THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision

Midterm Review

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

Midterm Details

Closed book, (simple) **calculators** allowed

Format similar to posted practice problems

- Part A: Multiple-part true/false — no partial credit
- Part B: Short answer — partial credit

No coding questions

No complex math questions

- Meaning problems will not involve numbers that are difficult to work with
- You can leave answers in non-simplified form (e.g., sqrt(3))

Midterm Review: Study materials

Lectures 1–11 **slides**

Lecture notes

(Optional) **readings** from Szeliski / Forsyth and Ponce

Assignments 1–2

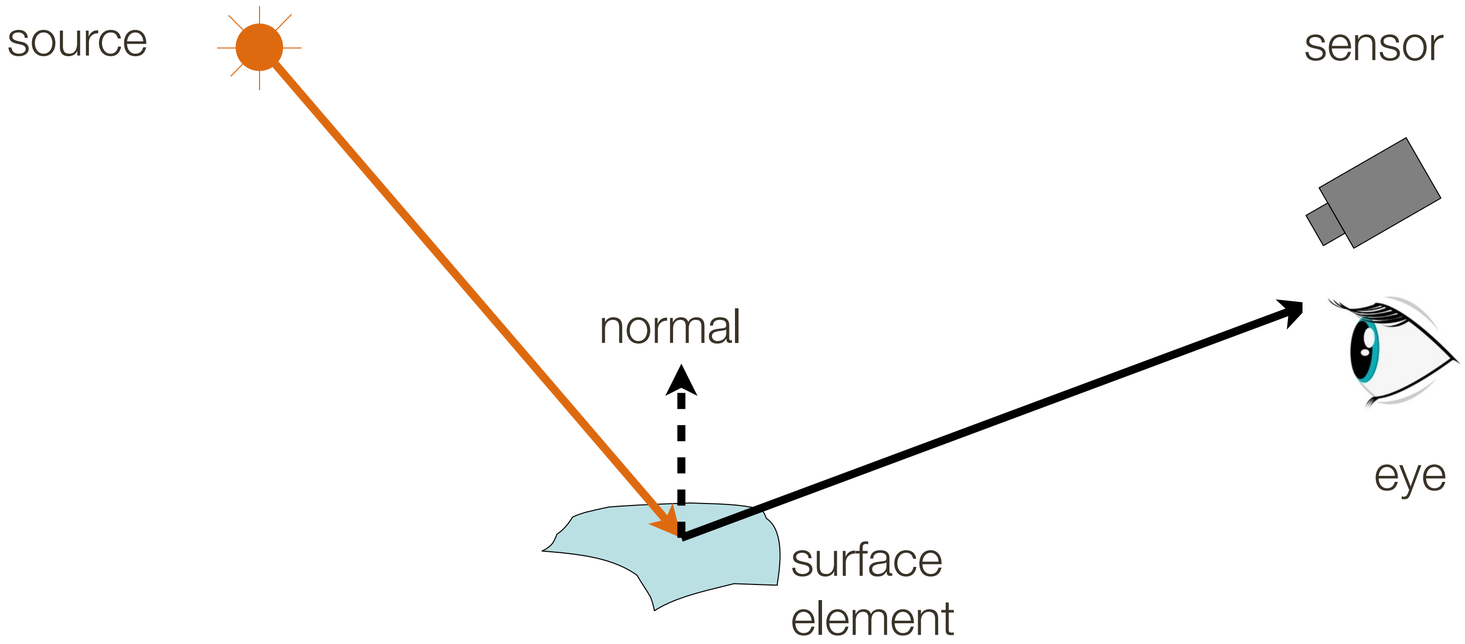
Practice quizzes on Canvas

Practice problems / solutions on Canvas

Overview: Image Formation, Cameras and Lenses

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

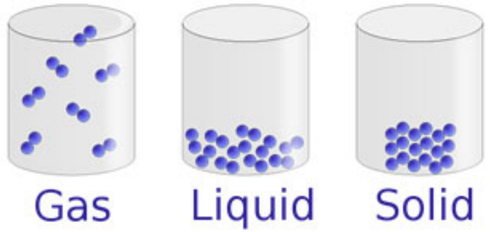
- **Lightening** condition
- Scene **geometry**
- **Surface** properties
- Camera **optics**



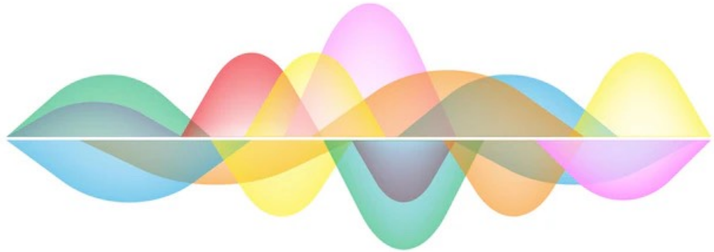
Sensor (or eye) **captures amount of light** reflected from the object

Light

Behaves like **particles**? photons



Behaves as **waves**?

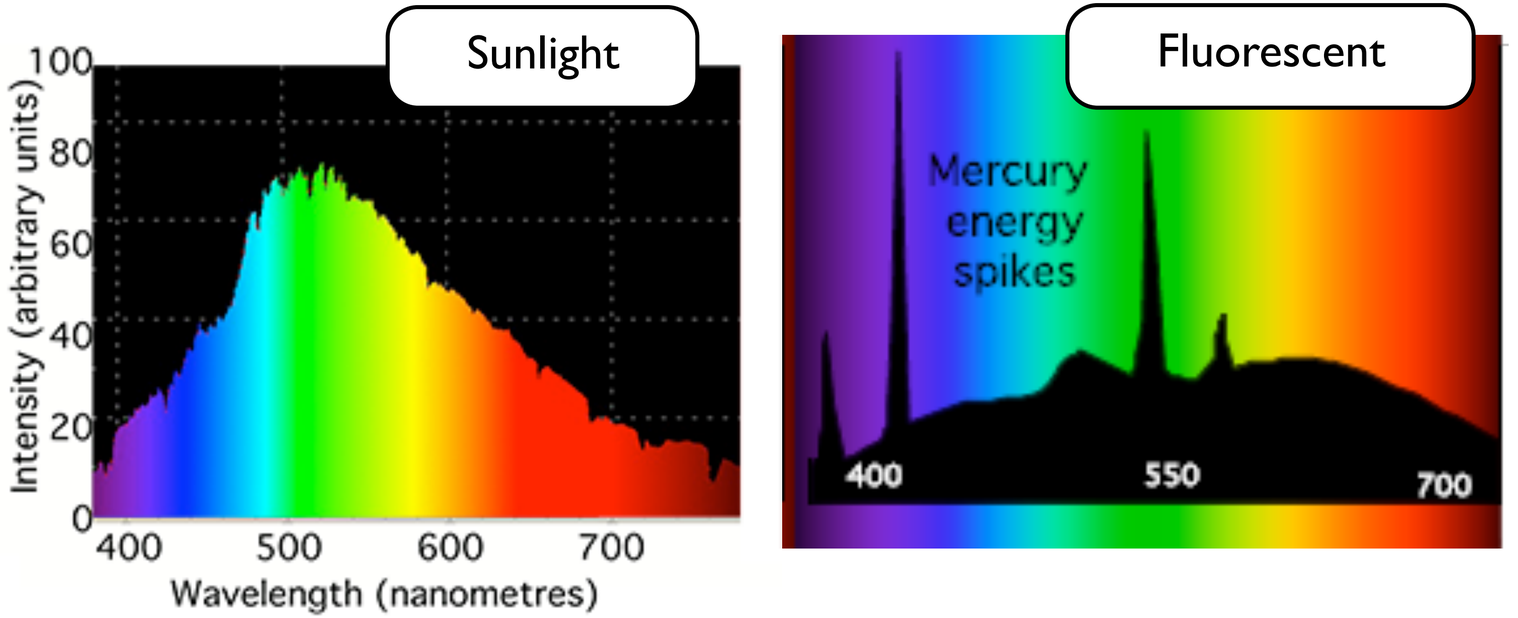


Wave-particle Duality: light exhibit particle or wave properties according to the experimental circumstances

Sir Isaac Newton

Christiaan Huygens

Spectral Power Distribution



The **spectral distribution of energy** in a light ray determines its colour

— e.g., you can have pure yellow or mixture of red and green

Surface **reflects** light energy according to a spectral distribution as well

The combination of **incident** and **reflectance** spectra determines **observed colour**

[scratchapixel.com]

Diffuse/Lambertian vs **Specular/Mirror** Surfaces

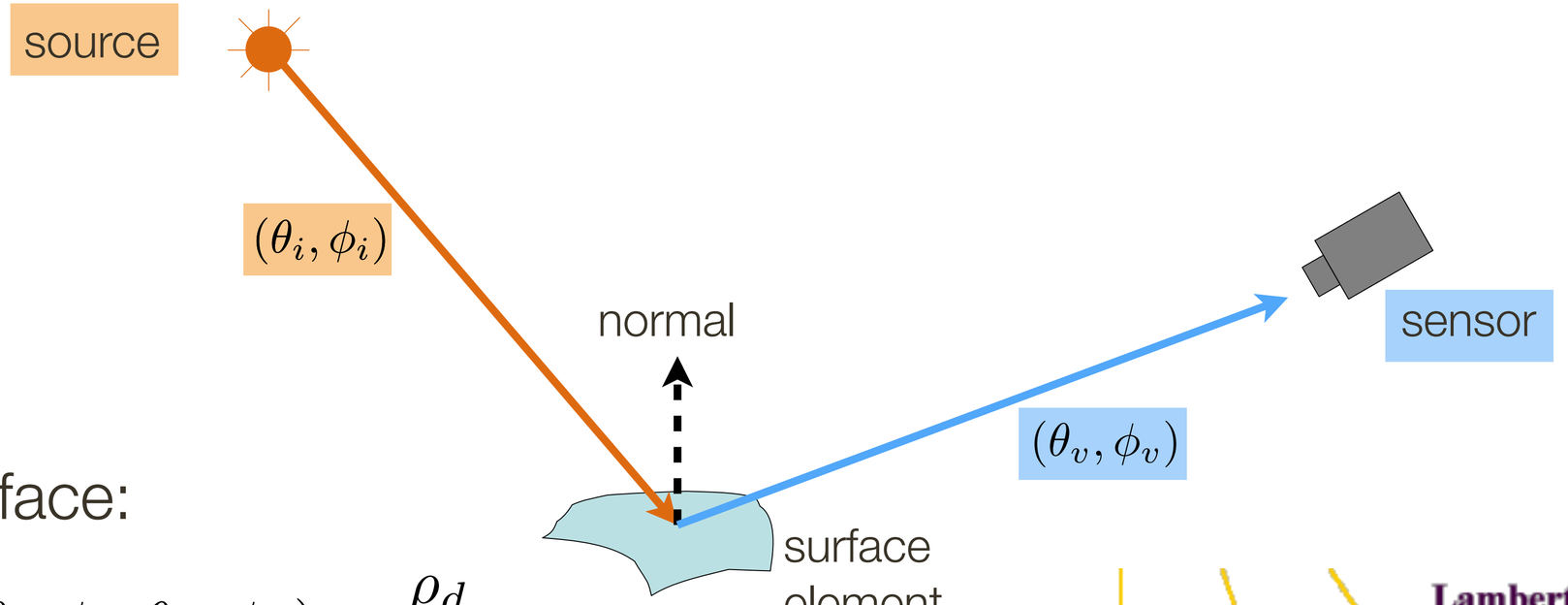
Diffuse/Lambertian vs **Specular/Mirror** Surfaces

Diffuse/Lambertian vs **Specular/Mirror** Surfaces

Diffuse/Lambertian vs **Specular/Mirror** Surfaces

(small) **Graphics** Review

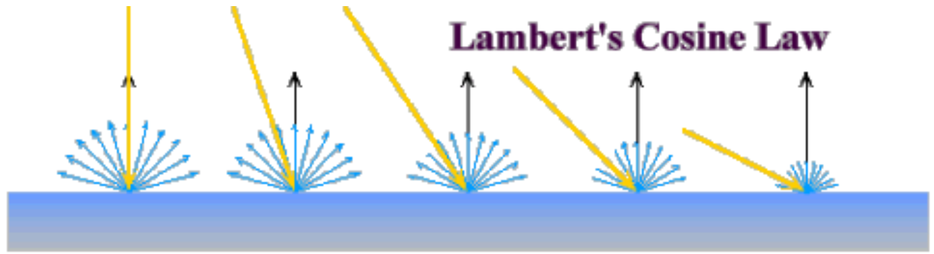
Surface reflection depends on both the **viewing** (θ_v, ϕ_v) and **illumination** (θ_i, ϕ_i) direction, with Bidirectional Reflection Distribution Function: $\mathbf{BRDF}(\theta_i, \phi_i, \theta_v, \phi_v)$



Lambertian surface:

$$\mathbf{BRDF}(\theta_i, \phi_i, \theta_v, \phi_v) = \frac{\rho_d}{\pi}$$

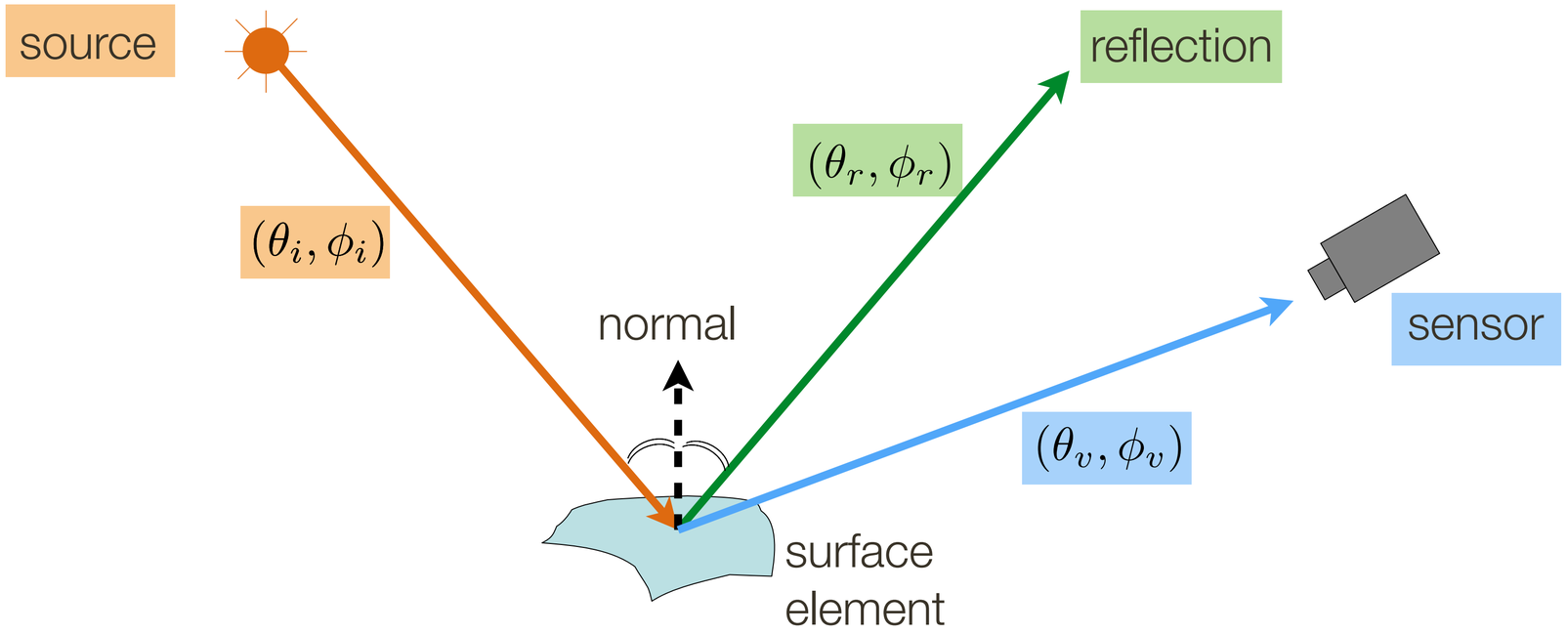
$$L = \frac{\rho_d}{\pi} I(\vec{i} \cdot \vec{n})$$



Slide adopted from: Ioannis (Yannis) Gkioulekas (CMU)

(small) **Graphics** Review

Surface reflection depends on both the **viewing** (θ_v, ϕ_v) and **illumination** (θ_i, ϕ_i) direction, with Bidirectional Reflection Distribution Function: $\mathbf{BRDF}(\theta_i, \phi_i, \theta_v, \phi_v)$

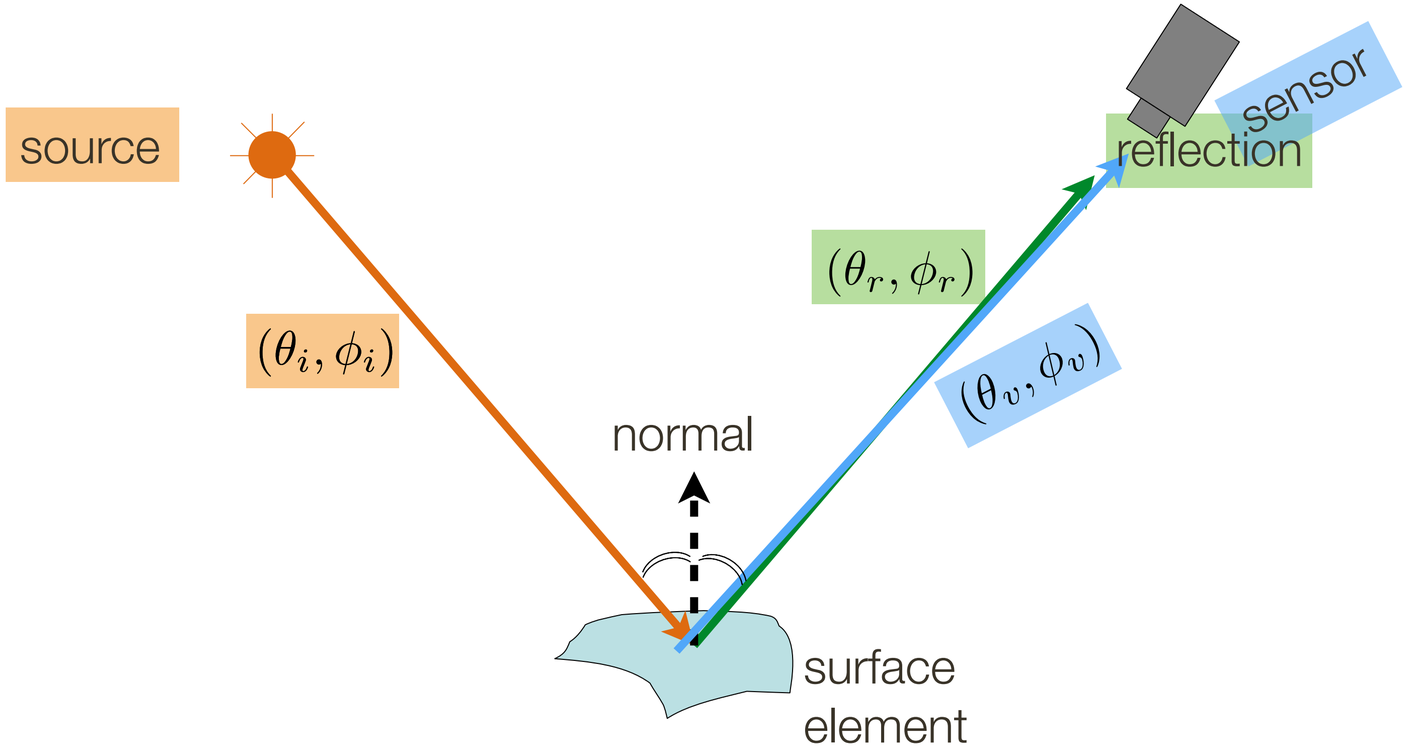


Mirror surface: all incident light reflected in one directions $(\theta_v, \phi_v) = (\theta_r, \phi_r)$

Slide adopted from: Ioannis (Yannis) Gkioulekas (CMU)

(small) **Graphics** Review

Surface reflection depends on both the **viewing** (θ_v, ϕ_v) and **illumination** (θ_i, ϕ_i) direction, with Bidirectional Reflection Distribution Function: $\mathbf{BRDF}(\theta_i, \phi_i, \theta_v, \phi_v)$

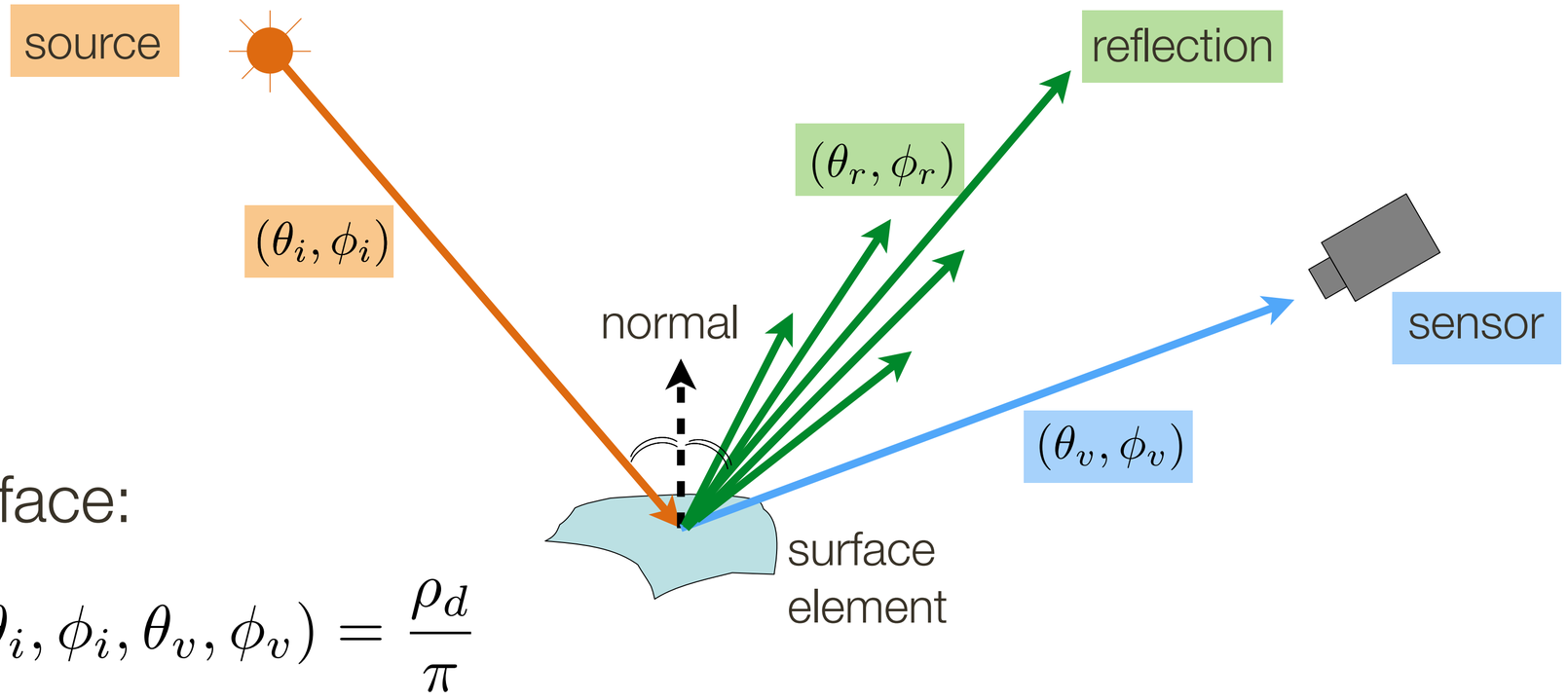


Mirror surface: all incident light reflected in one directions $(\theta_v, \phi_v) = (\theta_r, \phi_r)$

Slide adopted from: Ioannis (Yannis) Gkioulekas (CMU)

(small) **Graphics** Review

Surface reflection depends on both the **viewing** (θ_v, ϕ_v) and **illumination** (θ_i, ϕ_i) direction, with Bidirectional Reflection Distribution Function: $\mathbf{BRDF}(\theta_i, \phi_i, \theta_v, \phi_v)$



Lambertian surface:

$$\mathbf{BRDF}(\theta_i, \phi_i, \theta_v, \phi_v) = \frac{\rho_d}{\pi}$$

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

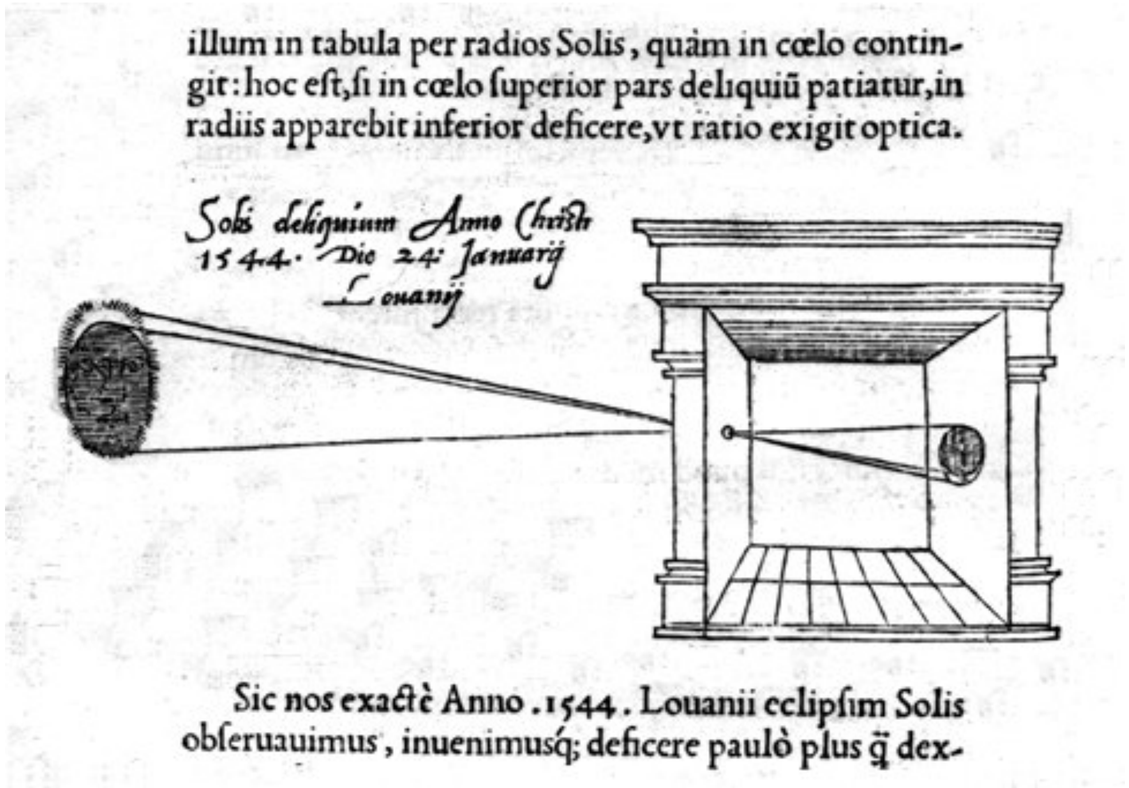
Slide adopted from: Ioannis (Yannis) Gkioulekas (CMU)

Diffuse vs **Specular** Surfaces

Diffuse vs **Specular** Surfaces

Diffuse vs **Specular** Surfaces

Camera Obscura (latin for “dark chamber”)



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”

Camera Obscura (latin for “dark chamber”)

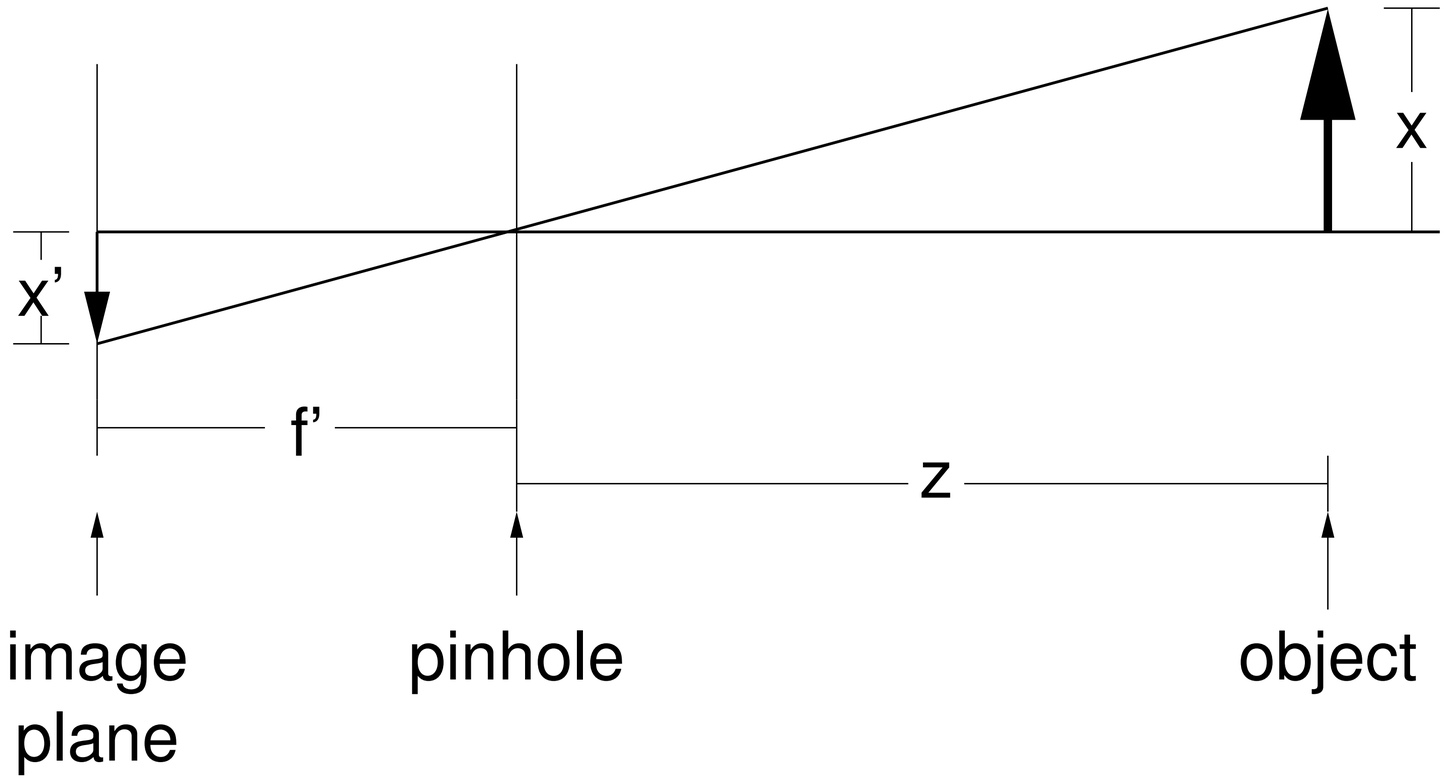
principles behind the pinhole camera or camera obscura were first mentioned by Chinese philosopher Mozi (Mo-Ti) (470 to 390 BCE)

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)

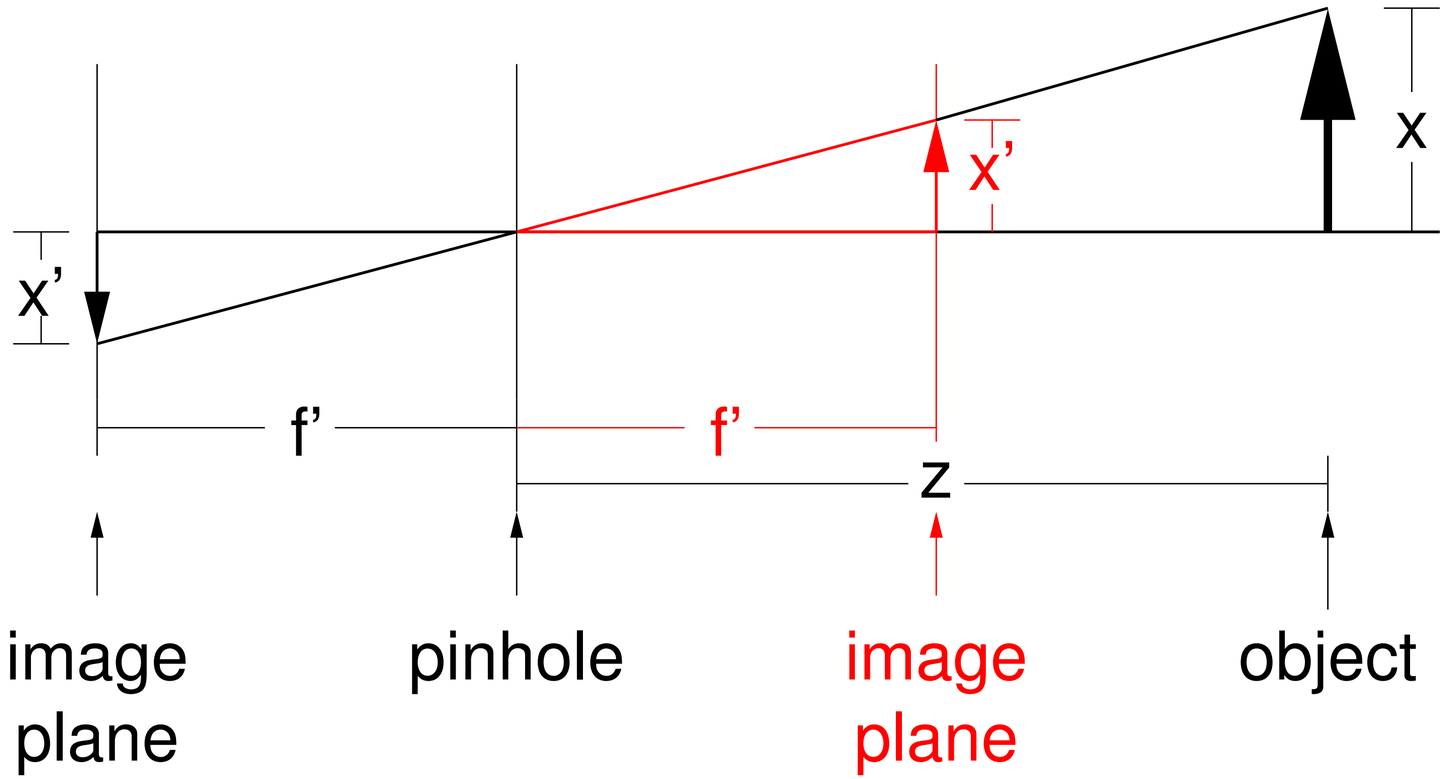
f’ is the **focal length** of the camera



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)

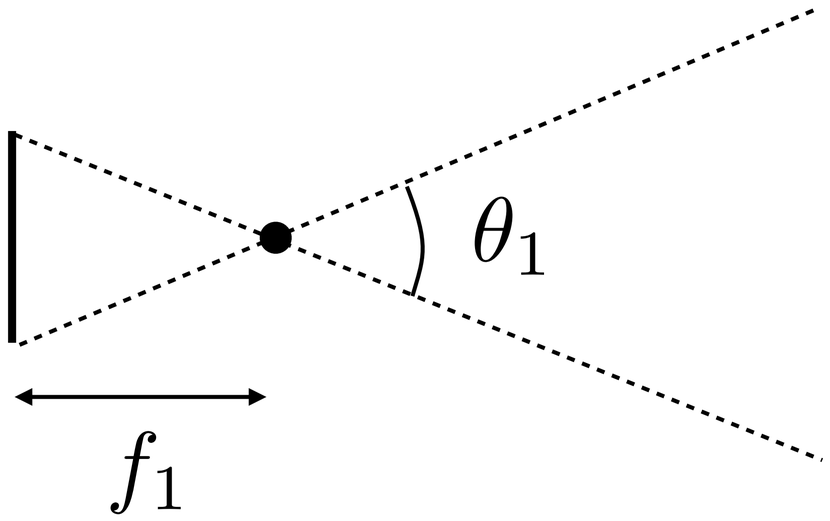
It is convenient to think of the **image plane** which is in from of the pinhole

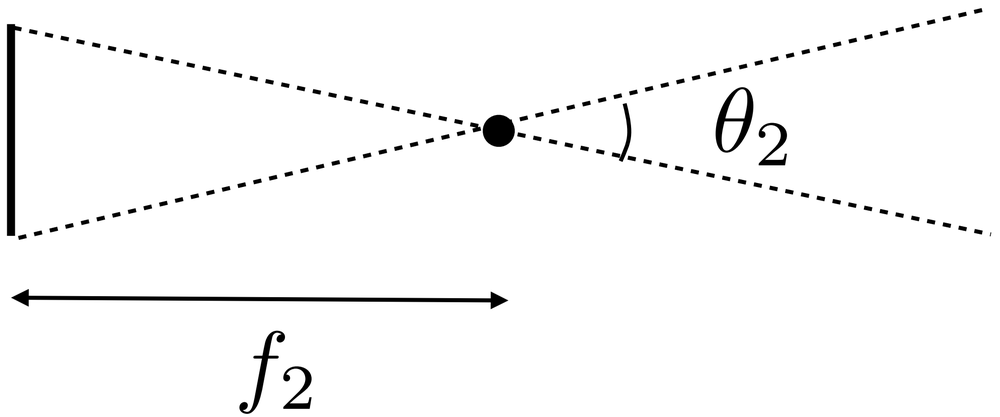


What happens if object moves towards the camera? Away from the camera?

Focal Length

For a fixed sensor size, focal length determines the **field of view** (FoV)





Properties of Projection

- **Points** project to **points**
- **Lines** project to **lines**
- **Planes** project to the **whole** or **half** image
- Angles are **not** preserved

Degenerate cases

- Line through focal point projects to a point
- Plane through focal point projects to a line

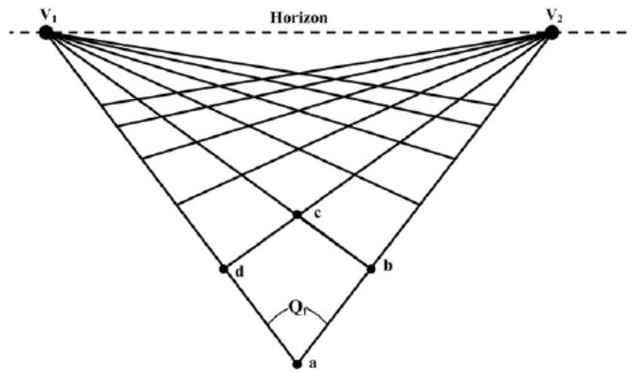
Vanishing Points

Each set of parallel lines meet at a different 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** for that plane

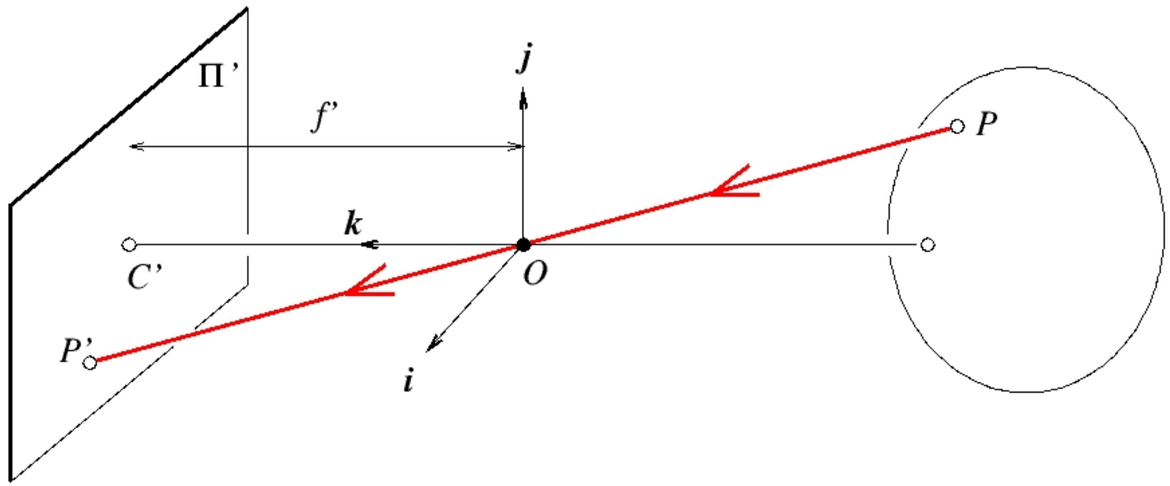


Good way to **spot fake images**

— scale and perspective do not work

— vanishing points behave badly

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

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

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

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.

Sample Question: Image Formation

True of **false**: For what imaging scenario would a weak perspective projection be an accurate approximation of the imaging process?

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

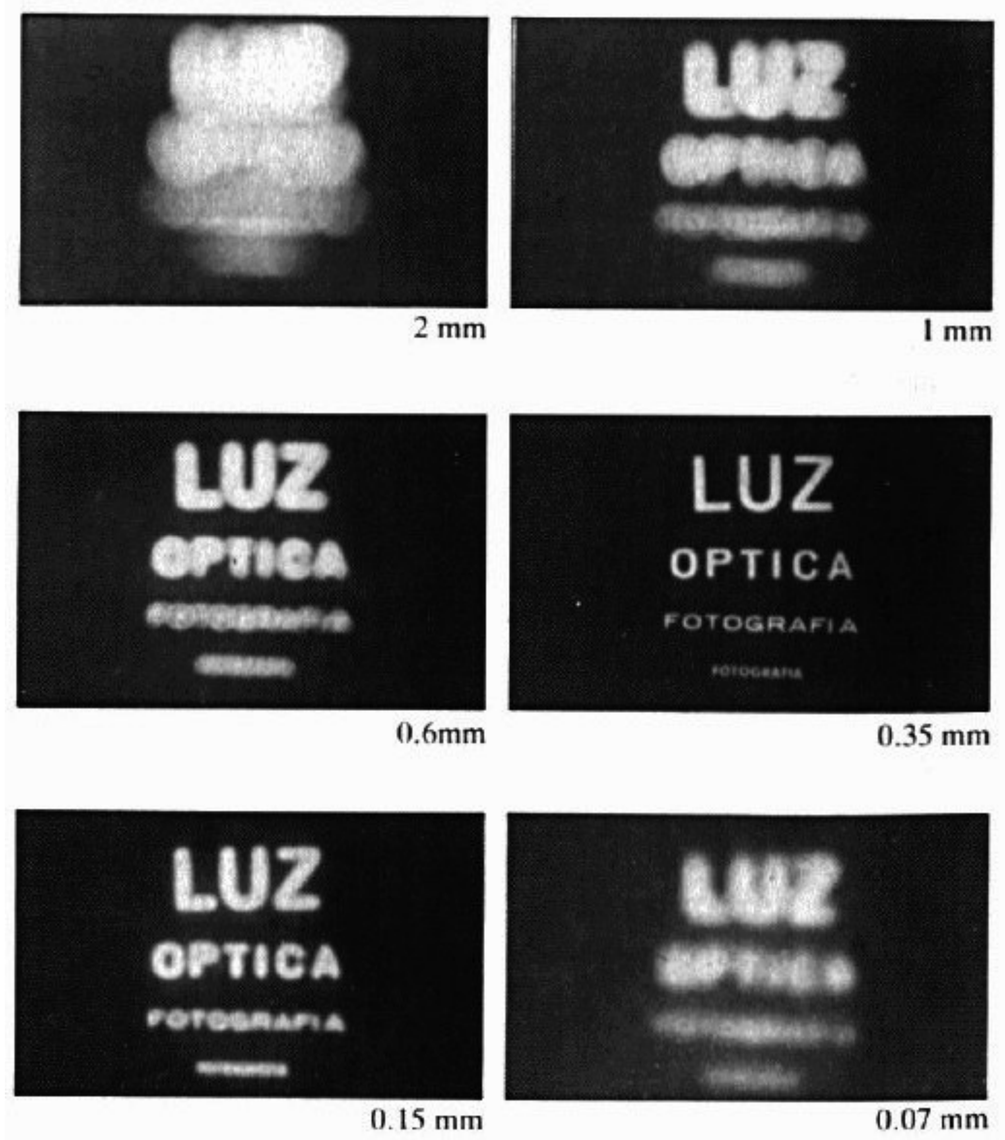
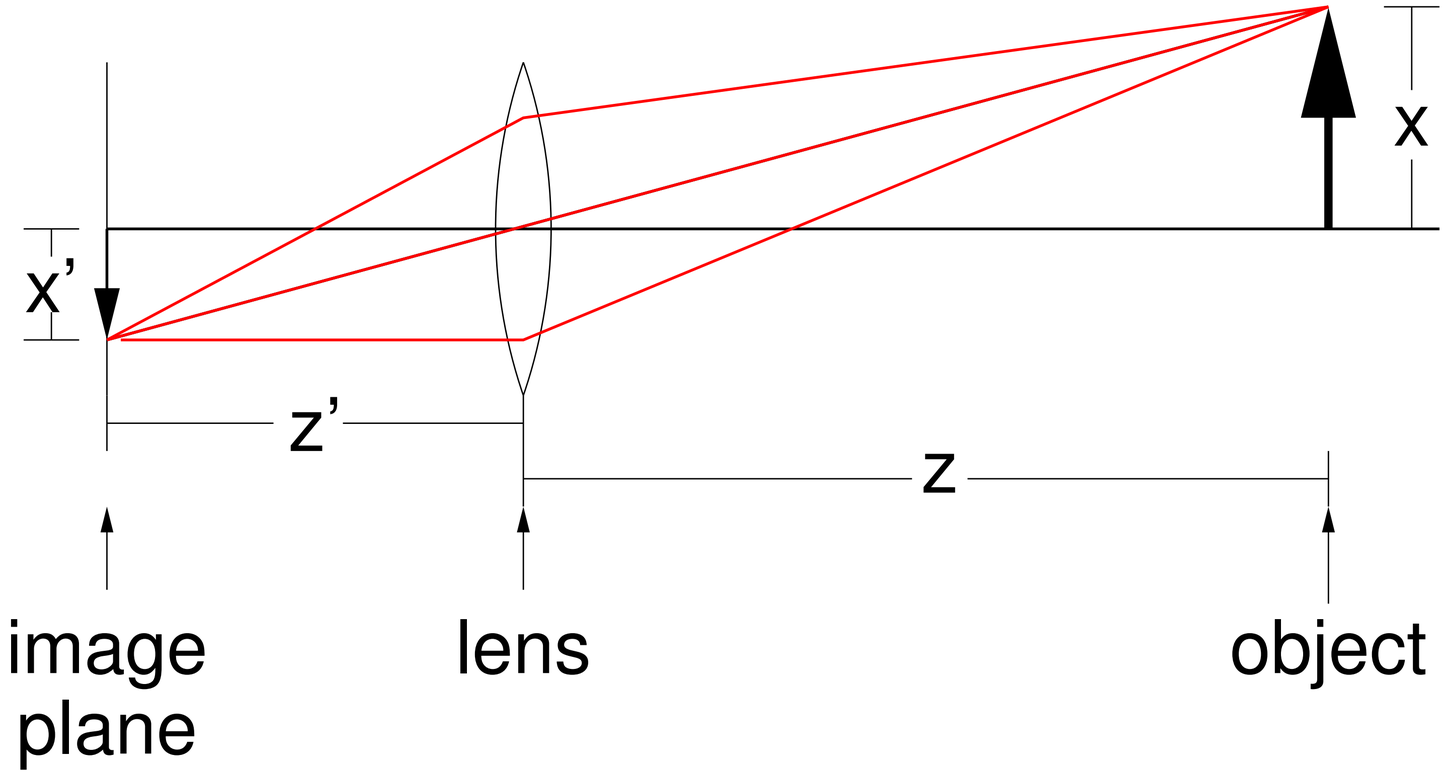


Image Credit: Credit: E. Hecht. “Optics,” Addison-Wesley, 1987

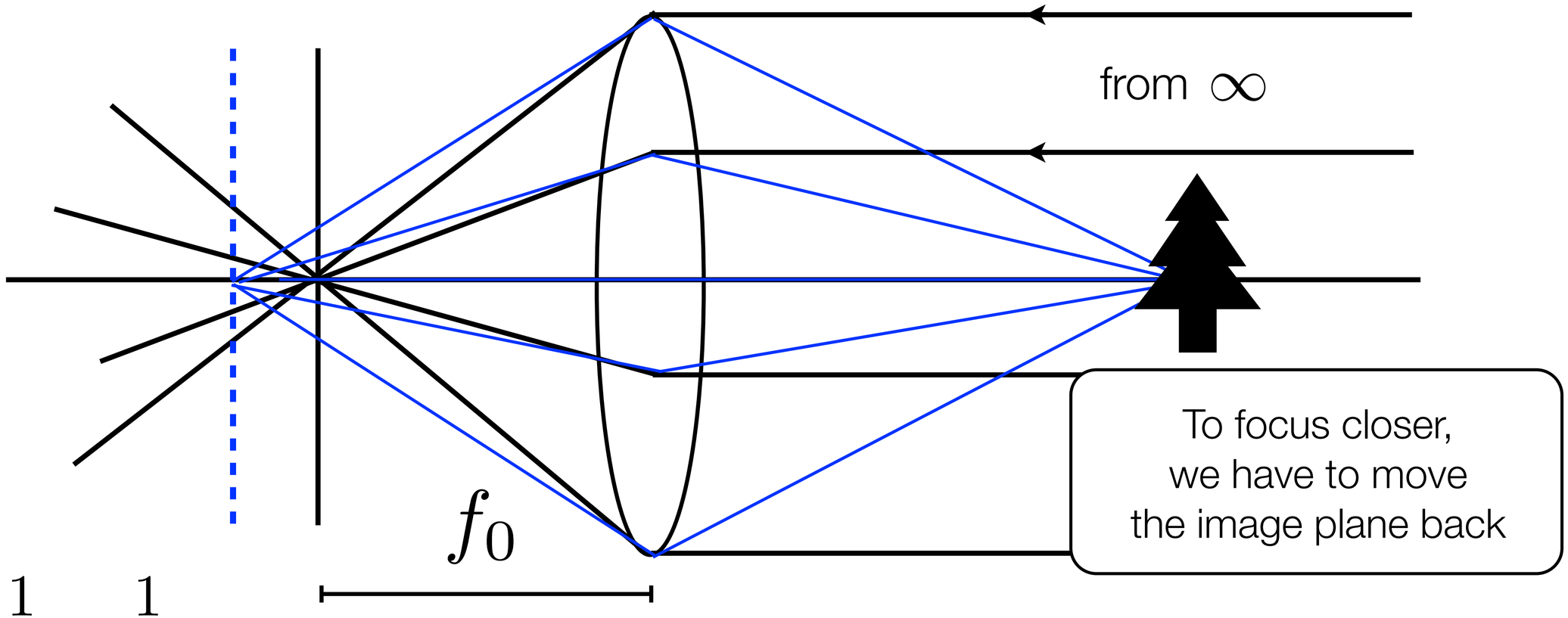
Pinhole Model (Simplified) **with Lens**



Lens Basics

A lens focuses parallel rays (from points at infinity) at focal length of the lens

Rays passing through the center of the lens are not bent



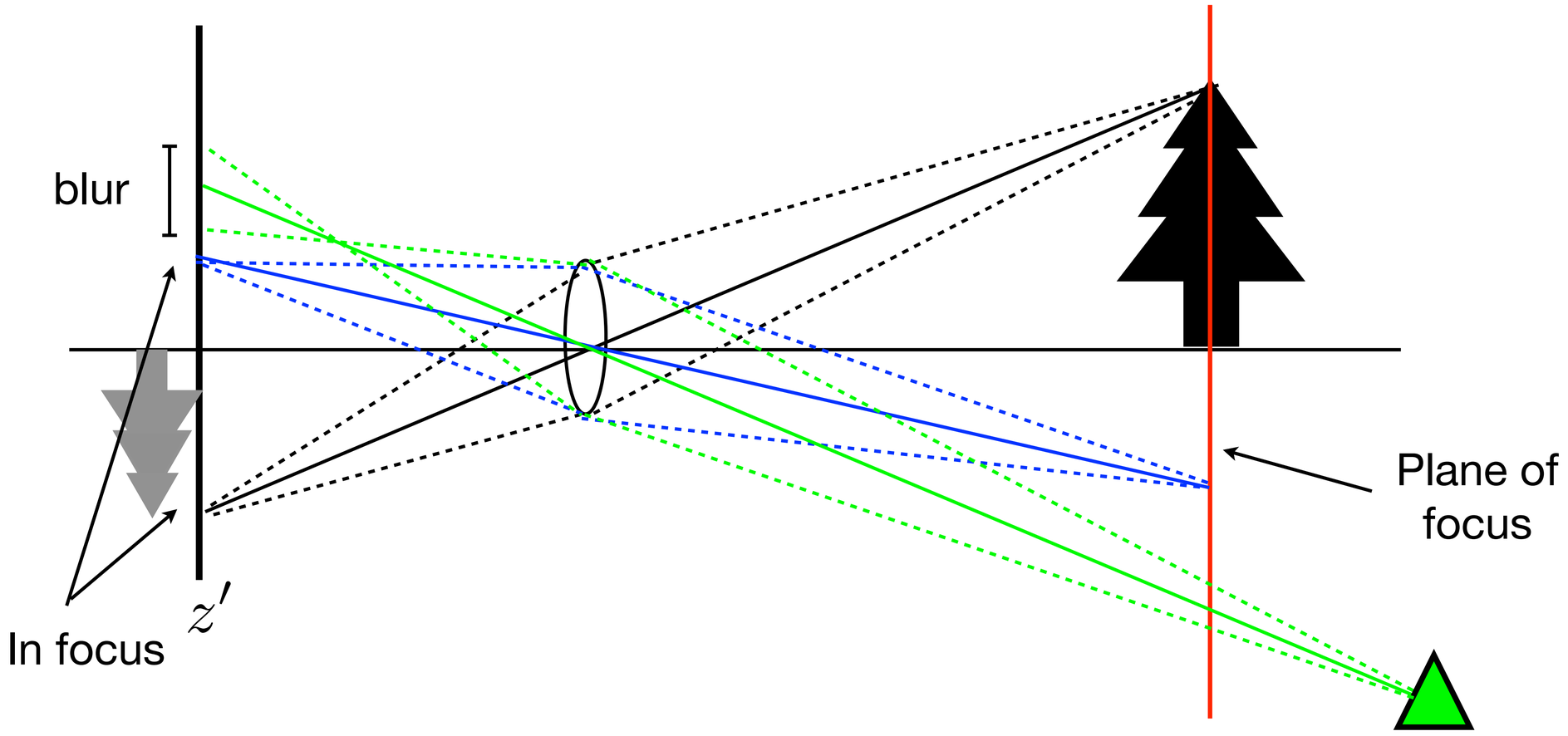
$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

Lens Basics

Lenses focus all rays from a (parallel to lens) plane in the world

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

focal length



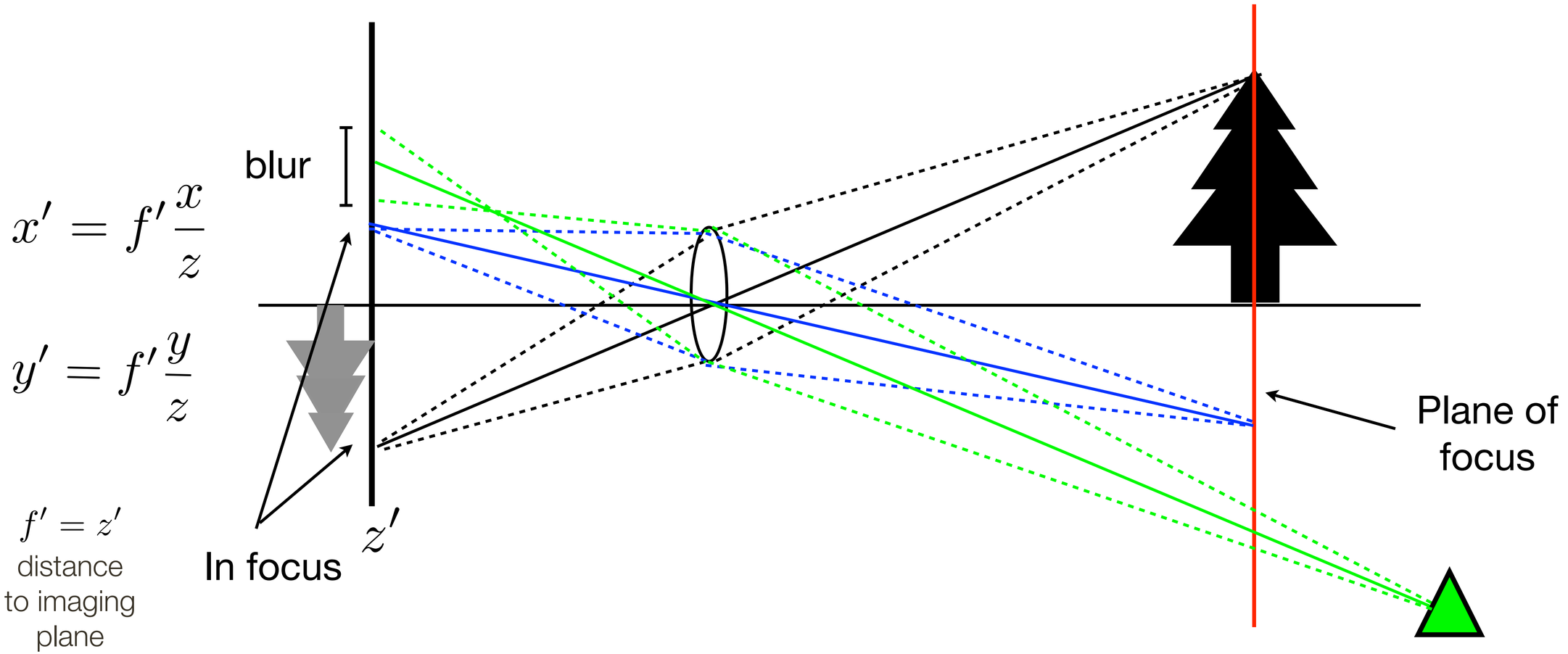
Objects off the plane are blurred depending on the distance

Lens Basics

Lenses focus all rays from a (parallel to lens) plane in the world

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

focal length



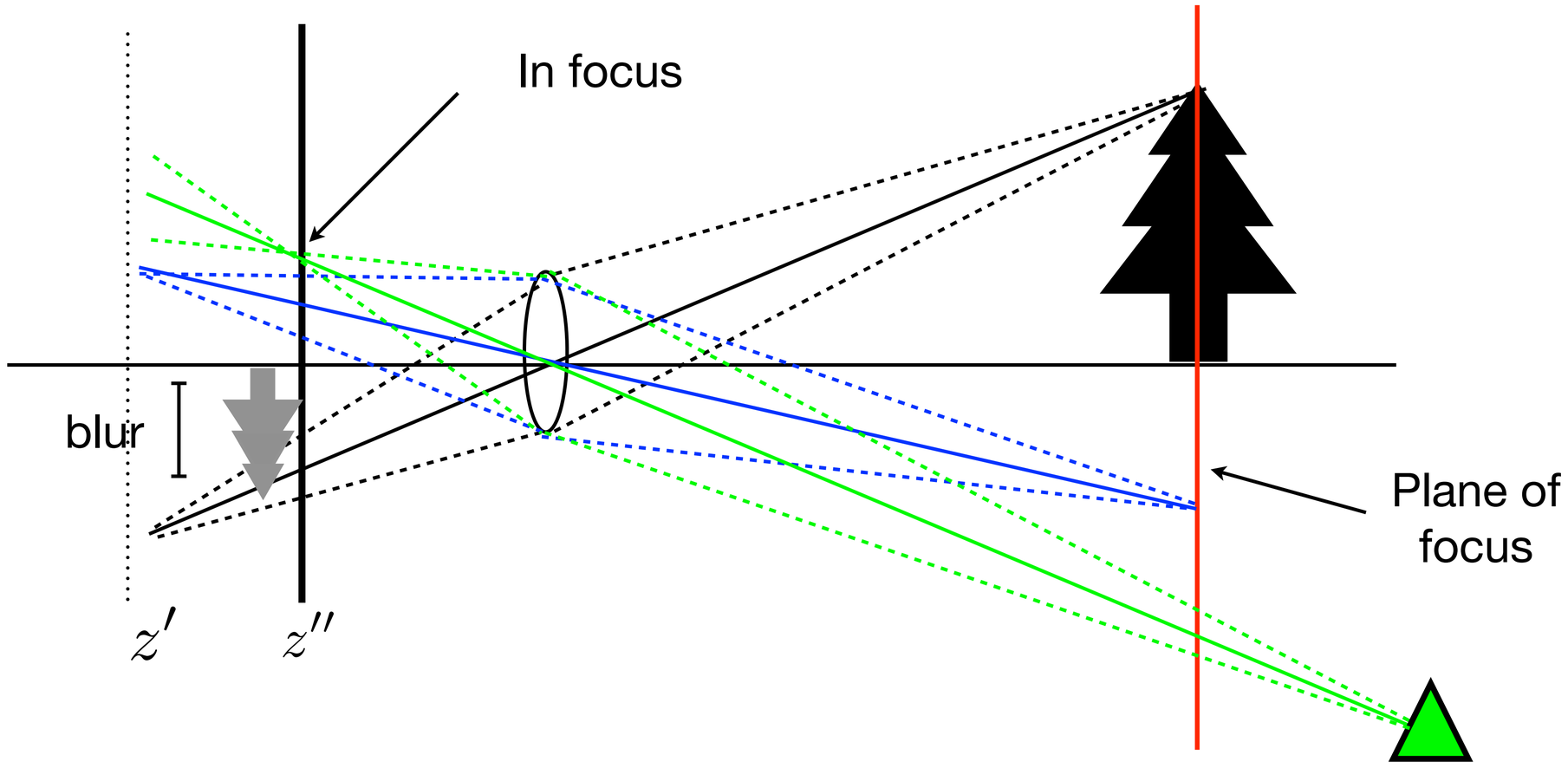
Objects off the plane are blurred depending on the distance

Lens Basics

Lenses focus all rays from a (parallel to lens) plane in the world

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

focal length



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

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

$f' = z''$
distance to imaging plane

Objects off the plane are blurred depending on the distance

Perspective Projection + **Thin Lens** Examples

Where would the focusing plane be for various positions of the object?

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

https://www.physicsclassroom.com/class/refrn/Lesson-5/Converging-Lenses-Object-Image-Relations

Perspective Projection + **Thin Lens** Examples

Where would the focusing plane be for various positions of the object?

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

$$z' = \frac{zf}{z+f}$$

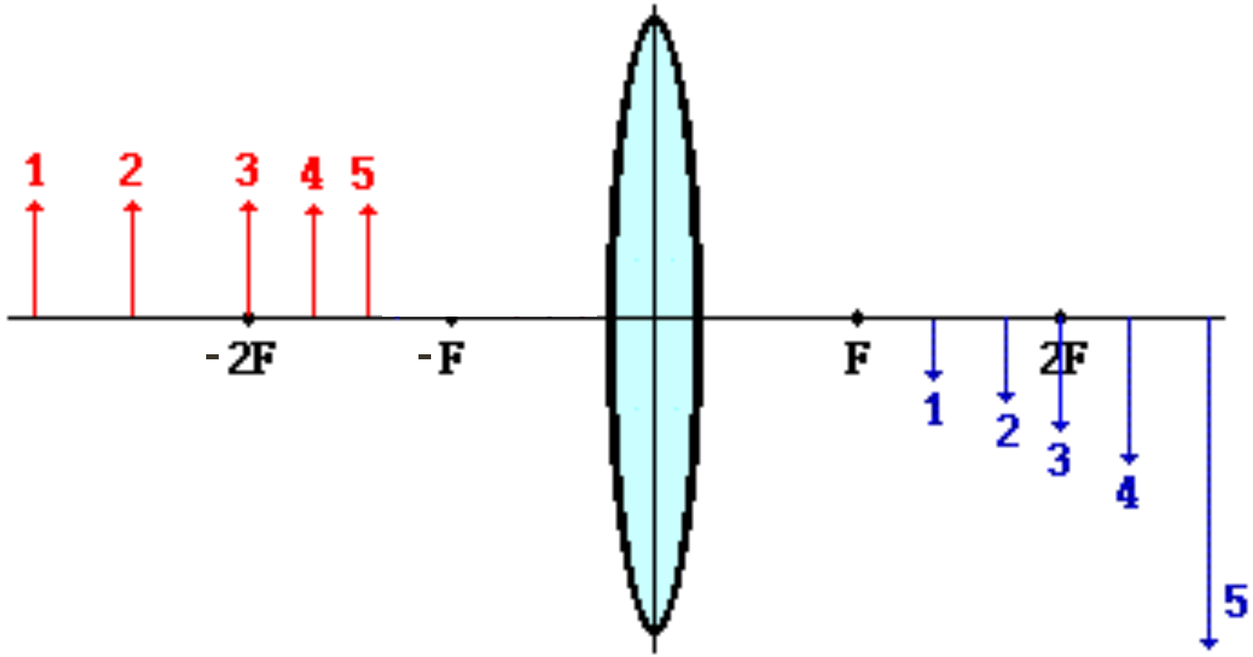
https://www.physicsclassroom.com/class/refrn/Lesson-5/Converging-Lenses-Object-Image-Relations

Perspective Projection + **Thin Lens** Examples

Where would the focusing plane be for various positions of the object?

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

$$z' = \frac{zf}{z+f}$$



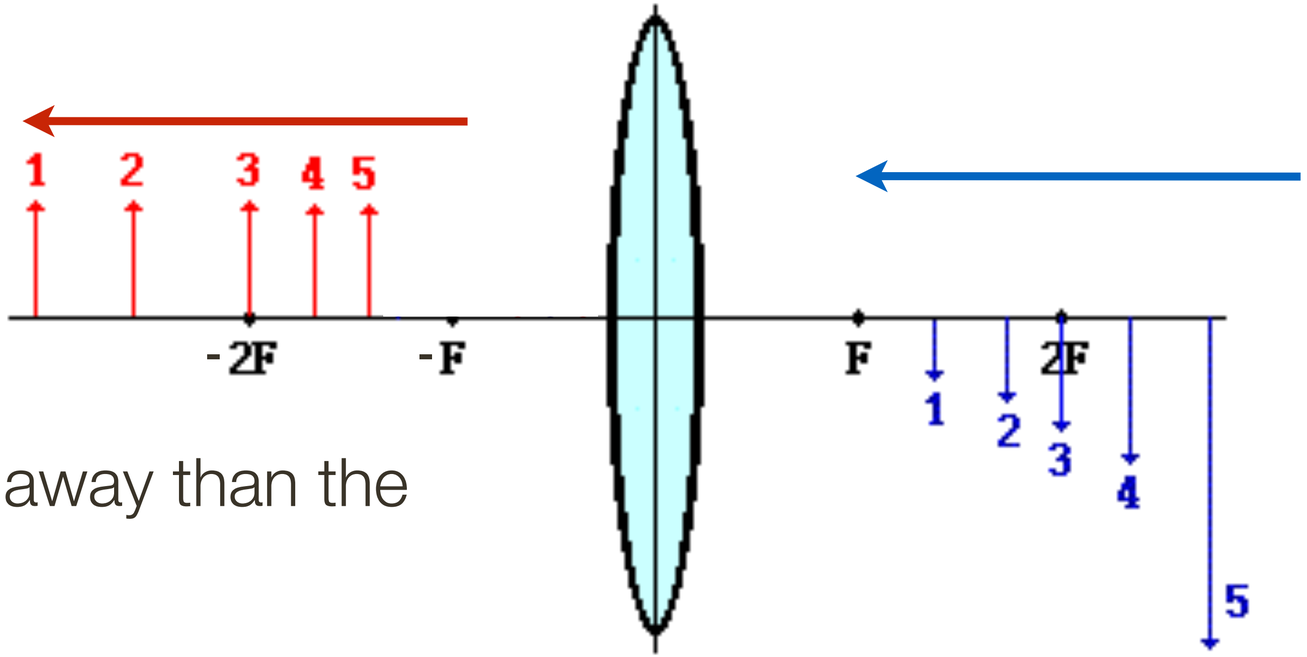
https://www.physicsclassroom.com/class/refrn/Lesson-5/Converging-Lenses-Object-Image-Relations

Perspective Projection + **Thin Lens** Examples

Where would the focusing plane be for various positions of the object?

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

$$z' = \frac{zf}{z+f}$$



Objects **further** away than the **focal length**

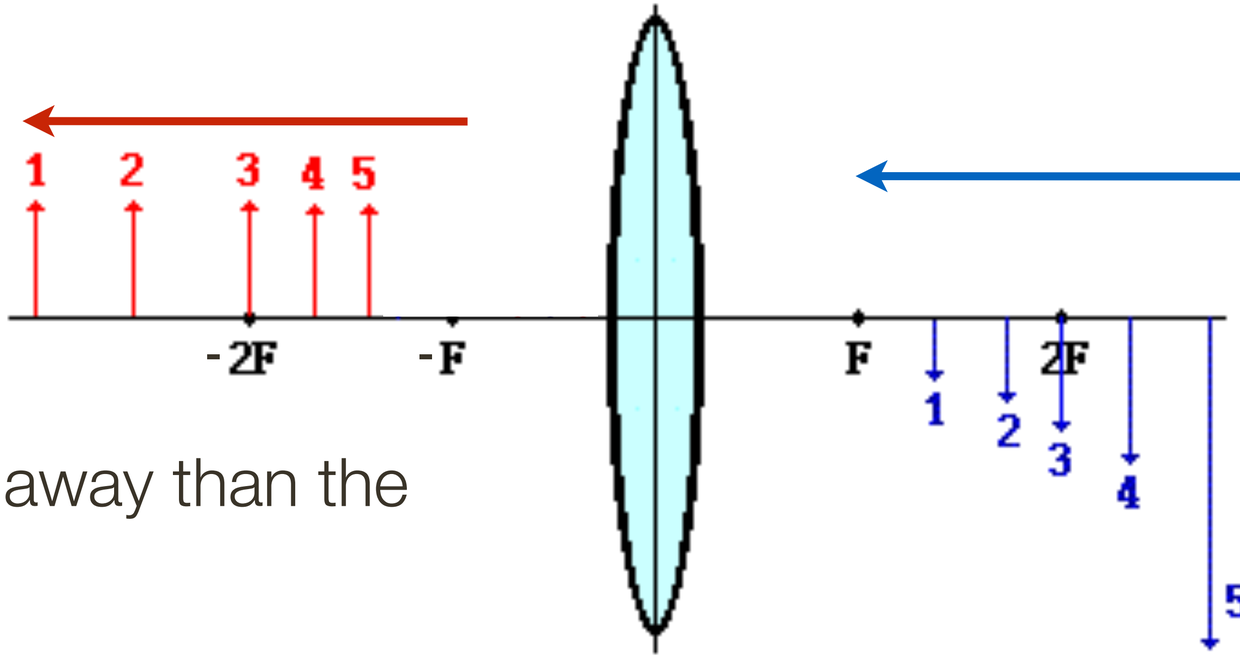
https://www.physicsclassroom.com/class/refrn/Lesson-5/Converging-Lenses-Object-Image-Relations

Perspective Projection + **Thin Lens** Examples

Where would the focusing plane be for various positions of the object?

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$$

$$z' = \frac{zf}{z+f}$$



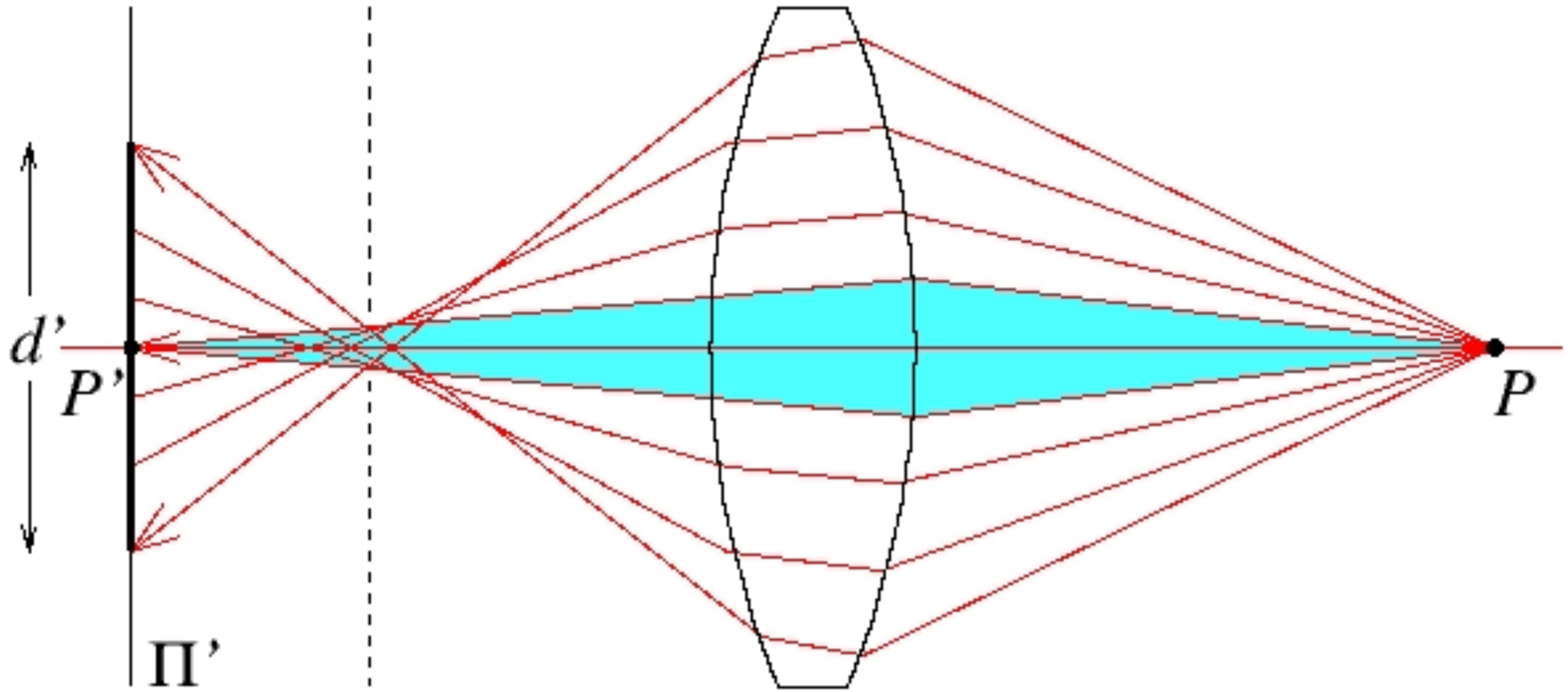
$$\lim_{z \to -\infty} \frac{zf}{z+f} = f$$

L'Hopital's Rule

Objects **further** away than the **focal length**

https://www.physicsclassroom.com/class/refrn/Lesson-5/Converging-Lenses-Object-Image-Relations

Spherical **Aberration**



Forsyth & Ponce (1st ed.) Figure 1.12a

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

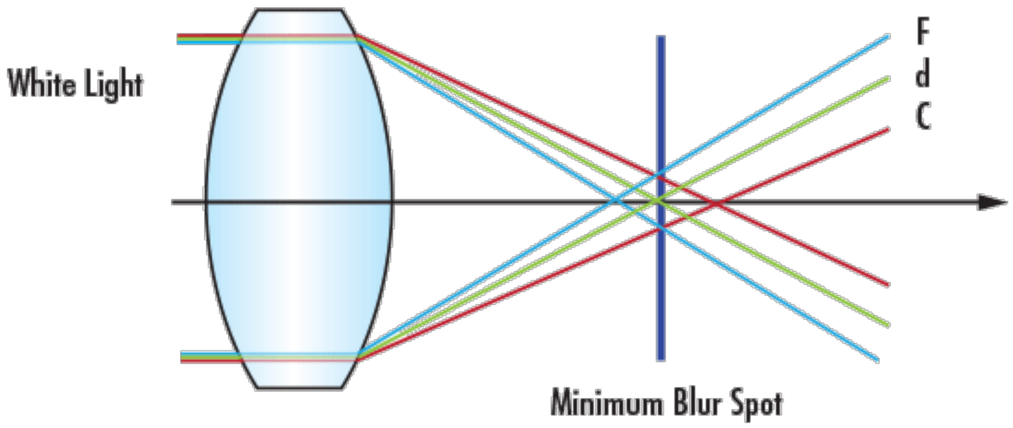
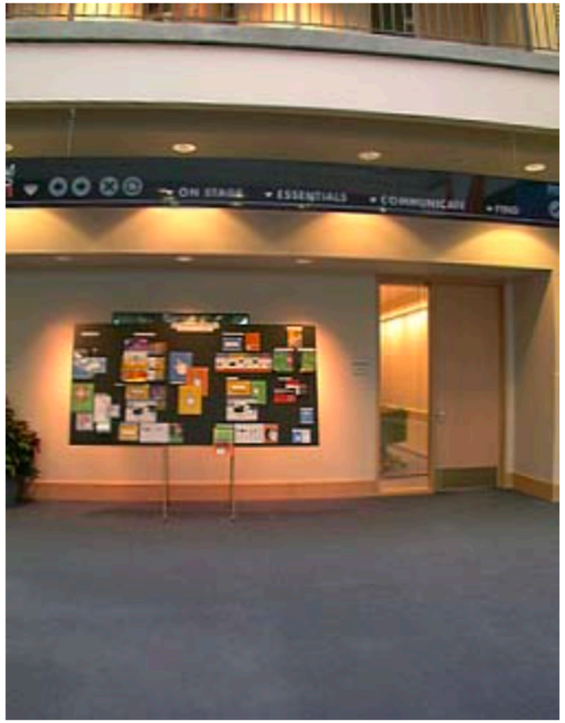
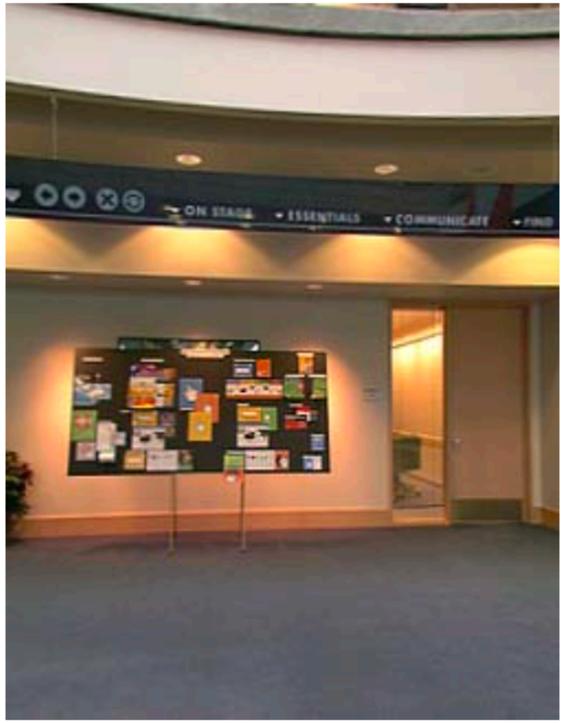


Image Credit: Trevor Darrell

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**

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

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 Filter **Example**

$F(X,Y)$

filter

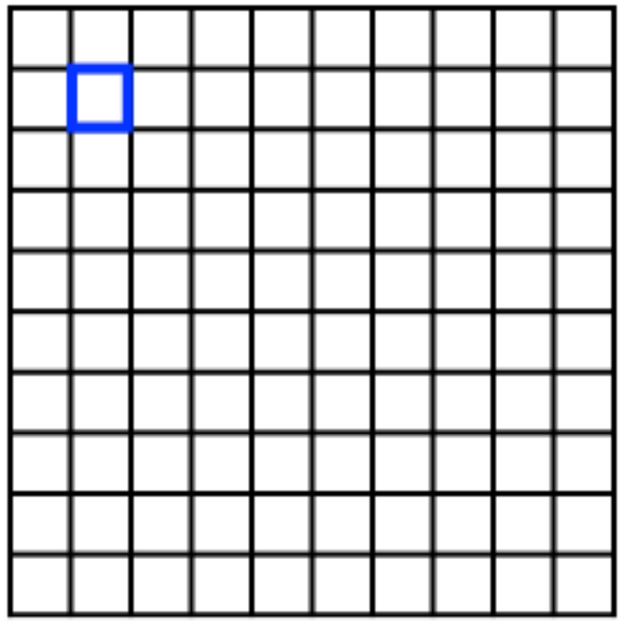
$\frac{1}{9}$

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

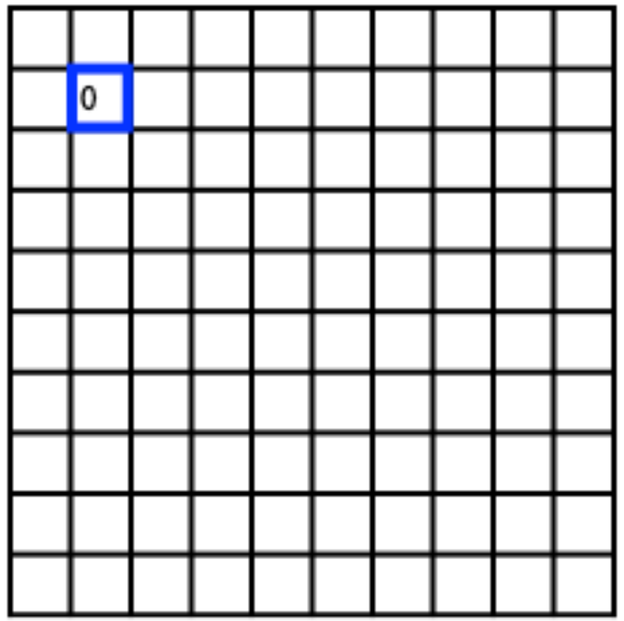
$\frac{1}{9}$

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 $I'(X,Y)$



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

output | filter | image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

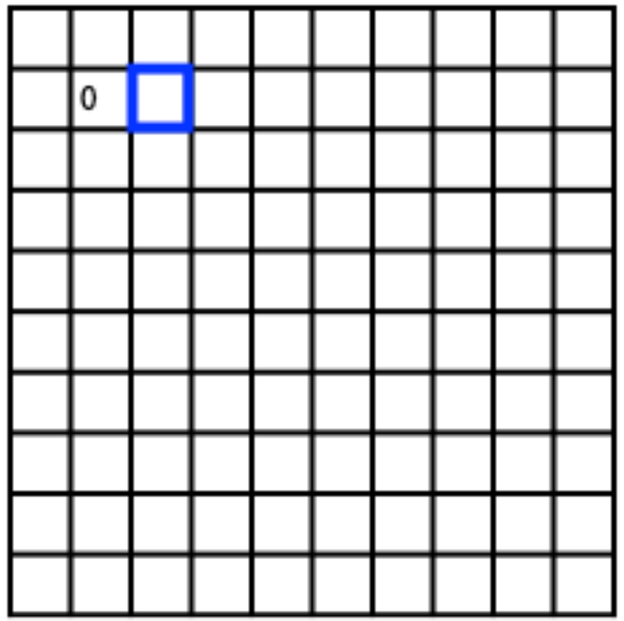
$\frac{1}{9}$

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 $I'(X,Y)$



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

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							

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

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							

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

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						

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

$I'(X,Y)$: 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				

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

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			

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

$I'(X,Y)$: 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		

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

$I'(X,Y)$: 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							

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

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	10	
	0	20	40						

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

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	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	

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

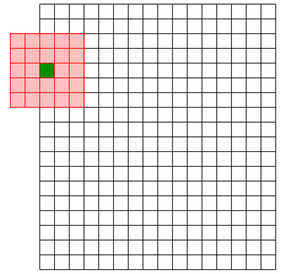
$I'(X,Y)$: output

$F(I,J)$: filter

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filters: **Boundary** Effects



Four 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

4. **Reflect boarder**: Copy rows/columns locally by reflecting over the edge

Sample Question: Filtering and Padding

When, specifically, would padding be **suboptimal** (i.e., you would opt for ignoring locations near boarders)? Why?

Linear **Filters**

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

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

output | filter | 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 Filters: **Properties**

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

Superposition: Let F_1 and F_2 be digital filters

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

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

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

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

An operation is **linear** if it satisfies both **superposition** and **scaling**

Linear 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 $G \otimes F = F \otimes G$

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)

Low-pass Filtering = “Smoothing”

Gaussian Filter

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

All of these filters are **Low-pass Filters**

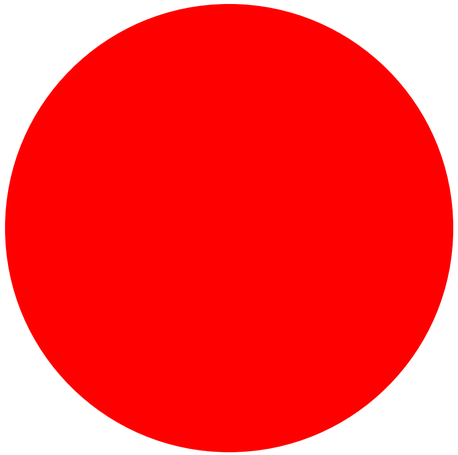
Low-pass filter: Low pass filter filters out all of the high frequency content of the image, only low frequencies remain

Other smoothing filters

Box Filter

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

Pillbox Filter



Smoothing

Smoothing with a box **doesn’t model lens defocus** well

— Smoothing with a box filter depends on direction

— Image in which the center point is 1 and every other point is 0

Smoothing with a (circular) **pillbox** is a better model for defocus (in geometric optics)

The **Gaussian** is a good general smoothing model

— for phenomena (that are the sum of other small effects)

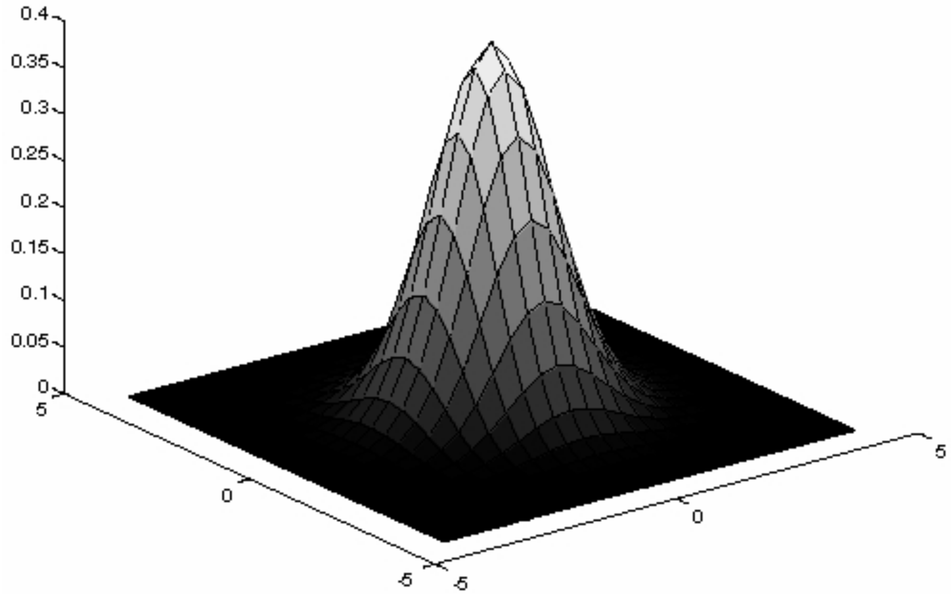
— whenever the Central Limit Theorem applies

Smoothing with a Gaussian

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

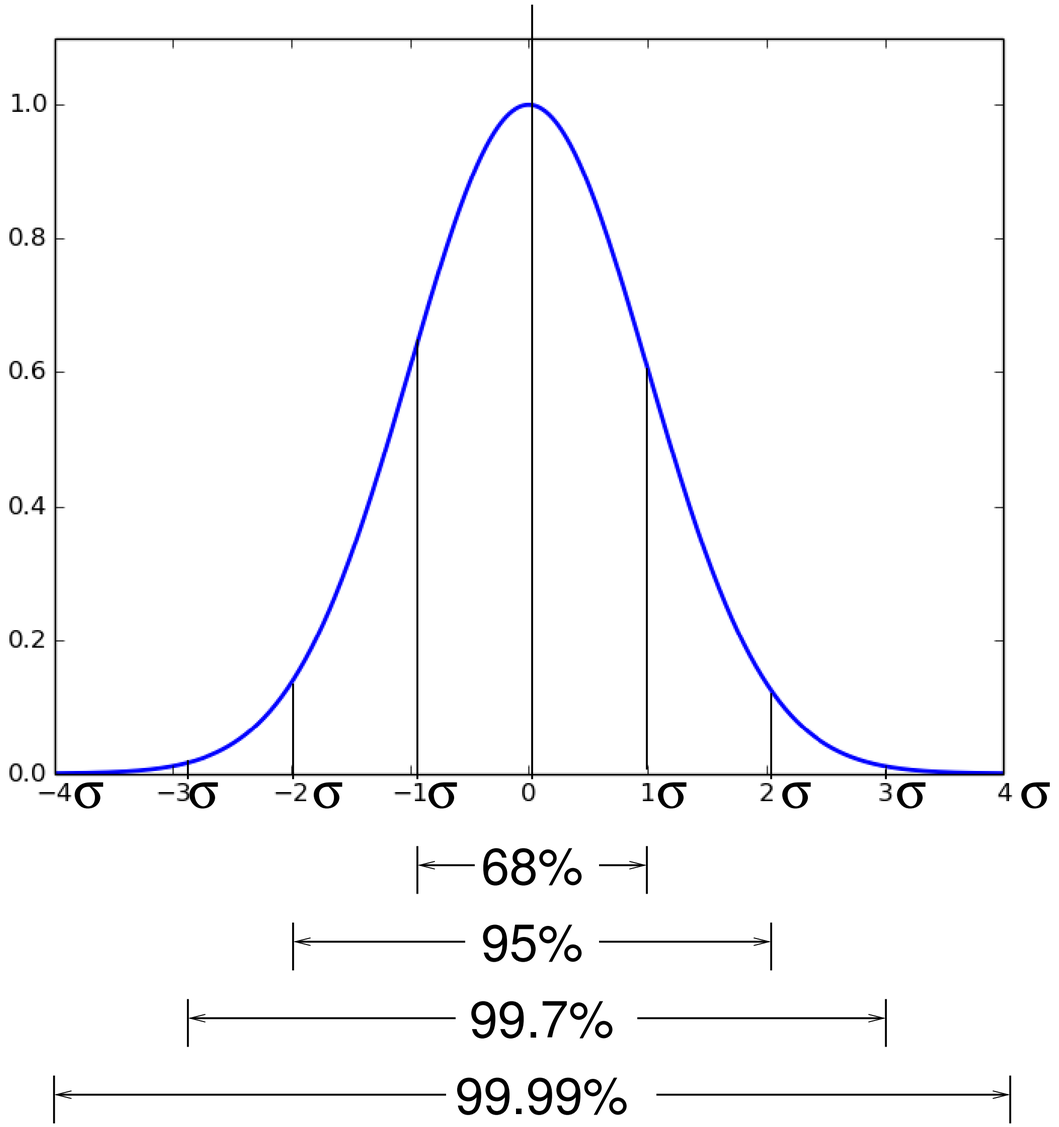
2D **Gaussian** (continuous case):

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



Forsyth & Ponce (2nd ed.)
Figure 4.2

Gaussian: Area Under the Curve



Efficient Implementation: **Separability**

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

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

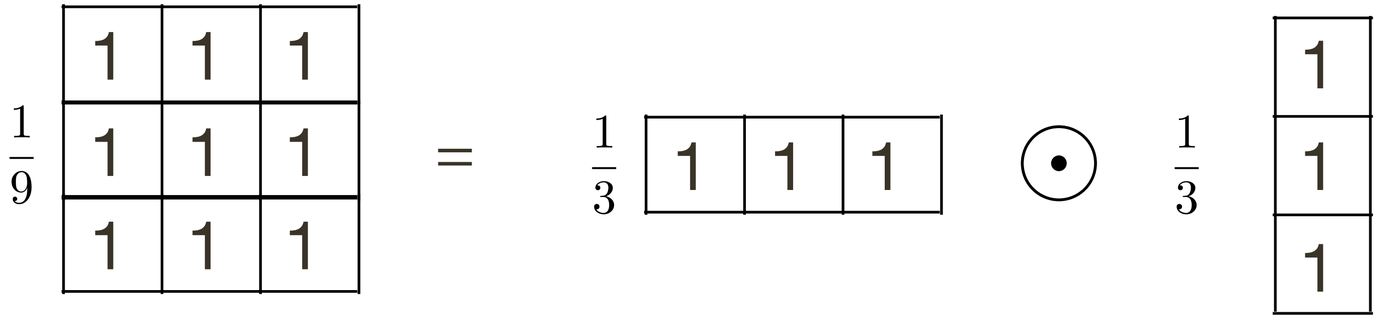
— Then, convolve each column with a 1D filter

— Aside: or vice versa

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

Separability: How do you know if filter is separable?

Mathematically: Rank of filter matrix is 1 (recall rank is number of linearly independent row vectors)



Efficient Implementation: **Separability**

Naive implementation of 2D **Gaussian**:

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

Separable 2D **Gaussian**:

At each pixel, (X, Y), there are	$2m$ multiplications
There are	$n \times n$ pixels in (X, Y)
Total:	$2m \times n^2$ multiplications

Speeding Up **Convolution** (The Convolution Theorem)

Convolution **Theorem**:

Let $$i'(x,y) = f(x,y) \otimes i(x,y)$$

then $$\mathcal{I}'(w_x, w_y) = \mathcal{F}(w_x, w_y)\ \mathcal{I}(w_x, w_y)$$

where $\mathcal{I}'(w_x, w_y)$, $\mathcal{F}(w_x, w_y)$, and $\mathcal{I}(w_x, w_y)$ are Fourier transforms of $i'(x,y)$, $f(x,y)$ and $i(x,y)$

At the expense of two **Fourier** transforms and one inverse Fourier transform, convolution can be reduced to (complex) multiplication

Speeding Up **Convolution** (The Convolution Theorem)

General implementation of **convolution**:

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

Convolution if FFT space:

Cost of FFT/IFFT for image: $\mathcal{O}(n^2 \log n)$

Cost of FFT/IFFT for filter: $\mathcal{O}(m^2 \log m)$

Cost of convolution: $\mathcal{O}(n^2)$

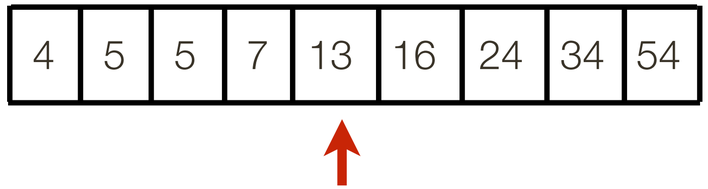
Note: not a function of filter size !!!

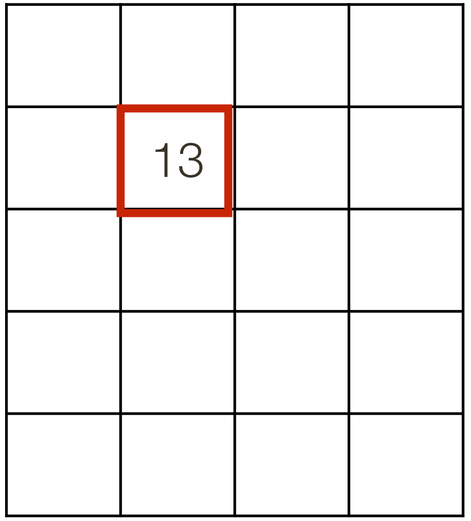
Median Filter

Take the **median value** of the pixels under the filter:

5	13	5	221
4	16	7	34
24	54	34	23
23	75	89	123
54	25	67	12

Image





Output

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: Convolution

Is the following filter applied as correlation shift invariant?

$$\begin{bmatrix} 1 & 32 & 7 \\ 4 & 2 & 7 \\ 1 & 4 & 7 \end{bmatrix}$$

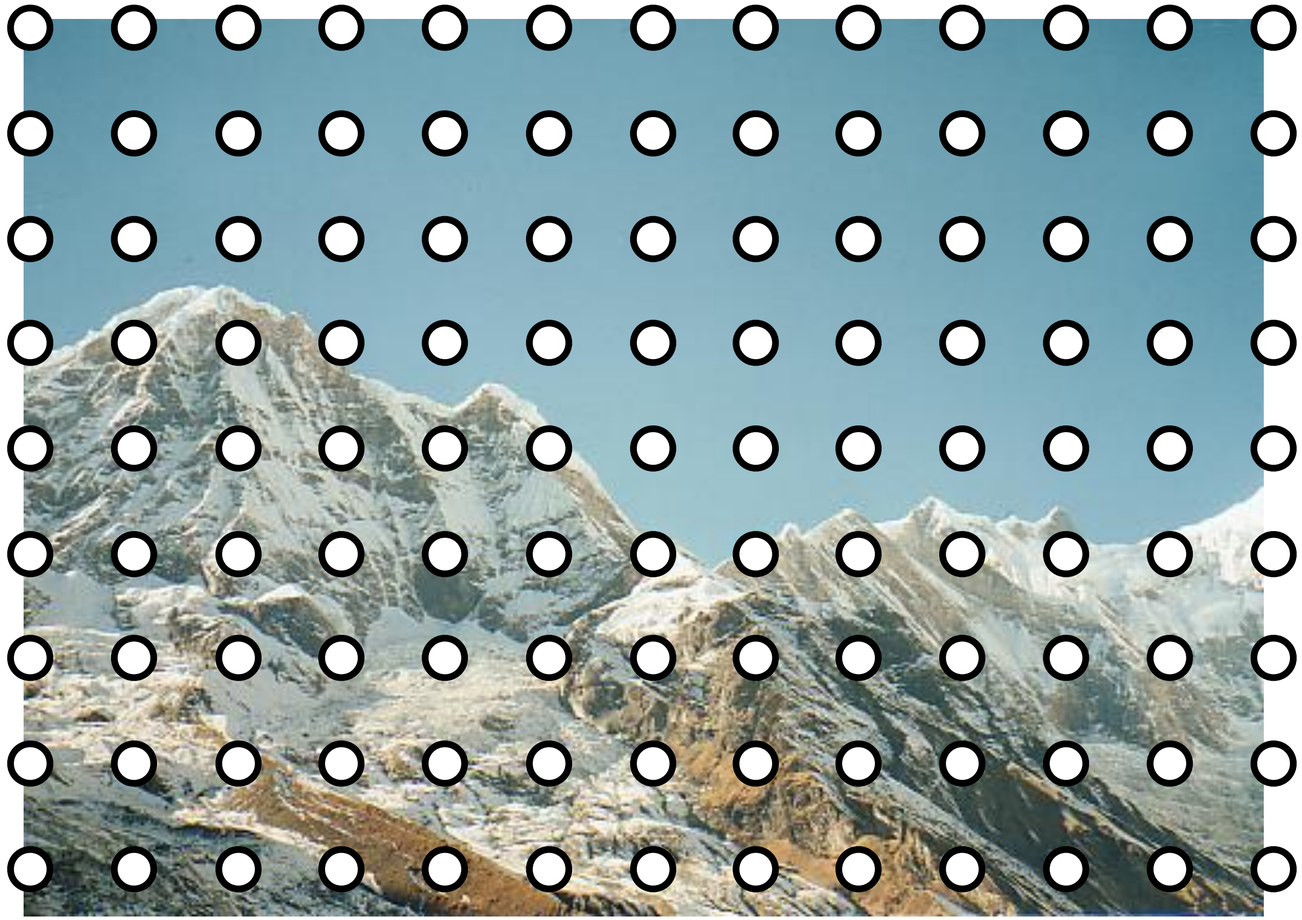
Sample Question: Filters

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

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

Resampling Images

- Naive method: form new image by selecting every *n*th pixel



Aliasing Example

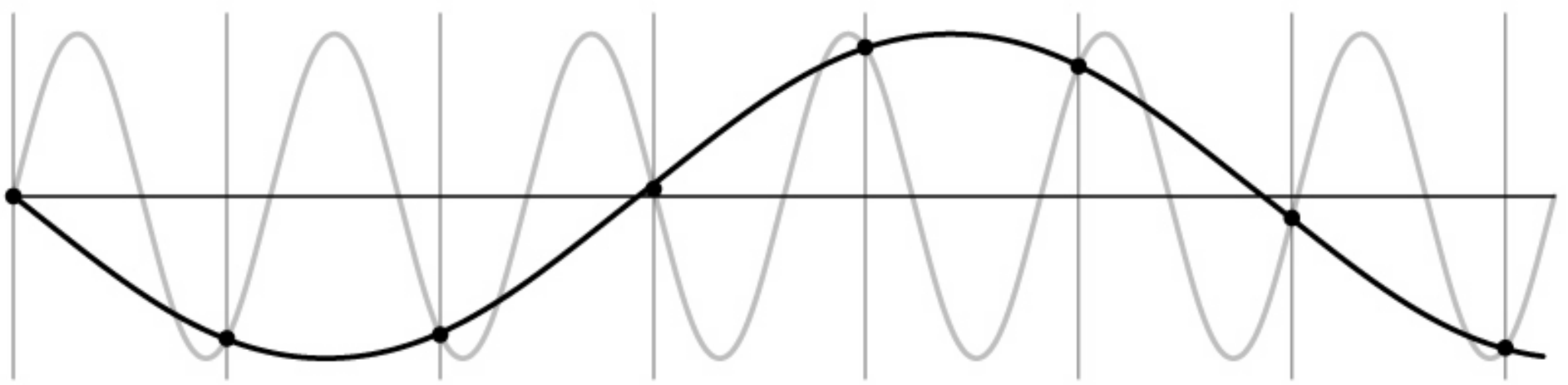
- Sampling every 5th pixel, while shifting rightwards 1 pixel at a time

Aliasing Example

- Sampling every 5th pixel, while shifting rightwards 1 pixel at a time

Example: A Simple Sine Wave

How do we discretize the signal?



Signal can be confused with one at lower frequency
— This is called “Aliasing”

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Nyquist Sampling Theorem

To avoid aliasing a signal must be sampled at twice the maximum frequency:

$$f_s > 2 \times f_{max}$$

where f_s is the sampling frequency, and f_{max} is the maximum frequency present in the signal

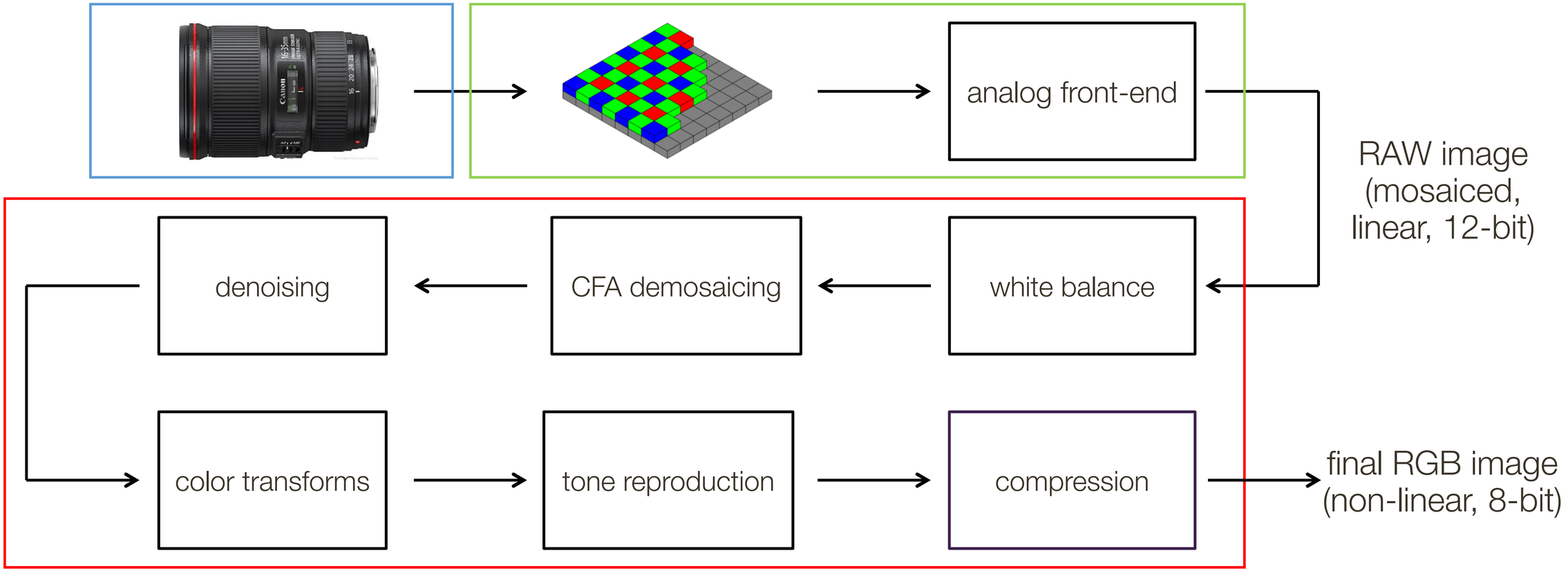
Futhermore, Nyquist's theorem states that a signal is **exactly recoverable** from its **samples** if sampled at the **Nyquist rate** (or higher)

Note: that a signal must be **bandlimited** for this to apply (i.e., it has a maximum frequency)

Back to **Processing in Camera**

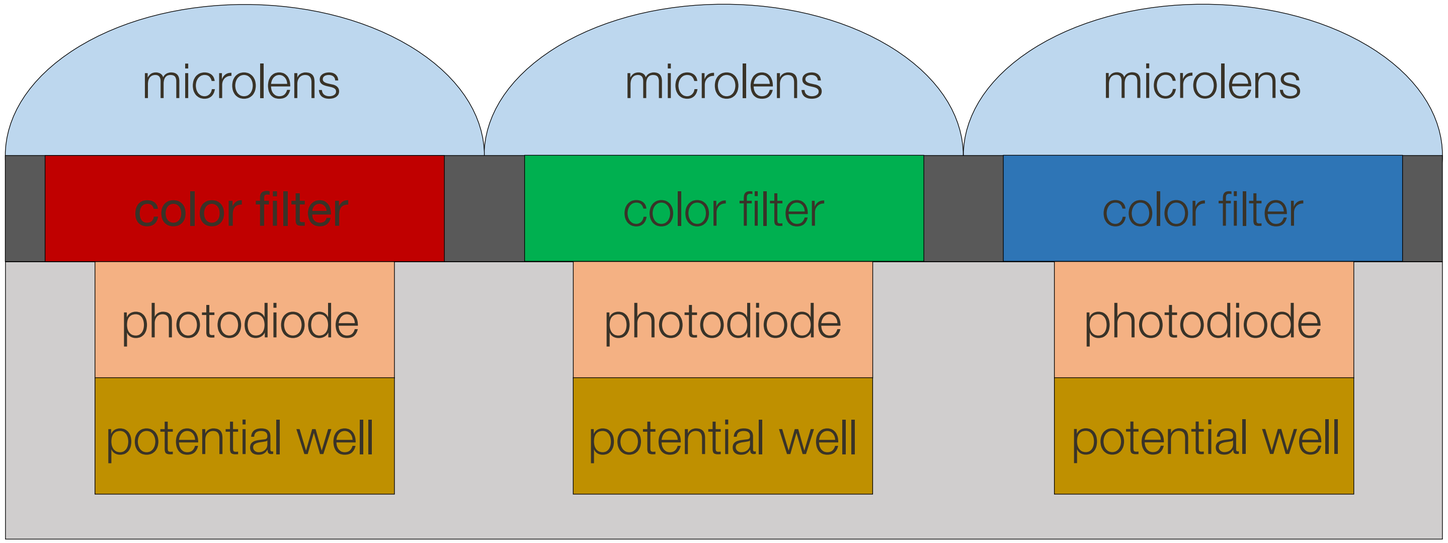
(in camera) **Image** Processing Pipeline

The sequence of image processing operations applied by the camera's image signal processor (ISP) to convert a RAW image into a "conventional" image.



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

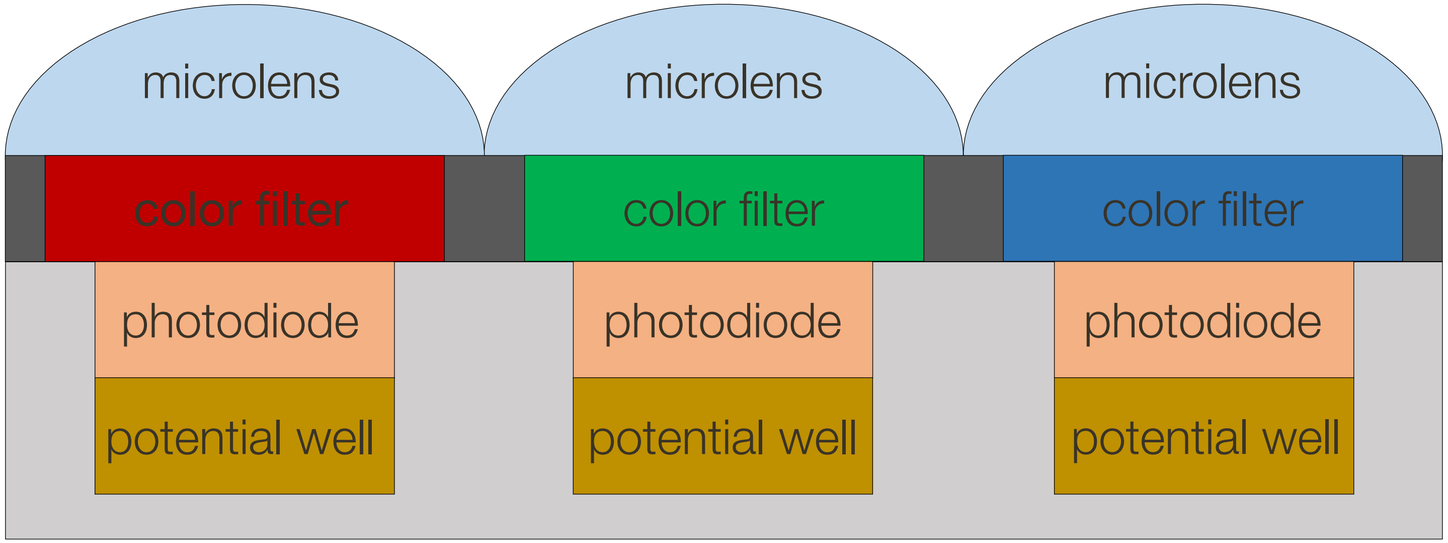
Color Filter Arrays (CFA)



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Color Filter Arrays (CFA)

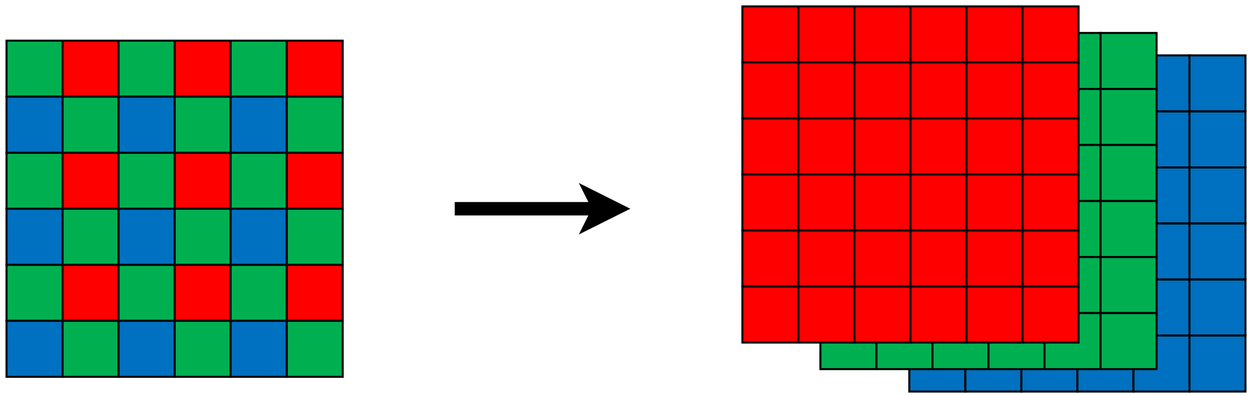
Implication: Only certain wavelengths of light are recorded at a given pixel



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

CFA **Demosicing**

Produce full RGB image from mosaiced sensor output



Interpolate from neighbors:

— Bilinear interpolation (needs 4 neighbors)

— Bicubic interpolation (needs more neighbors, may overblur)

— Edge-aware interpolation (e.g., Bilateral)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

(in camera) **White** balance

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

(in camera) **White** balance

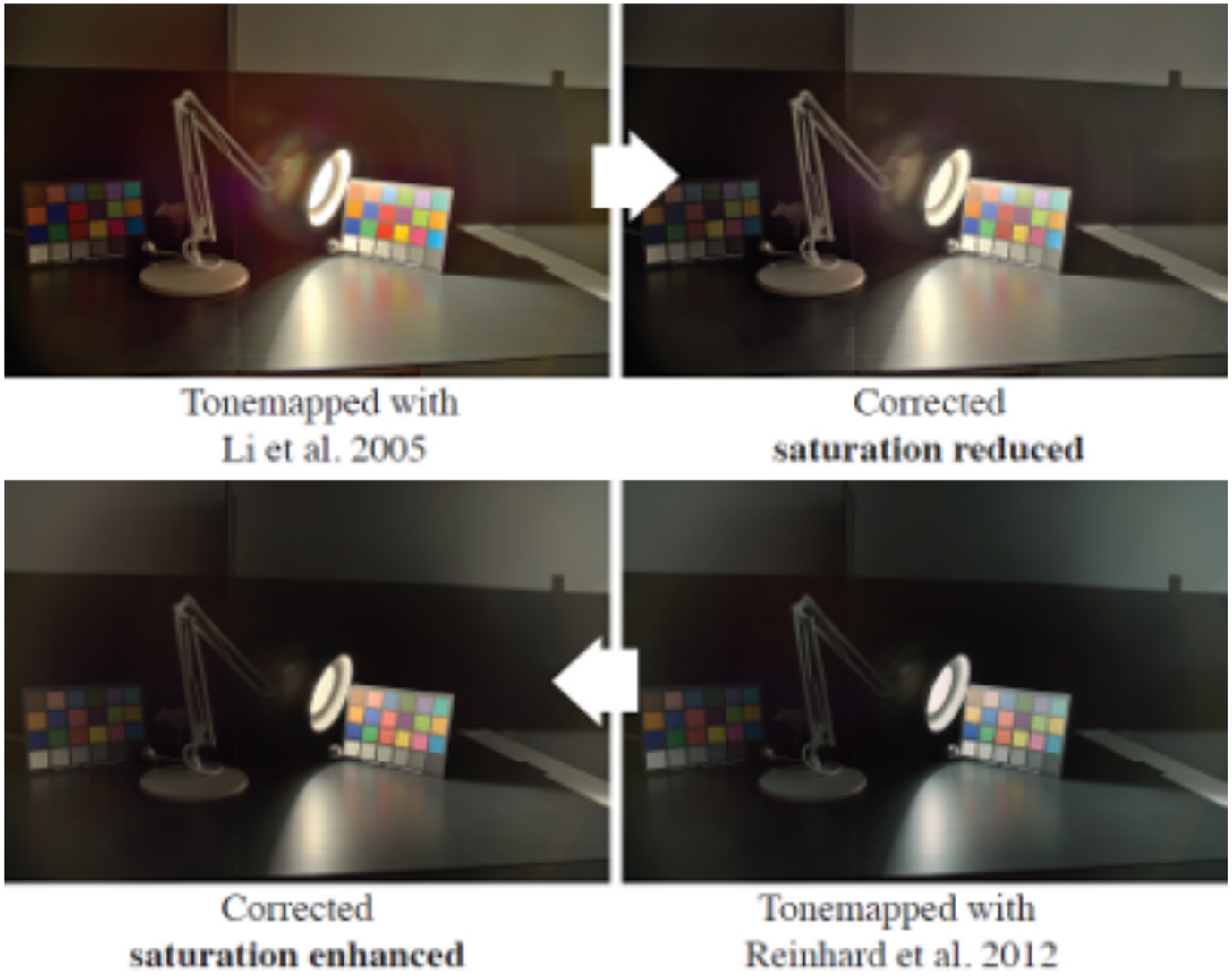
R: 200
G: 255
B: 190

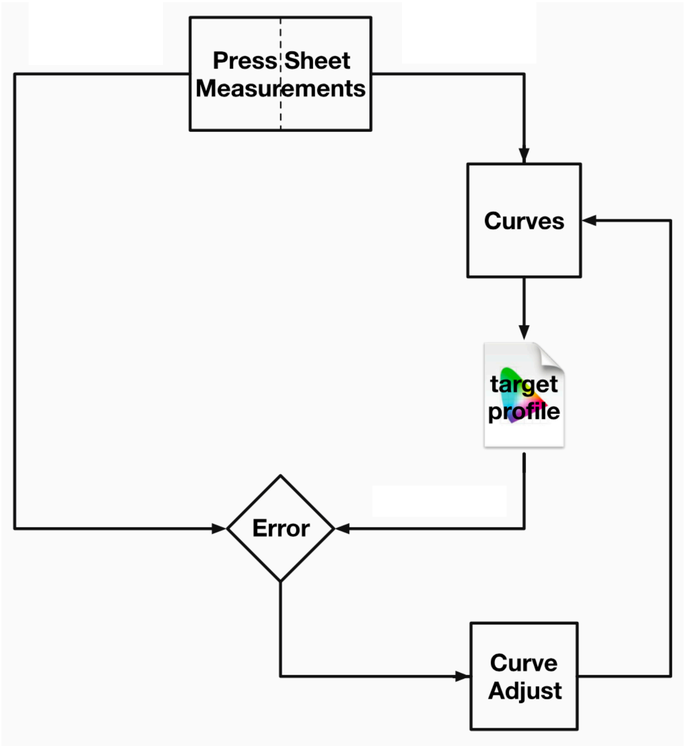
→

R-correction: + 55
G-correction: + 0
B-correction: + 65

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

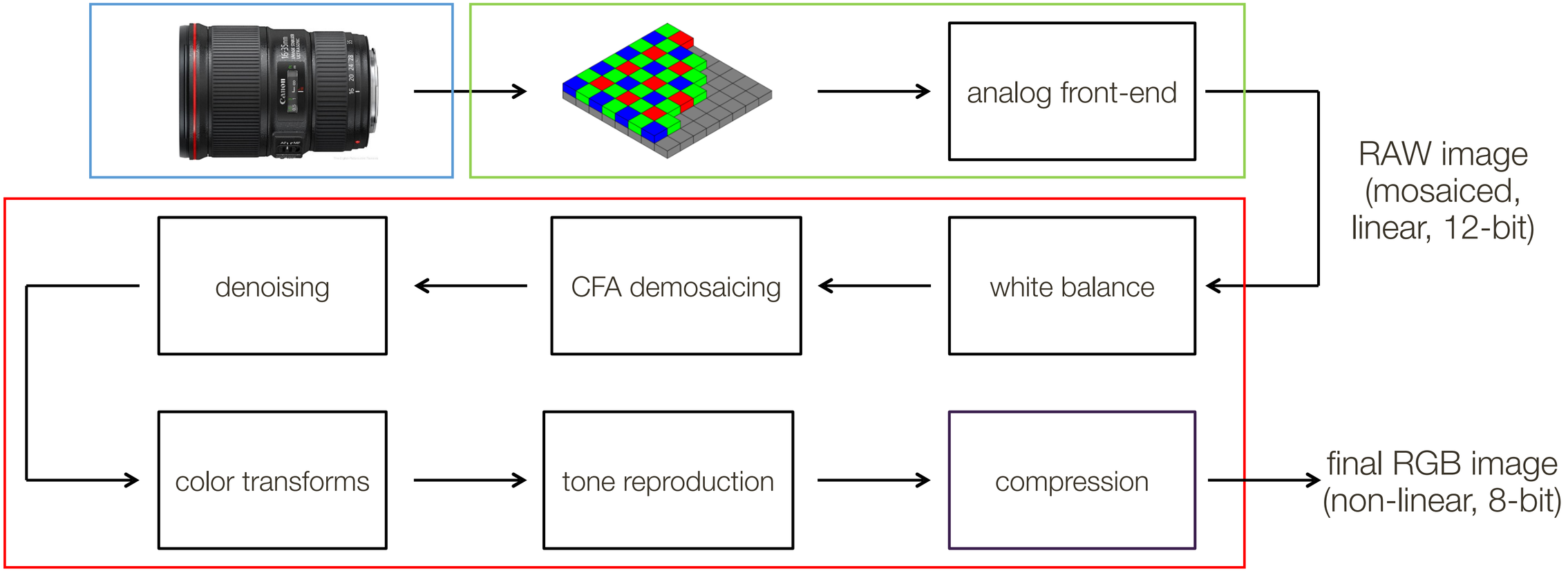
(in camera) **Tone** reproduction





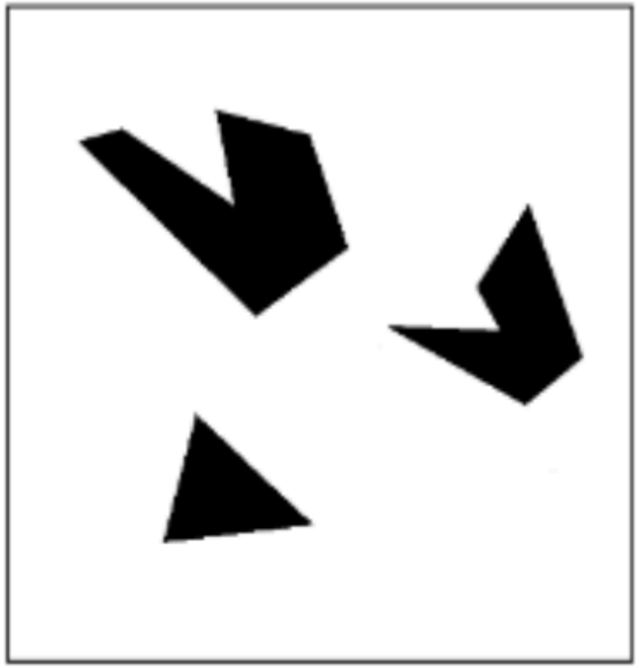
(in camera) **Image** Processing Pipeline

The sequence of image processing operations applied by the camera's image signal processor (ISP) to convert a RAW image into a "conventional" image.



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Template Matching



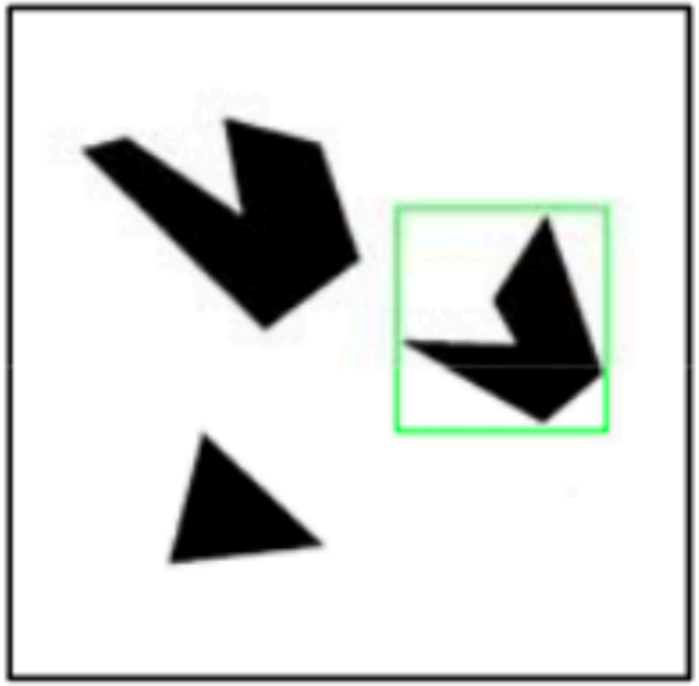
Scene

Template (mask)

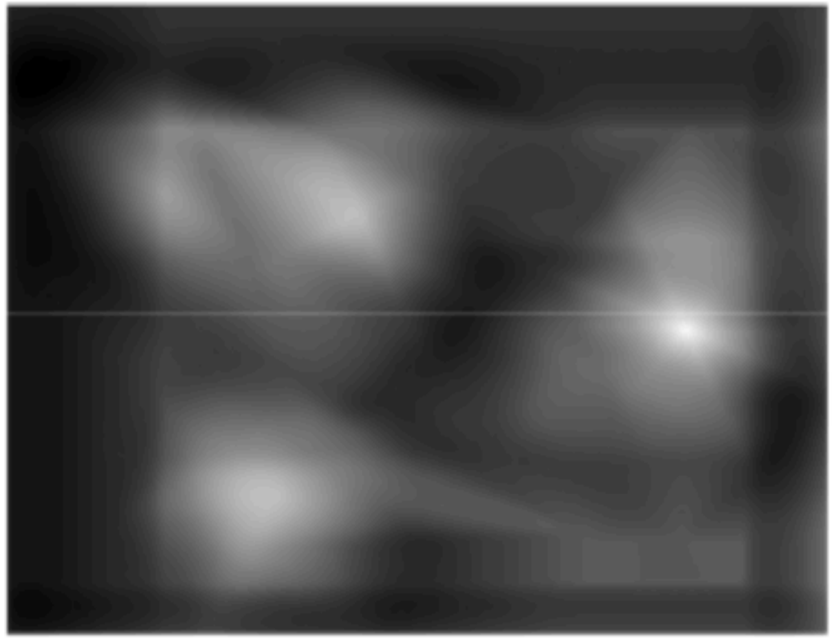
A toy example

Slide Credit: Kristen Grauman

Template Matching



Detected template



Correlation map

Slide Credit: Kristen Grauman

Template Matching

Similarity measures between a filter $\mathbf{J}$ local image region $\mathbf{I}$

Correlation, CORR = $\mathbf{I} \cdot \mathbf{J} = \mathbf{I}^T \mathbf{J}$

Normalised Correlation, NCORR = $\dfrac{\mathbf{I}^T \mathbf{J}}{|\mathbf{I}||\mathbf{J}|} = \cos\theta$

Sum Squared Difference, SSD = $|\mathbf{I} - \mathbf{J}|^2$

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

Minimising SSD and maximizing Normalized Correlation are equivalent if $|\mathbf{I}| = |\mathbf{J}| = 1$

Template Matching

Convolve image with template, find local maxima

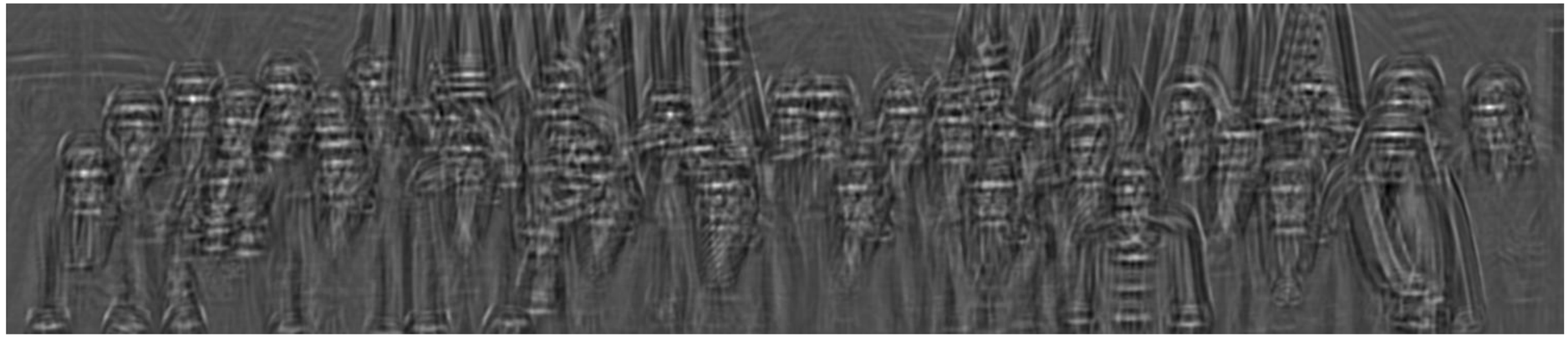
Template Matching

Convolve image with template, find local maxima

$*$  $\rightarrow$

Template Matching

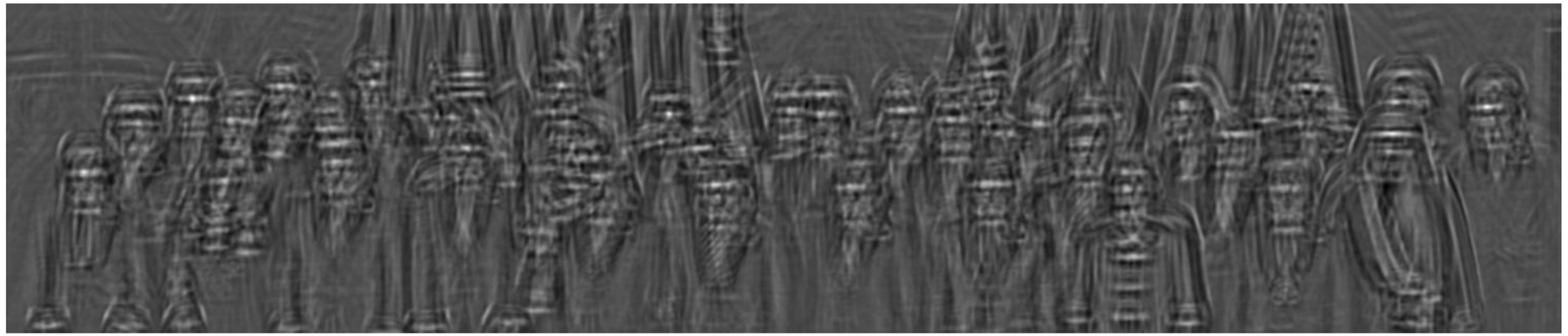
Convolve image with template, find local maxima



Template Matching

Convolve image with template, find local maxima

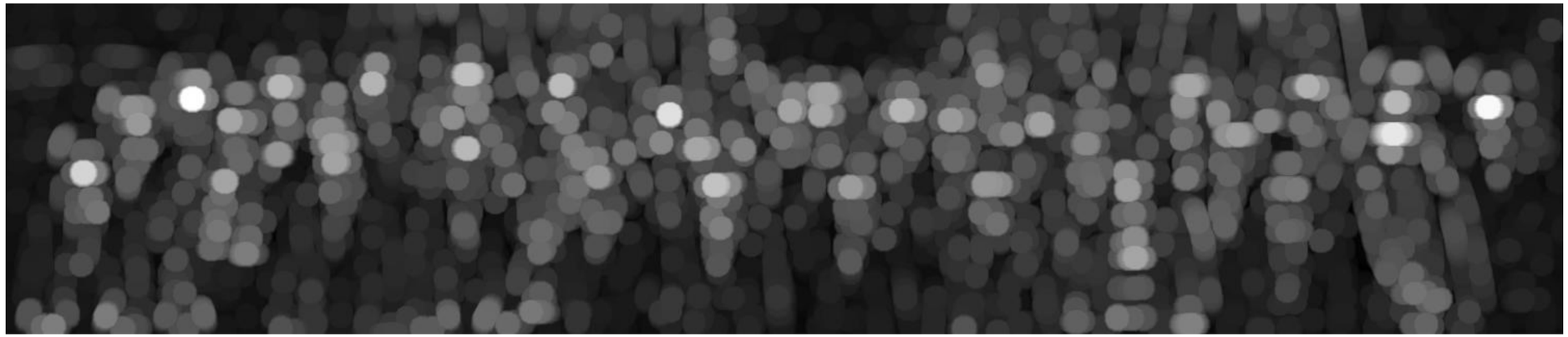
Non-max suppress



Template Matching

Convolve image with template, find local maxima

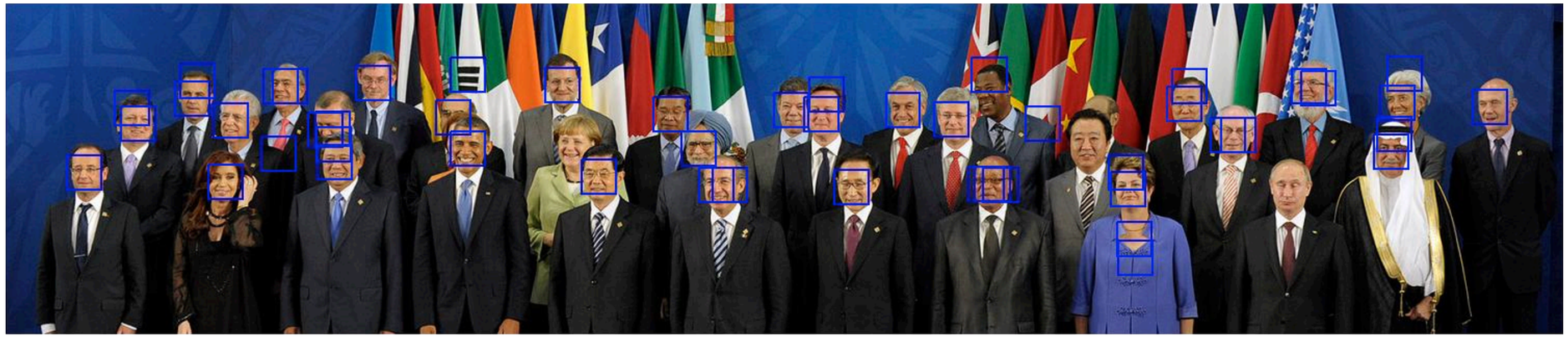
Non-max suppress



Template Matching

Convolve image with template, find local maxima

Non-max suppress
+ threshold



Template Matching

When might **template matching fail**?

— Different scales 

— Different orientation 

— Lighting conditions 

— Left vs. Right hand 

— Partial Occlusions 

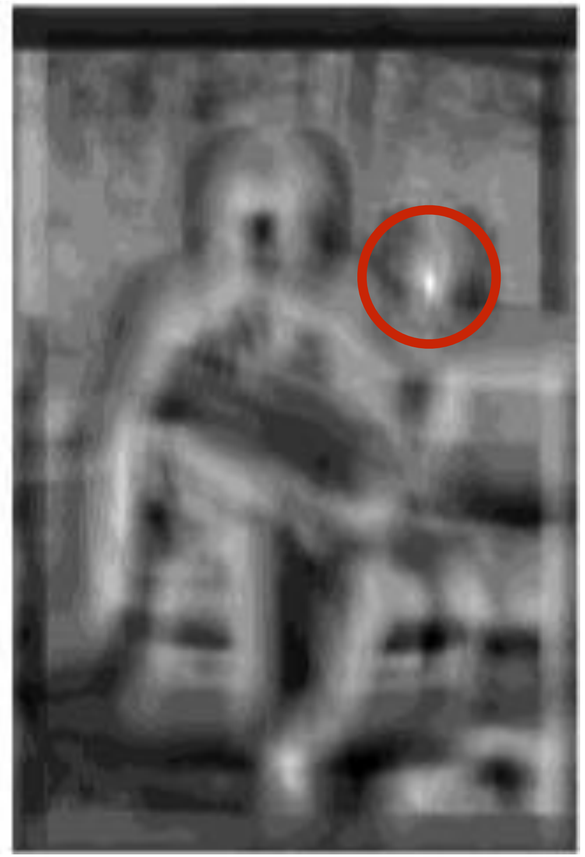
— Different Perspective

— Motion / blur

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.

Sample Question: Template Matching

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

$$[x_1, x_2, x_3, ..., x_N]$$

Sample Question: Template Matching

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

$$[x_1, x_2, x_3, ..., x_N]$$

After (L2) normalization:

$$\frac{[x_1, x_2, x_3, ..., x_N]}{\sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_N^2}}$$

Sample Question: Template Matching

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

$$[x_1, x_2, x_3, ..., x_N]$$

$$[kx_1, kx_2, kx_3, ..., kx_N]$$

After (L2) normalization:

$$\frac{[x_1, x_2, x_3, ..., x_N]}{\sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_N^2}}$$

Sample Question: Template Matching

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

$$[x_1, x_2, x_3, ..., x_N]$$

$$[kx_1, kx_2, kx_3, ..., kx_N]$$

After (L2) normalization:

$$\frac{[x_1, x_2, x_3, ..., x_N]}{\sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_N^2}}$$

After (L2) normalization:

$$\frac{[kx_1, kx_2, kx_3, ..., kx_N]}{\sqrt{k^2x_1^2 + k^2x_2^2 + k^2x_3^2 + ... + k^2x_N^2}}$$

Sample Question: Template Matching

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

$$[x_1, x_2, x_3, ..., x_N]$$

$$[kx_1, kx_2, kx_3, ..., kx_N]$$

After (L2) normalization:

$$\frac{[x_1, x_2, x_3, ..., x_N]}{\sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_N^2}}$$

After (L2) normalization:

$$\frac{[kx_1, kx_2, kx_3, ..., kx_N]}{\sqrt{k^2x_1^2 + k^2x_2^2 + k^2x_3^2 + ... + k^2x_N^2}}$$

$$\frac{k[x_1, x_2, x_3, ..., x_N]}{\sqrt{k^2}\sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_N^2}}$$

Sample Question: Template Matching

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

$$[x_1, x_2, x_3, ..., x_N]$$

$$[kx_1, kx_2, kx_3, ..., kx_N]$$

After (L2) normalization:

$$\frac{[x_1, x_2, x_3, ..., x_N]}{\sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_N^2}}$$

After (L2) normalization:

$$\frac{[kx_1, kx_2, kx_3, ..., kx_N]}{\sqrt{k^2x_1^2 + k^2x_2^2 + k^2x_3^2 + ... + k^2x_N^2}}$$

$$\frac{\cancel{k}[x_1, x_2, x_3, ..., x_N]}{\cancel{\sqrt{k^2}}\sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_N^2}}$$

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)

$G1$

Blur with a Gaussian kernel, then select every 2nd pixel

$$I_s(x, y) = I(x, y) * g_\sigma(x, y)$$

$G1$

blur

Blur with a Gaussian kernel, then select every 2nd pixel

$$I_s(x, y) = I(x, y) * g_\sigma(x, y)$$

$G1$

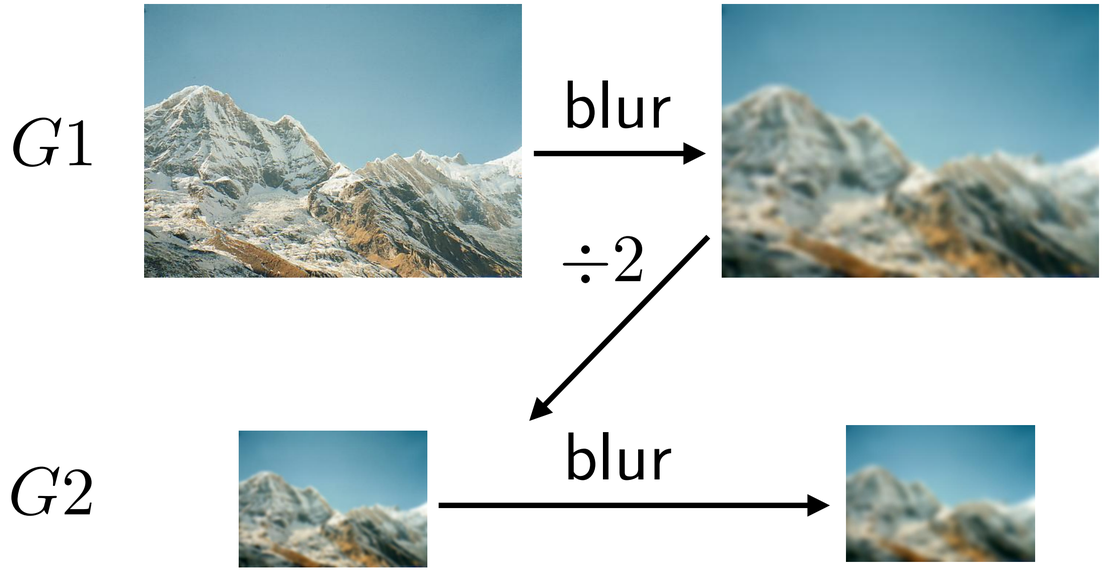
blur

$\div 2$

$G2$

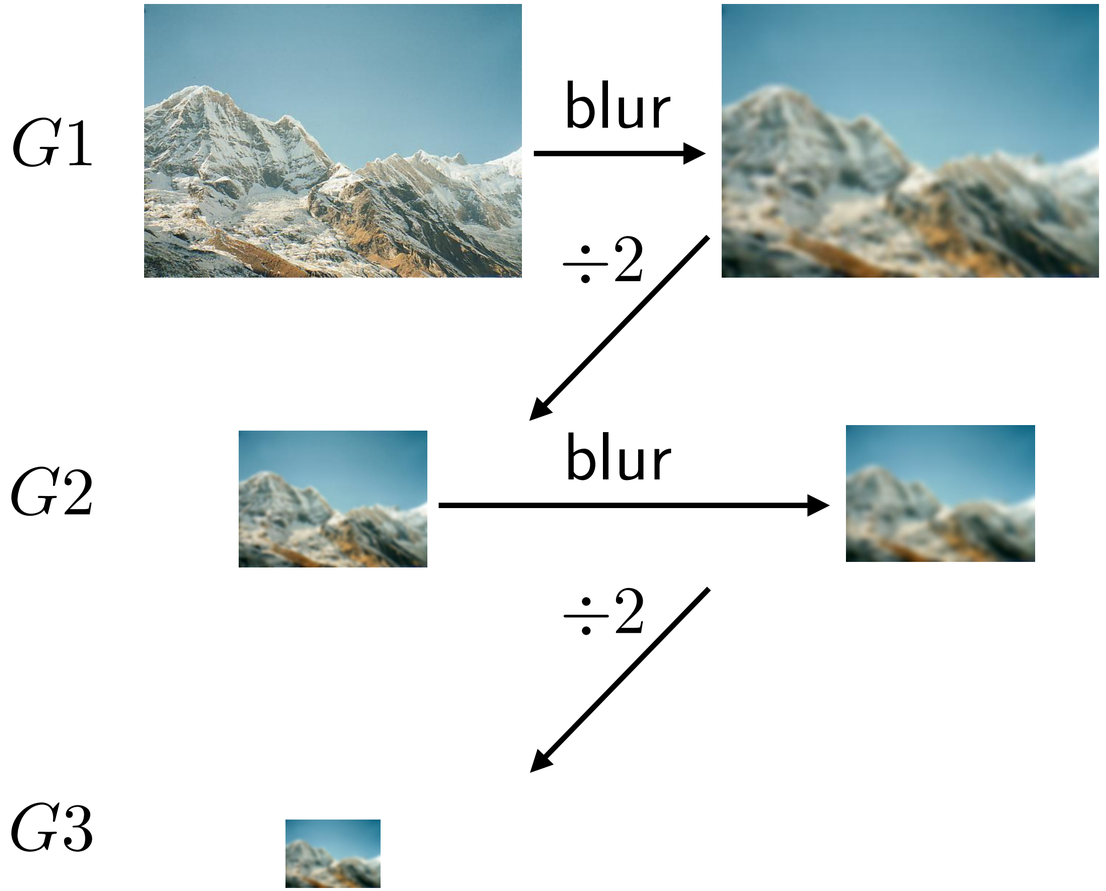
Blur with a Gaussian kernel, then select every 2nd pixel

$$I_s(x, y) = I(x, y) * g_\sigma(x, y)$$



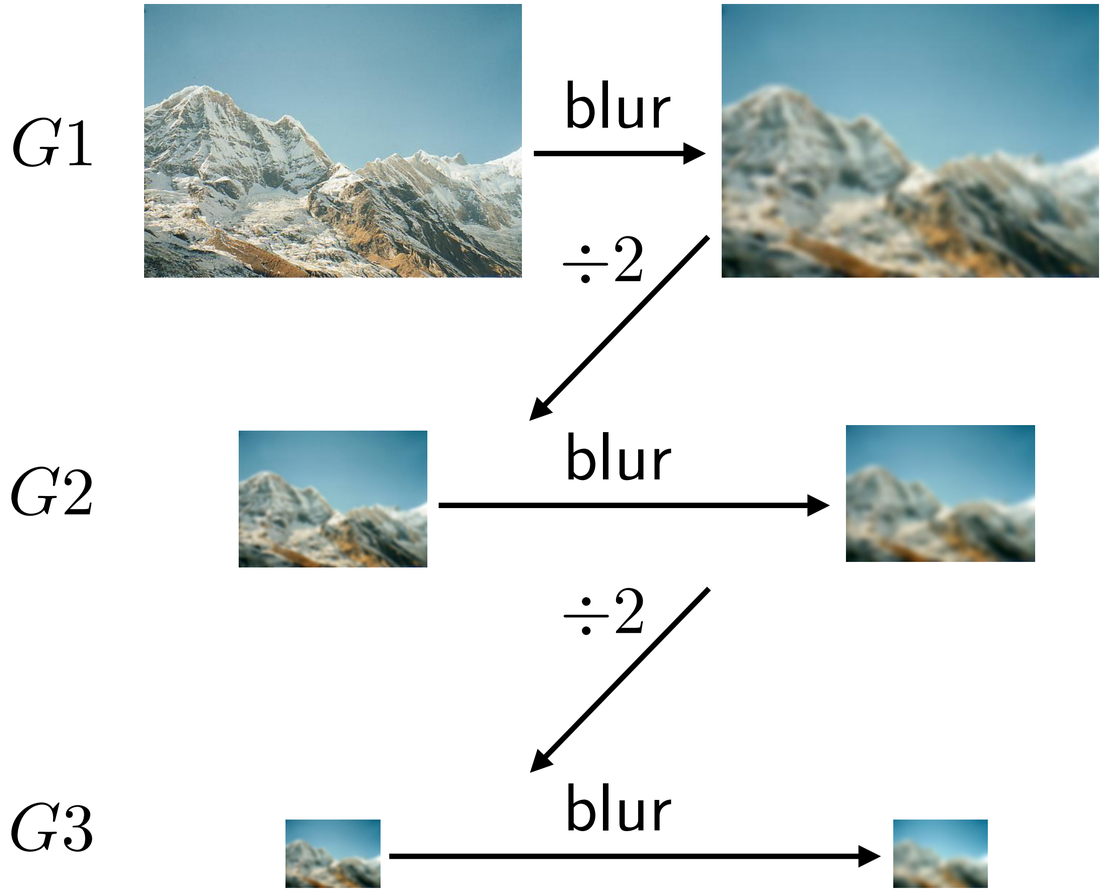
Blur with a Gaussian kernel, then select every 2nd pixel

$$I_s(x, y) = I(x, y) * g_\sigma(x, y)$$



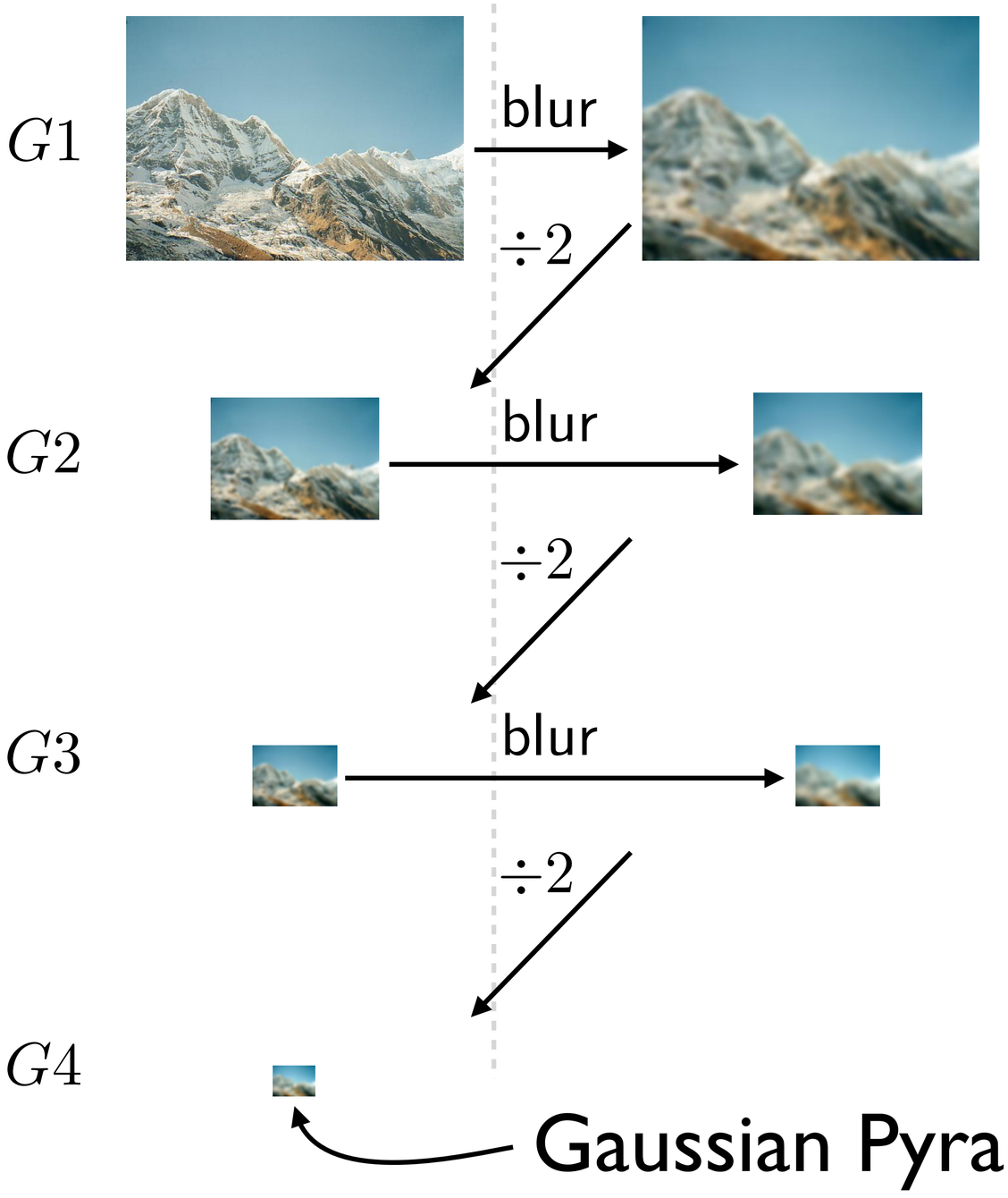
Blur with a Gaussian kernel, then select every 2nd pixel

$$I_s(x, y) = I(x, y) * g_\sigma(x, y)$$



Blur with a Gaussian kernel, then select every 2nd pixel

$$I_s(x, y) = I(x, y) * g_\sigma(x, y)$$



Blur with a Gaussian kernel, then select every 2nd pixel

$$I_s(x,y) = I(x,y) * g_\sigma(x,y)$$

$G1$ 

$G2$ 

$G3$ 

$G4$ 

Gaussian Pyramid

Laplacian Pyramid

$G1$

$G2$

$G3$

$G4$

$L4$

Gaussian Pyramid

Laplacian Pyramid

$G1$

$G2$

$G3$

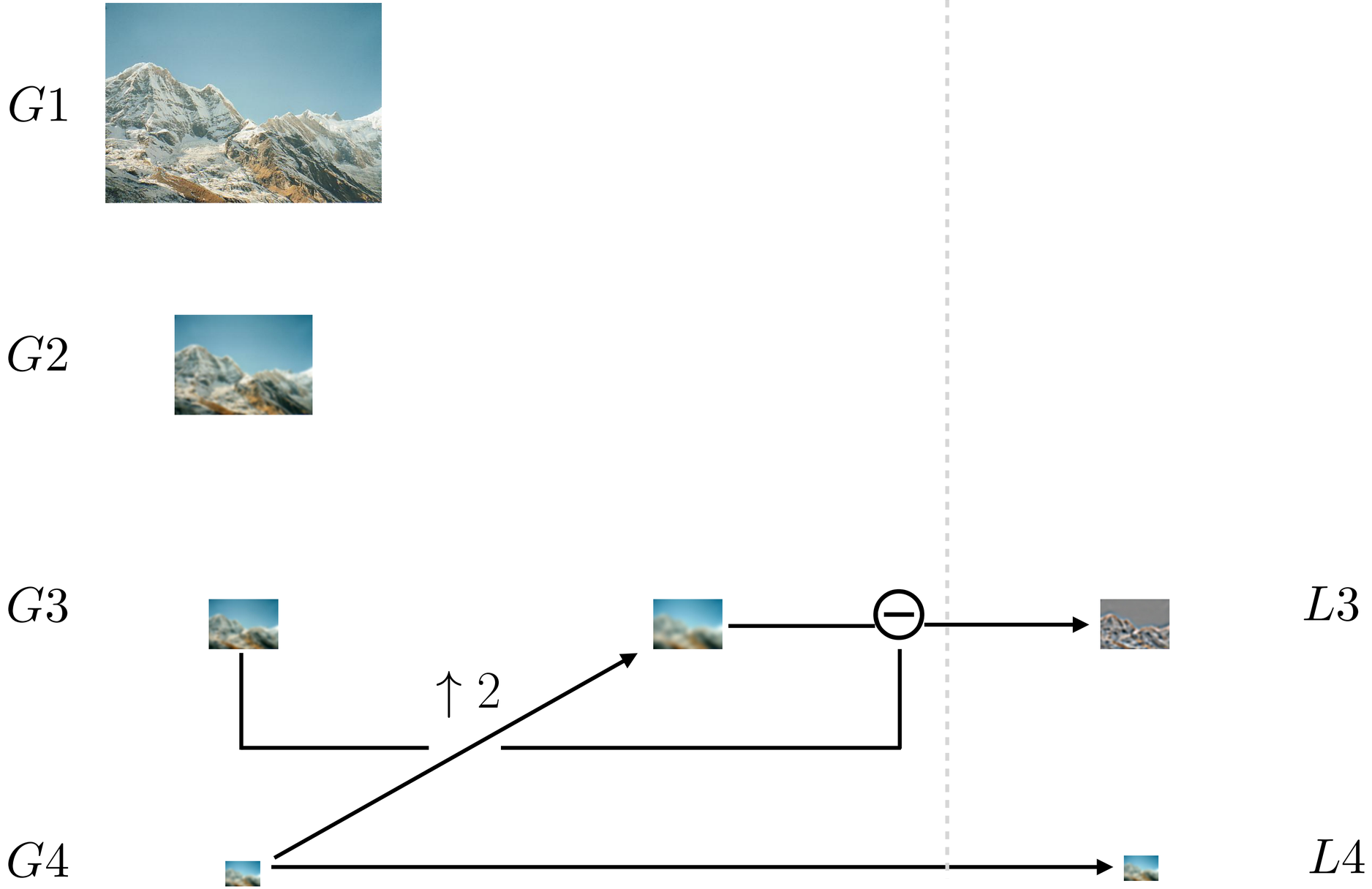
$\uparrow 2$

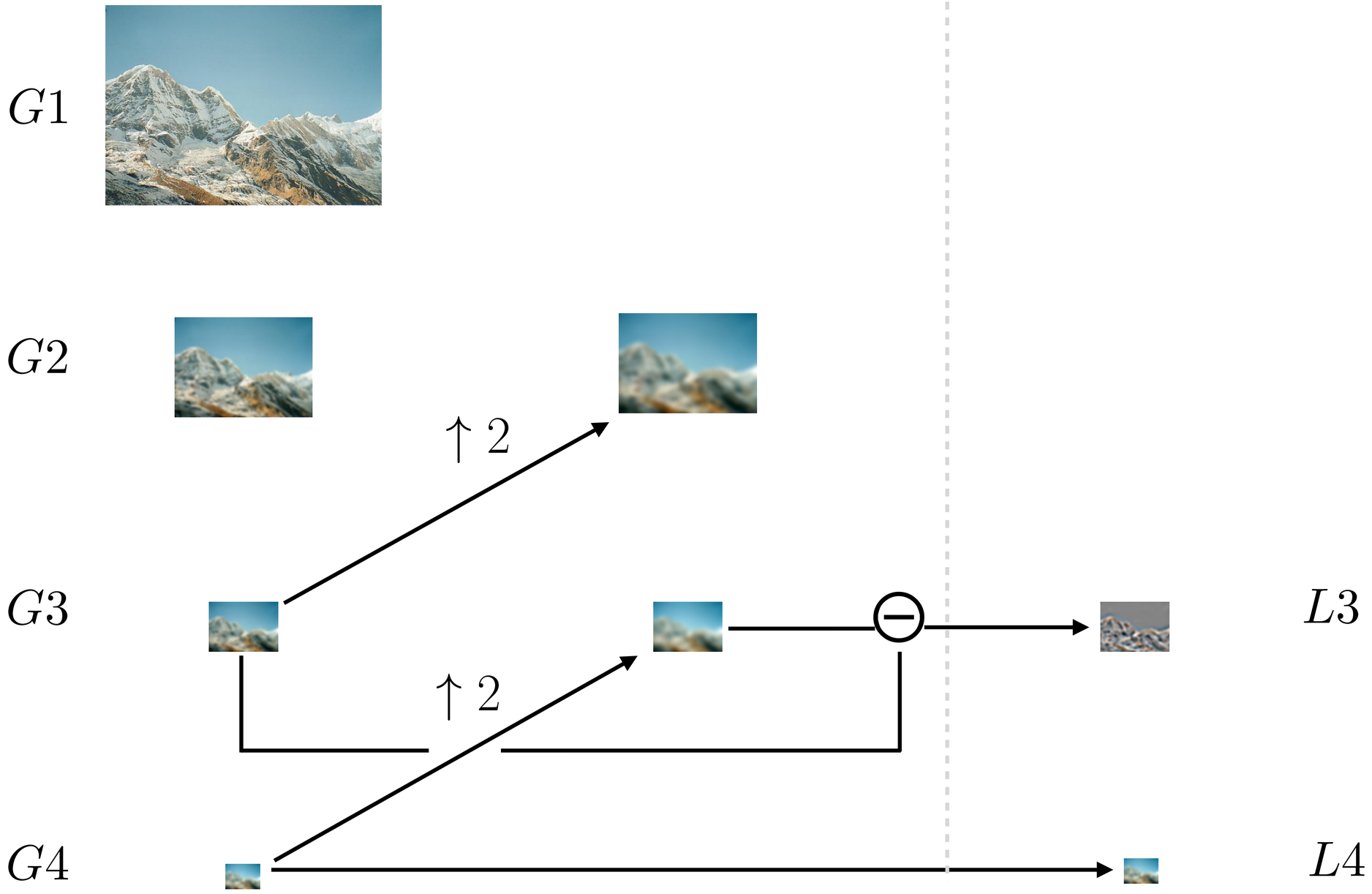
$G4$

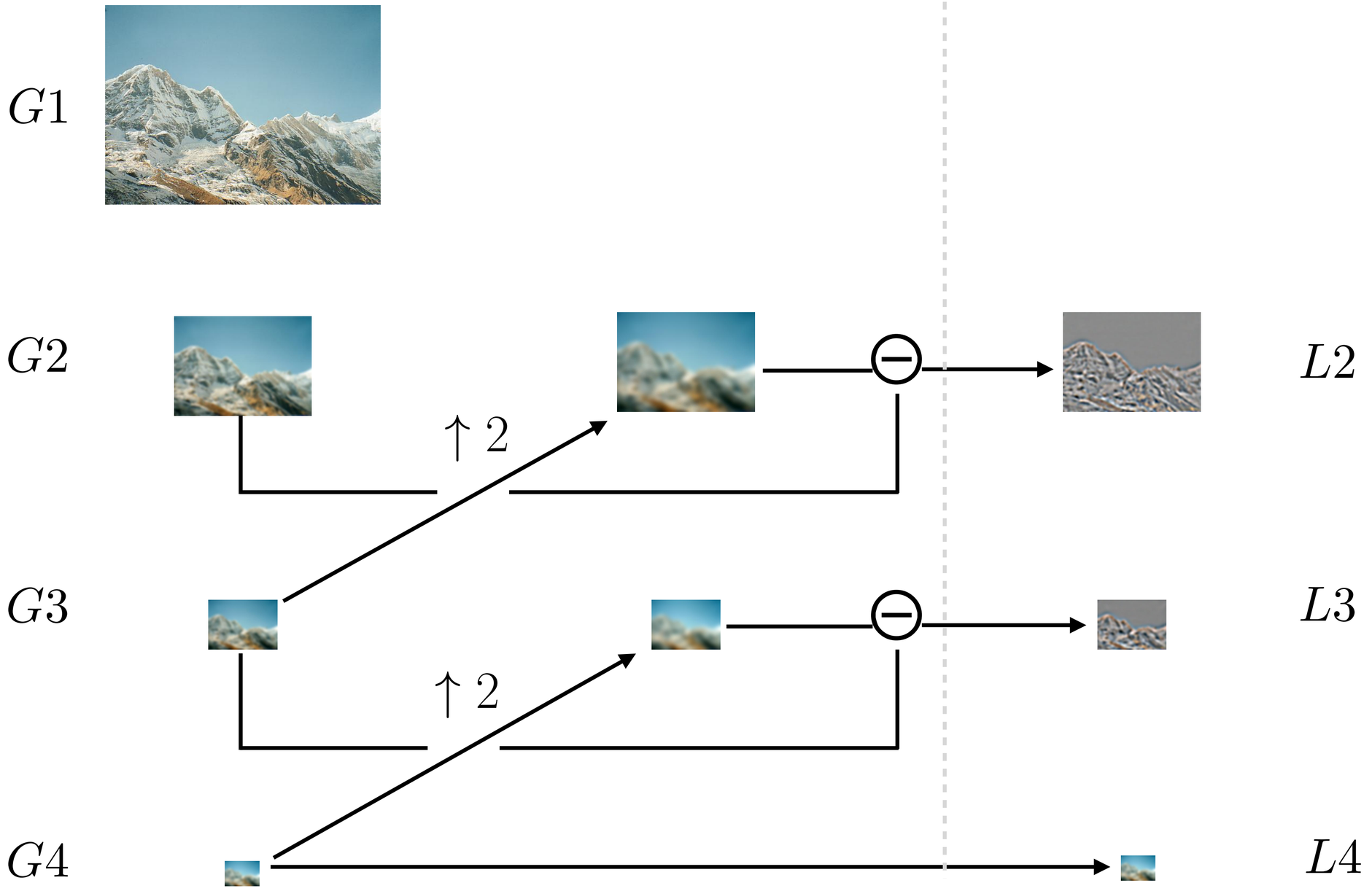
$L4$

Gaussian Pyramid

Laplacian Pyramid

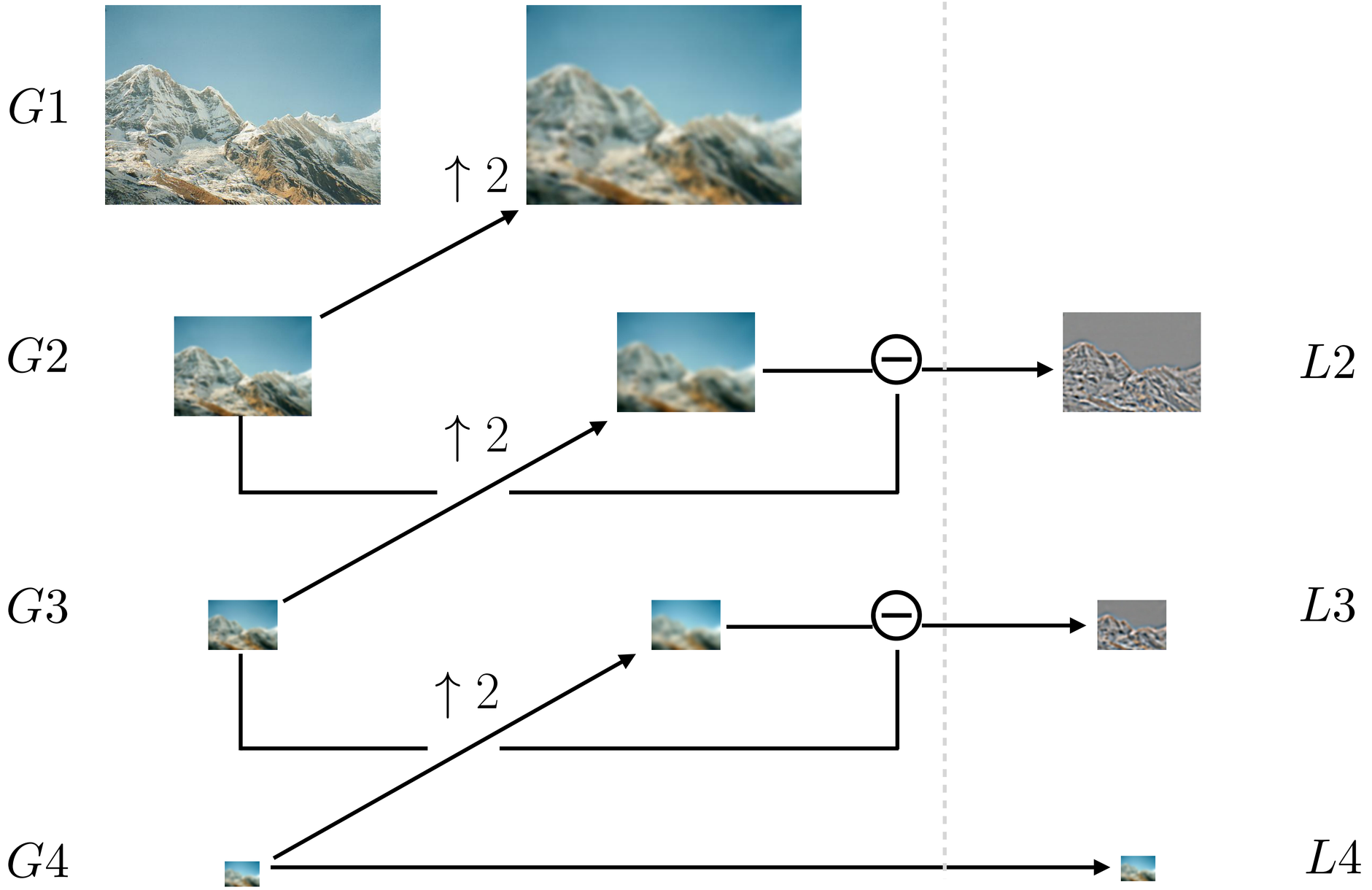

G1
G2
G3
G4
↑ 2
L3
L4
Gaussian Pyramid
Laplacian Pyramid


G1
G2
G3
G4
↑ 2
↑ 2
L3
L4
Gaussian Pyramid
Laplacian Pyramid



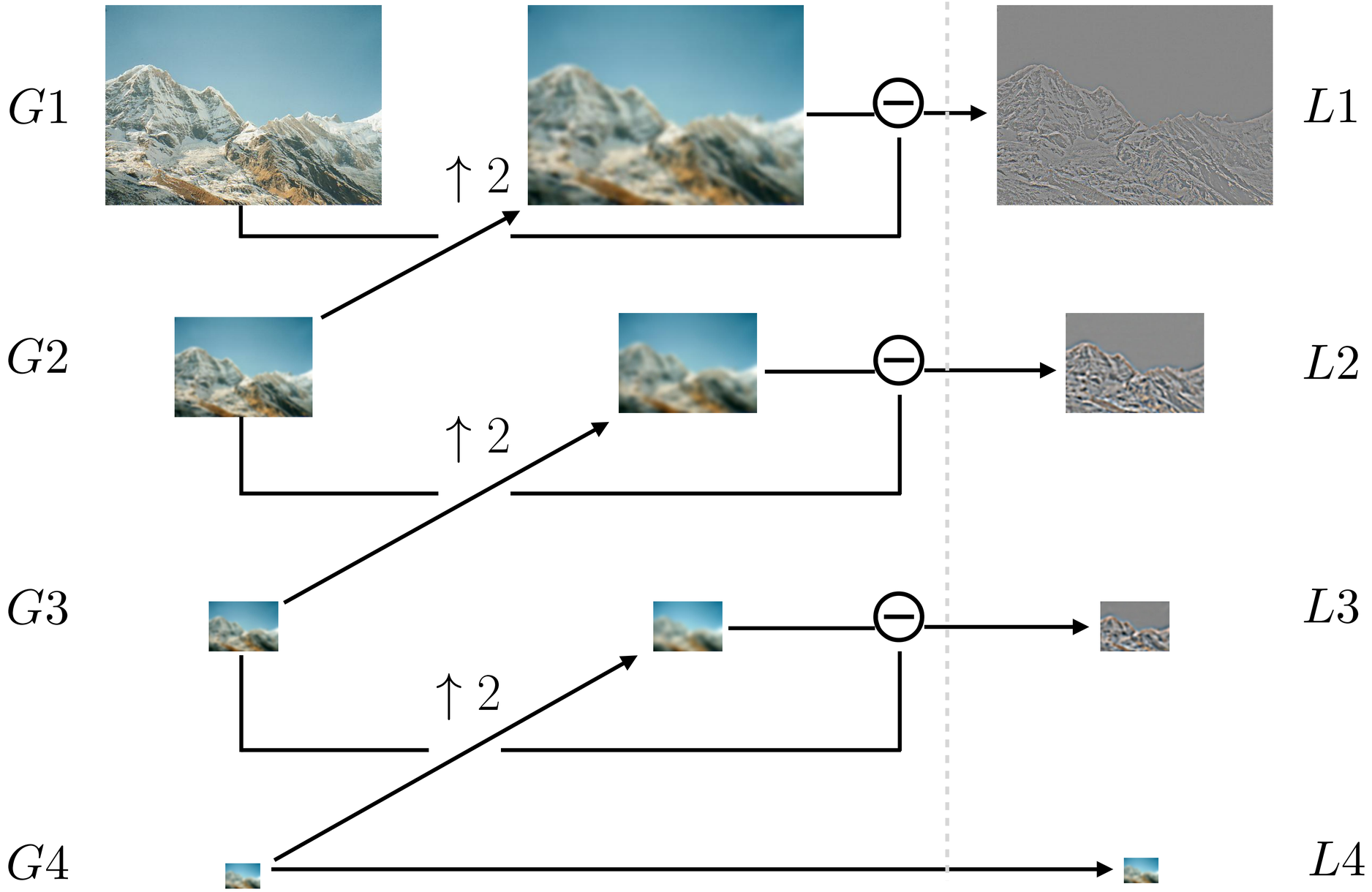
Gaussian Pyramid

Laplacian Pyramid



Gaussian Pyramid

Laplacian Pyramid



Gaussian Pyramid

Laplacian Pyramid

$G1$

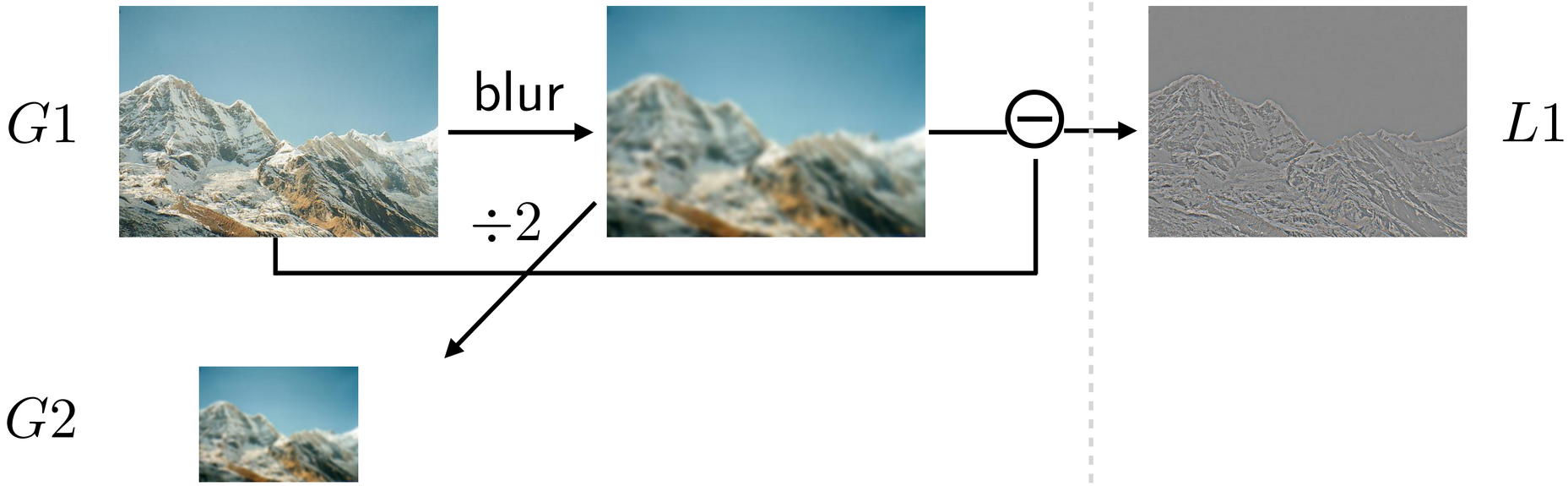
Laplacian Pyramid

$G1$

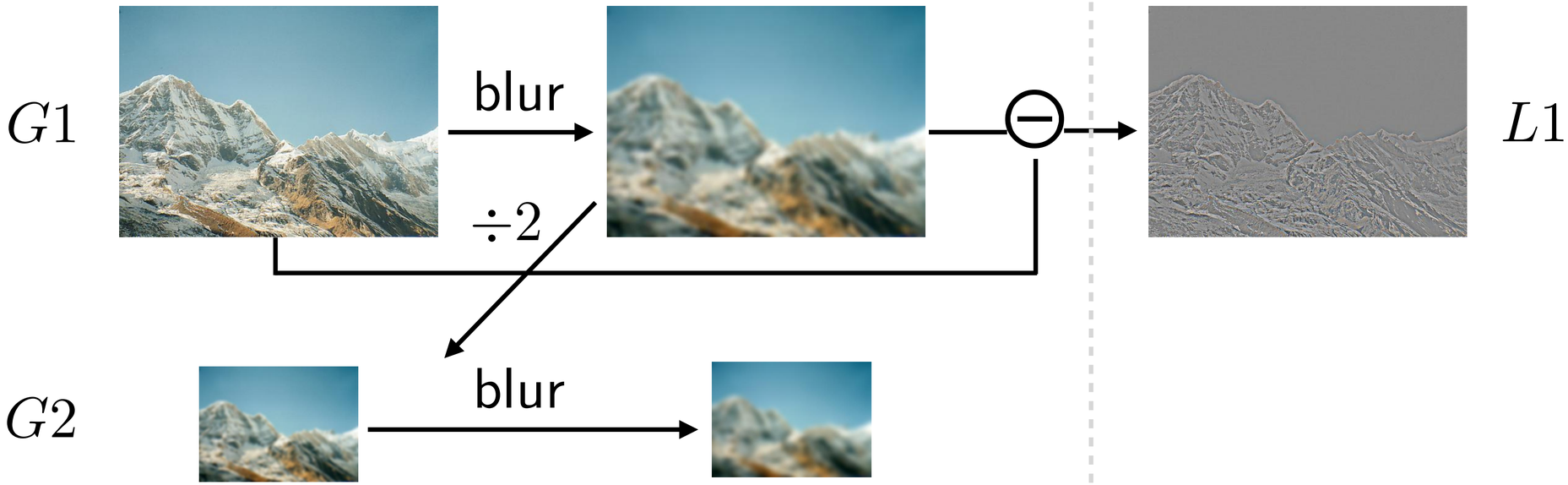
blur

Laplacian Pyramid

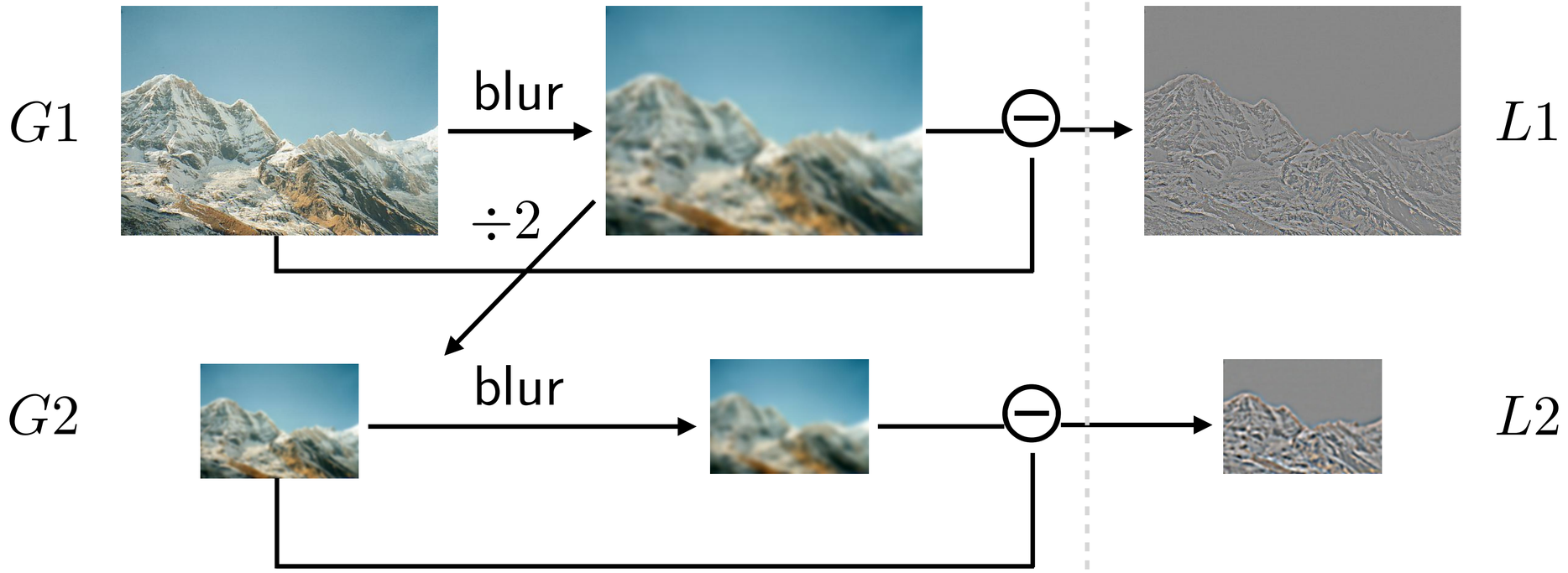
Laplacian Pyramid



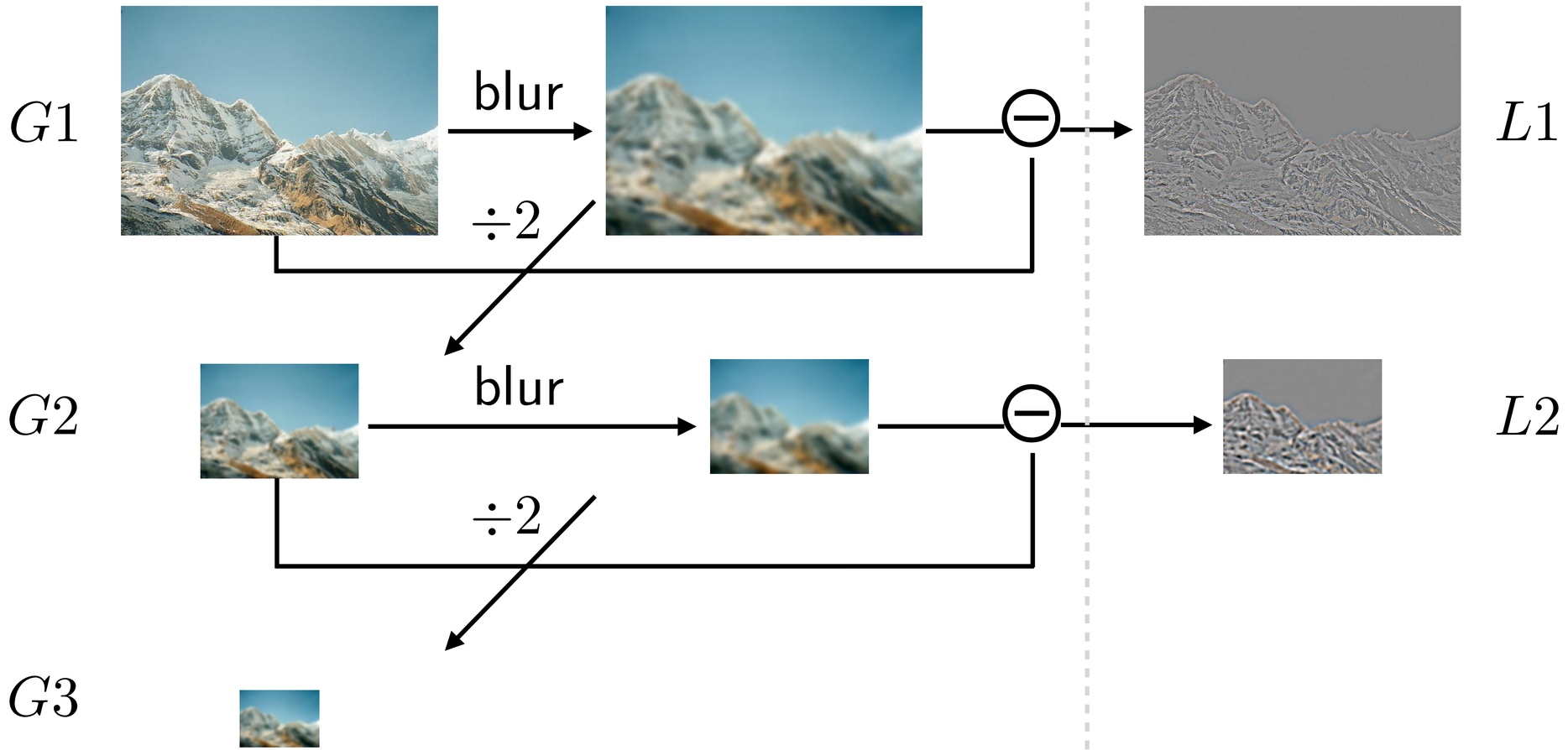
Laplacian Pyramid



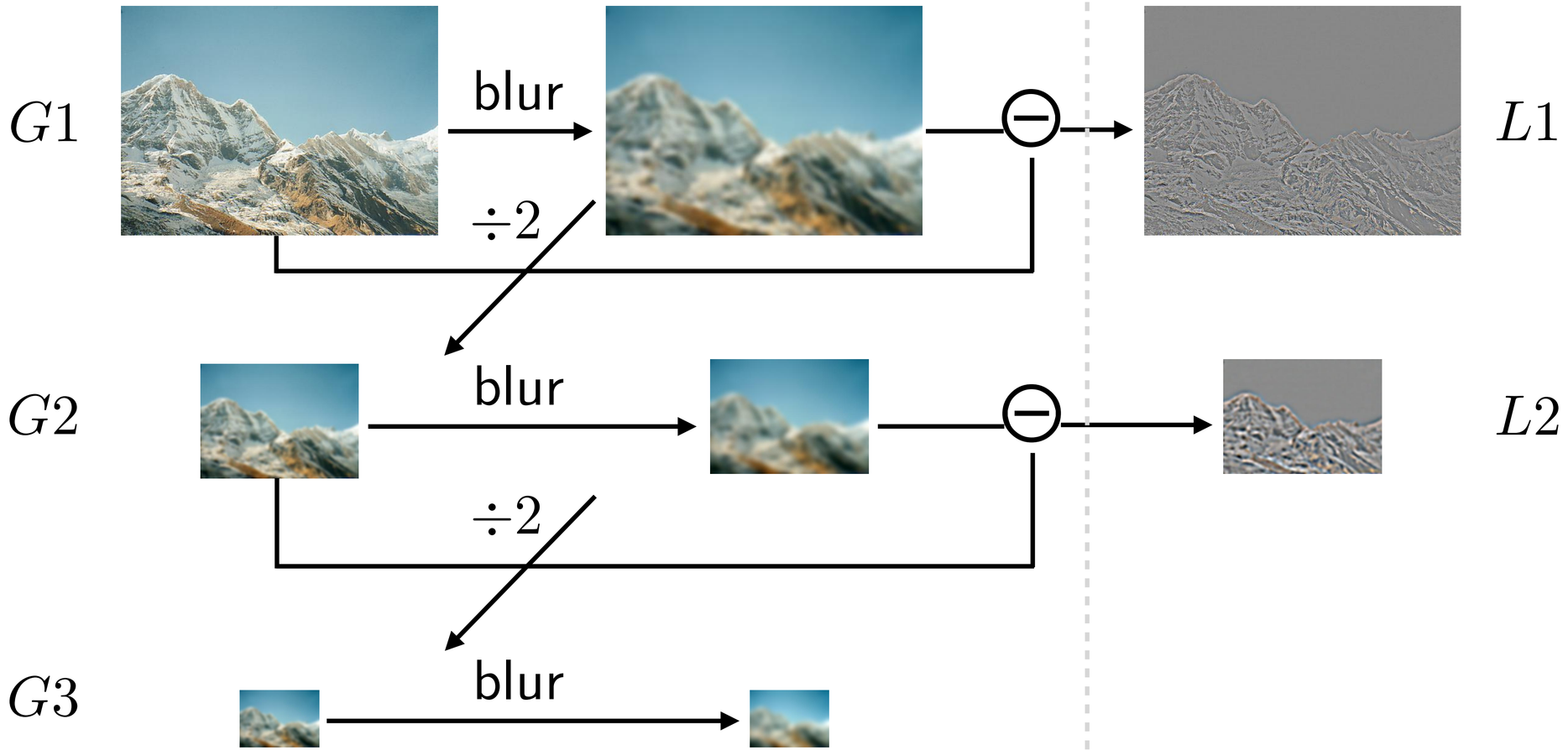
Laplacian Pyramid



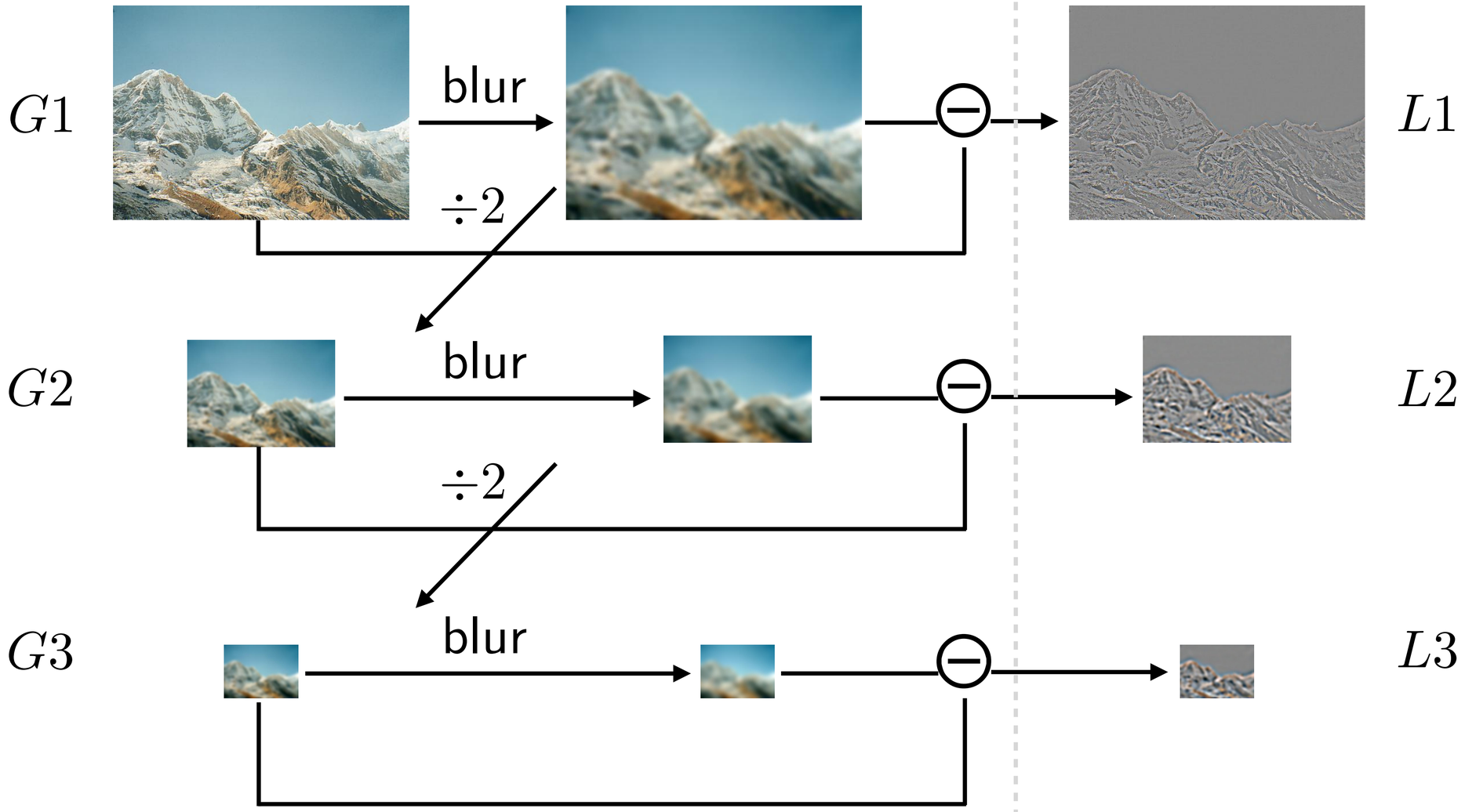
Laplacian Pyramid



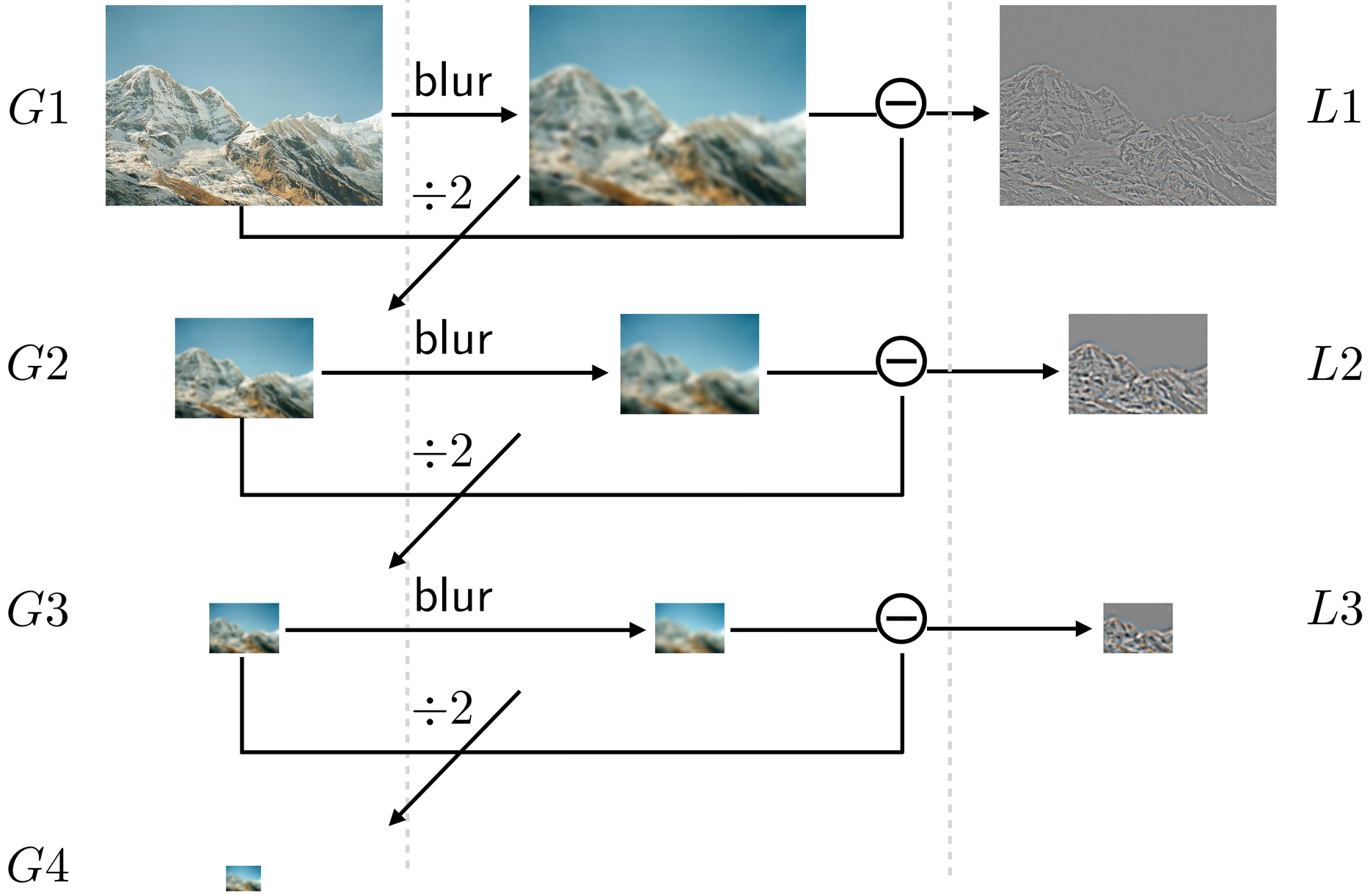
Laplacian Pyramid



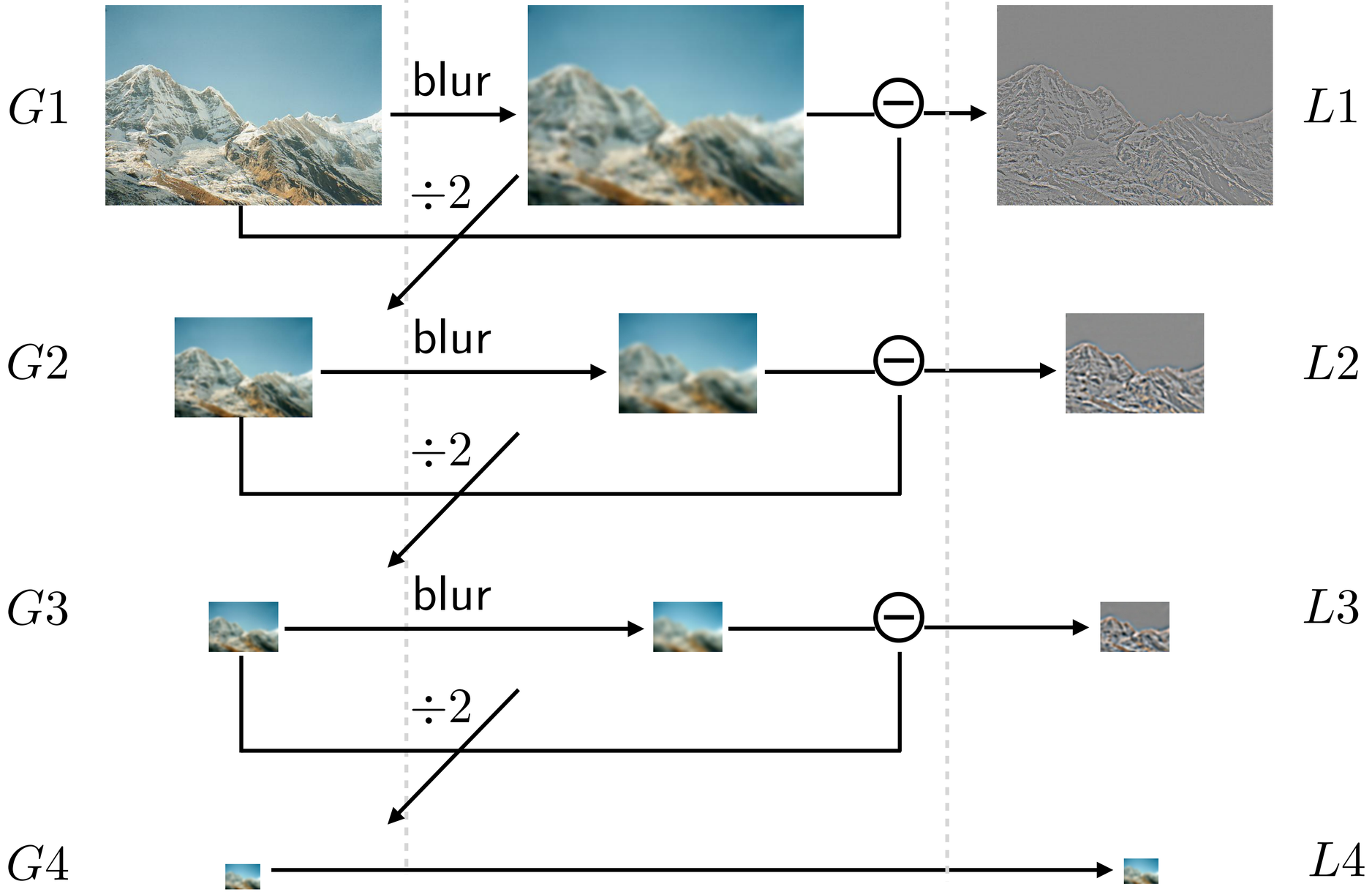
Laplacian Pyramid



Laplacian Pyramid



Laplacian Pyramid

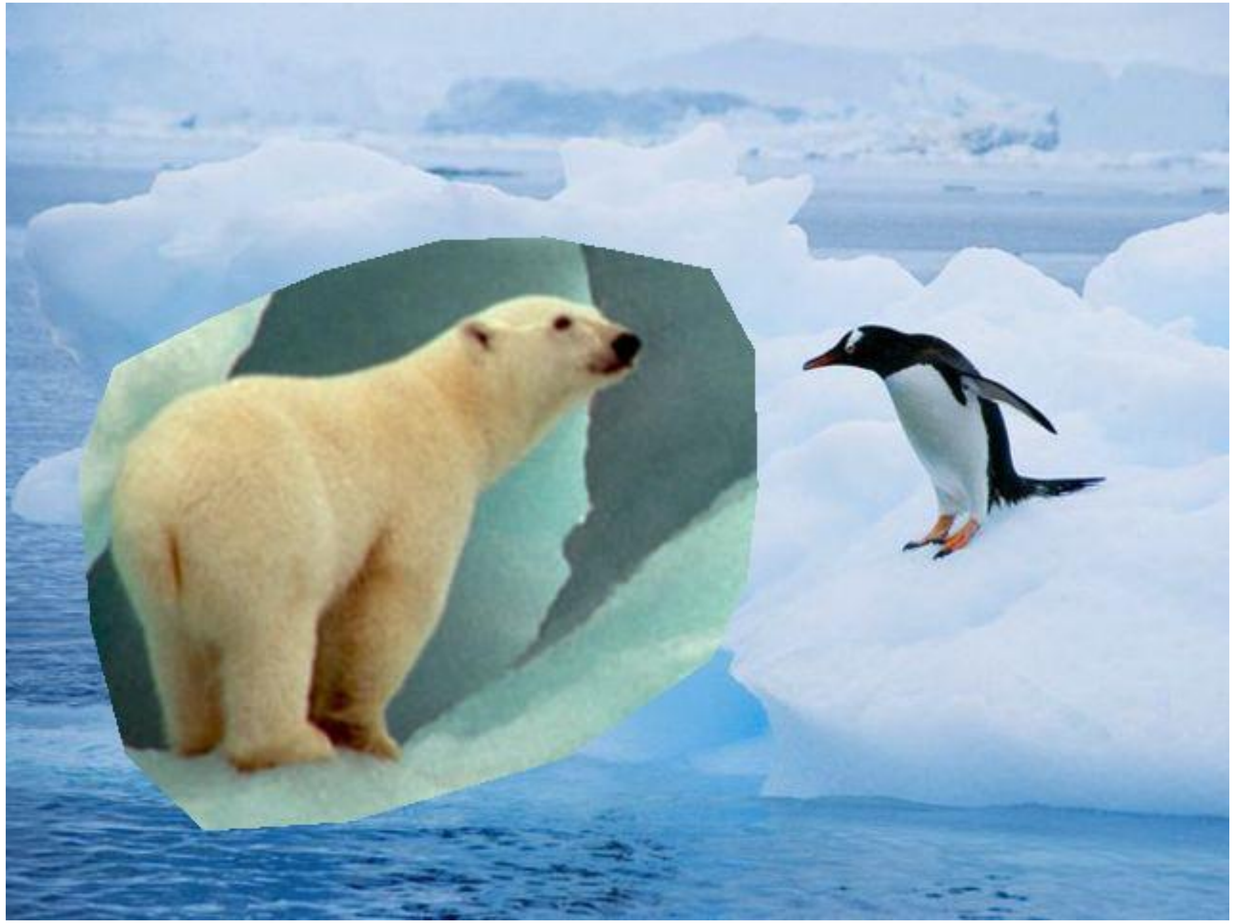


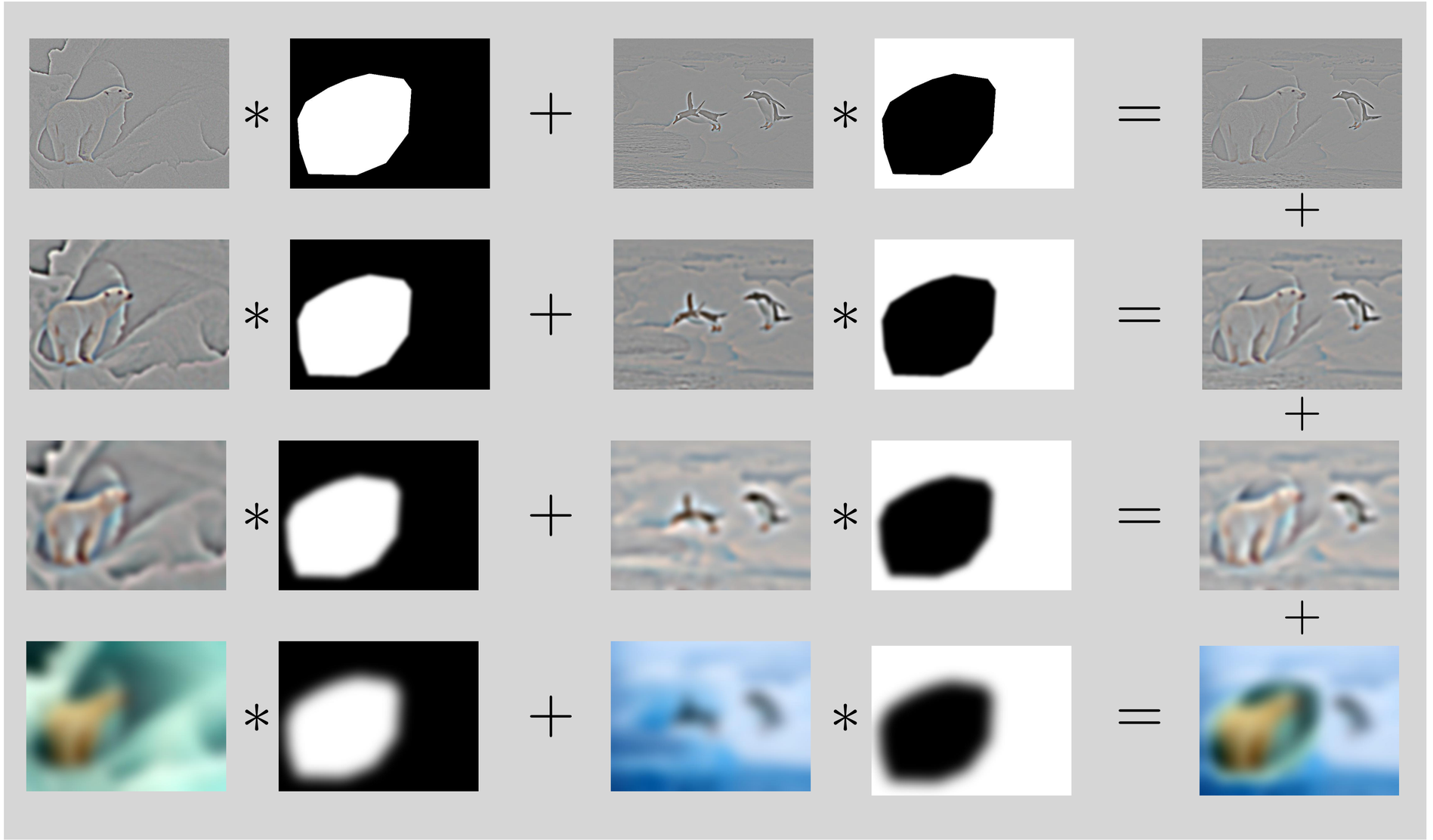
Laplacian Pyramid

Sample Question: Scaled Representations

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

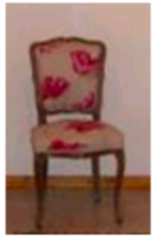
Pyramid Blending



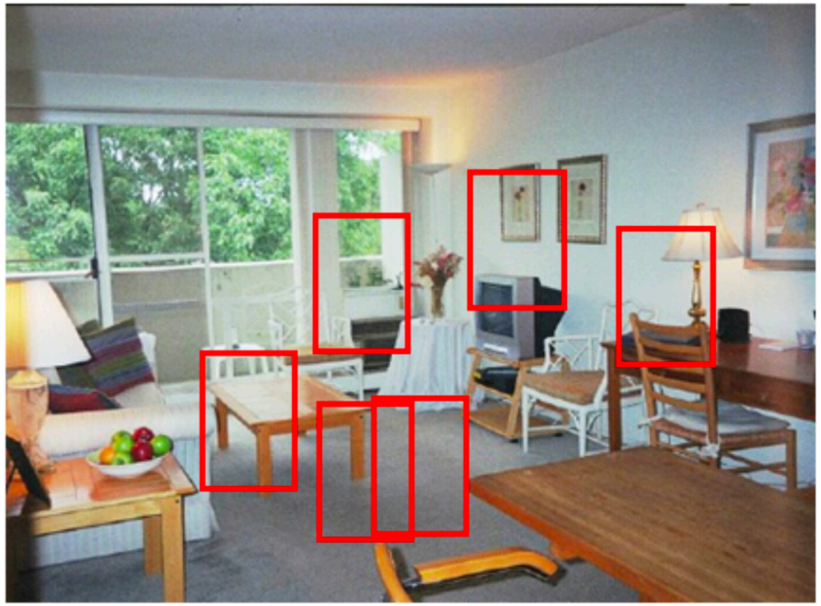


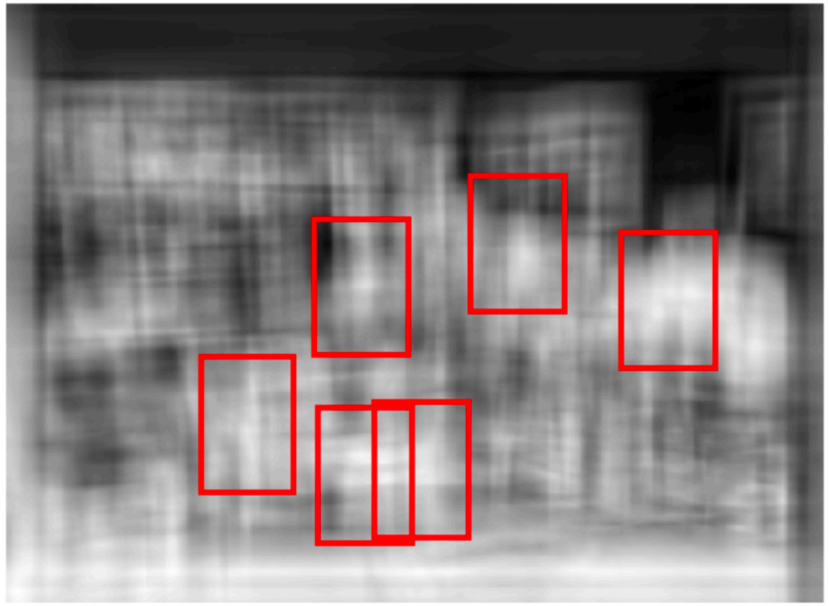
Step 2: blend lower frequency bands over larger spatial ranges, high frequency bands over small spatial ranges

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**

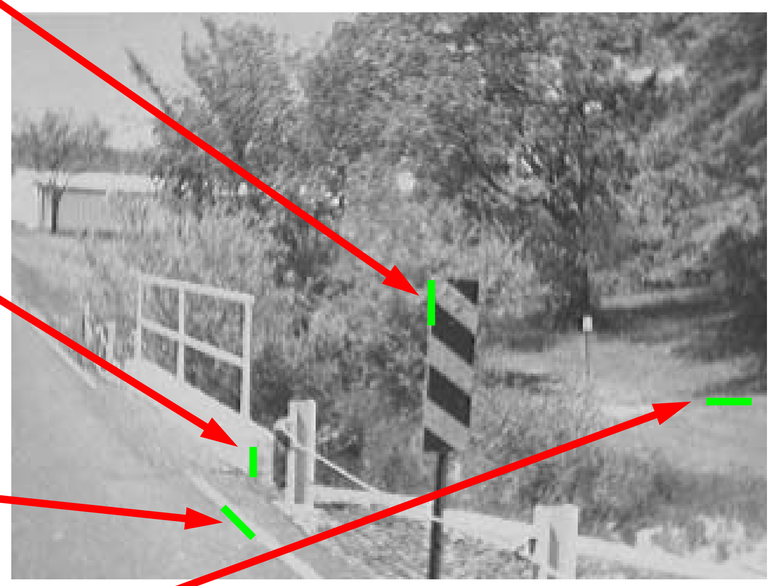
A similar definition (and approximation) holds for $\frac{\partial f}{\partial y}$

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

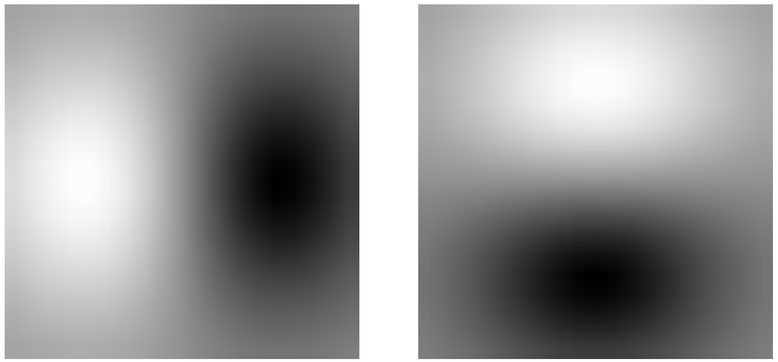
We can use **derivative of Gaussian** filters

— 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

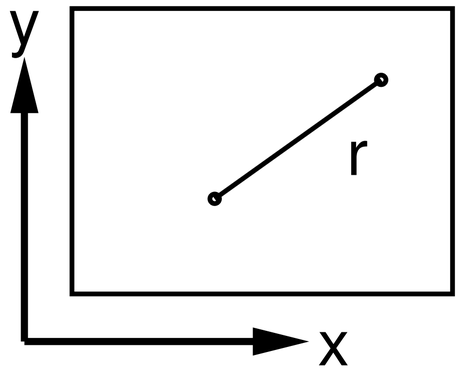
Let $I_x(X, Y)$ and $I_y(X, Y)$ be estimates of the partial derivatives in the x and y directions, respectively.

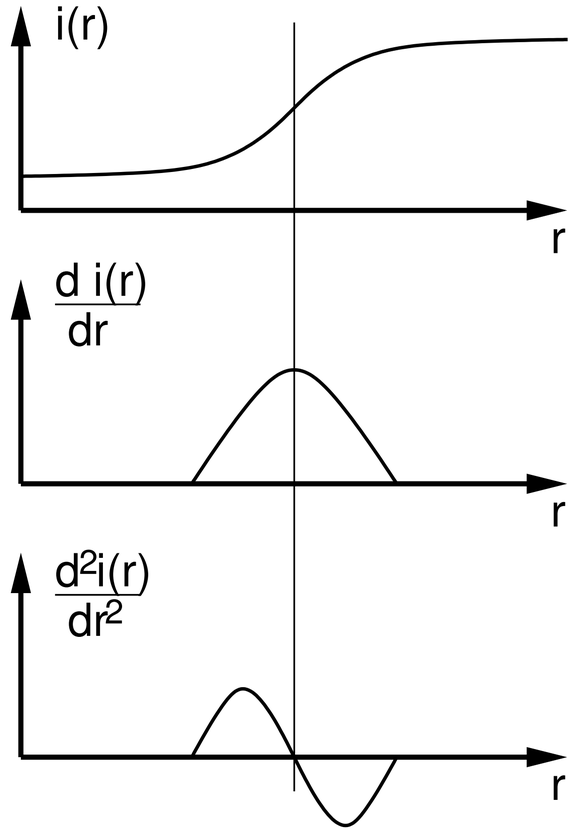
Call these estimates I_x and I_y (for short) The vector $[I_x, I_y]$ is the **gradient**

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

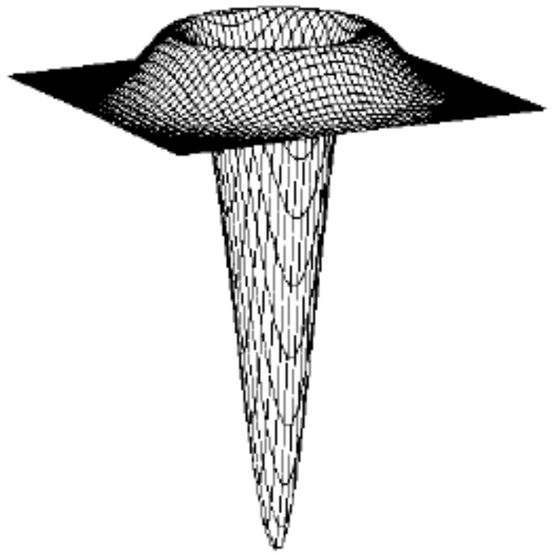
$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

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

Sample Question: Edges

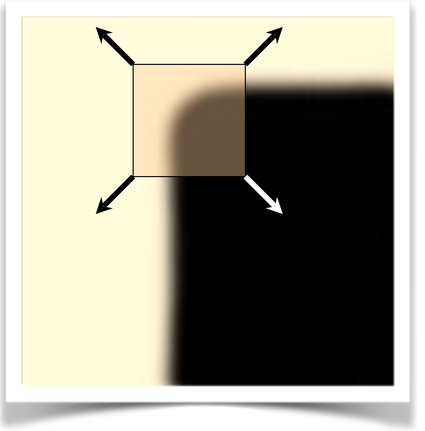
Why is non-maximum suppression applied in the Canny edge detector?

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**?



"**corner**":
significant change
in all directions

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.

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Autocorrelation

Szeliski, Figure 4.5

Corner Detection

Edge detectors perform poorly at corners

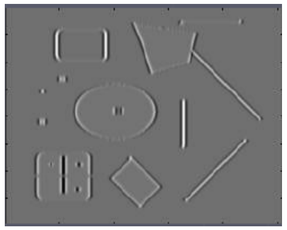
Observations:

— The gradient is ill defined exactly at a corner

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

Harris Corner Detection

$$I_x = \frac{\partial I}{\partial x} \qquad I_y = \frac{\partial I}{\partial y}$$



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

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

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

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

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

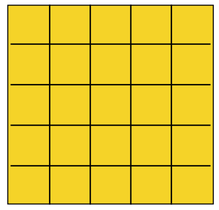
Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

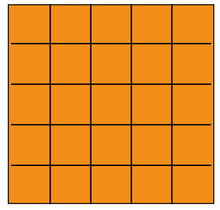
Gradient with respect to x, times gradient with respect to y

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

$$\sum_{p \in P} I_x I_y = \text{sum}(\; I_x = \frac{\partial I}{\partial x} \; .* \; I_y = \frac{\partial I}{\partial y} \;)$$



array of x gradients



array of y gradients

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

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

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

$$\mathbf{C} = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix}$$

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

$$\mathbf{C} = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix} => \lambda_1 = 1.4384; \lambda_2 = 5.5616$$

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

$$\mathbf{C} = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix} => \lambda_1 = 1.4384; \lambda_2 = 5.5616$$

$$\det(\mathbf{C}) - 0.04\text{trace}^2(\mathbf{C}) = 6.04$$

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

$$\mathbf{C} = \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix} => \lambda_1 = 3; \lambda_2 = 0$$

$$\det(\mathbf{C}) - 0.04\mathrm{trace}^2(\mathbf{C}) = -0.36$$

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

$$\mathbf{C} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} => \lambda_1 = 3; \lambda_2 = 2$$

$$\det(\mathbf{C}) - 0.04\text{trace}^2(\mathbf{C}) = 5$$

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

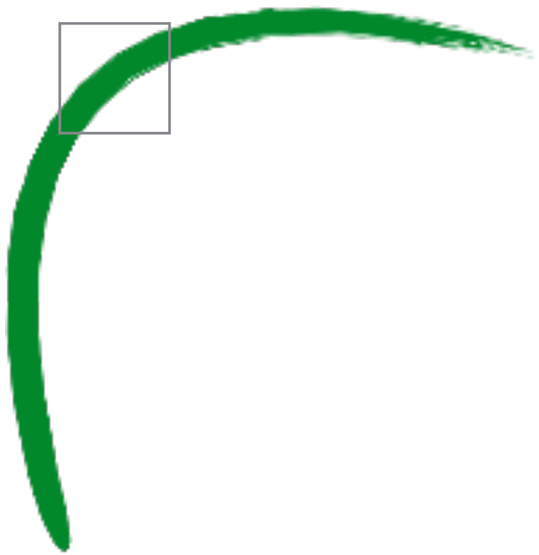
0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Properties: NOT Invariant to Scale Changes

edge!



corner!

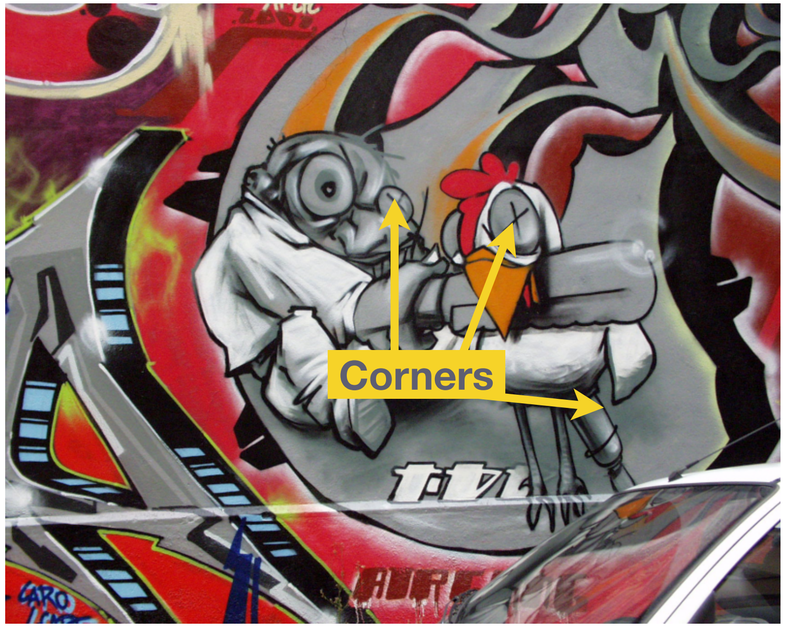
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Blobs features

Blobs are circular regions in the image

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

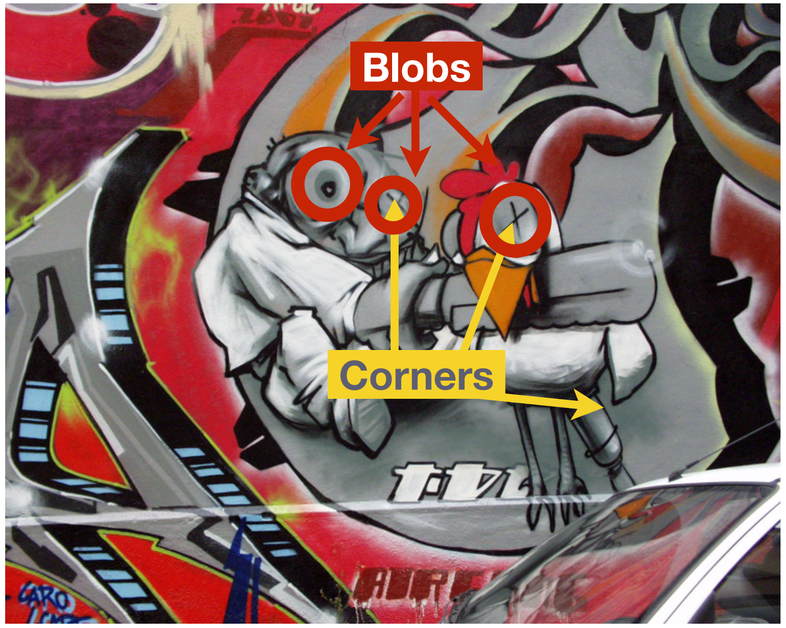
Blobs features



Blobs are circular regions in the image

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

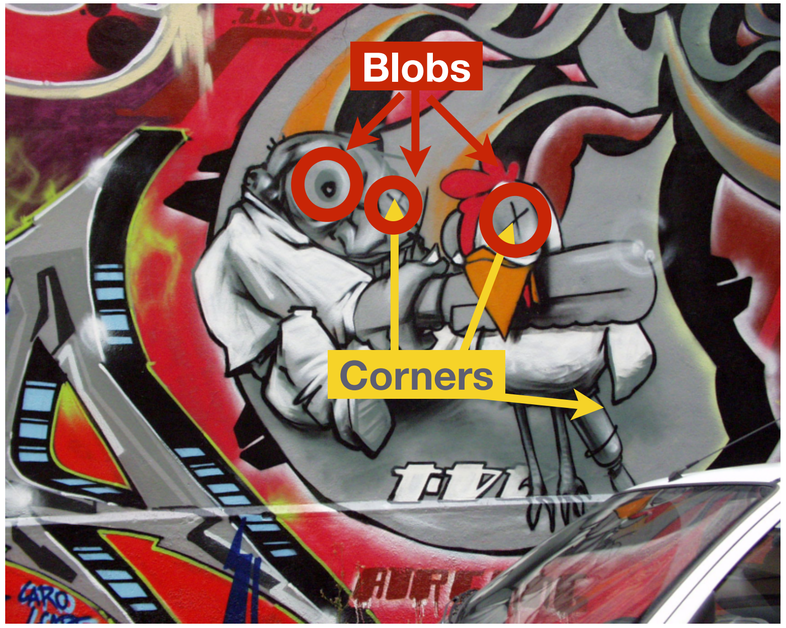
Blobs features



Blobs are circular regions in the image

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

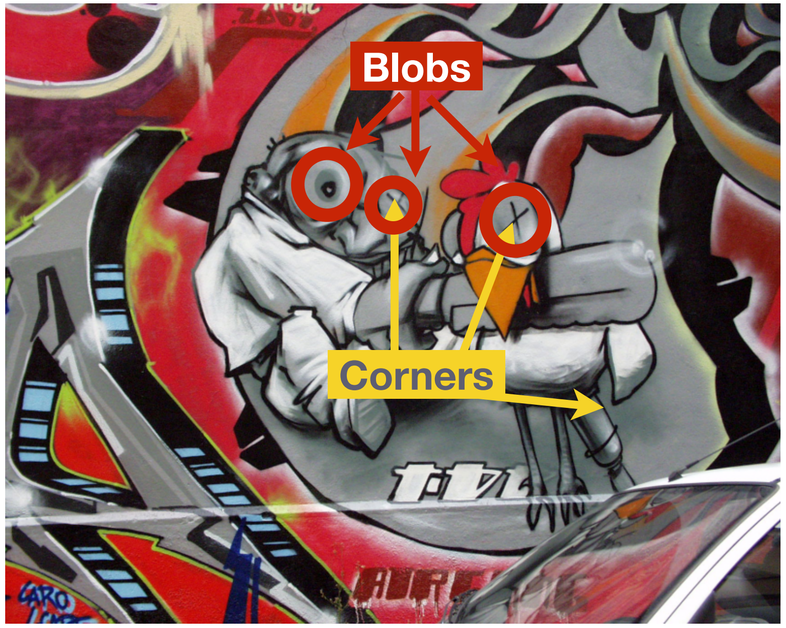
Blobs features



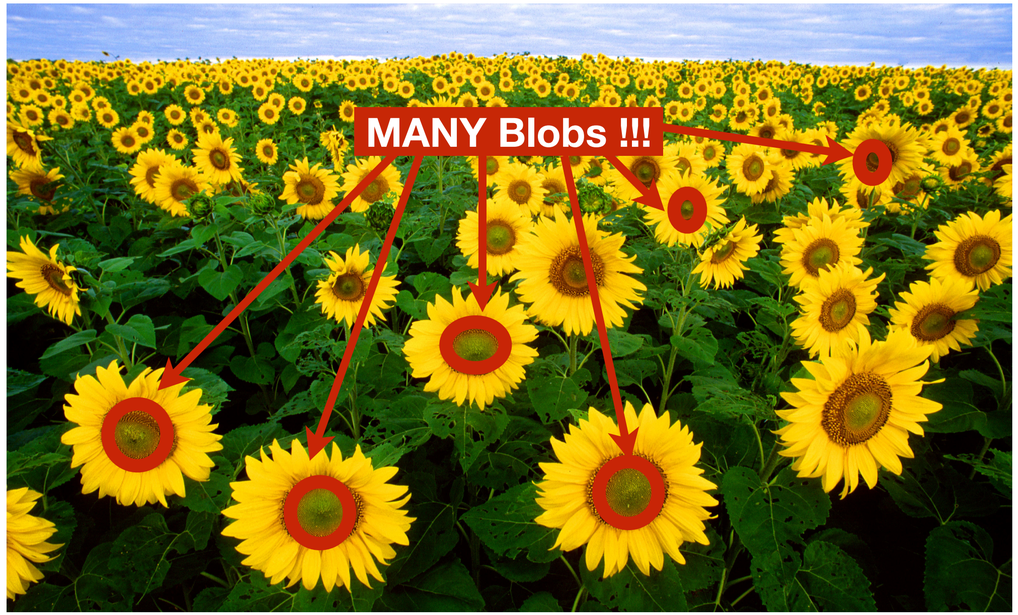
Blobs are circular regions in the image

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Blobs features



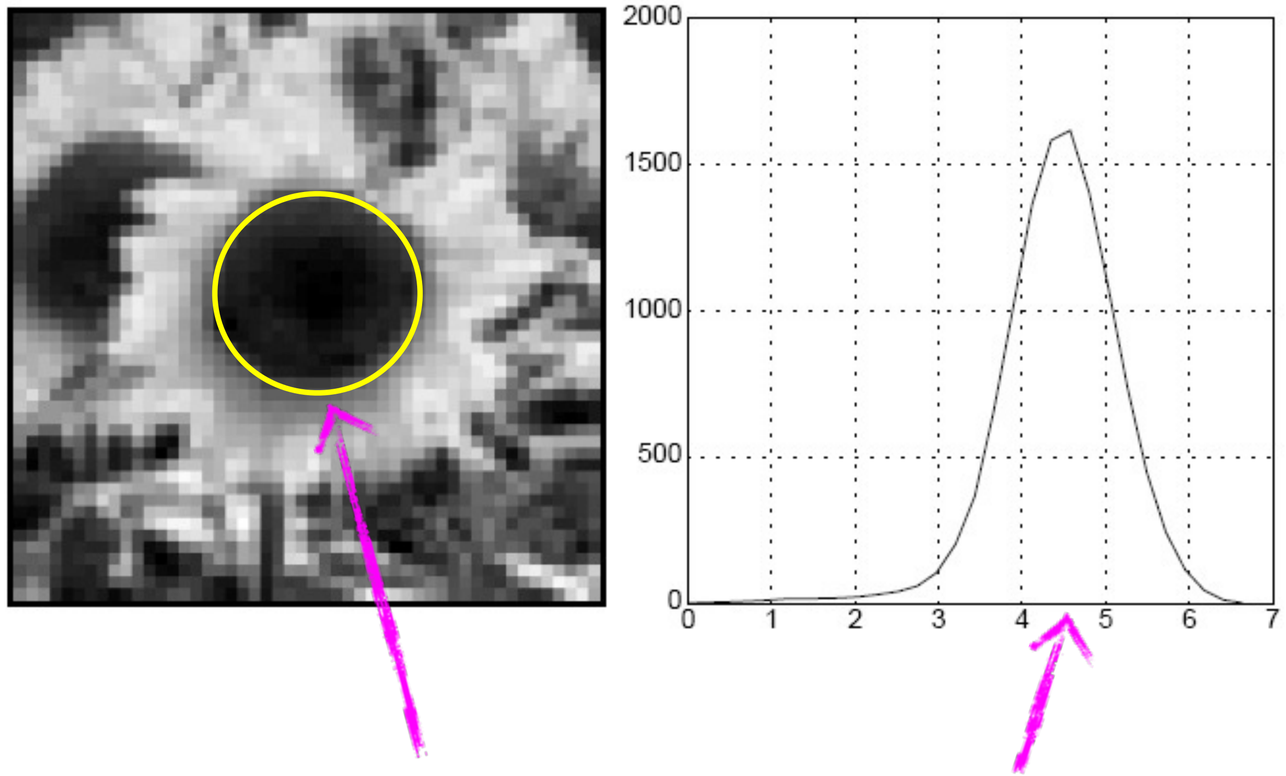
Blobs are circular regions in the image



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Characteristic Scale

characteristic scale - the scale that produces peak filter response



characteristic scale

we need to search over characteristic scales

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Implementation

For each level of the Gaussian pyramid

compute feature response (e.g. Harris, Laplacian)

For each level of the Gaussian pyramid

    if local maximum and cross-scale

    save scale and location of feature (x, y, s)

Implementation

```
For each level of the Gaussian pyramid

compute feature response (e.g. Harris, Laplacian)

For each level of the Gaussian pyramid

    if local maximum and cross-scale

    save scale and location of feature (x, y, s)
```

(for Harris you can also save orientation θ based on top eigenvector)

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

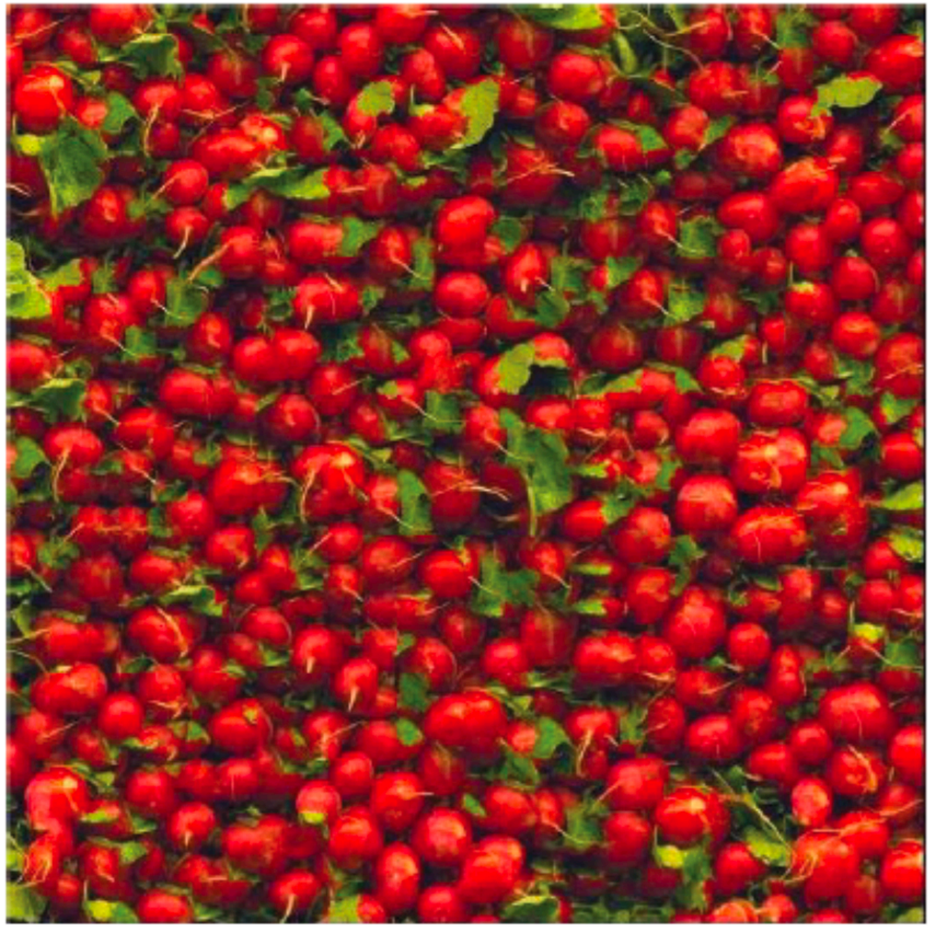
— We synthesize regions of texture that fit in and look convincing

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**

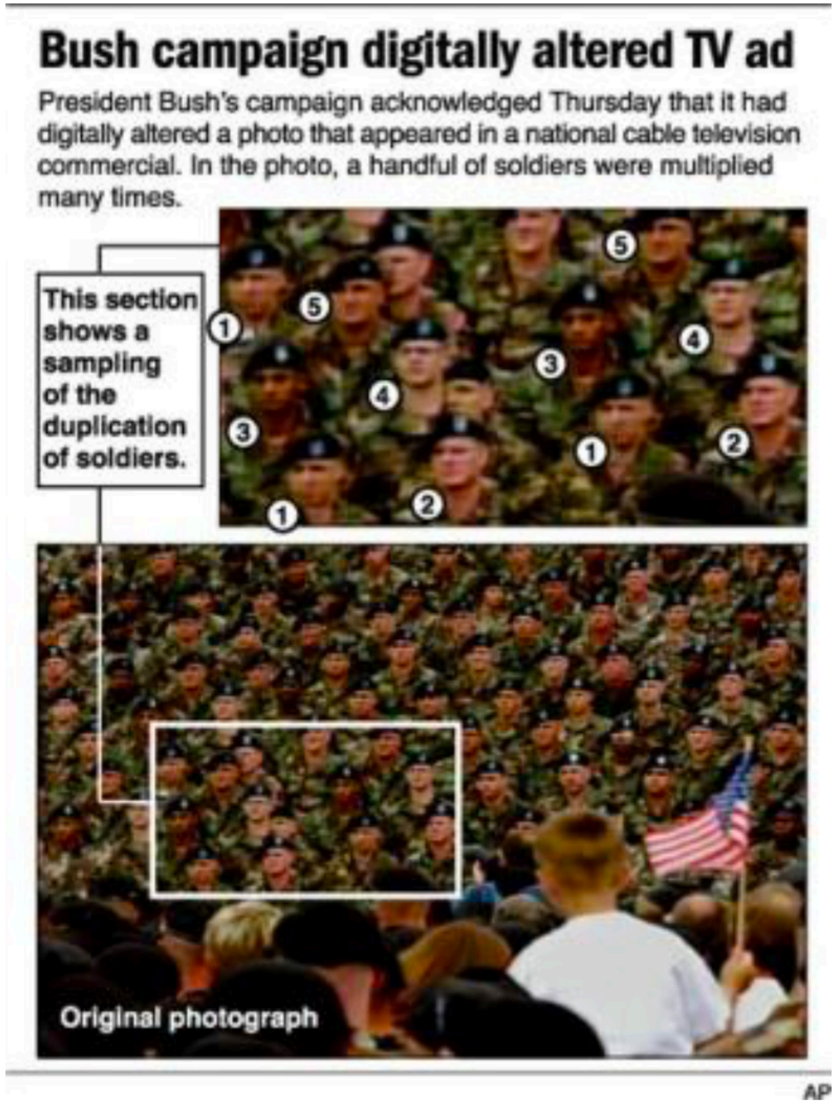
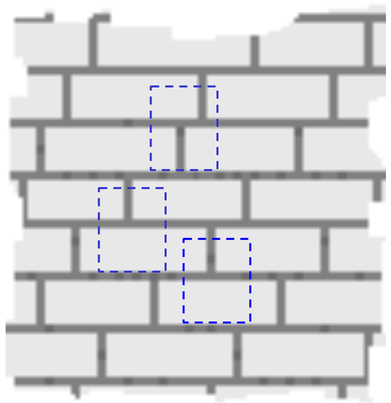


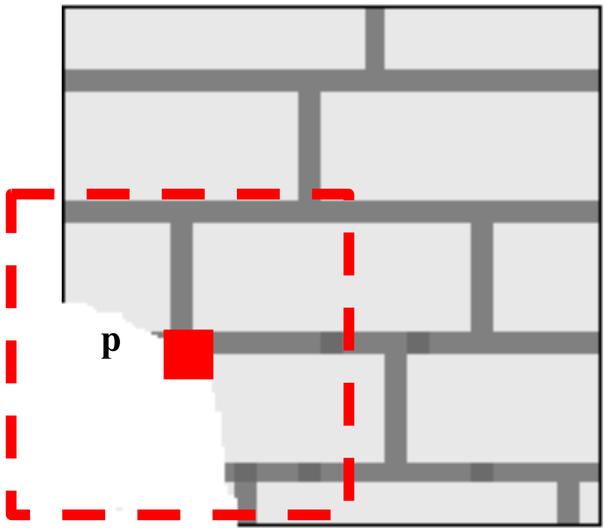
Photo Credit: Associated Pres

Efros and Leung: Synthesizing One Pixel



Infinite sample image

SAMPLE

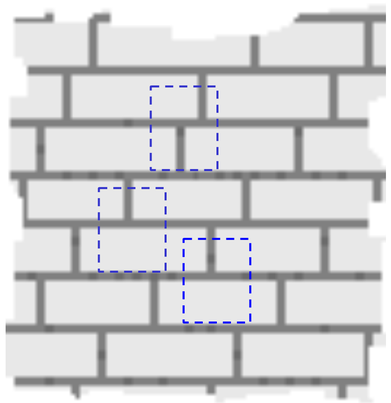


— What is **conditional** probability distribution of *p*, given the neighbourhood window?

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

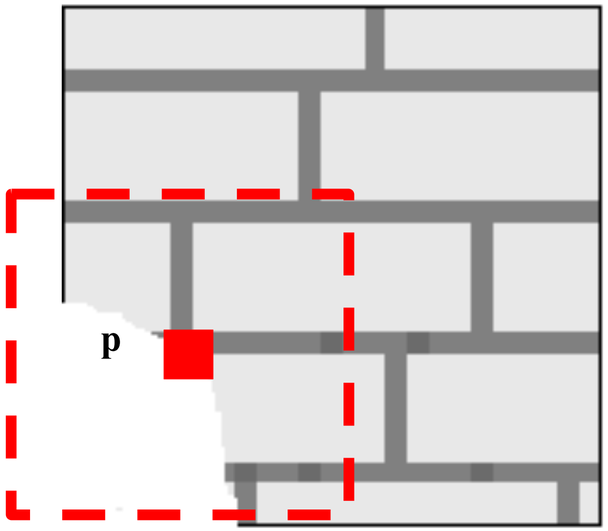
— To **synthesize** *p*, pick one match at random

Efros and Leung: Synthesizing One Pixel



Infinite sample image

SAMPLE



— 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

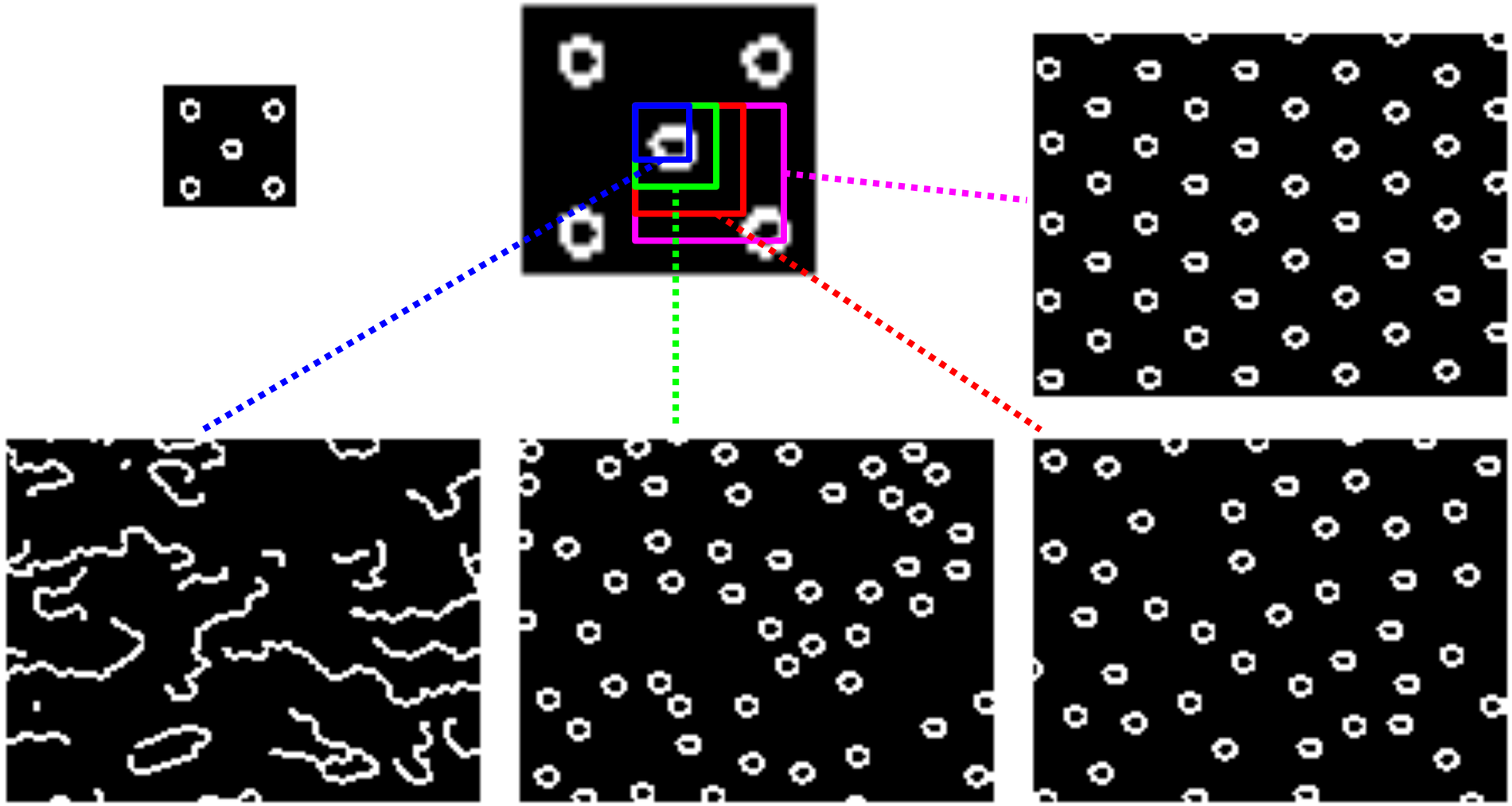
— In the case of hole-filling, start from the edges of the hole

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