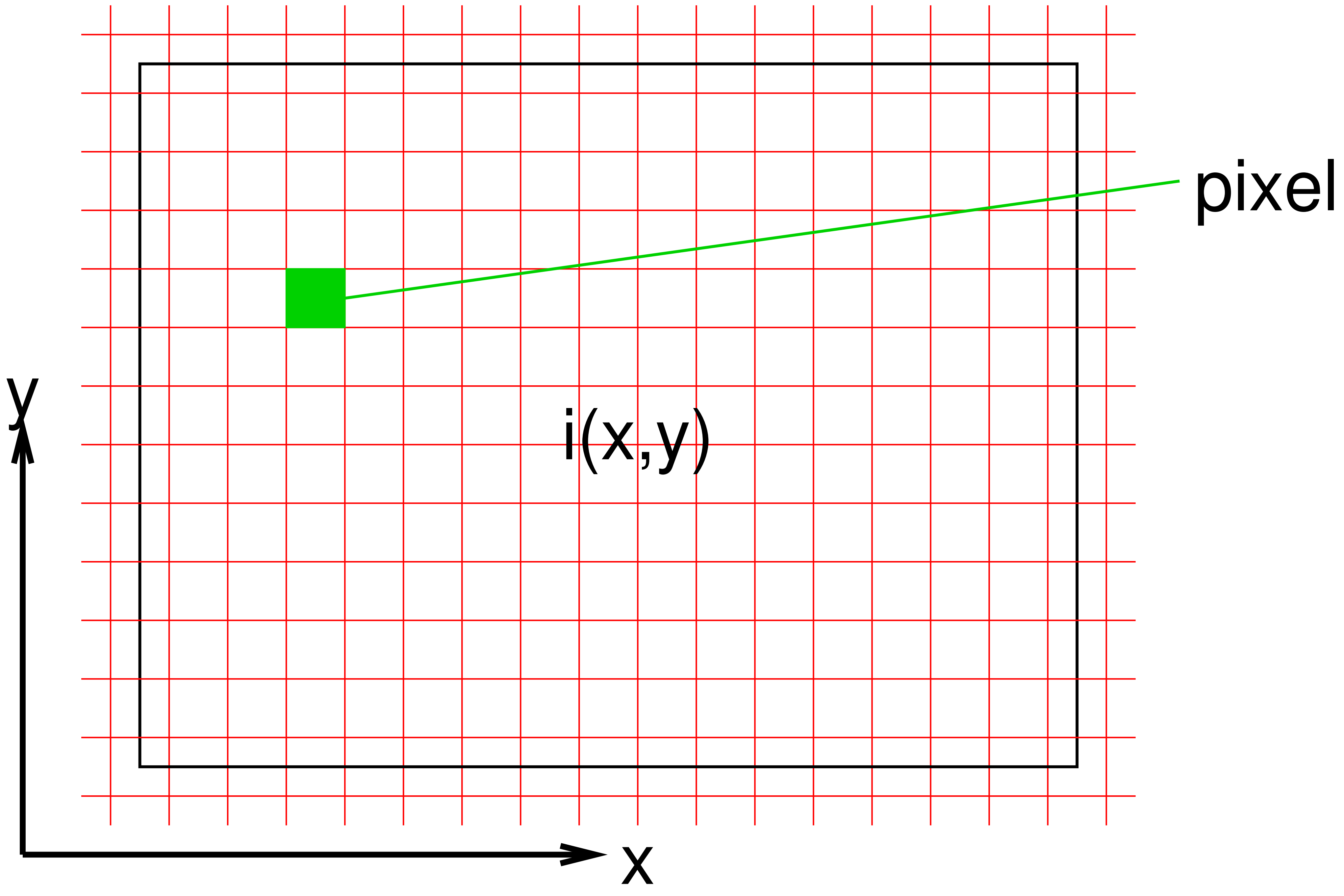
Idea: Superimpose (regular) grid on continuous image



Sample the underlying continuous image according to the **tessellation** imposed by the grid

Discrete Case

Each grid cell is called a picture element (**pixel**)

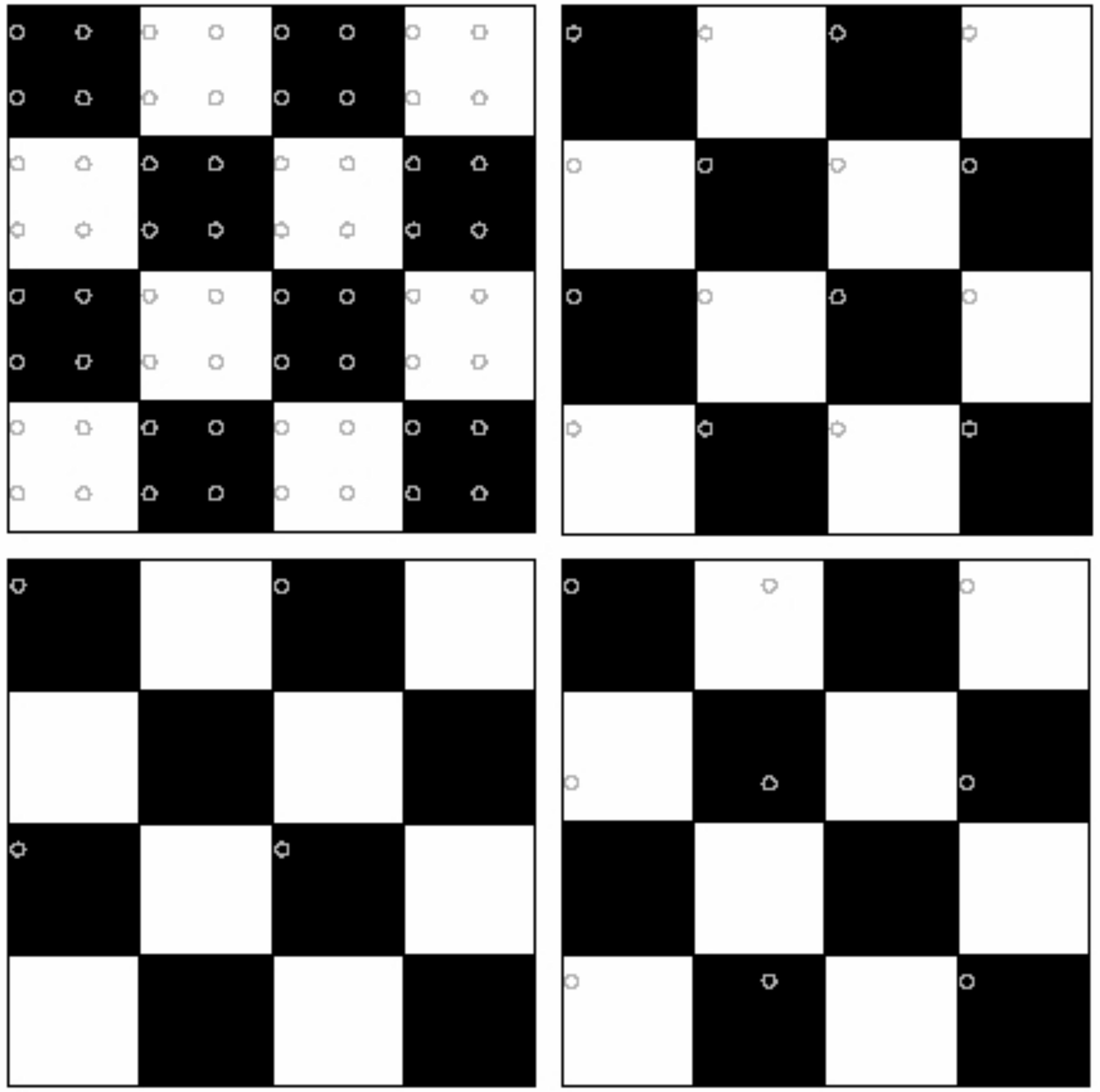


Denote the discrete image as $I(X, Y)$

We can store the pixels in a matrix or array

Sampling

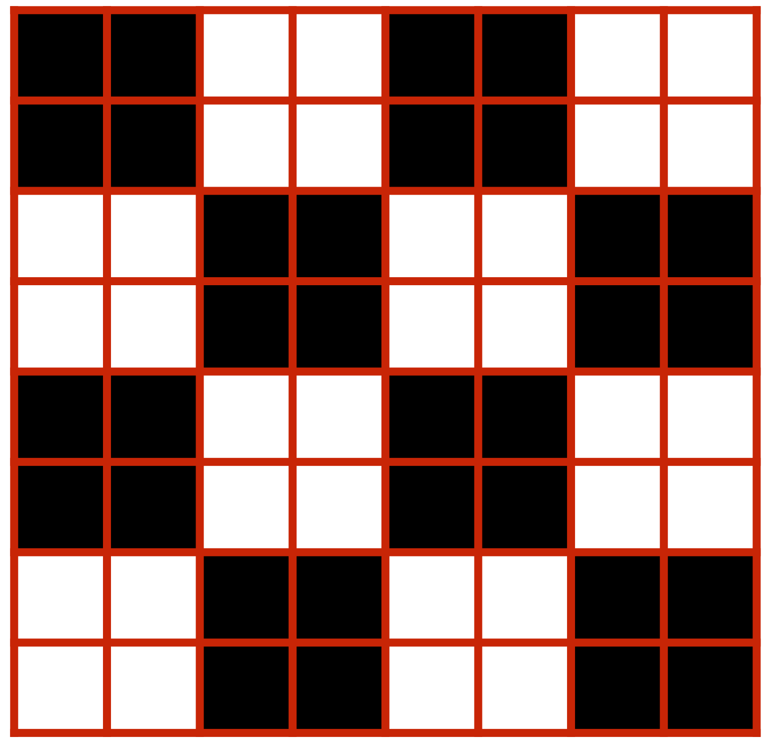
It is clear that *some* information may be lost when we work on a discrete pixel grid.

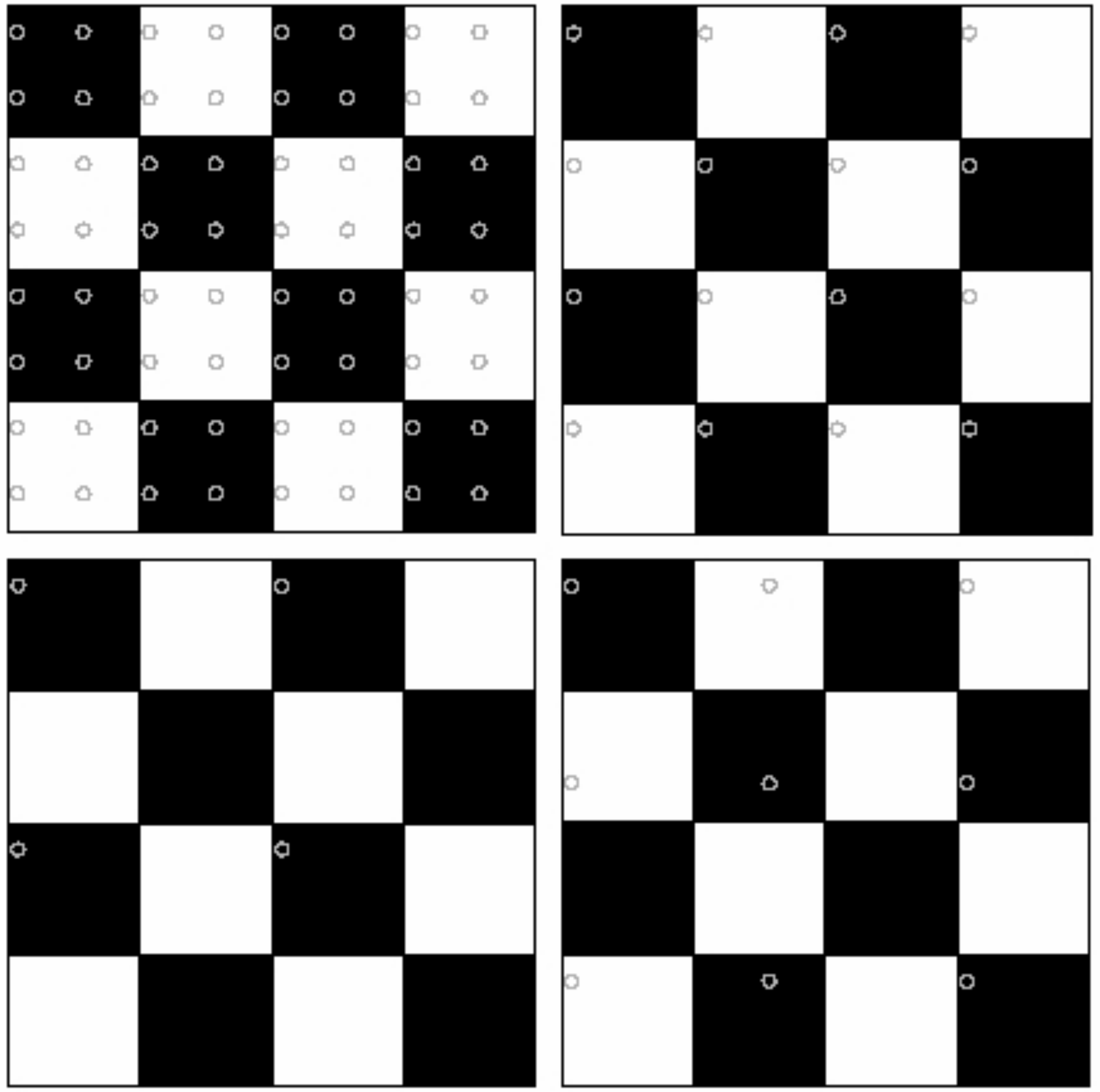


Forsyth & Ponce (2nd ed.) Figure 4.7

Sampling

It is clear that *some* information may be lost when we work on a discrete pixel grid.

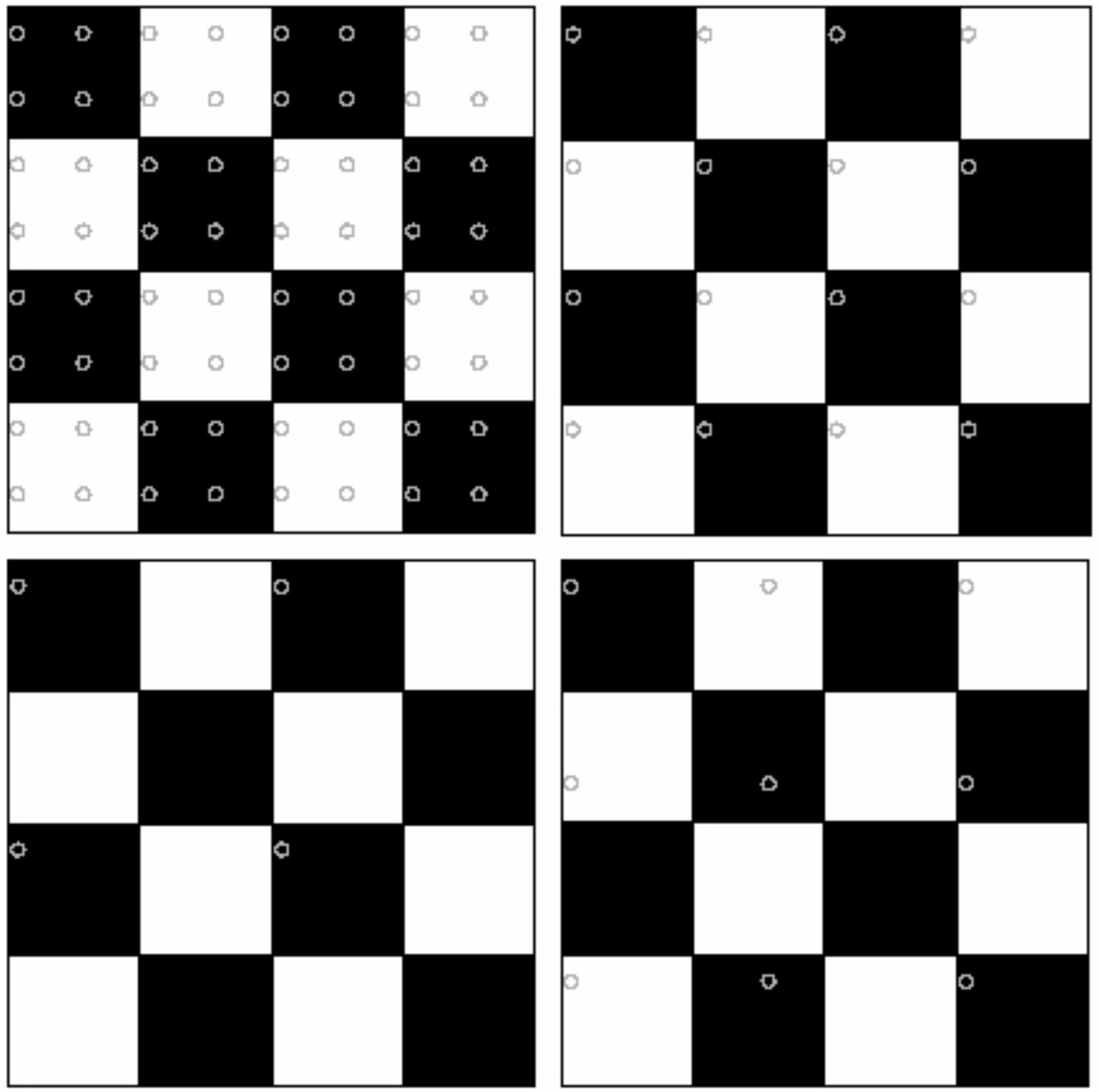




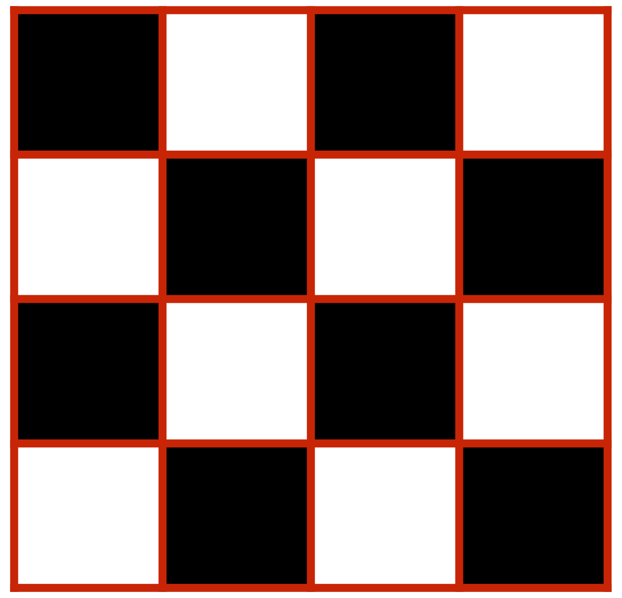
Forsyth & Ponce (2nd ed.) Figure 4.7

Sampling

It is clear that *some* information may be lost when we work on a discrete pixel grid.

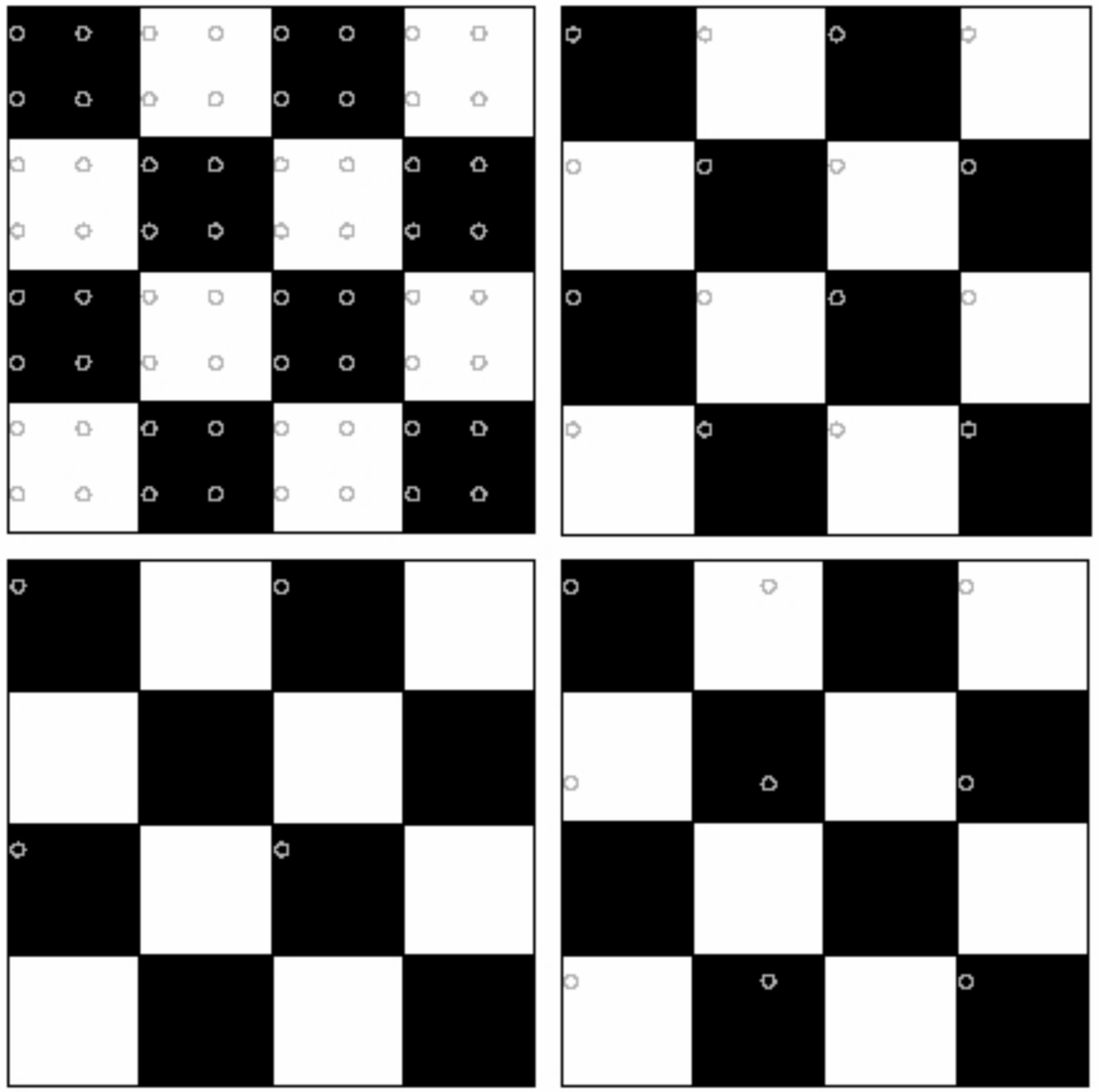


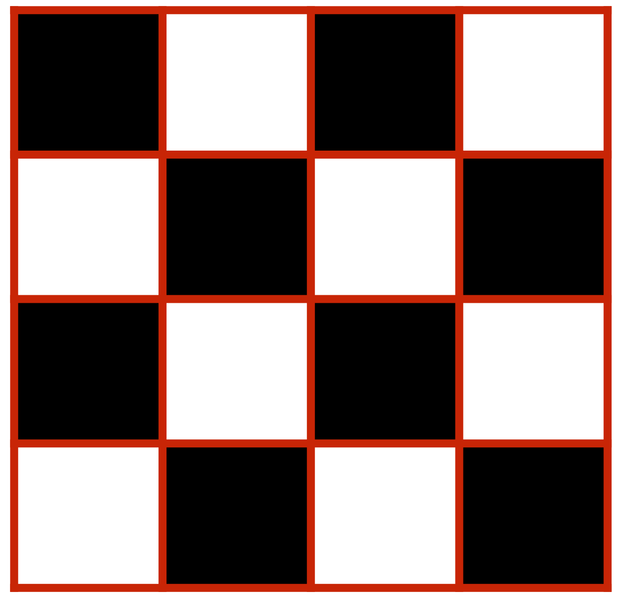
Forsyth & Ponce (2nd ed.) Figure 4.7

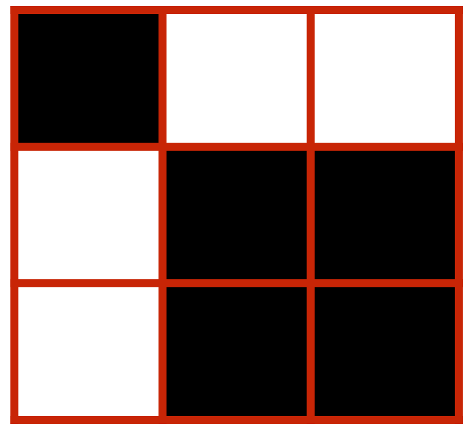


Sampling

It is clear that *some* information may be lost when we work on a discrete pixel grid.



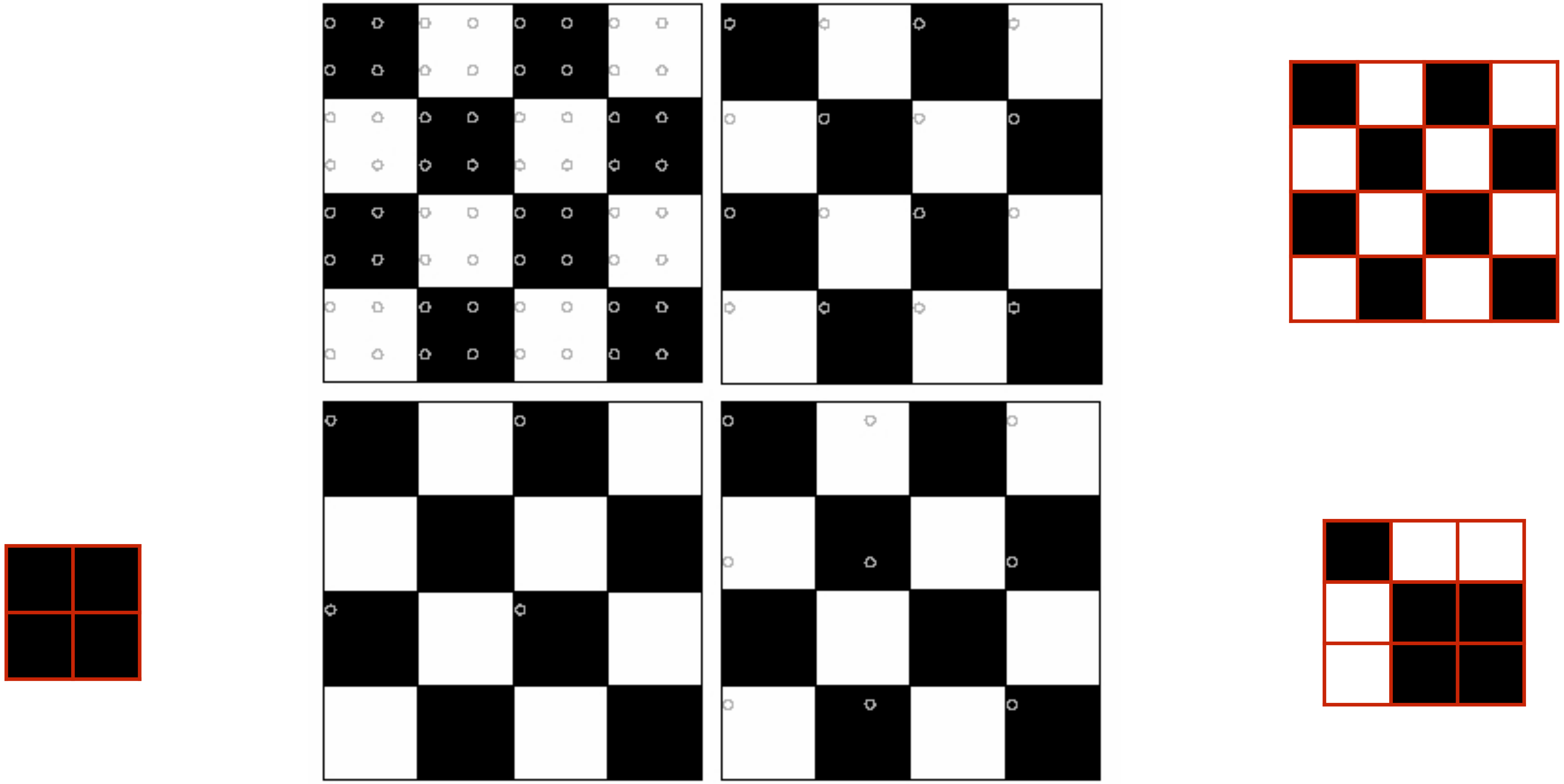




Forsyth & Ponce (2nd ed.) Figure 4.7

Sampling

It is clear that *some* information may be lost when we work on a discrete pixel grid.



Forsyth & Ponce (2nd ed.) Figure 4.7

Sampling Theory

Exact reconstruction requires constraint on the rate at which $i(x, y)$ can change between samples

— “rate of change” means derivative

— the formal concept is **bandlimited signal**

— “bandlimit” and “constraint on derivative” are linked

An image is bandlimited if it has some maximum **spatial frequency**

A fundamental result (**Sampling Theorem**) is:

> For bandlimited signals, if you sample regularly at or above twice the maximum frequency (called the **Nyquist rate**), then you can reconstruct the original signal exactly

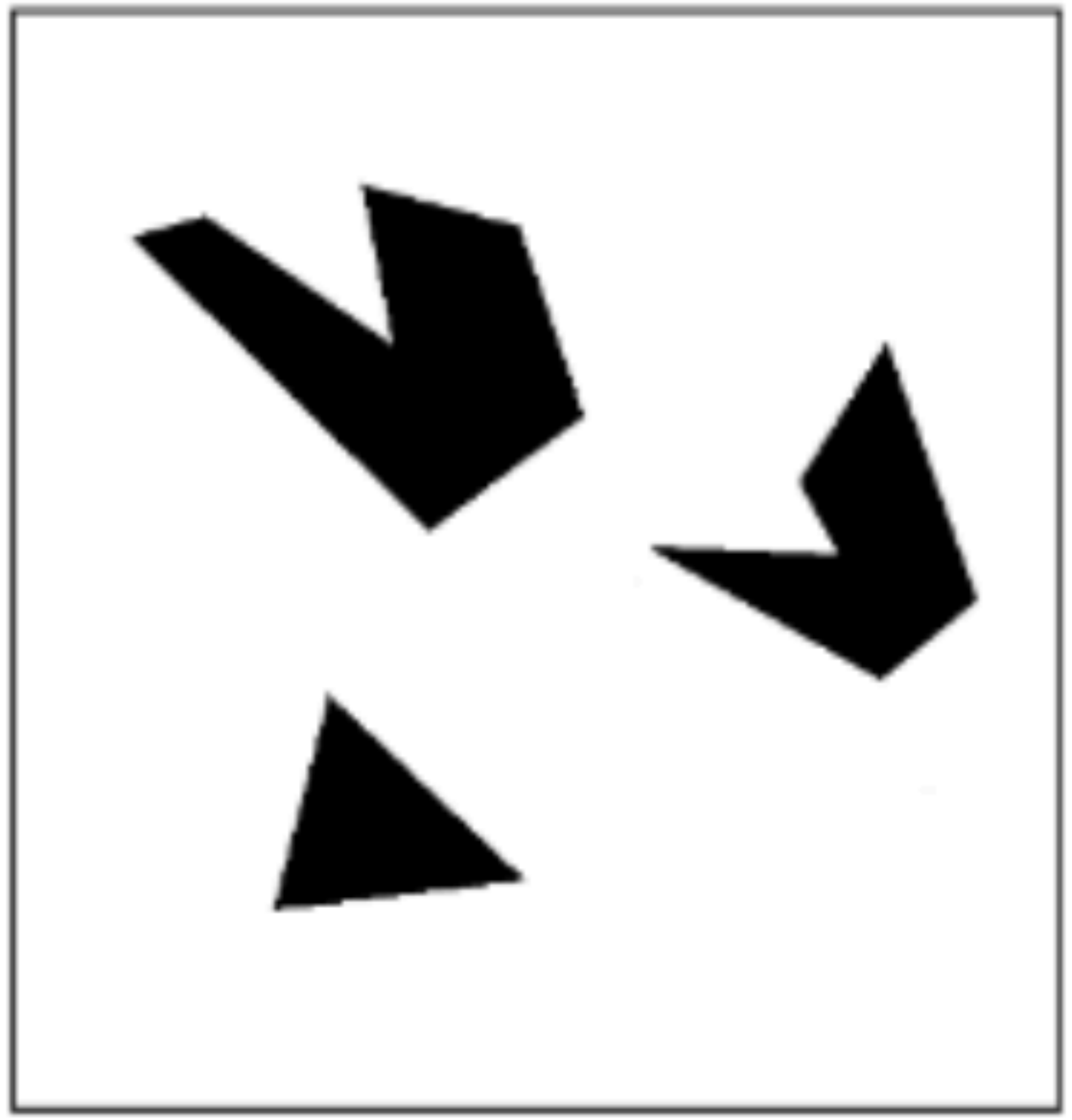
Sampling Theory

Sometimes undersampling is unavoidable, and there is a trade-off between “things missing” and “artifacts.”

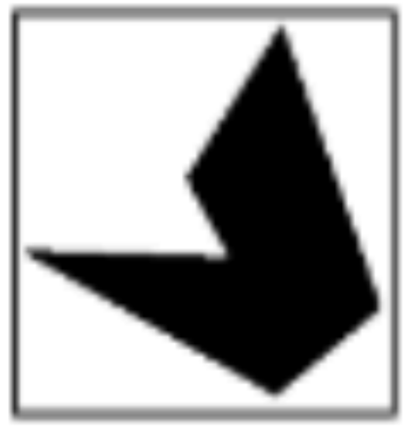
Medical imaging: usually try to maximize information content, tolerate some artifacts

Computer graphics: usually try to minimize artifacts, tolerate some information missing

Template Matching



Scene



Template (mask)

A toy example

Slide Credit: Kristen Grauman

Template Matching

We can think of convolution/**correlation** as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

Template Matching

We can think of convolution/**correlation** as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

Template

0	0	0
0	1	0
0	1	1

Template Matching

We can think of convolution/**correlation** as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

Image Patch 1

0	0	0
0	1	0
0	1	1

Image Patch 2

1	0	1
0	1	0
0	0	0

Template

0	0	0
0	1	0
0	1	1

Template Matching

We can think of convolution/**correlation** as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

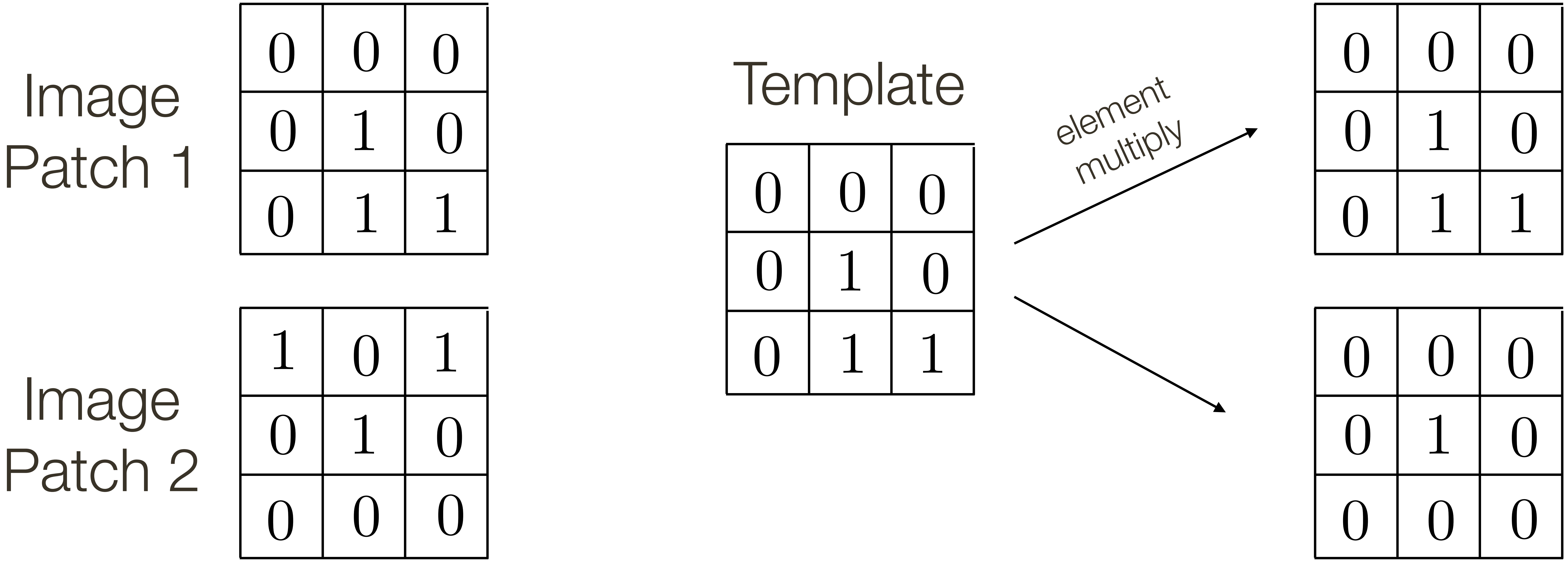


Image Patch 1

0	0	0
0	1	0
0	1	1

Image Patch 2

1	0	1
0	1	0
0	0	0

Template

0	0	0
0	1	0
0	1	1

0	0	0
0	1	0
0	1	1

0	0	0
0	1	0
0	0	0

Template Matching

We can think of convolution/**correlation** as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

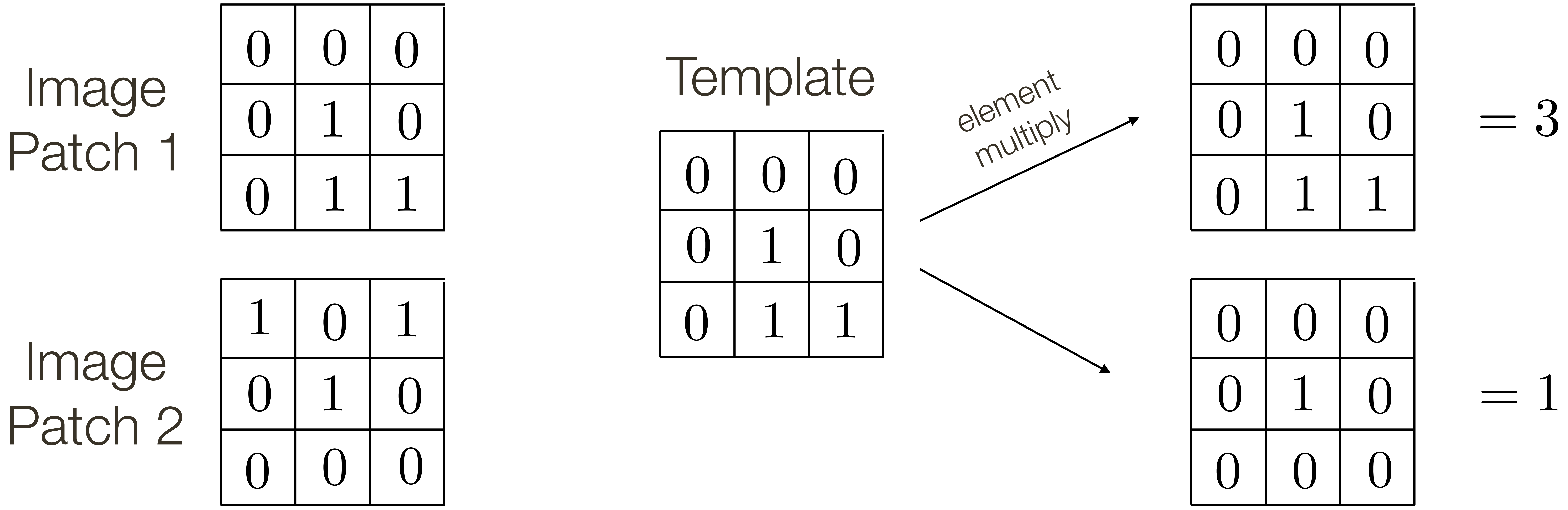


Image Patch 1

0	0	0
0	1	0
0	1	1

Image Patch 2

1	0	1
0	1	0
0	0	0

Template

0	0	0
0	1	0
0	1	1

0	0	0
0	1	0
0	1	1

$= 3$

0	0	0
0	1	0
0	0	0

$= 1$

Template Matching

We can think of convolution/**correlation** as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

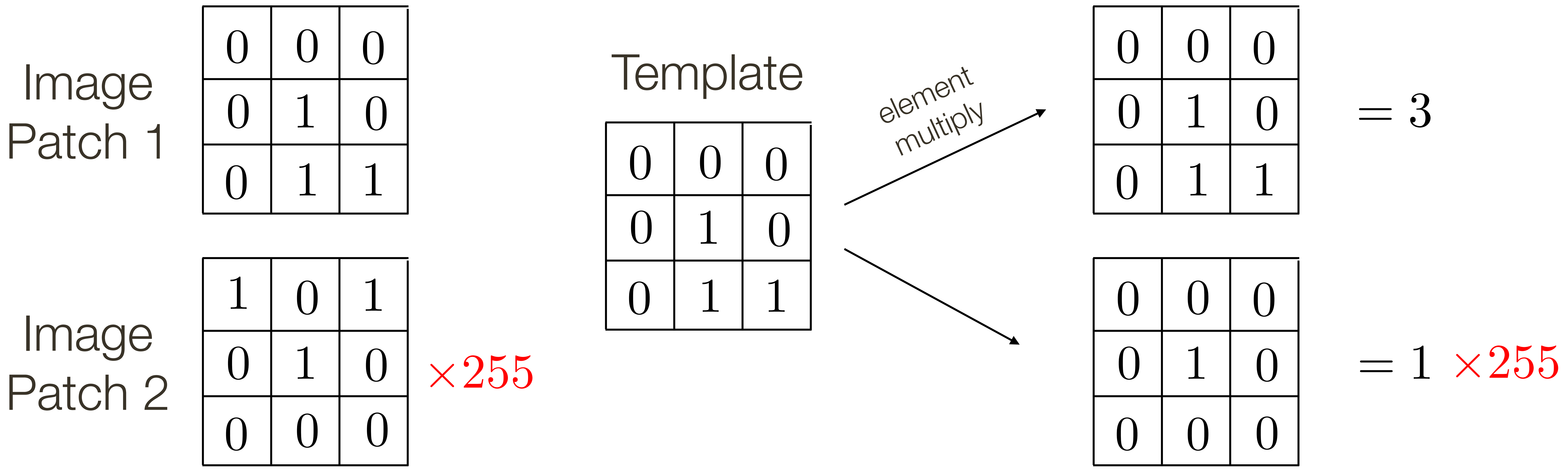


Image Patch 1

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Image Patch 2

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \times 255$$

Template

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = 3$$

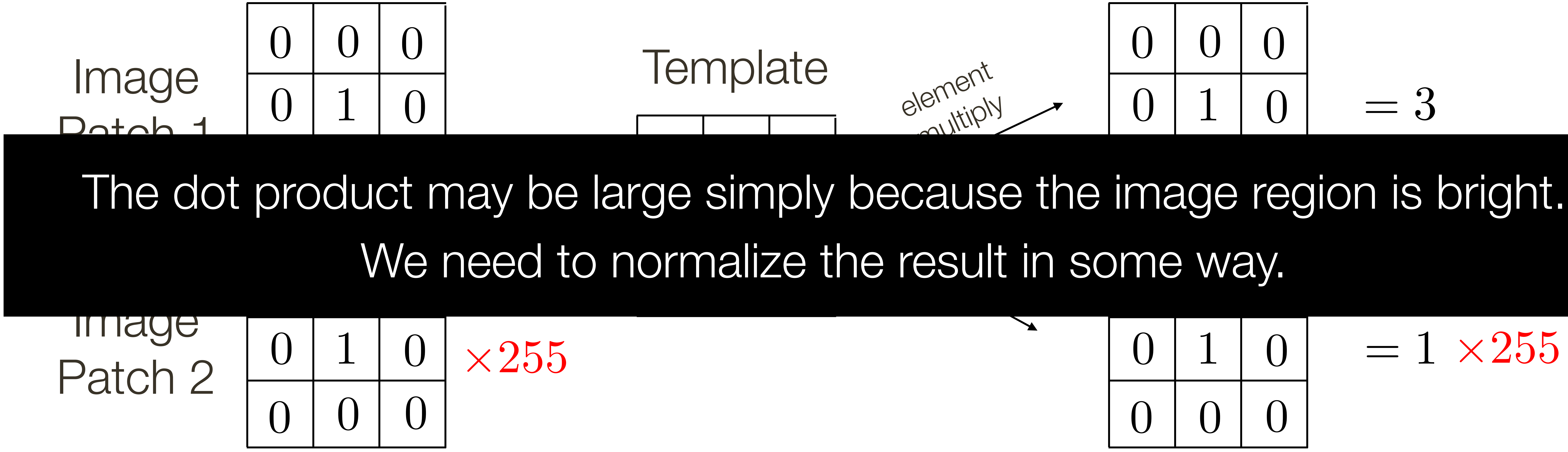
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = 1 \times 255$$

Template Matching

We can think of convolution/**correlation** as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.



The dot product may be large simply because the image region is bright.

We need to normalize the result in some way.

Template Matching

Let a and b be vectors. Let θ be the angle between them. We know

$$\cos\theta = \frac{a \cdot b}{|a||b|} = \frac{a \cdot b}{\sqrt{(a \cdot a)(b \cdot b)}} = \frac{a}{|a|}\frac{b}{|b|}$$

where · is dot product and | | is vector magnitude

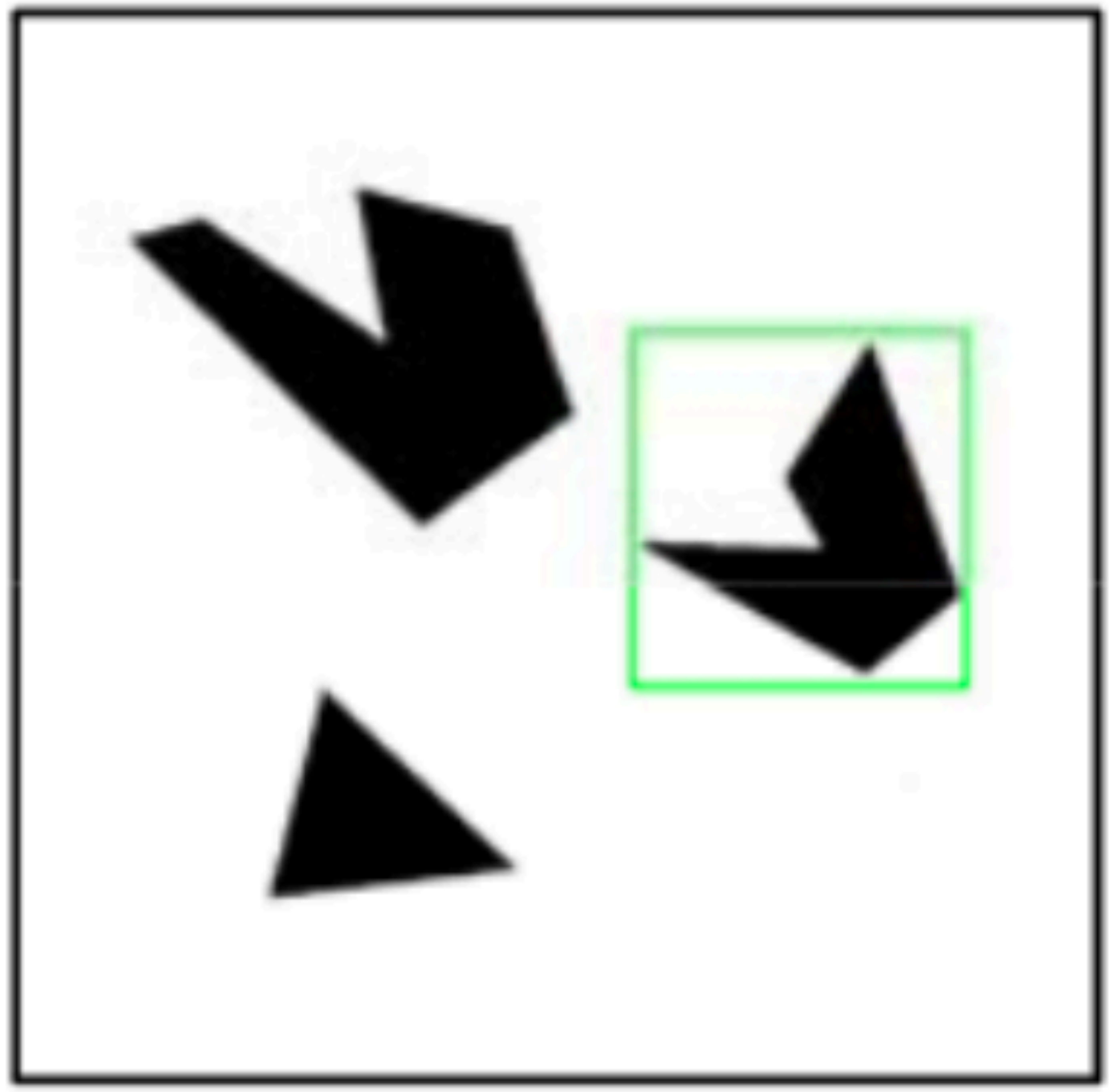
Correlation is a dot product

Correlation measures similarity between the filter and each local image region

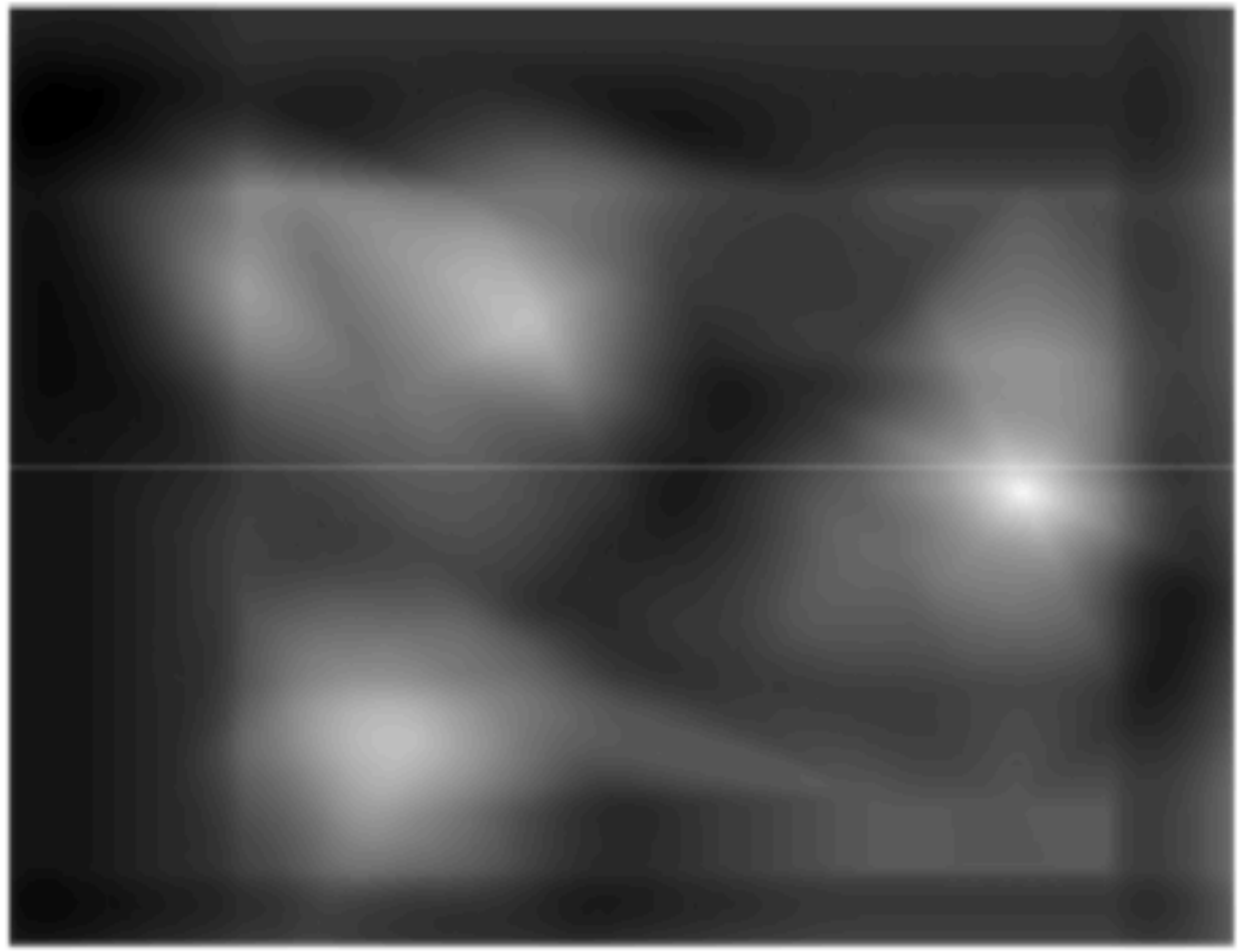
Normalized correlation varies between –1 and 1

Normalized correlation attains the value 1 when the filter and image region are identical (up to a scale factor)

Template Matching



Detected template



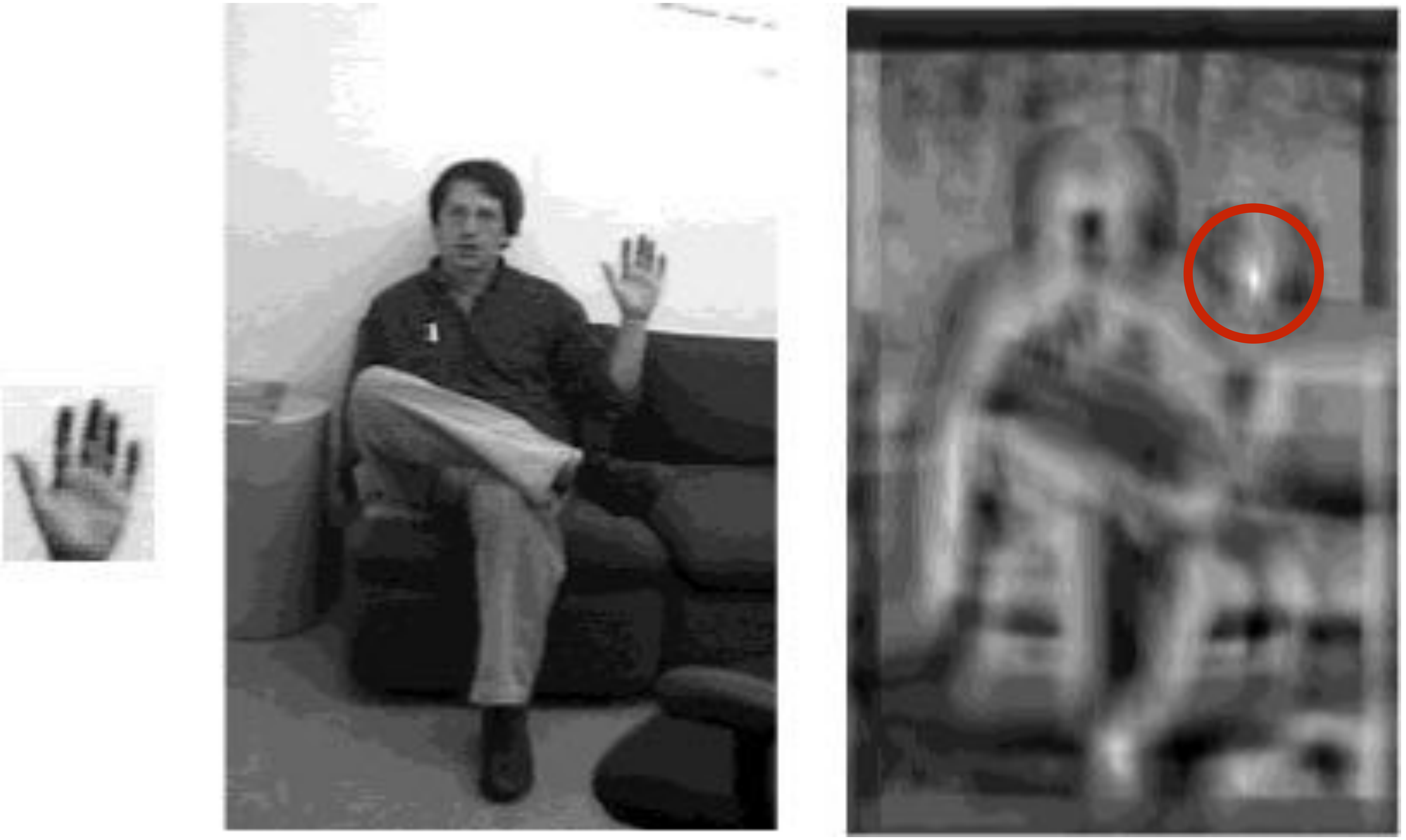
Correlation map

Slide Credit: Kristen Grauman

Example 1:

Template (left), image (middle), normalized correlation (right)

Note peak value at the true position of the hand



Credit: W. Freeman et al., “Computer Vision for Interactive Computer Graphics,” IEEE Computer Graphics and Applications, 1998

Sample Question: Template Matching

True or **false**: Normalized correlation is robust to a constant scaling in the image brightness.

Scaled Representations: Goals

to find **template matches** at all scales

— template size constant, image scale varies

— finding hands or faces when we don’t know what size they are in the image

efficient search for image–to–image correspondences

— look first at coarse scales, refine at finer scales

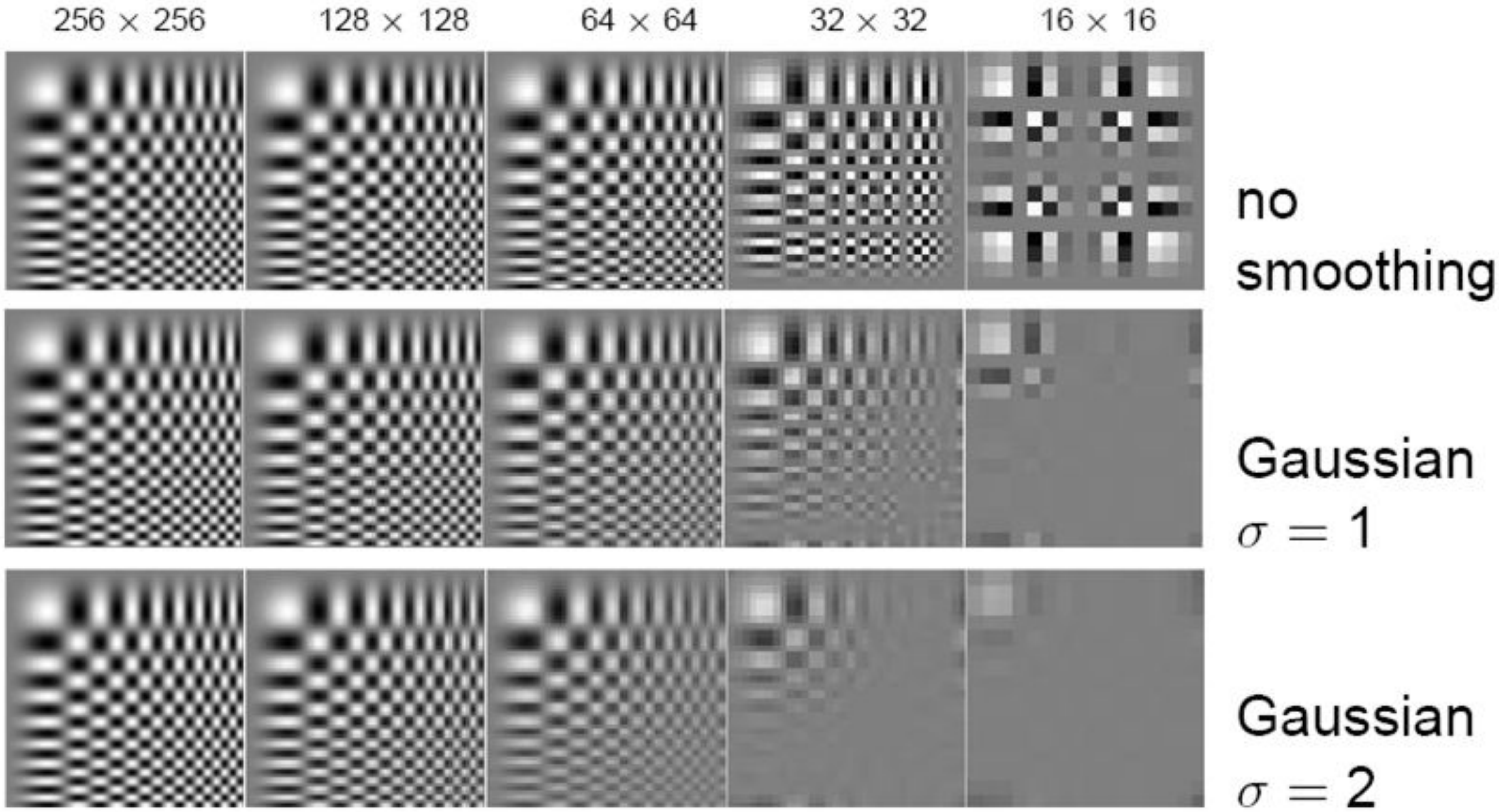
— much less cost (but may miss best match)

to examine all **levels of detail**

— find edges with different amounts of blur

— find textures with different spatial frequencies (i.e., different levels of detail)

Shrinking the Image



Forsyth & Ponce (2nd ed.) Figure 4.12-4.14 (top rows)

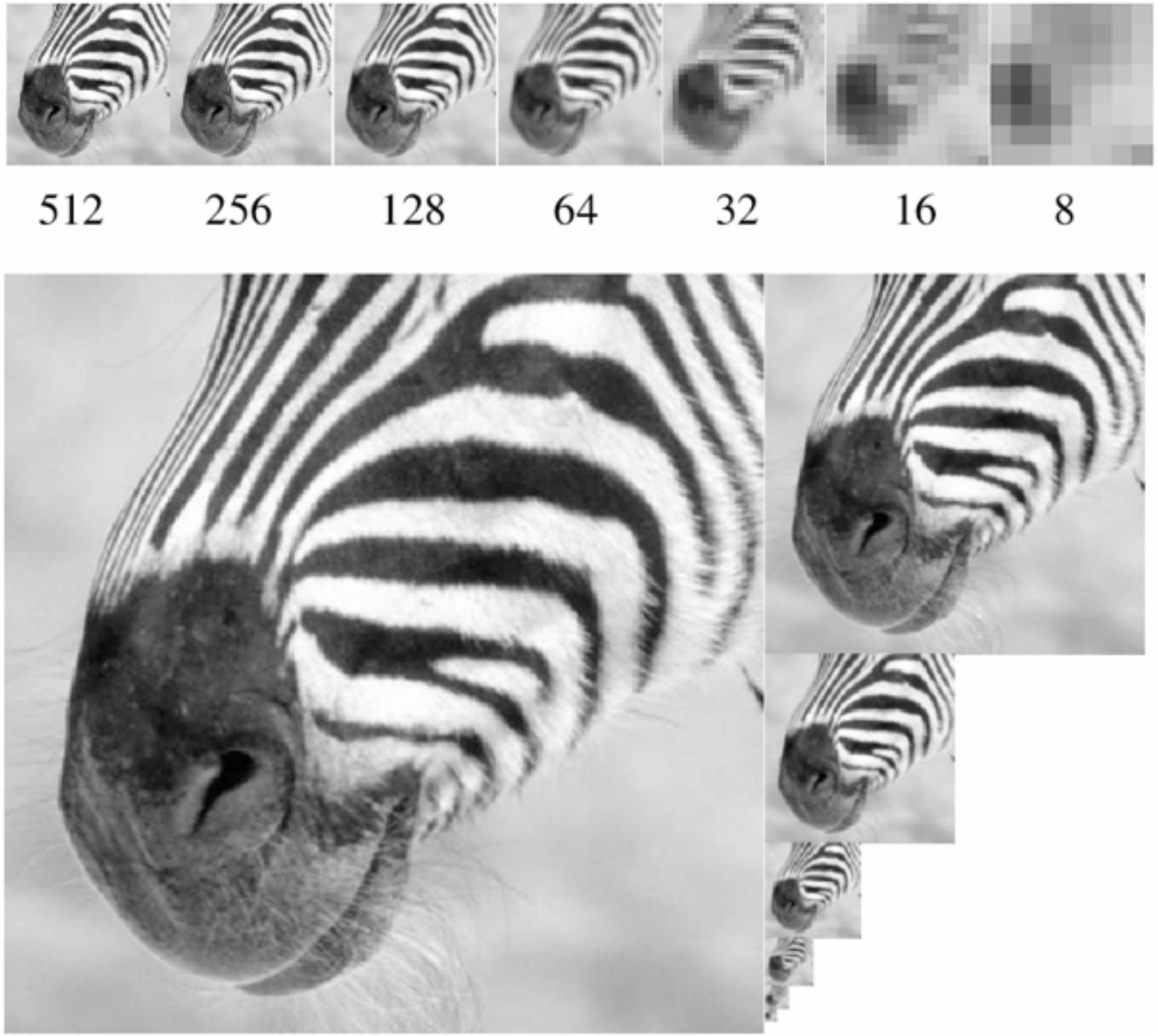
Image **Pyramid**

An **image pyramid** is a collection of representations of an image. Typically, each layer of the pyramid is half the width and half the height

of the previous layer.

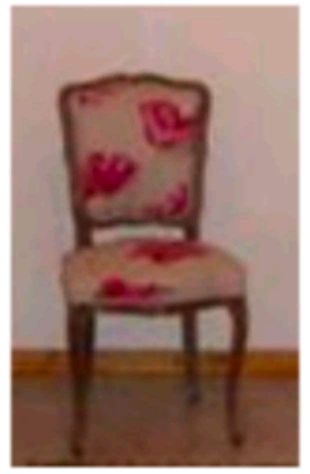
In a **Gaussian pyramid**, each layer is smoothed by a Gaussian filter and resampled to get the next layer

Gaussian Pyramid

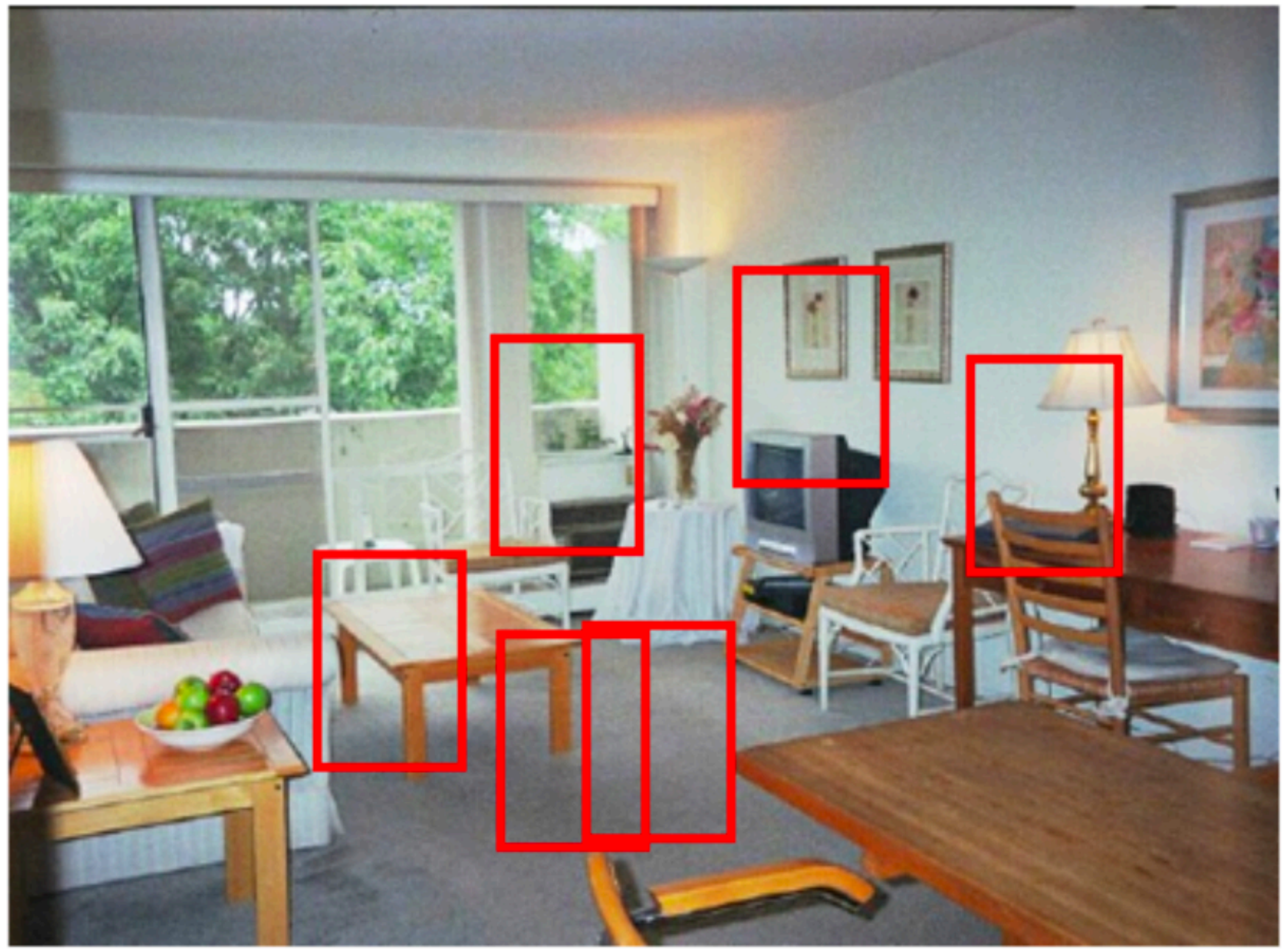


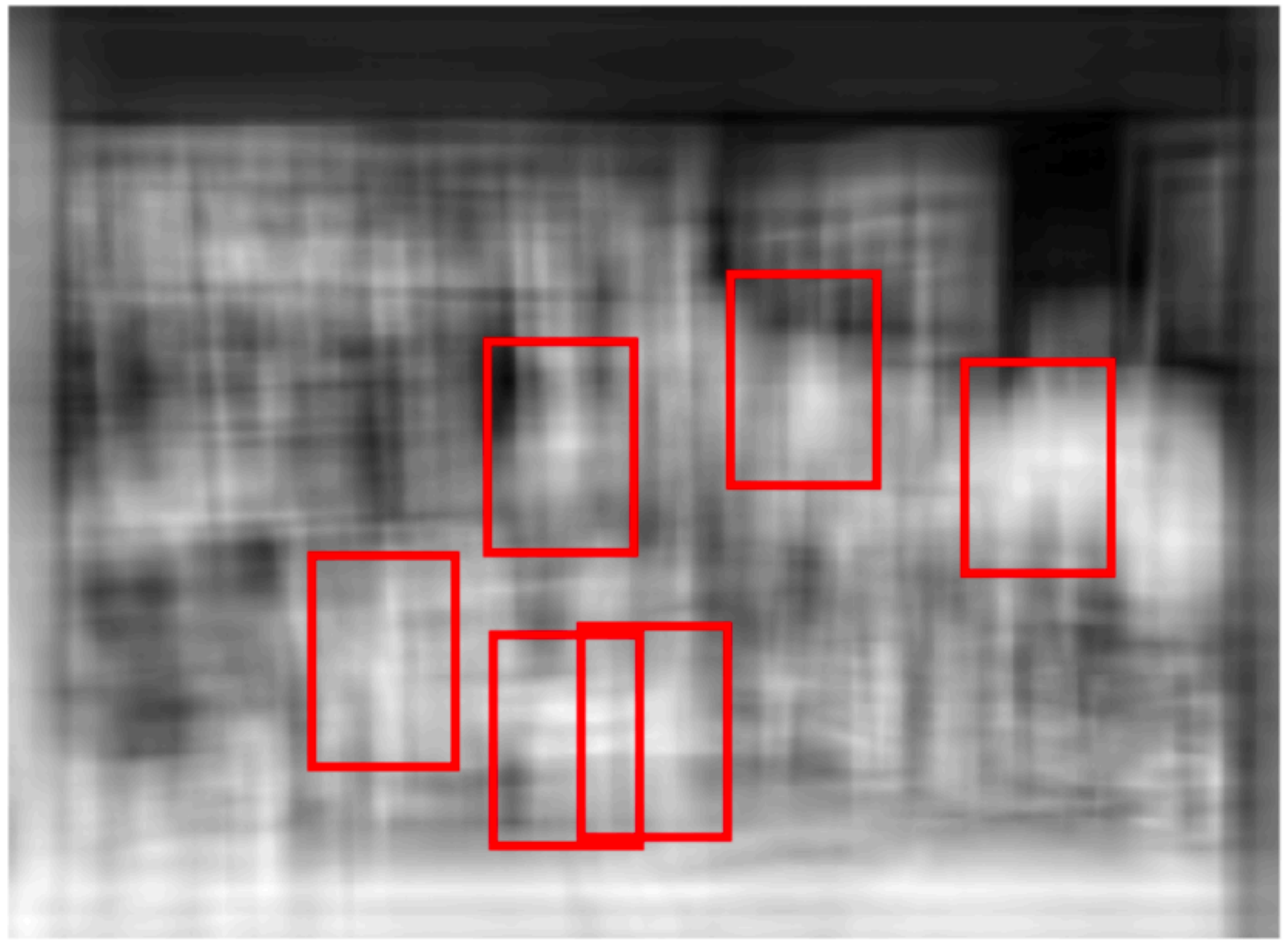
Forsyth & Ponce (2nd ed.) Figure 4.17

From Template Matching to **Local Feature Detection**



Find the chair in this image





Pretty much garbage
Simple template matching is not going to make it

Slide Credit: Li Fei-Fei, Rob Fergus, and Antonio Torralba

Estimating **Derivatives**

Recall, for a 2D (continuous) function, f(x,y)

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}$$

Differentiation is linear and shift invariant, and therefore can be implemented as a convolution

A (discrete) approximation is

$$\frac{\partial f}{\partial x} \approx \frac{F(X+1,y) - F(x,y)}{\Delta x}$$

-1	1

Estimating **Derivatives**

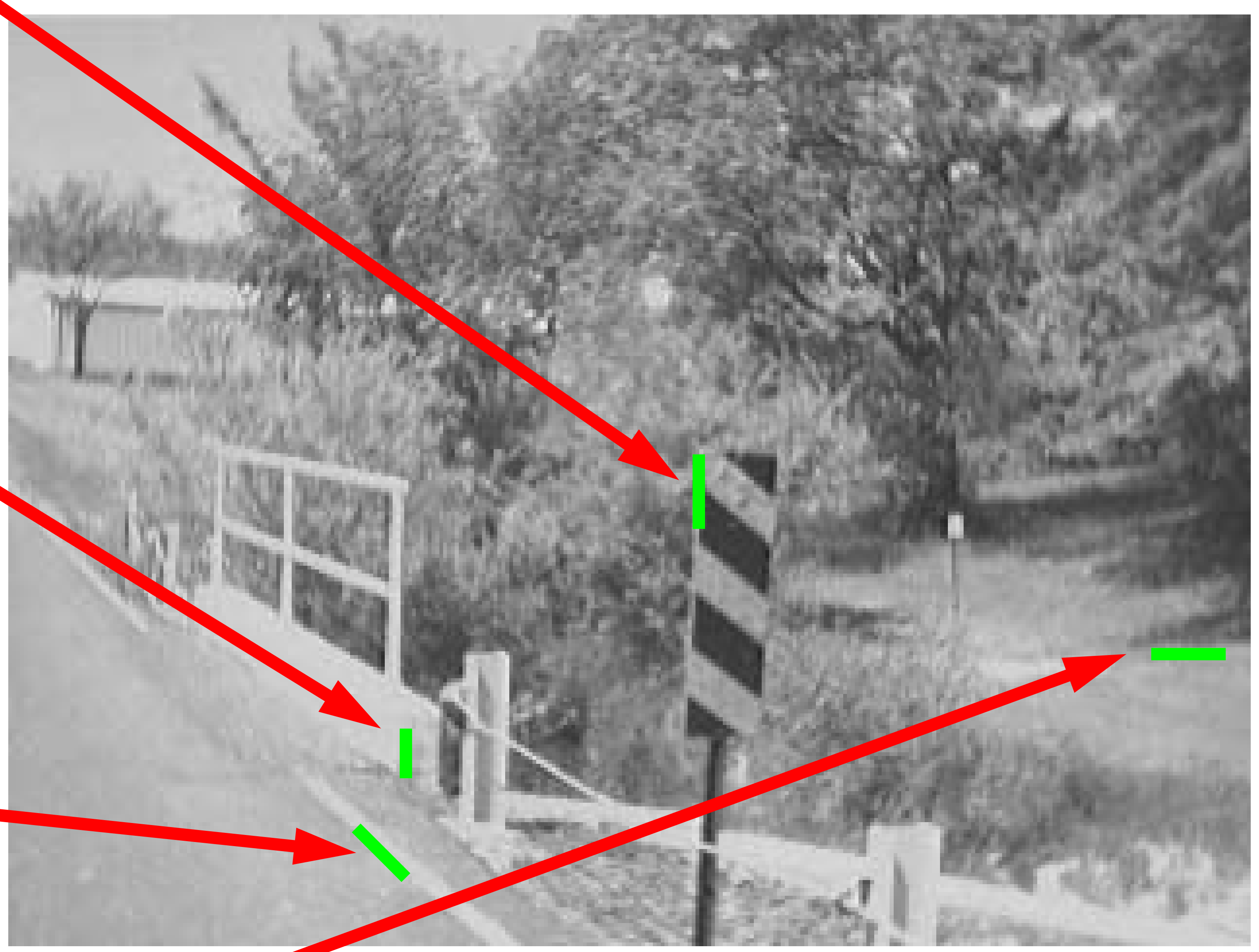
A similar definition (and approximation) holds for $\frac{\partial f}{\partial y}$

Image **noise** tends to result in pixels not looking exactly like their neighbours, so simple “finite differences” are sensitive to noise.

The usual way to deal with this problem is to **smooth** the image prior to derivative estimation.

What Causes **Edges**?

- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)



Slide Credit: Christopher Rasmussen

Smoothing and Differentiation

Edge: a location with high gradient (derivative)

Need smoothing to reduce noise prior to taking derivative

Need two derivatives, in x and y direction

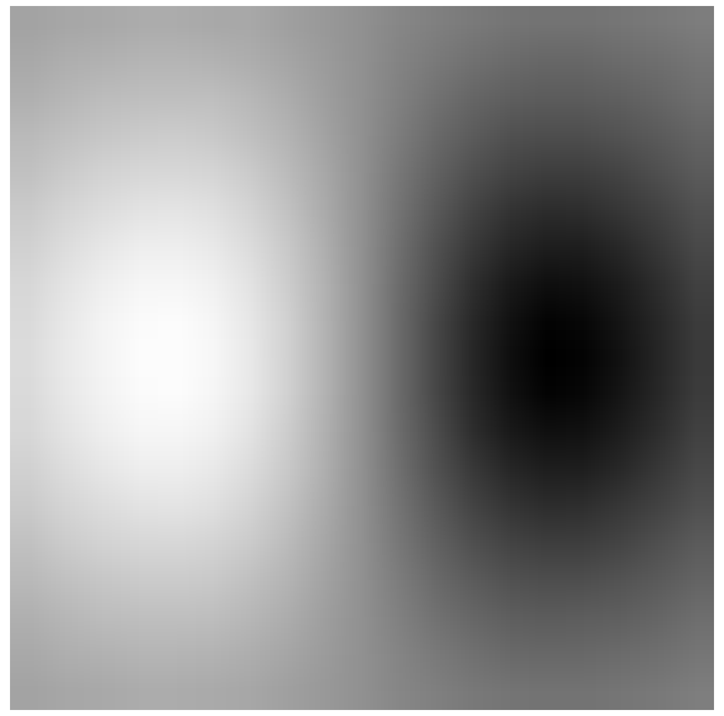
We can use **derivative of Gaussian** filters

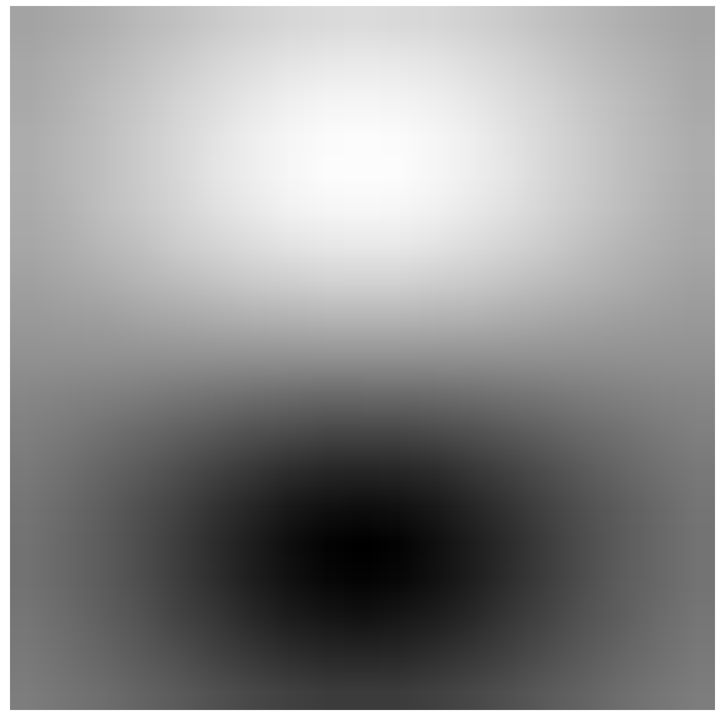
— because differentiation is convolution, and

— convolution is associative

Let $\otimes$ denote convolution

$$D \otimes (G \otimes I(X,Y)) = (D \otimes G) \otimes I(X,Y)$$





Gradient **Magnitude**

Let $I(X,Y)$ be a (digital) image

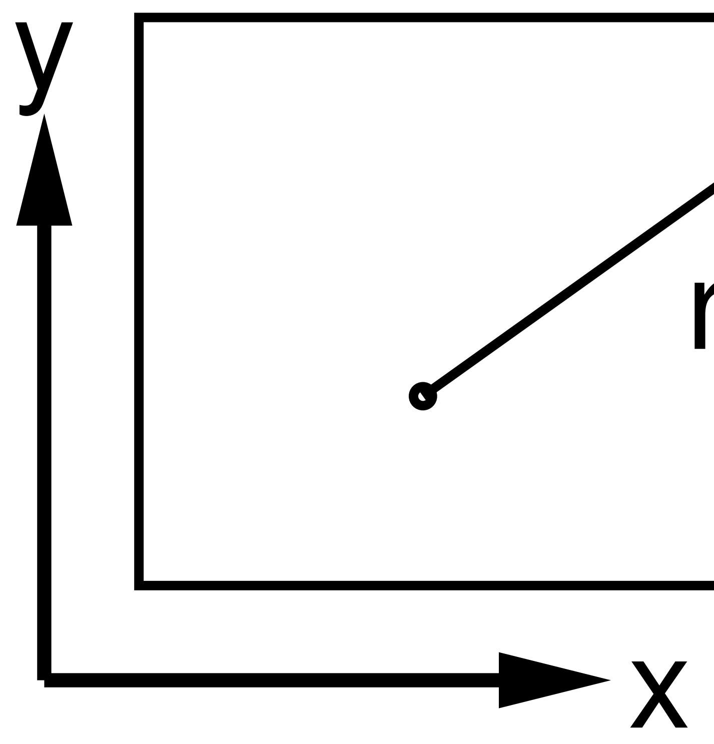
Let $I_x(X,Y)$ and $I_y(X,Y)$ be estimates of the partial derivatives in the x and y directions, respectively.

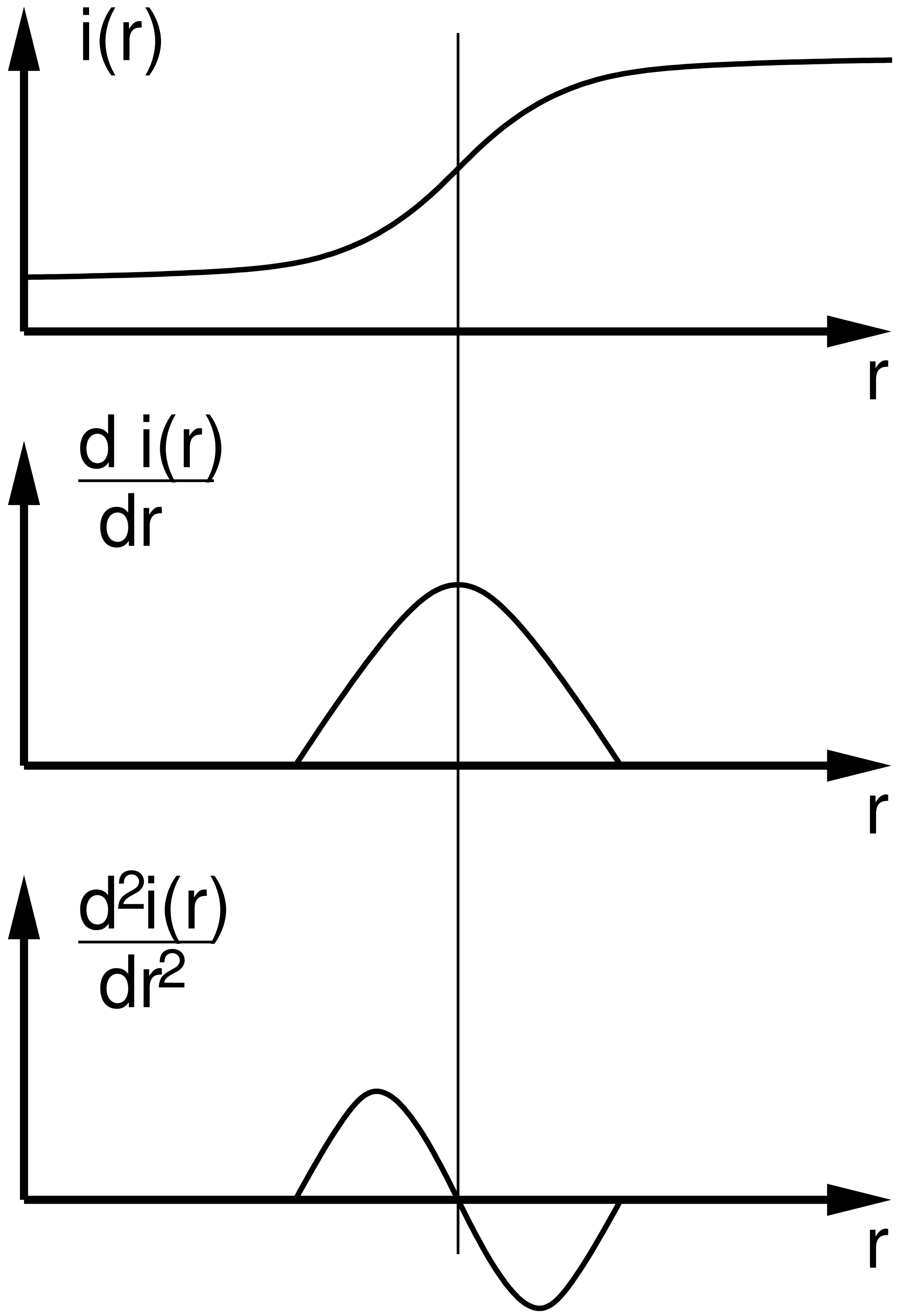
Call these estimates I_x and I_y (for short) The vector $[I_x, I_y]$ is the **gradient**

The scalar $\sqrt{I_x^2 + I_y^2}$ is the **gradient magnitude**

The **gradient direction** is given by: $\theta = \tan^{-1}\left(\frac{I_y}{I_x}\right)$

Two Generic Approaches for **Edge** Detection





Two generic approaches to **edge point detection**:

— (significant) local extrema of a first derivative operator

— zero crossings of a second derivative operator

Marr / Hildreth **Laplacian of Gaussian**

A "**zero crossings** of a second derivative operator" approach

Steps:

1. Gaussian for smoothing

2. Laplacian (∇^2) for differentiation where

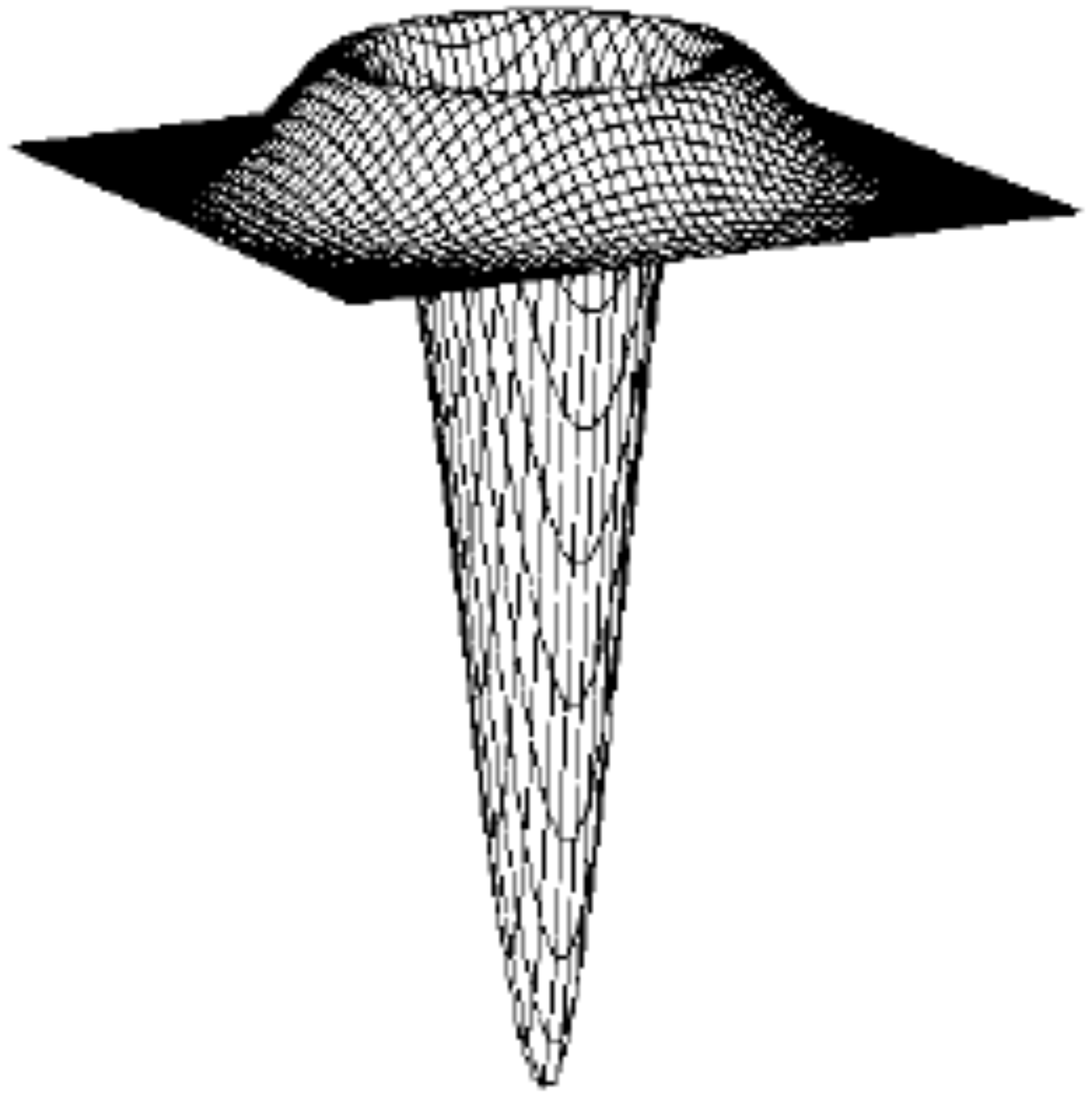
$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

3. Locate zero-crossings in the Laplacian of the Gaussian ($\nabla^2 G$) where

$$\nabla^2 G(x,y) = \frac{-1}{2\pi\sigma^4}\left[2 - \frac{x^2+y^2}{\sigma^2}\right]\exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

Marr / Hildreth **Laplacian of Gaussian**

Here's a 3D plot of the Laplacian of the Gaussian ($\nabla^2 G$)



. . . with its characteristic "Mexican hat" shape

Canny Edge Detector

Steps:

1. Apply **directional derivatives** of Gaussian

2. Compute **gradient magnitude** and **gradient direction**

3. **Non-maximum** suppression
   — thin multi-pixel wide “ridges” down to single pixel width

4. **Linking** and thresholding
   — Low, high edge-strength thresholds
   — Accept all edges over low threshold that are connected to edge over high threshold

Edge **Hysteresis**

One way to deal with broken edge chains is to use hysteresis

Hysteresis: A lag or momentum factor

Idea: Maintain two thresholds $\mathbf{k}_{high}$ and $\mathbf{k}_{low}$

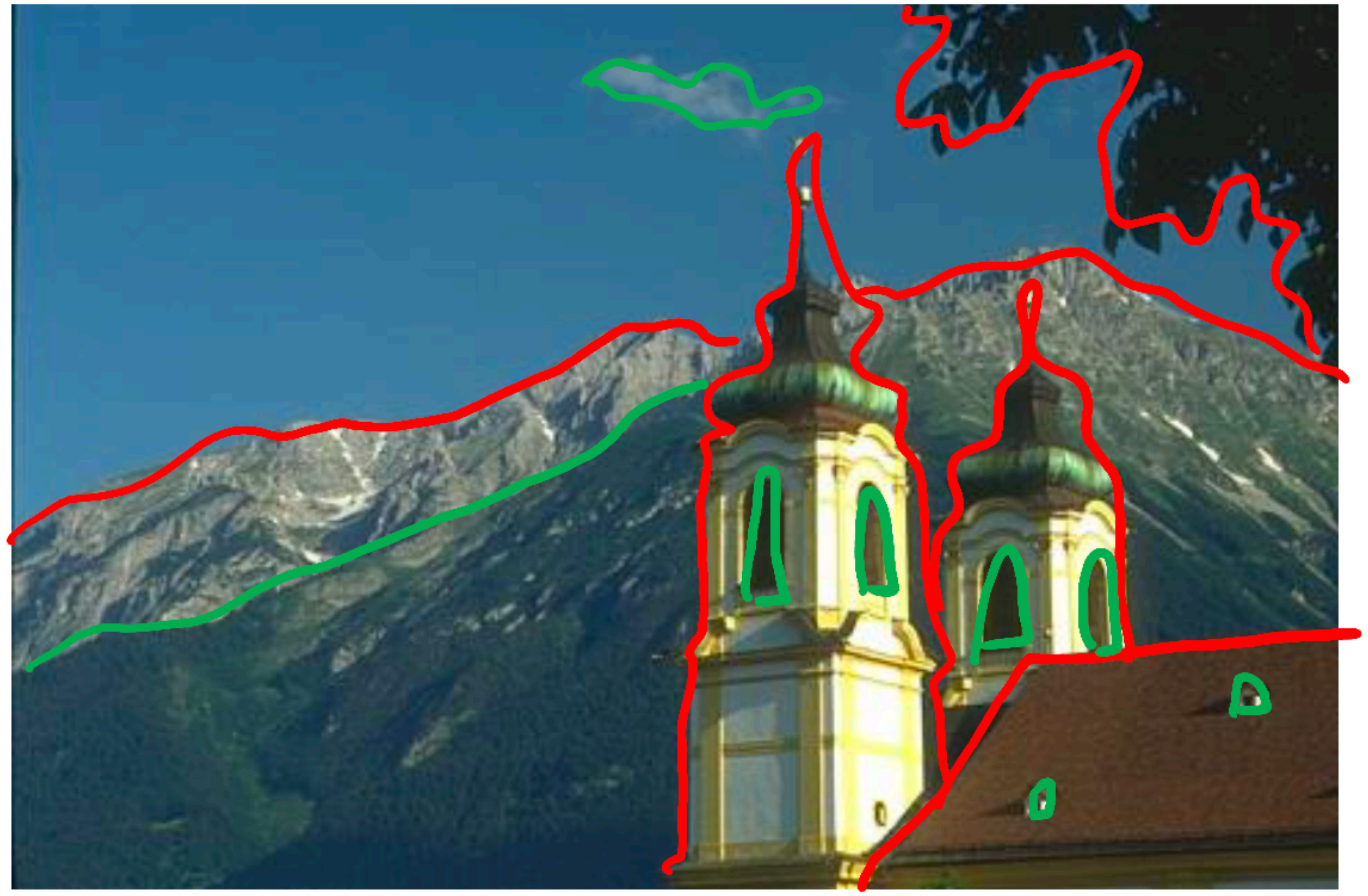
— Use khigh to find strong edges to start edge chain

— Use klow to find weak edges which continue edge chain

Typical ratio of thresholds is (roughly):

$$\frac{\mathbf{k}_{high}}{\mathbf{k}_{low}} = 2$$

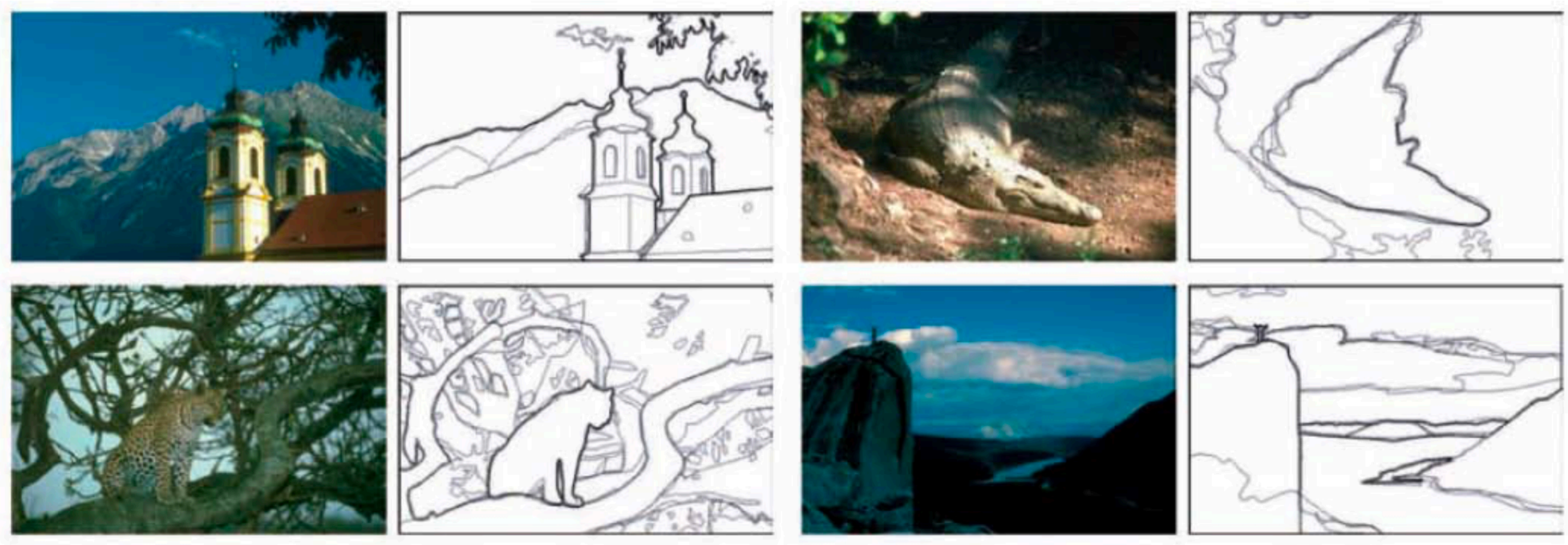
How do humans perceive **boundaries**?



"Divide the image into some number of segments, where the segments represent ’things’ or ’parts of things’ in the scene. The number of segments is up to you, as it depends on the image. Something between 2 and 30 is likely to be appropriate. It is important that all of the segments have approximately equal importance."

(Martin et al. 2004)

How do humans perceive **boundaries**?



Each image shows multiple (4-8) human-marked boundaries. Pixels are darker where more humans marked a boundary.

Figure Credit: Szeliski Fig. 4.31. **Original**: Martin et al. 2004

Sample Question: Edges

Why is non-maximum suppression applied in the Canny edge detector?

What is a **corner**?

Image Credit: John Shakespeare, Sydney Morning Herald

We can think of a corner as any locally distinct 2D image feature that (hopefully) corresponds to a distinct position on an 3D object of interest in the scene.

Why are corners **distinct**?

A corner can be **localized reliably**.

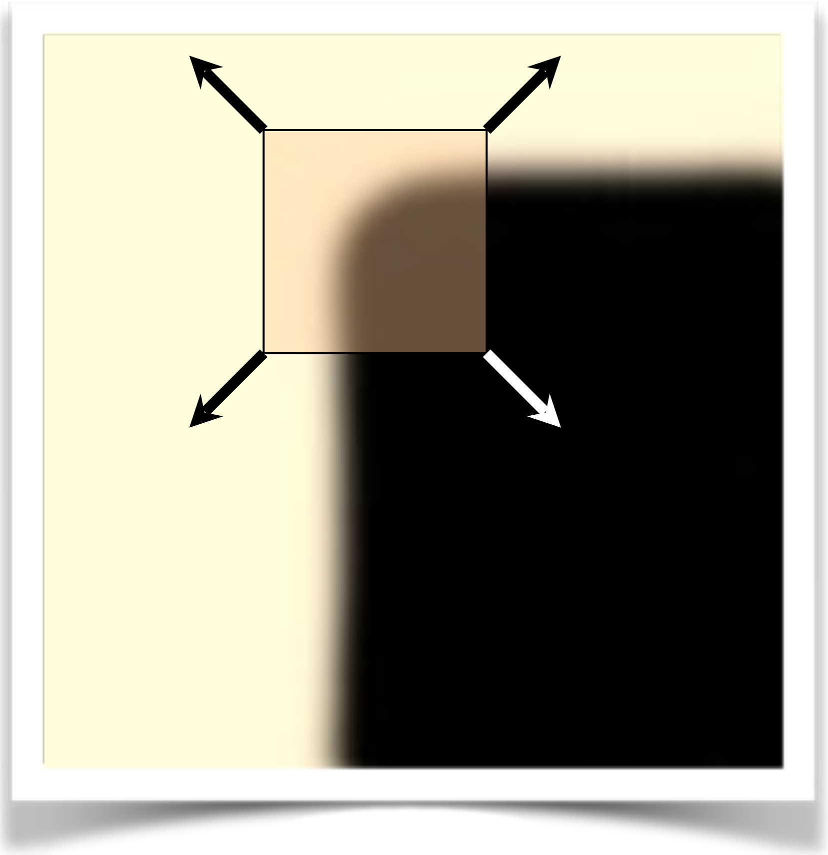
Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

— Place a small window over an edge. If you slide the window in the direction of the edge, the image in the window will not change

→ Cannot estimate location along an edge (a.k.a., **aperture** problem)

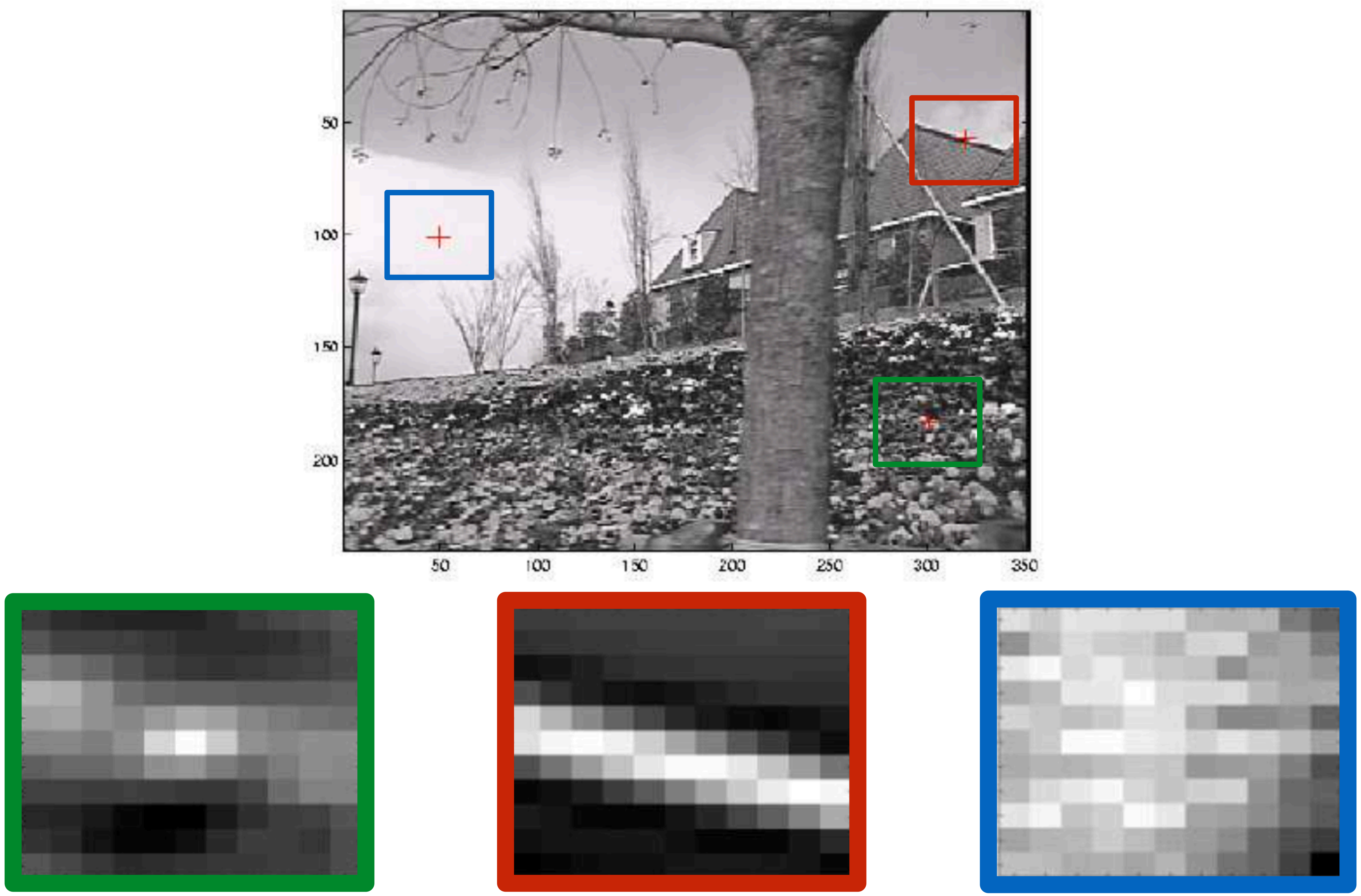
— Place a small window over a corner. If you slide the window in any direction, the image in the window changes.



“**corner**”:
significant change
in all directions

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Autocorrelation



Szeliski, Figure 4.5

Corner Detection

Edge detectors perform poorly at corners

Observations:

— The gradient is ill defined exactly at a corner

— Near a corner, the gradient has two (or more) distinct values

Harris Corner Detection

1. Compute image gradients over small region

2. Compute the covariance matrix

3. Compute eigenvectors and eigenvalues

4. Use threshold on eigenvalues to detect corners

$$I_x = \frac{\partial I}{\partial x}$$

$$I_y = \frac{\partial I}{\partial y}$$

$$\begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix}$$

Slide Adopted: Ioannis (Yannis) Gkioulekas (CMU)

2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

Gradient with respect to x, times gradient with respect to y

$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix}$$

Matrix is **symmetric**

Harris Corner Detection

— Filter image with **Gaussian**

— Compute magnitude of the x and y **gradients** at each pixel

— Construct C in a window around each pixel

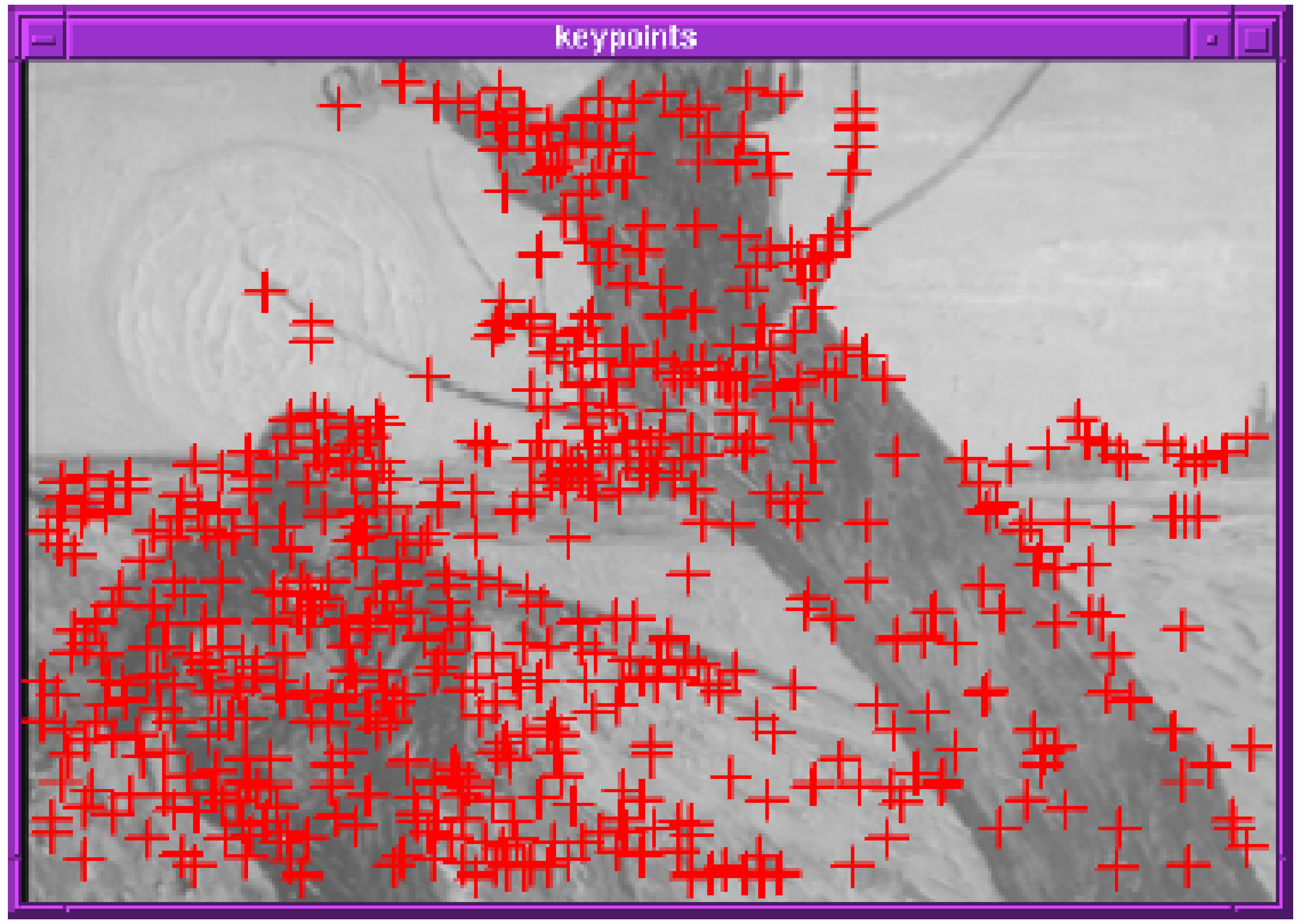
  — Harris uses a **Gaussian window**

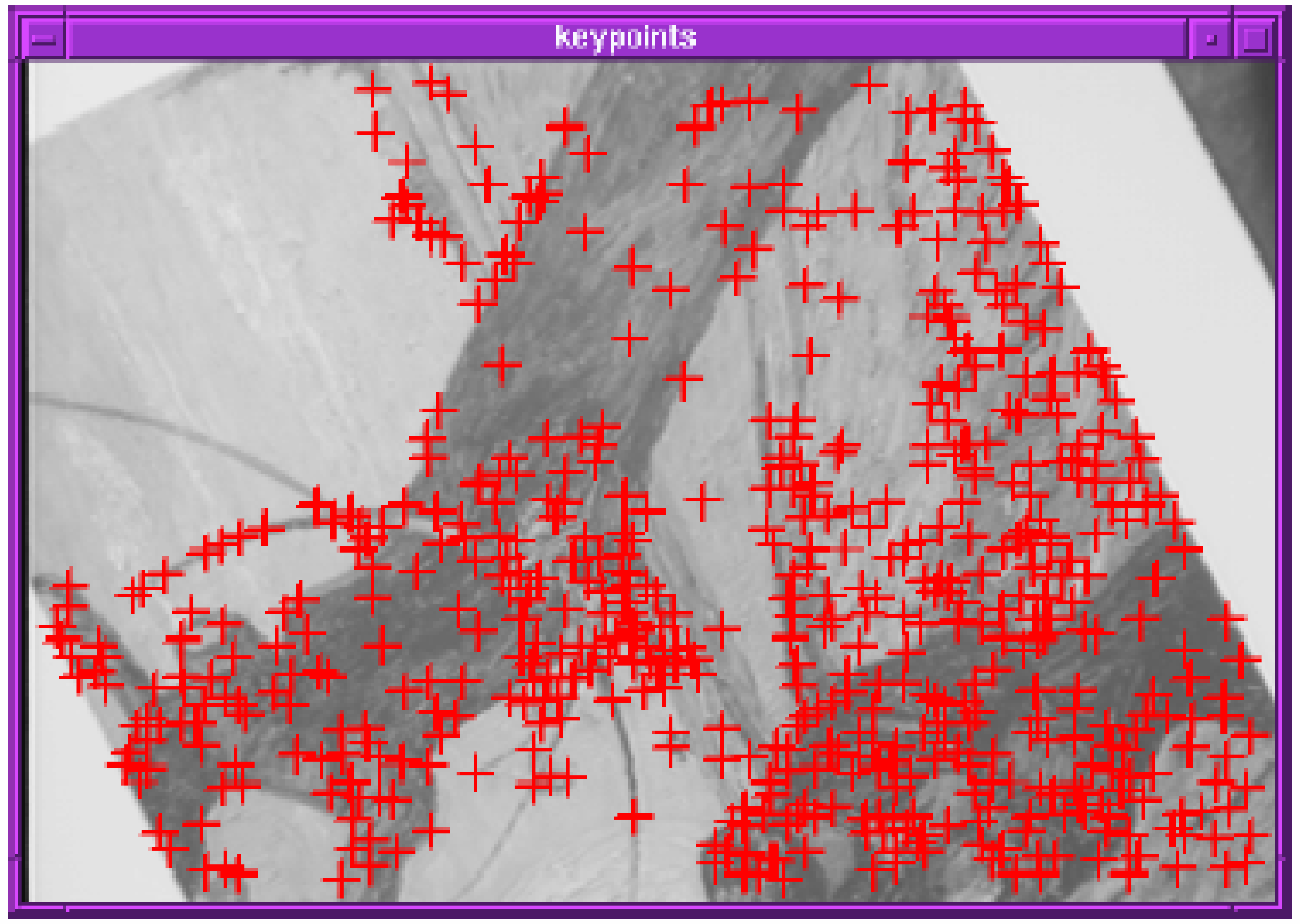
— Solve for product of the λ's

— If λ's both are big (product reaches local maximum above threshold) then we have a corner

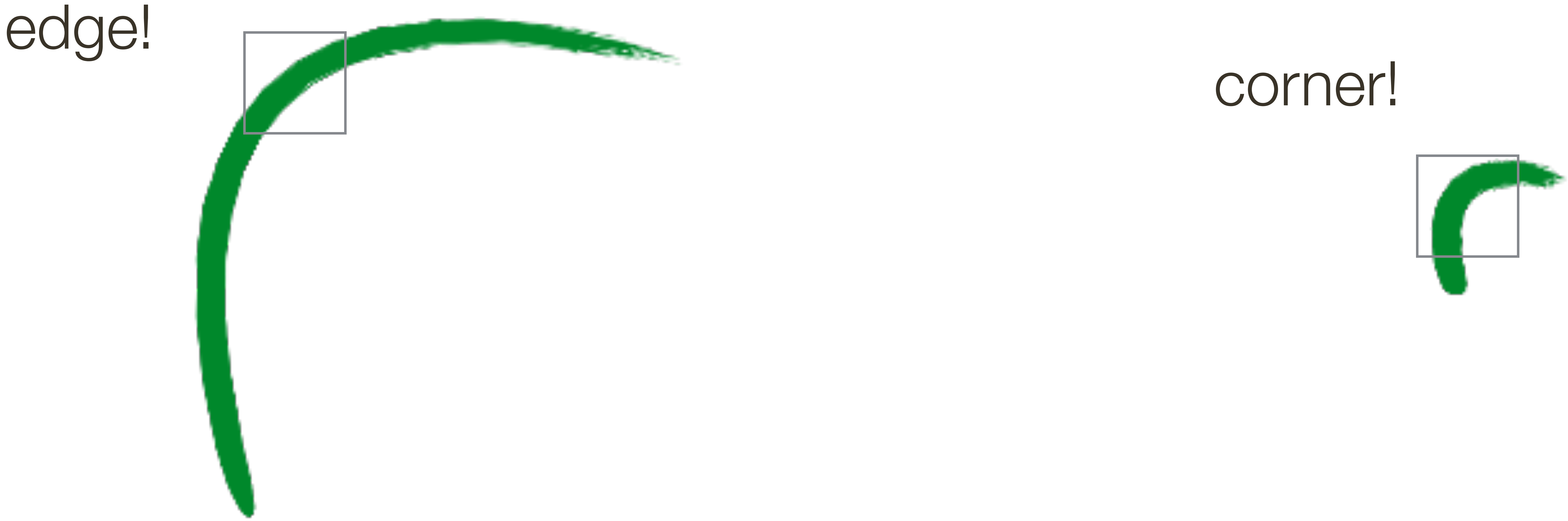
  — Harris also checks that ratio of λs is not too high

Harris Corner Detection





Properties: NOT Invariant to Scale Changes



edge!

corner!

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Sample Questions: Corners

The Harris corner detector is stable under some image transformations (features are considered stable if the same locations on an object are typically selected in the transformed image).

True or **false**: The Harris corner detector is stable under image blur.

Texture

We will look at two main questions:

1. How do we represent texture?
   → Texture **analysis**

2. How do we generate new examples of a texture?
   → Texture **synthesis**

Texture **Synthesis**

Why might we want to synthesize texture?

1. To fill holes in images (**inpainting**)

— Art directors might want to remove telephone wires. Restorers might want to remove scratches or marks.

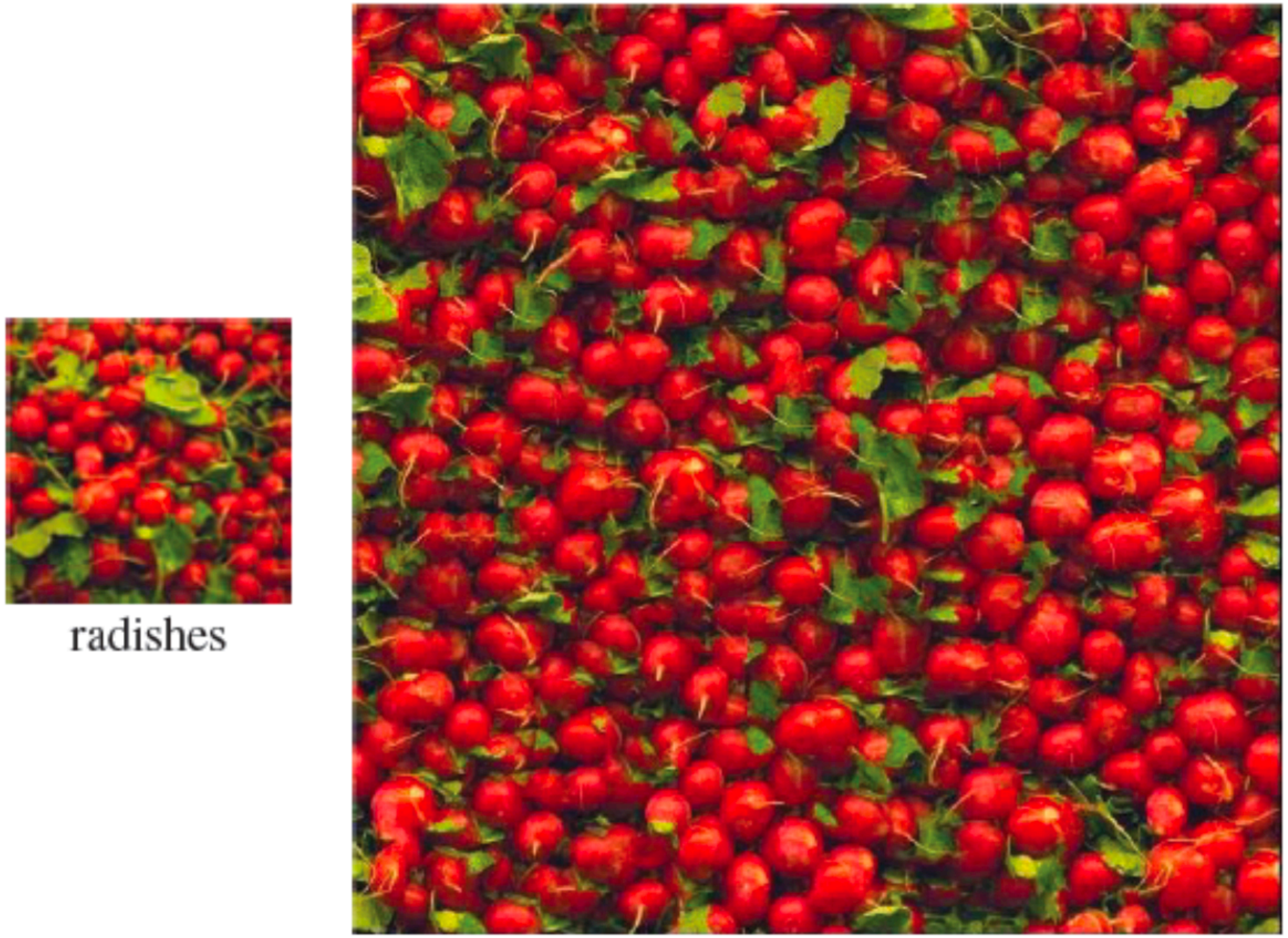
— We need to find something to put in place of the pixels that were removed

— We synthesize regions of texture that fit in and look convincing

2. To produce large quantities of texture for computer graphics

— Good textures make object models look more realistic

Texture **Synthesis**

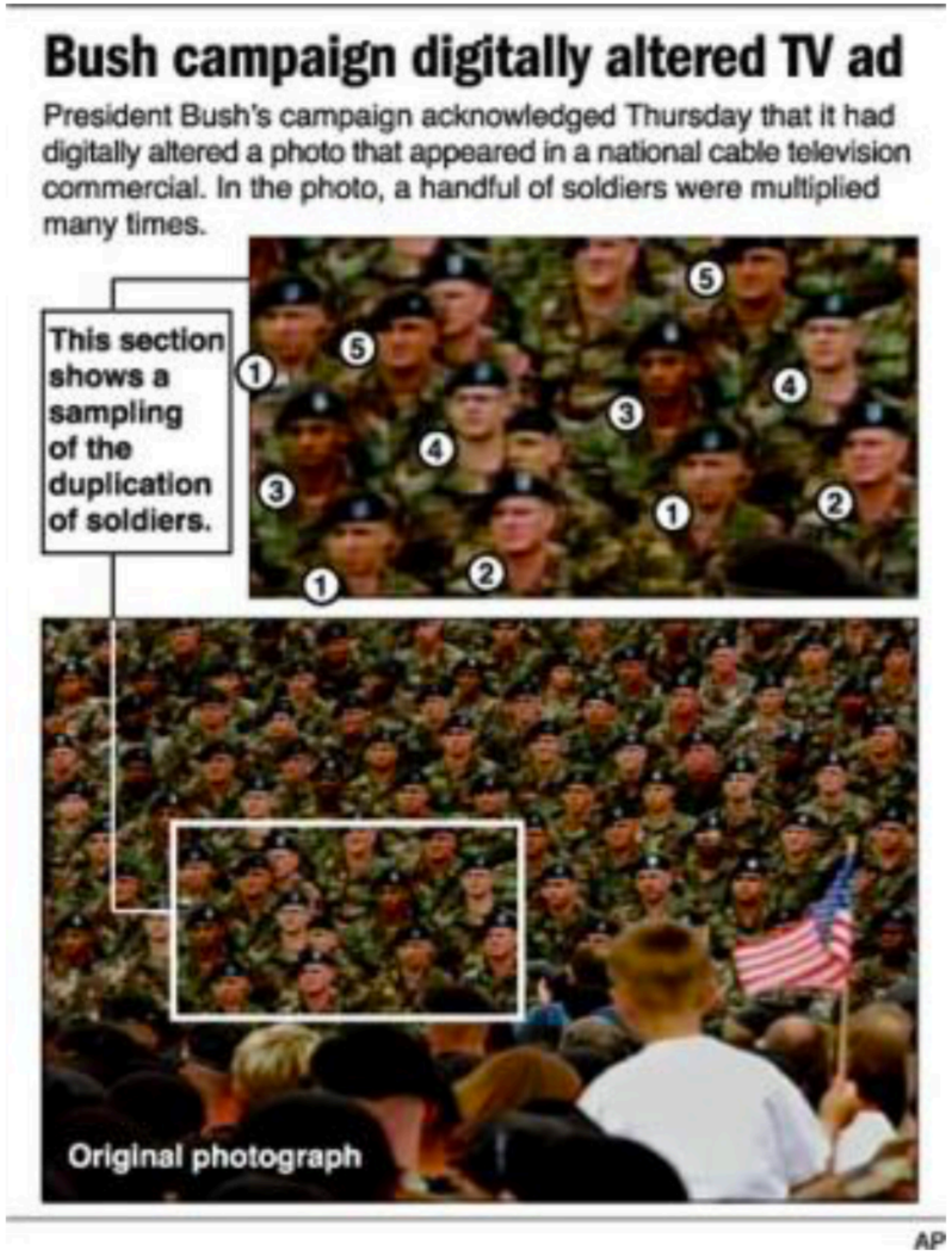


radishes

lots more radishes

Szeliski, Fig. 10.49

Texture **Synthesis**



Bush campaign digitally altered TV ad

President Bush's campaign acknowledged Thursday that it had digitally altered a photo that appeared in a national cable television commercial. In the photo, a handful of soldiers were multiplied many times.

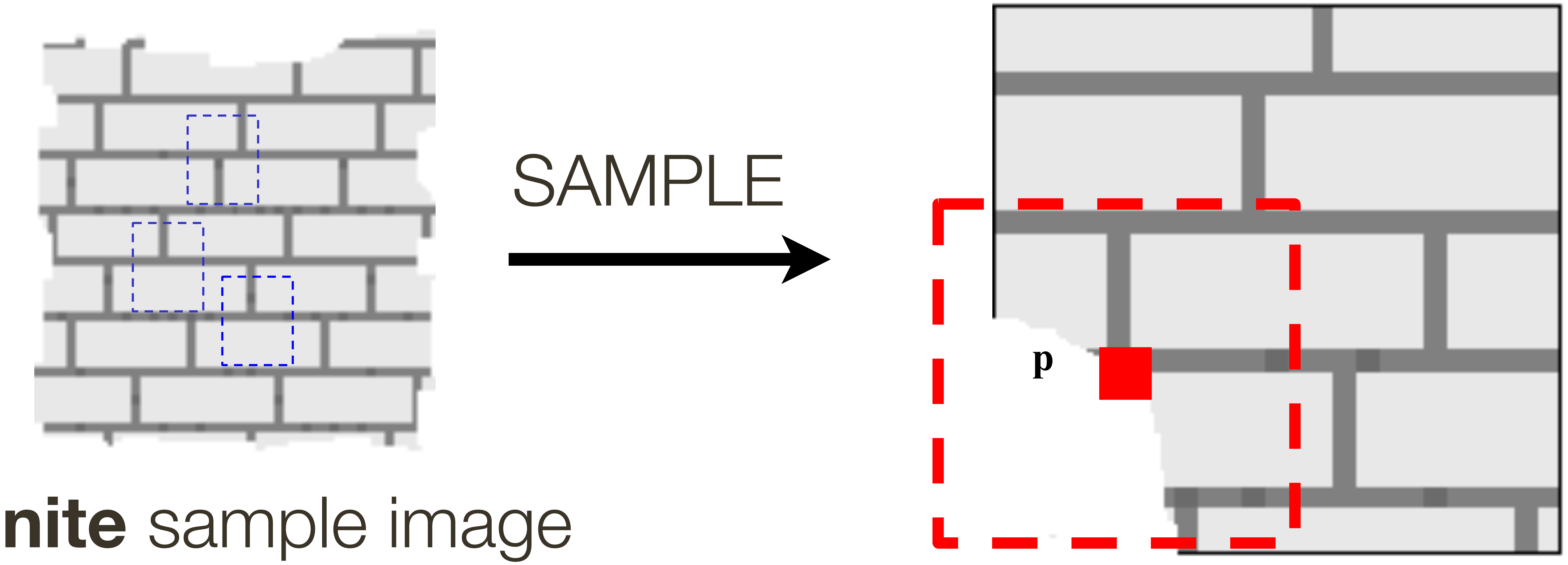
This section shows a sampling of the duplication of soldiers.

Original photograph

AP

Photo Credit: Associated Pres

Efros and Leung: Synthesizing One Pixel



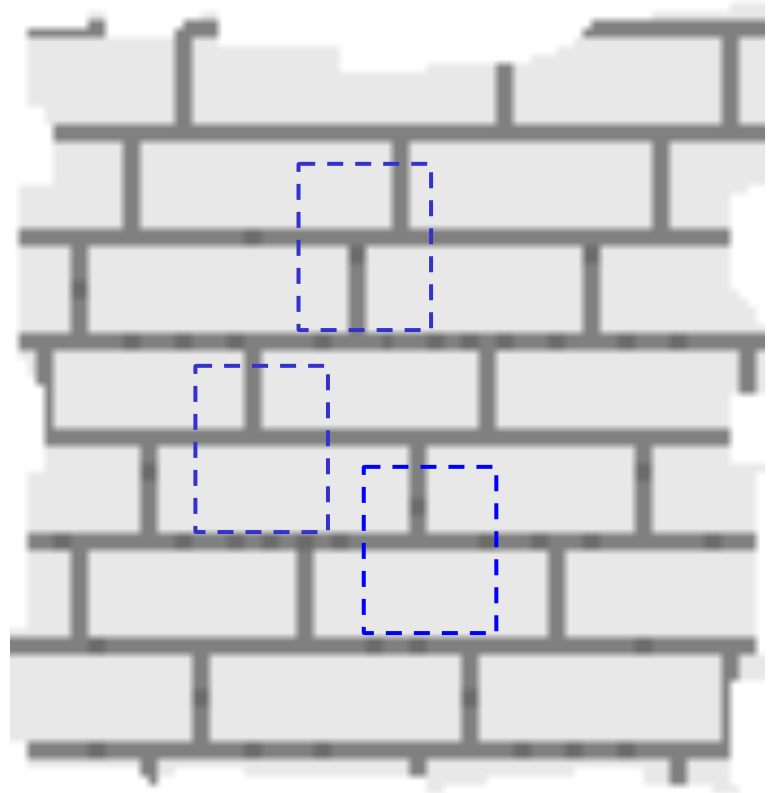
Infinite sample image

— What is **conditional** probability distribution of *p*, given the neighbourhood window?

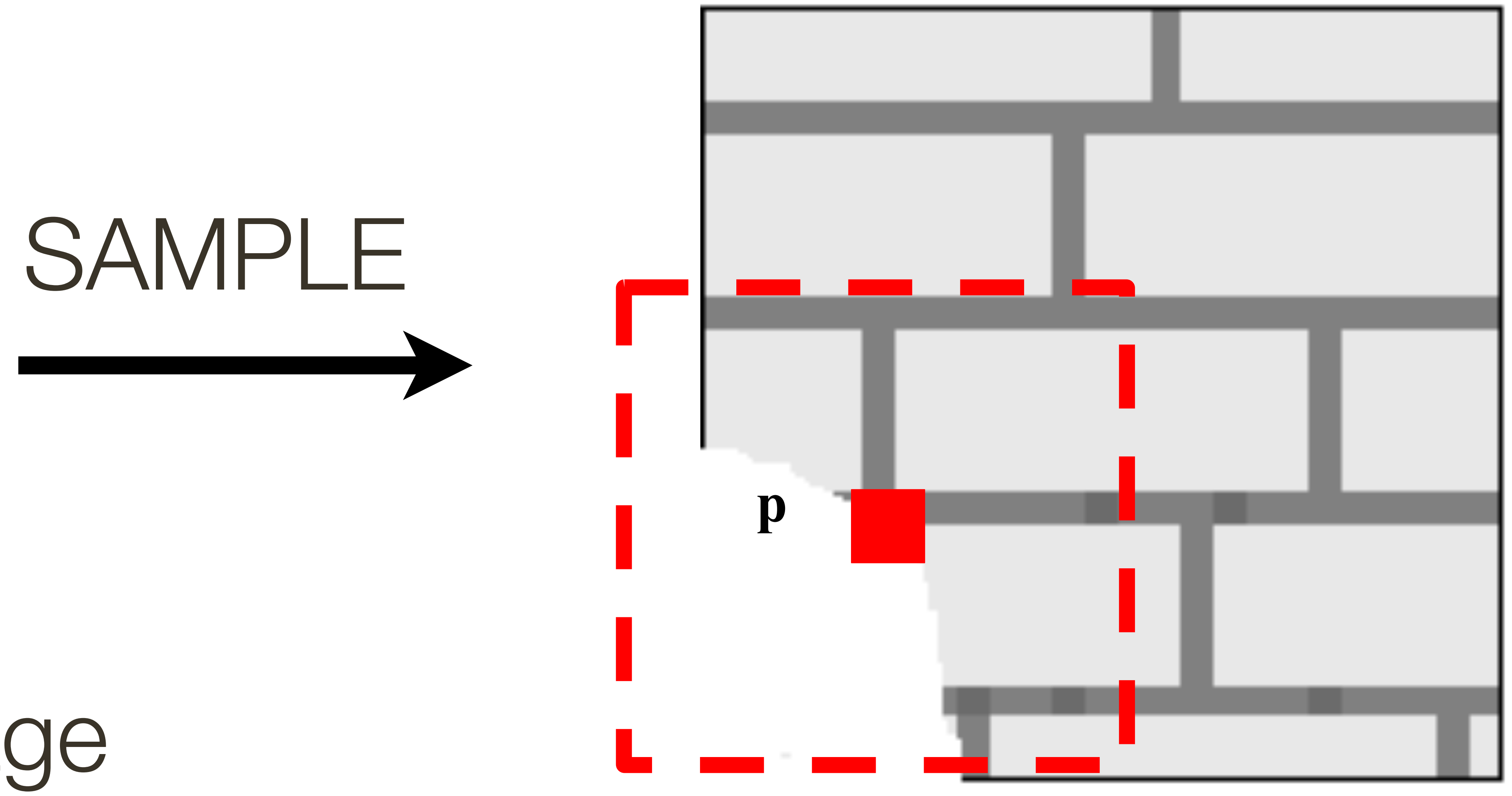
— Directly search the input image for all such neighbourhoods to produce a **histogram** for *p*

— To **synthesize** *p*, pick one match at random

Efros and Leung: Synthesizing One Pixel



Infinite sample image



— Since the sample image is finite, an exact neighbourhood match might not be present

— Find the **best match** using SSD error, weighted by Gaussian to emphasize local structure, and take all samples within some distance from that match

Efros and Leung: Synthesizing Many Pixels

For multiple pixels, "grow" the texture in layers

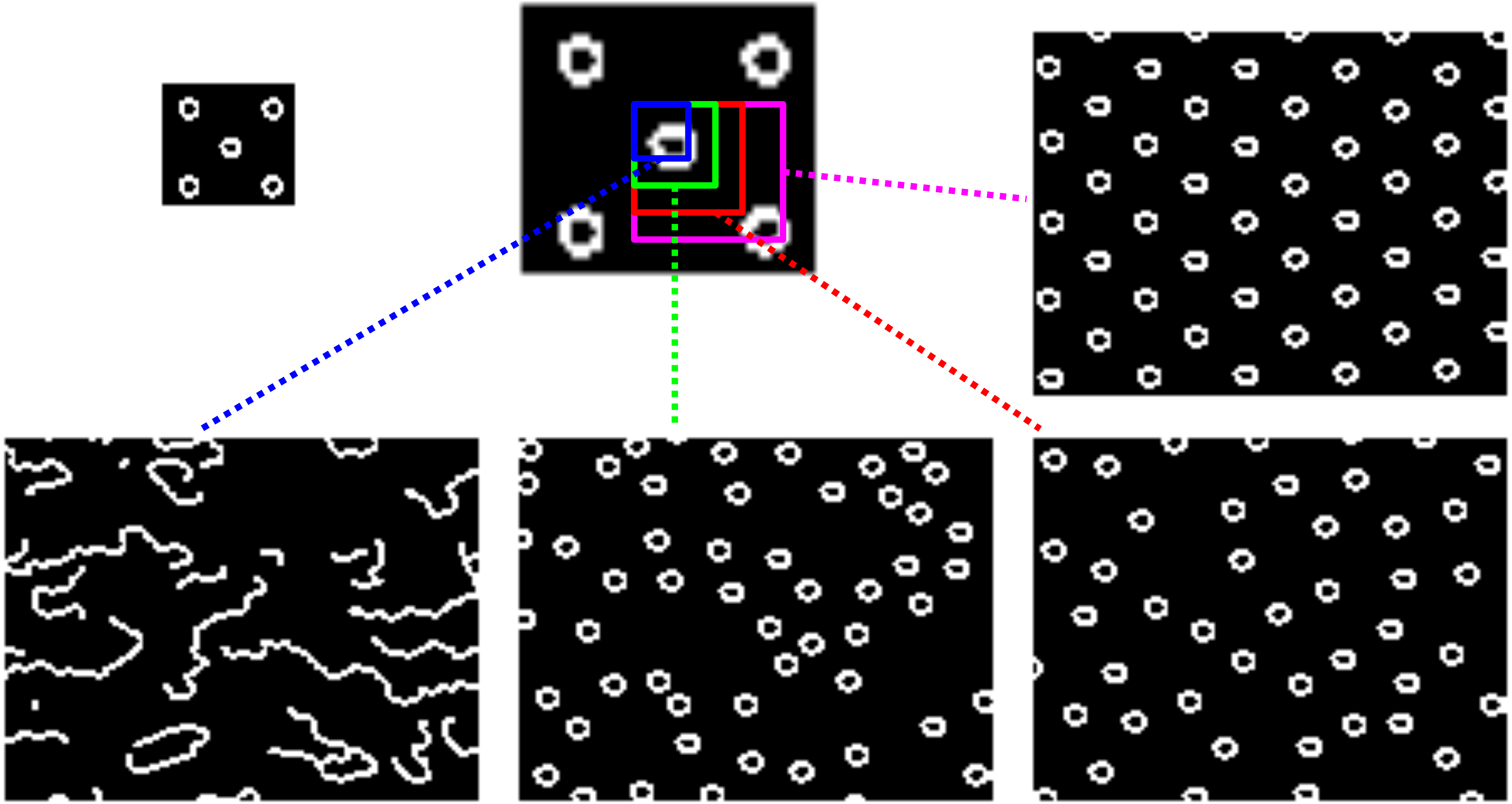
— In the case of hole-filling, start from the edges of the hole

For an interactive demo, see

https://una-dinosauria.github.io/efros-and-leung-js/

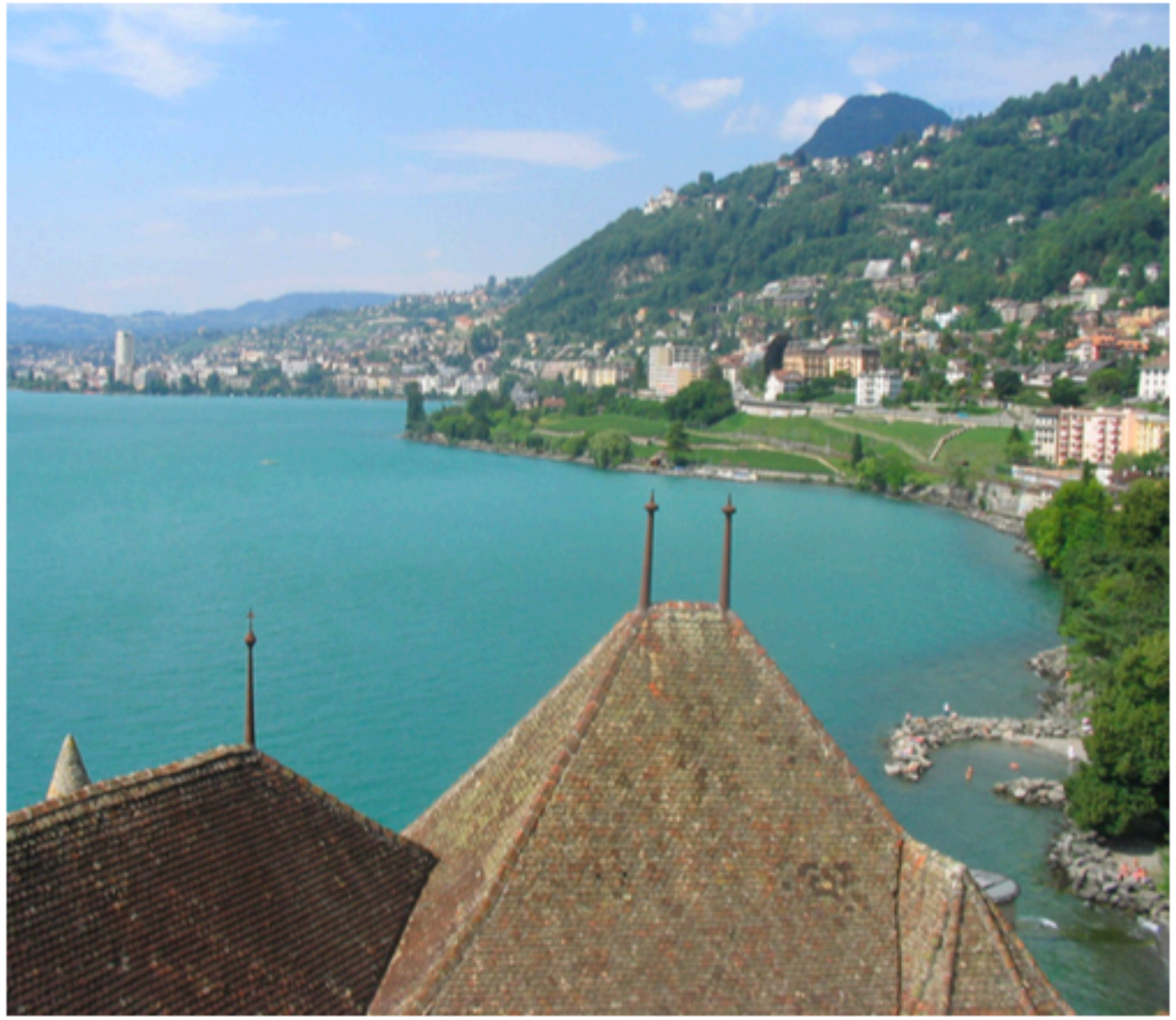
(written by Julieta Martinez, a previous CPSC 425 TA)

Randomness Parameter

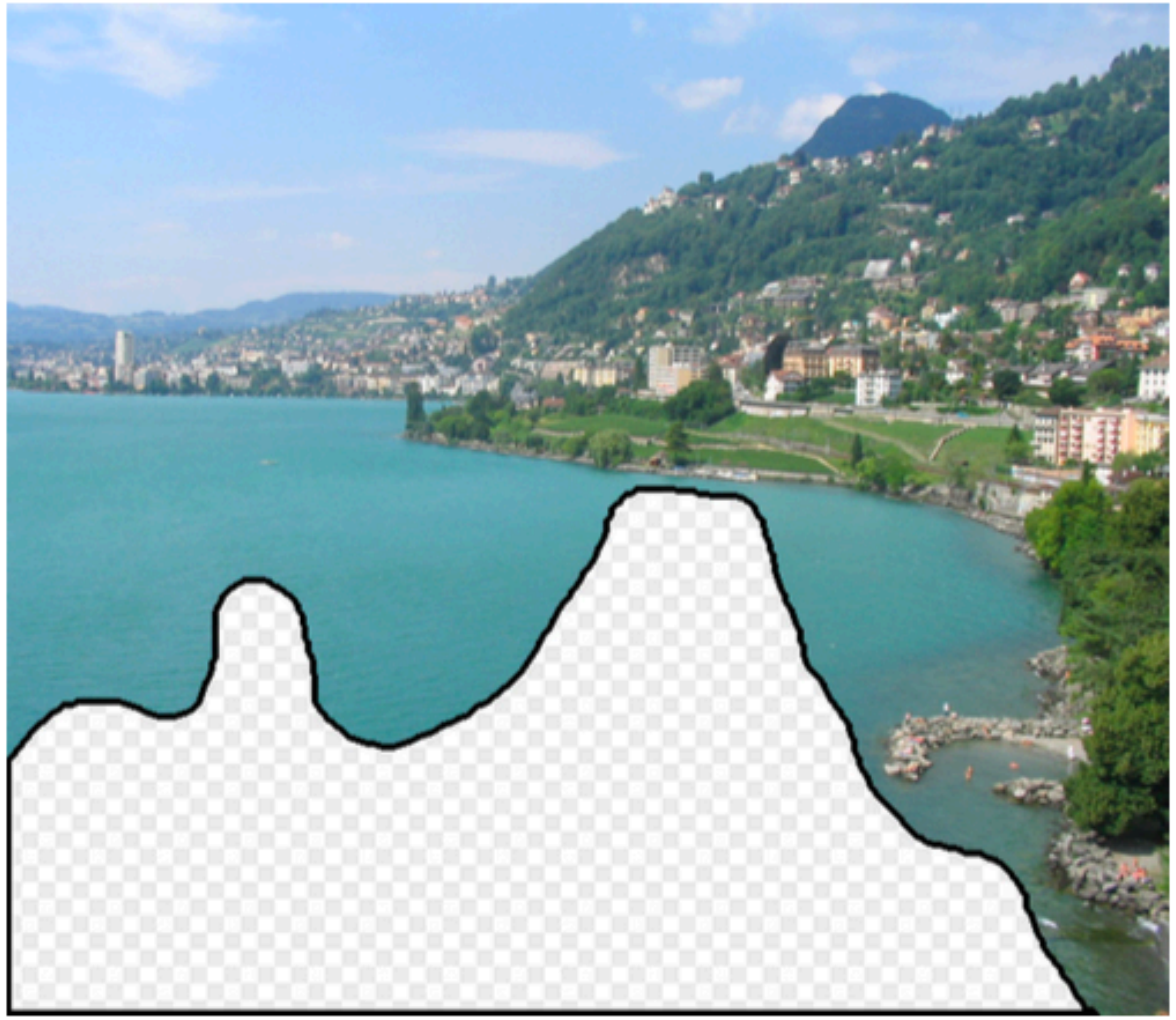


Slide Credit: http://graphics.cs.cmu.edu/people/efros/research/NPS/efros-iccv99.ppt

“**Big** Data” Meets Inpainting



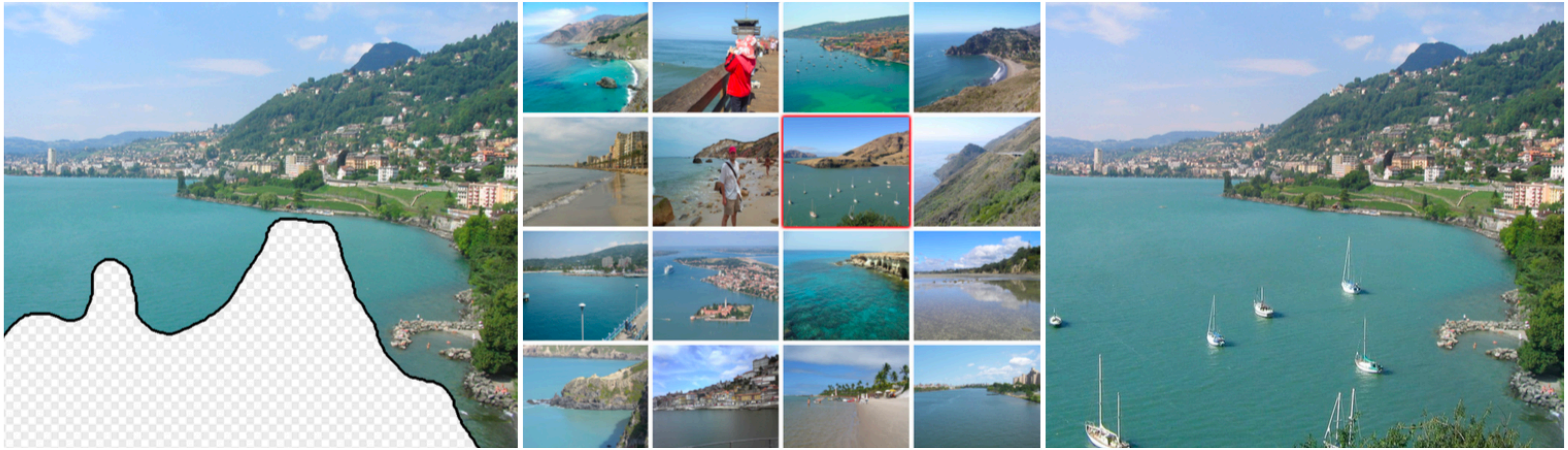
Original Image



Input

Figure Credit: Hays and Efros 2007

“**Big** Data” Meets Inpainting



Input

Scene Matches

Output

Figure Credit: Hays and Efros 2007

“**Big** Data” Meets Inpainting

Algorithm sketch (Hays and Efros 2007):

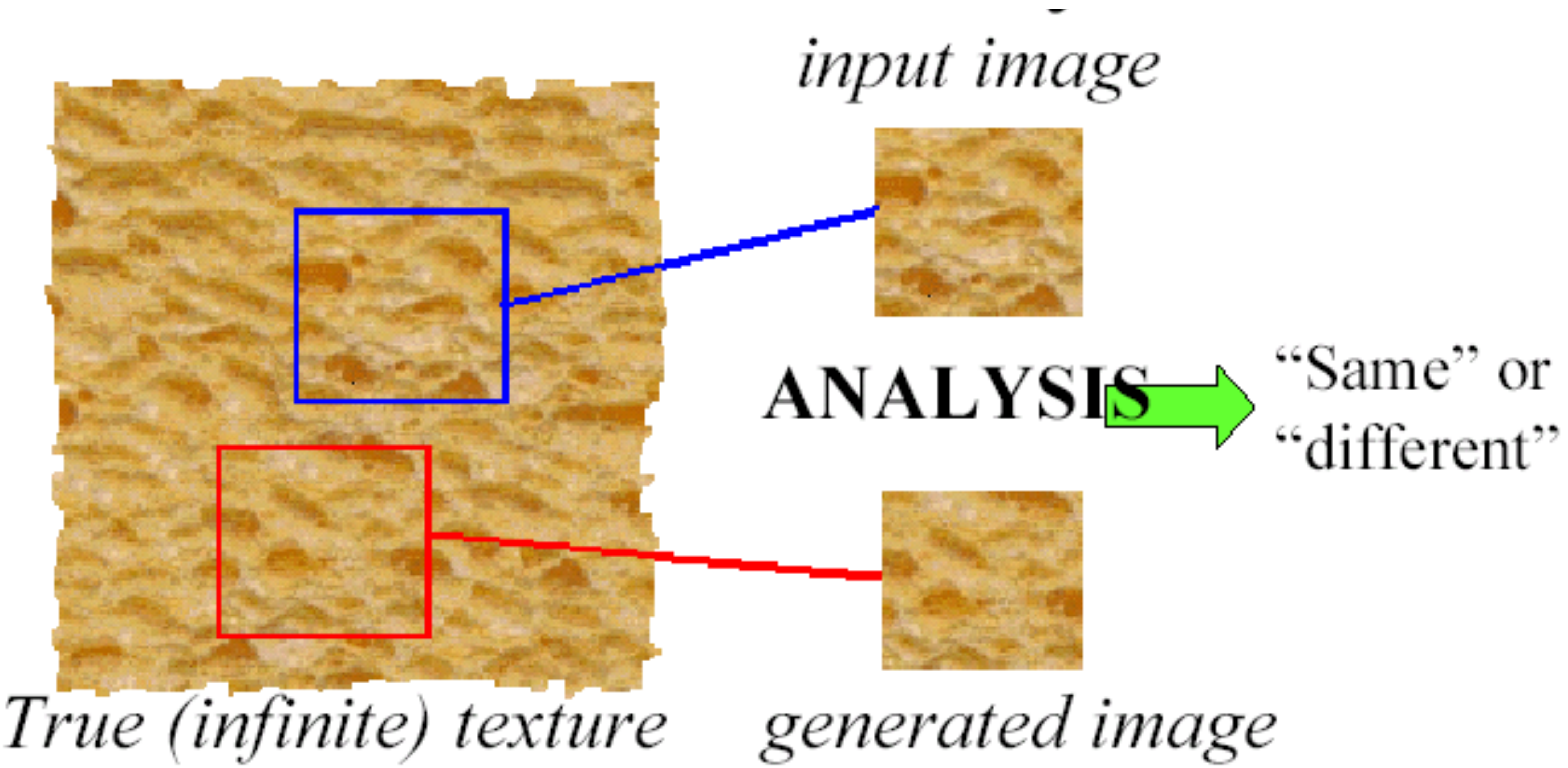
1. Create a short list of a few hundred “best matching" images based on global image statistics

2. Find patches in the short list that match the context surrounding the image region we want to fill

3. Blend the match into the original image

Purely **data-driven**, requires no manual labeling of images

Goal of Texture **Analysis**



Compare textures and decide if they're mae of the same "stuff"

Credit: Bill Freeman

Texture **Representation**

Observation: Textures are made up of generic sub-elements, repeated over a region with similar statistical properties

Idea: Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region

Texture **Representation**

Observation: Textures are made up of generic sub-elements, repeated over a region with similar statistical properties

Idea: Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region

Question: What filters should we use?

Answer: Human vision suggests spots and oriented edge filters at a variety of different orientations and scales

Texture **Representation**

Observation: Textures are made up of generic sub-elements, repeated over a region with similar statistical properties

Idea: Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region

Question: What filters should we use?

Answer: Human vision suggests spots and oriented edge filters at a variety of different orientations and scales

Question: How do we "summarize"?

Answer: Compute the mean or maximum of each filter response over the region
— Other statistics can also be useful

Texture **Representation**

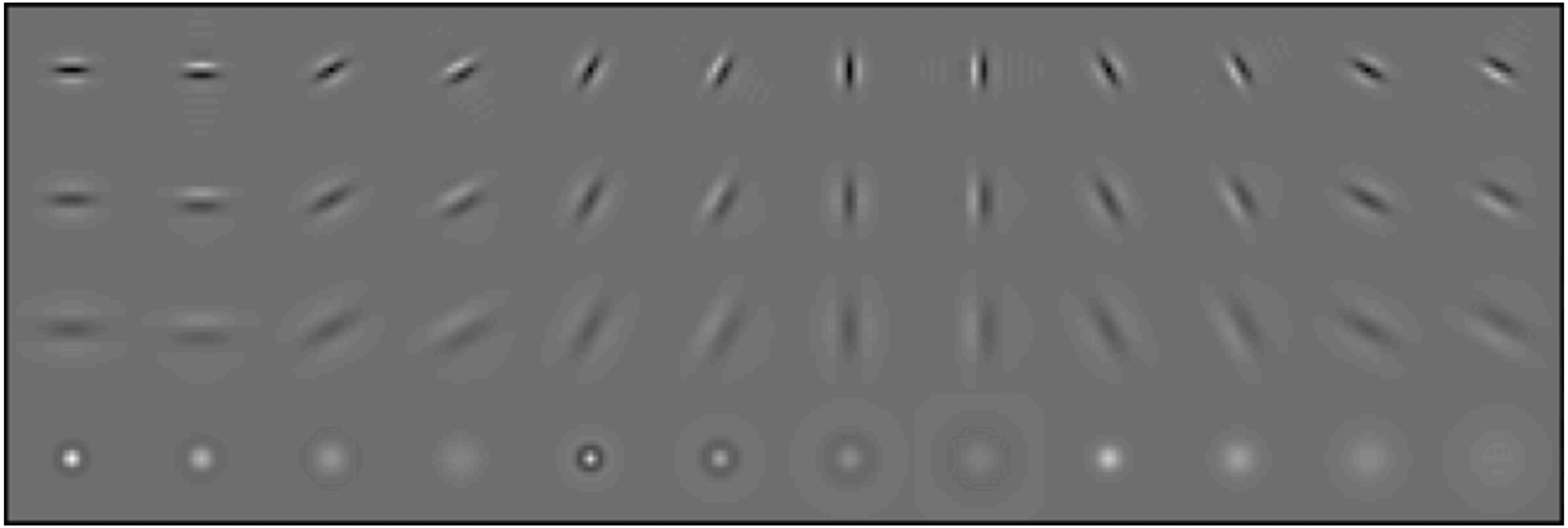
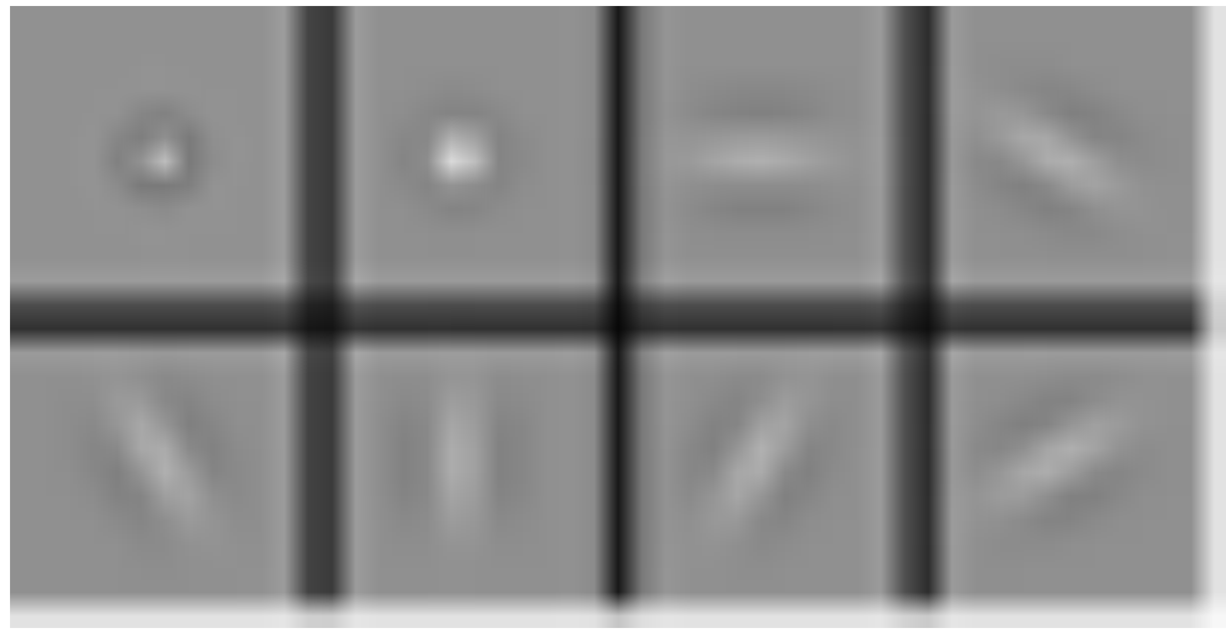
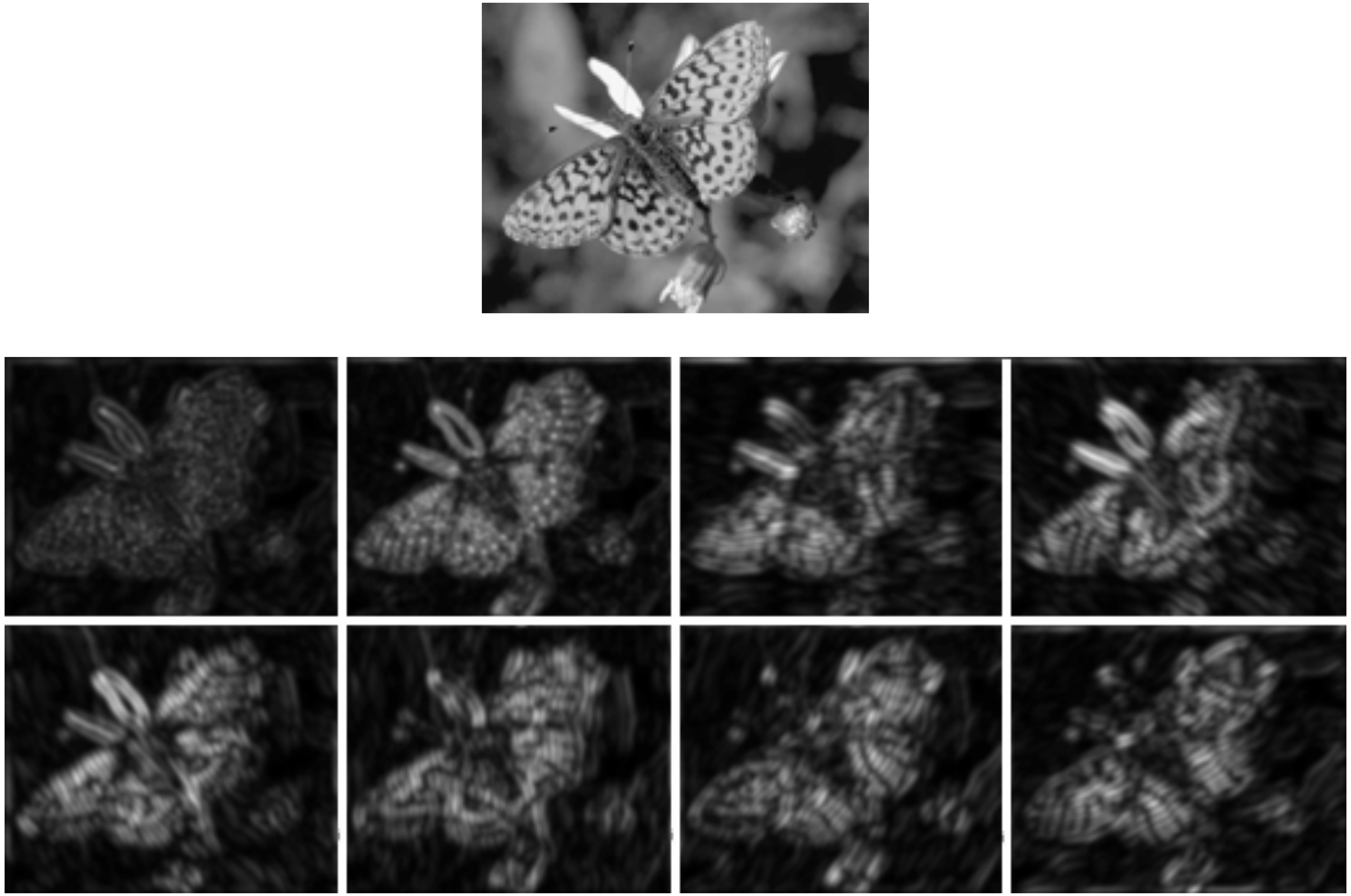


Figure Credit: Leung and Malik, 2001

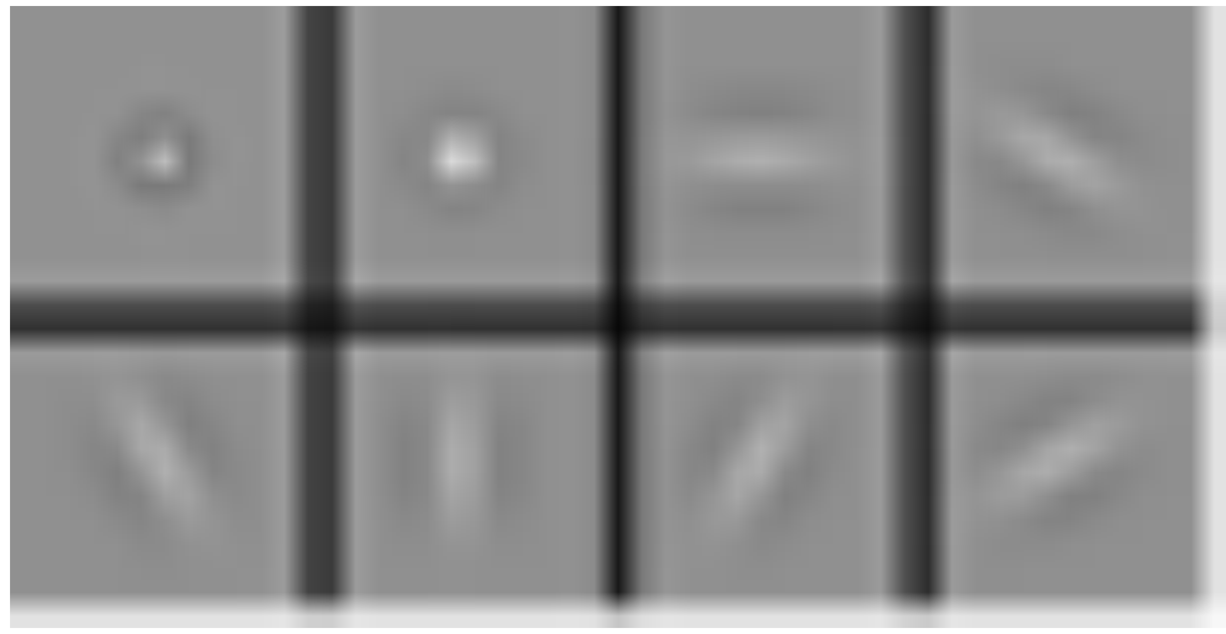
Spots and Bars (Fine Scale)

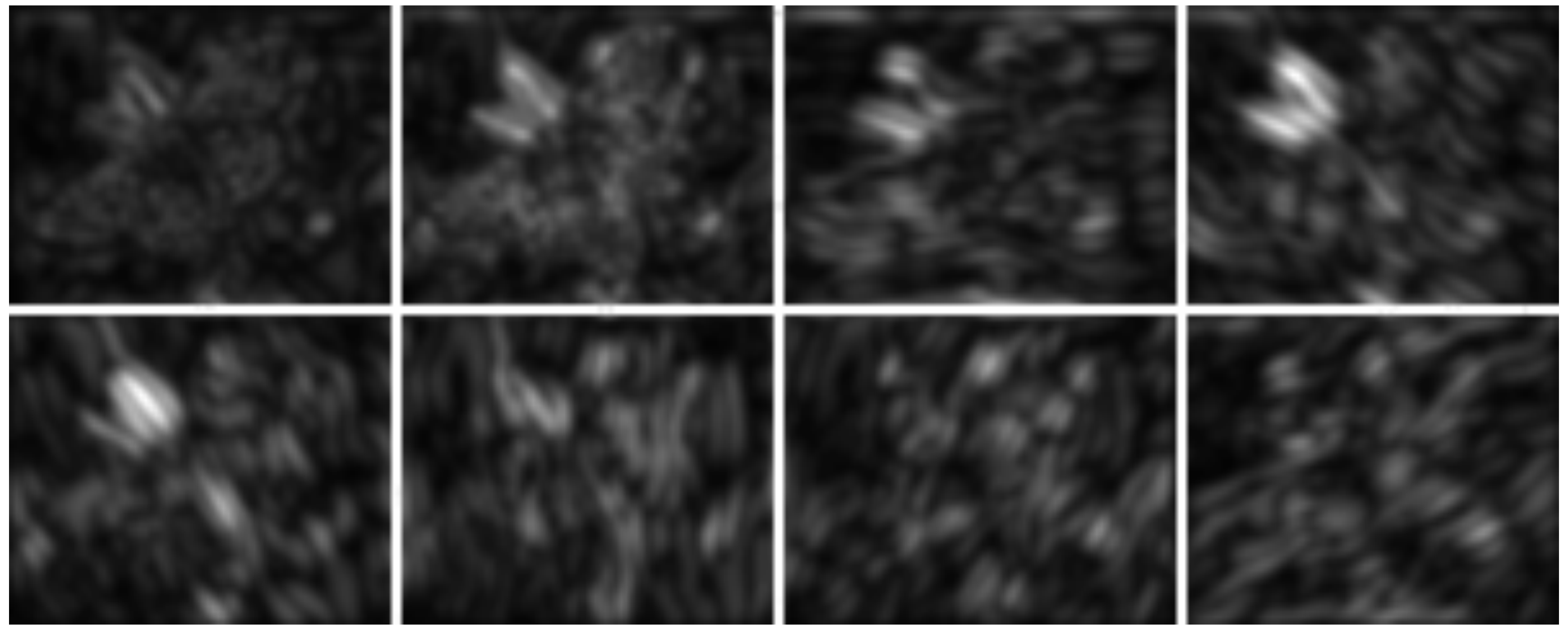




Forsyth & Ponce (1st ed.) Figures 9.3–9.4

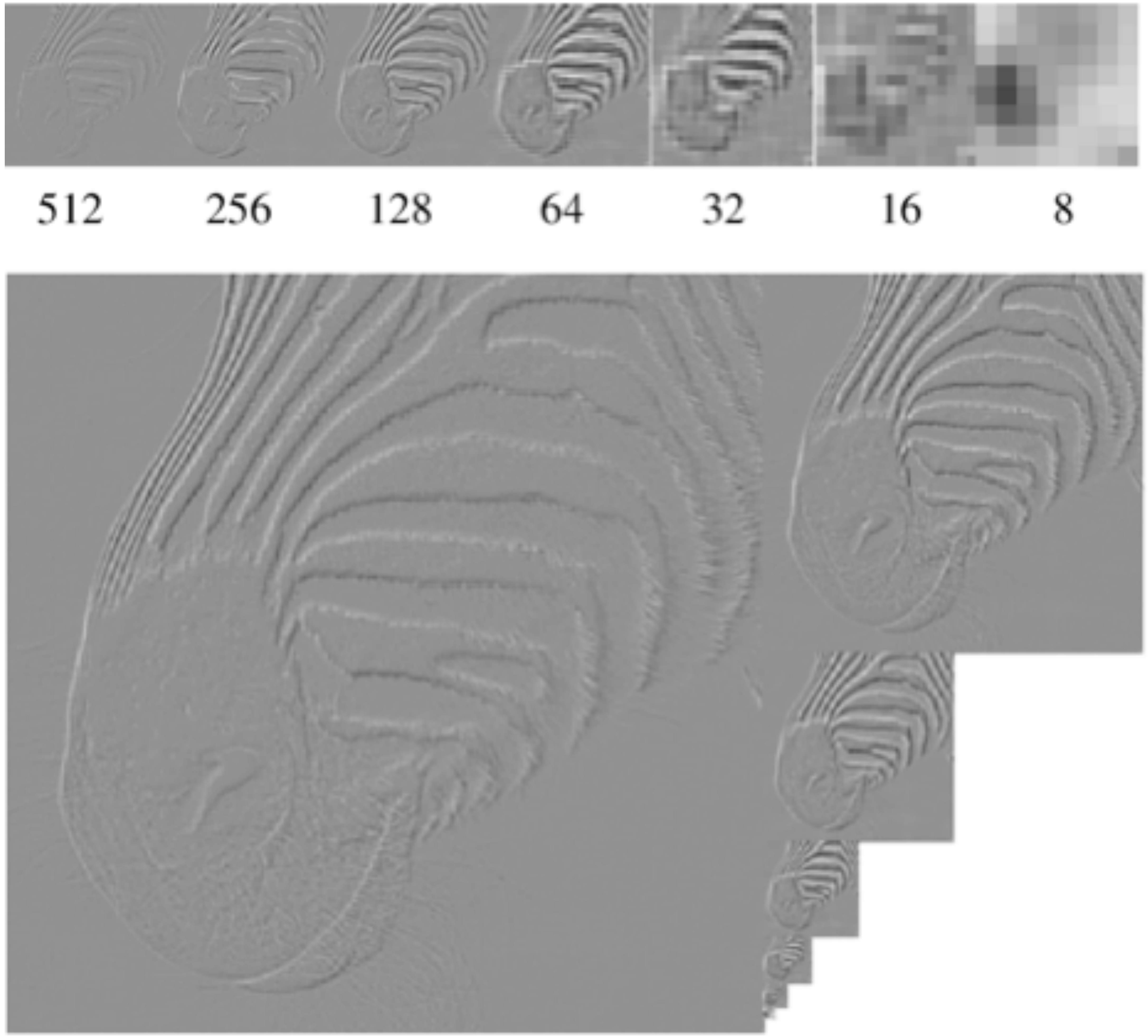
Spots and Bars (Coarse Scale)





Forsyth & Ponce (1st ed.) Figures 9.3 and 9.5

Laplacian Pyramid



Laplacian Pyramid

Building a **Laplacian** pyramid:

— Create a Gaussian pyramid

— Take the difference between one Gaussian pyramid level and the next (before subsampling)

Properties

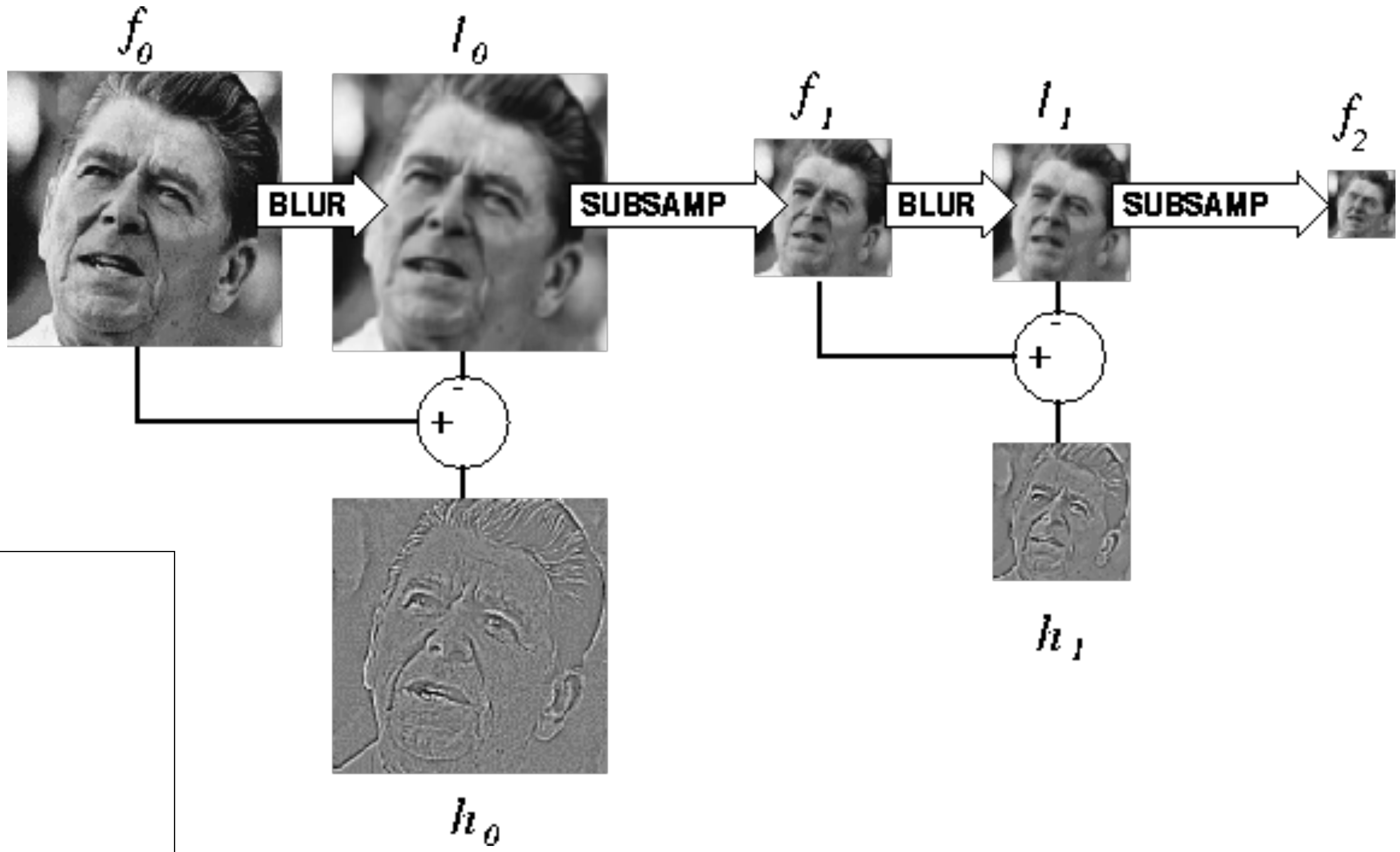
— Also known as the difference-of-Gaussian (DOG) function, a close approximation to the Laplacian

— It is a band pass filter – each level represents a different band of spatial frequencies

Reconstructing the original image:

— Reconstruct the Gaussian pyramid starting at top

Constructing a **Laplacian** Pyramid

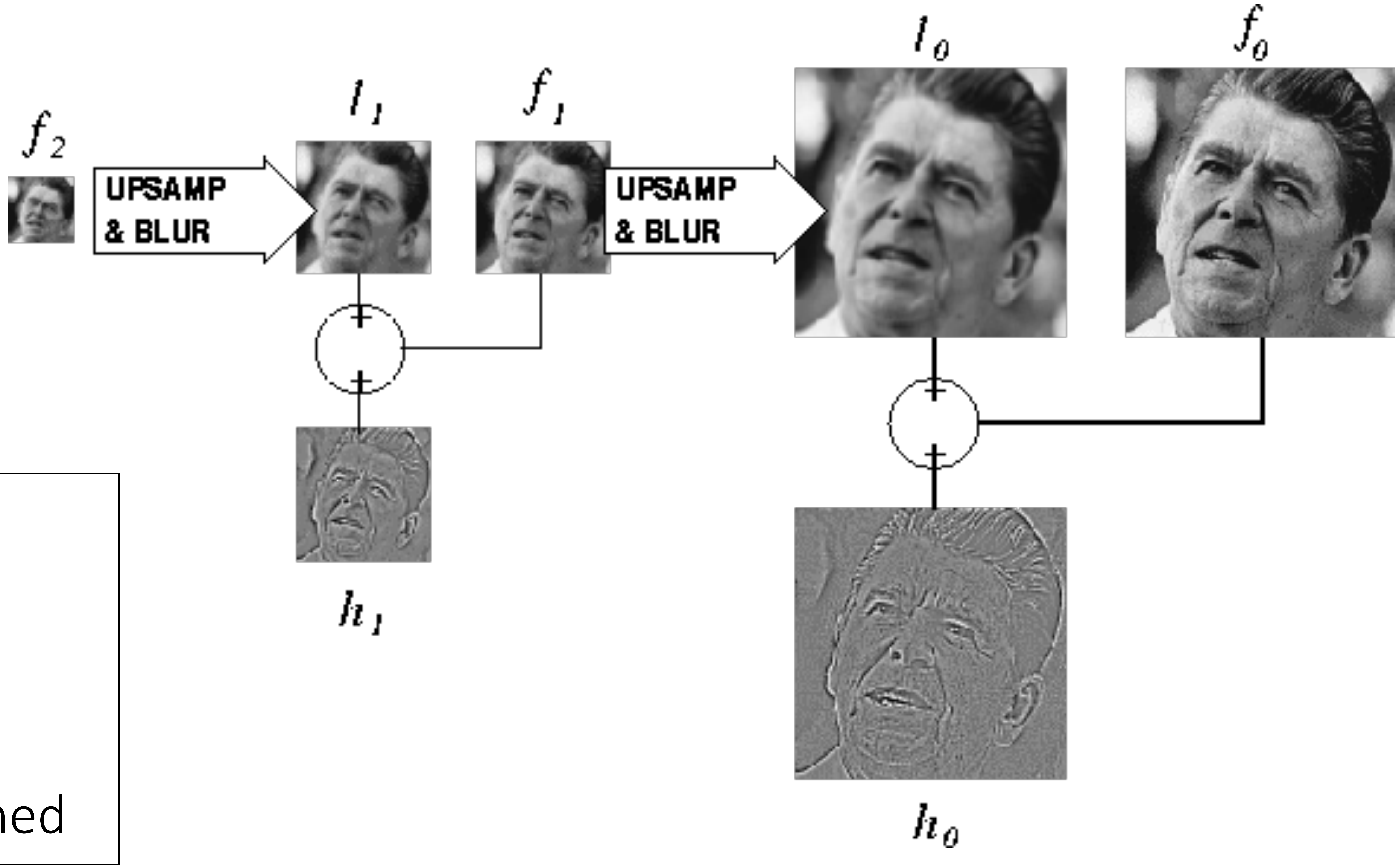


Algorithm

```
repeat:
    filter
    compute residual
    subsample
until min resolution reached
```

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Reconstructing the Original Image

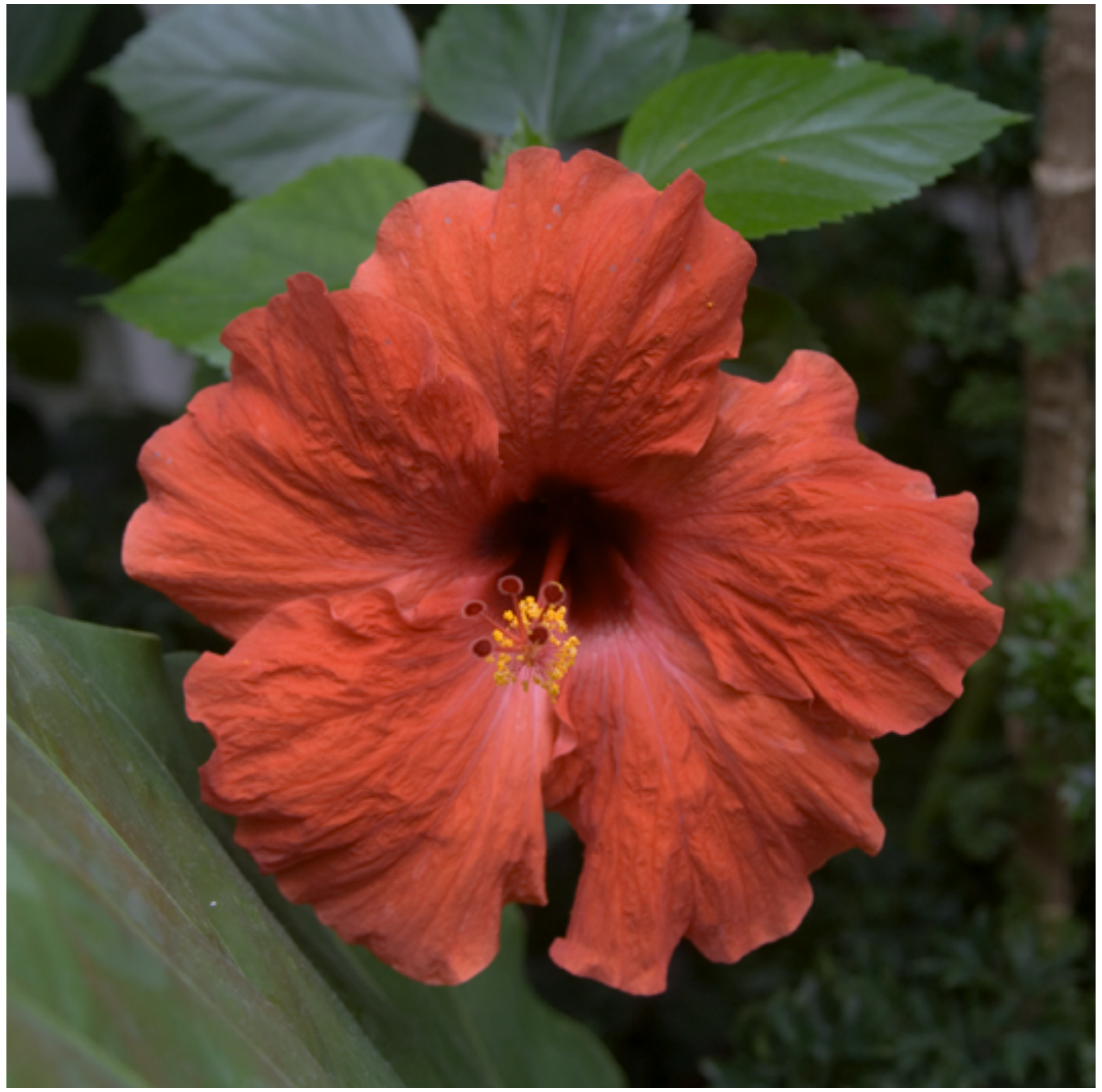


Algorithm

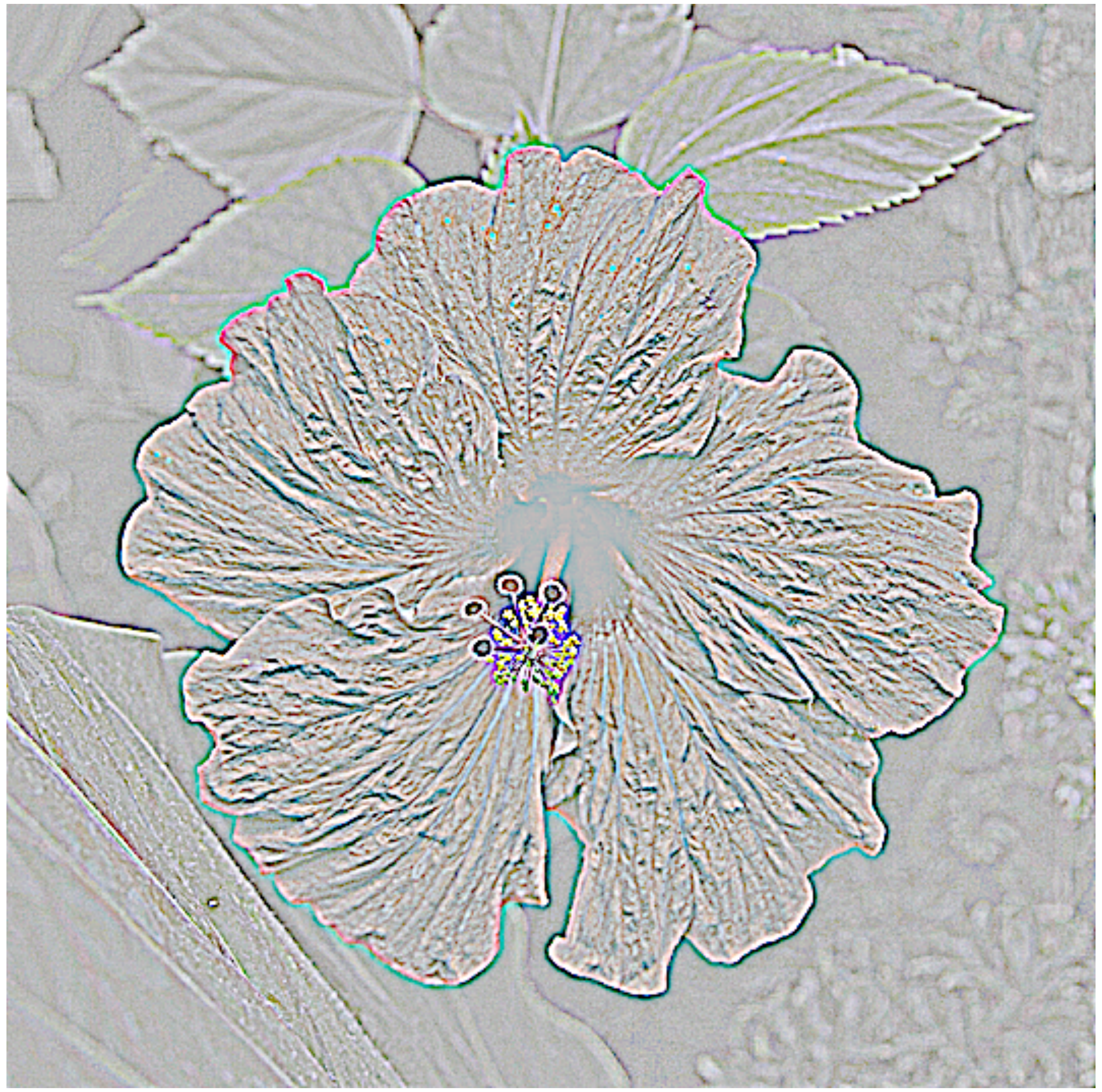
```
repeat:
    upsample
    sum with residual
until orig resolution reached
```

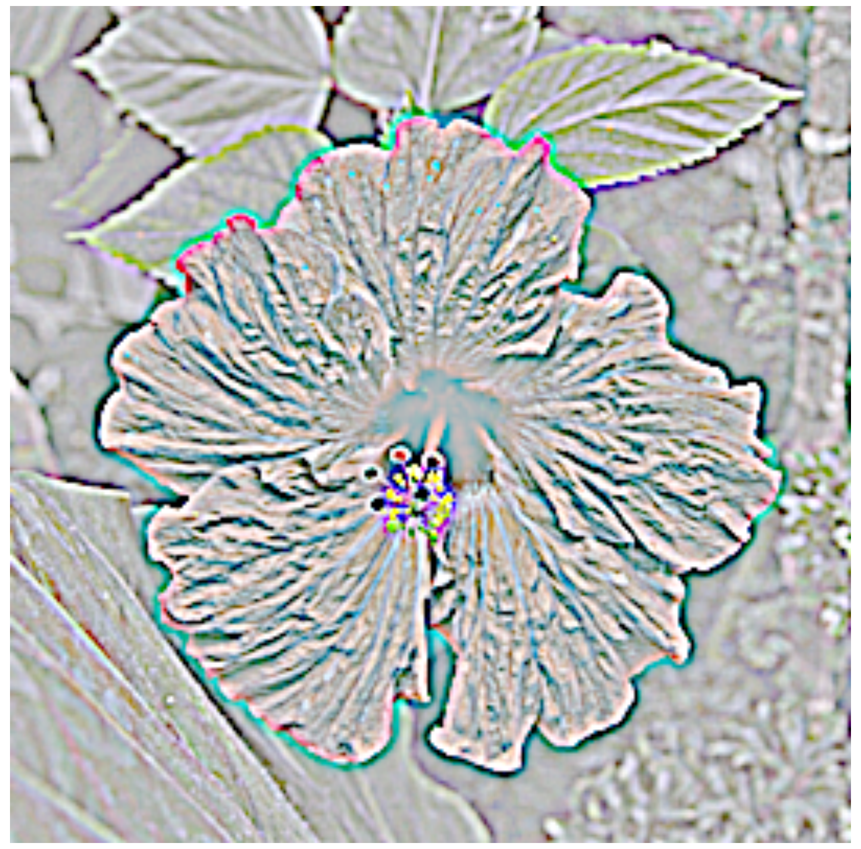
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

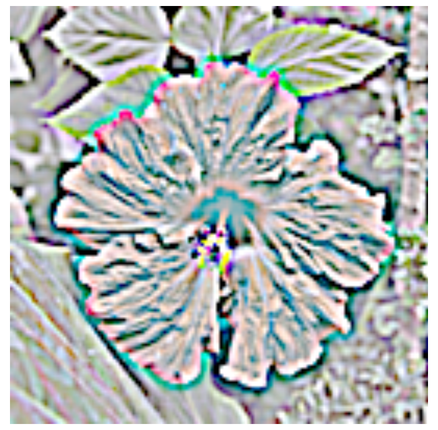
Gaussian vs **Laplacian** Pyramid



Shown in opposite order for space



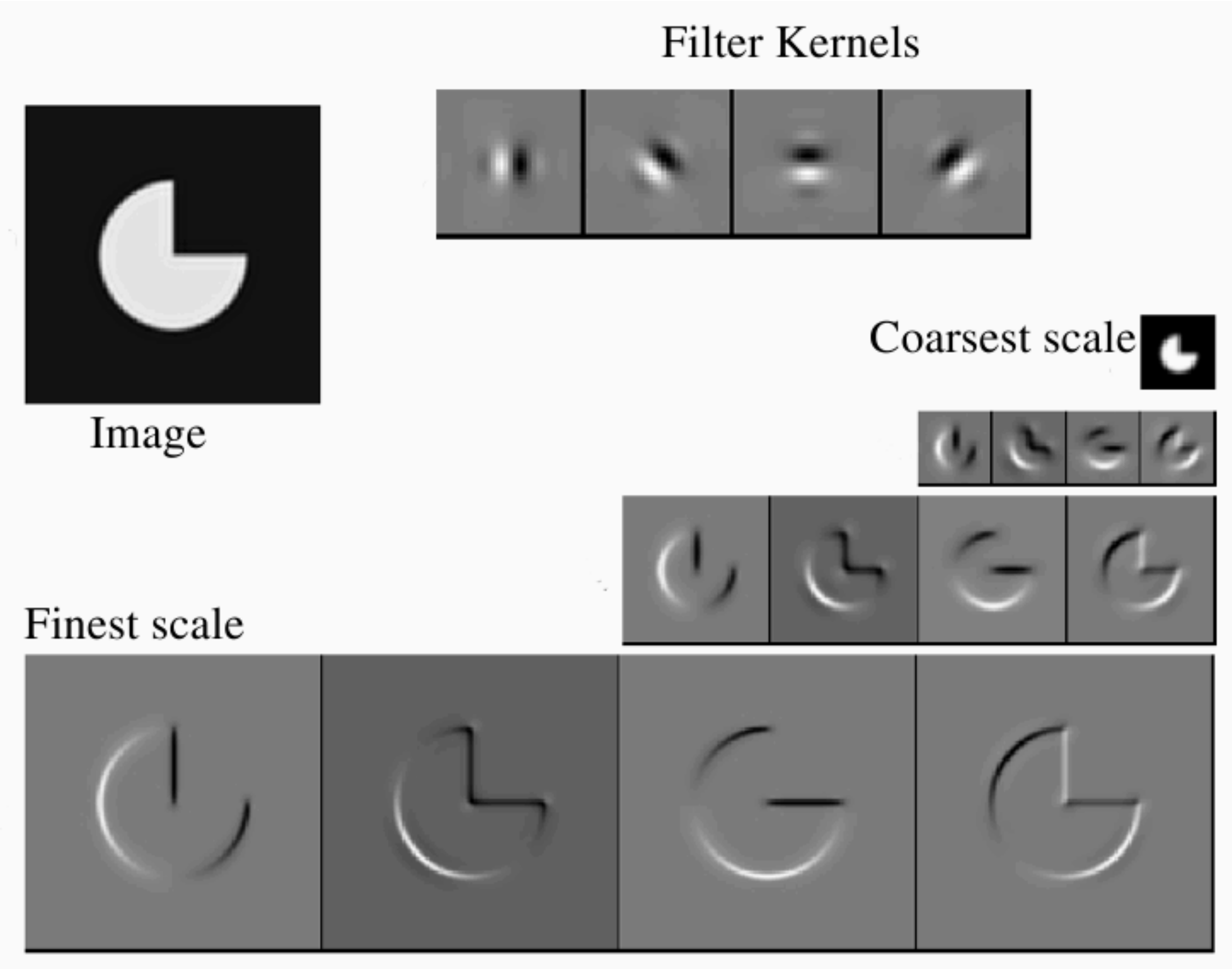




Which one takes more space to store?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Oriented Pyramids



Forsyth & Ponce (1st ed.) Figure 9.13

Final Texture Representaation

Steps:

1. Form a Laplacian and oriented pyramid (or equivalent set of responses to filters at different scales and orientations)

2. Square the output (makes values positive)

3. Average responses over a neighborhood by blurring with a Gaussian

4. Take statistics of responses

— Mean of each filter output

— Possibly standard deviation of each filter

Sample Question: Texture

How does the top-most image in a Laplacian pyramid differ from the others?