Computer Vision

Kumar Bipin

BE, MS, PhD (MMTU, IISc, IIIT-Hyderabad) (Robotics Control and Computer Vision)

Motorola, STMicroeletronics, Tata Elxsi (Technical Manager)

Image Formation

Image: Projection of 3D scene onto 2D plane. We need to understand the geometric and photometric relation between the scene and its image.

Topics:

(1) Pinhole and Perspective Projection

(2) Image formation using Lenses

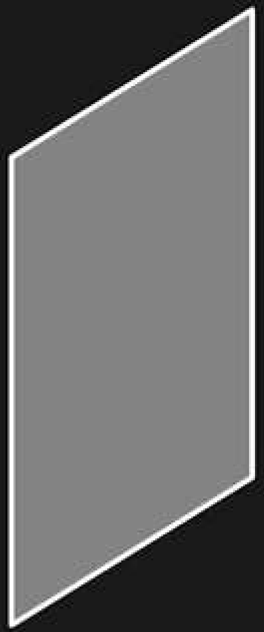
(3) Lens Related Issues

(4) Wide Angle Cameras

(5) Animal Eyes

© 2020 Shree K. Nayar

Image Formation



Screen

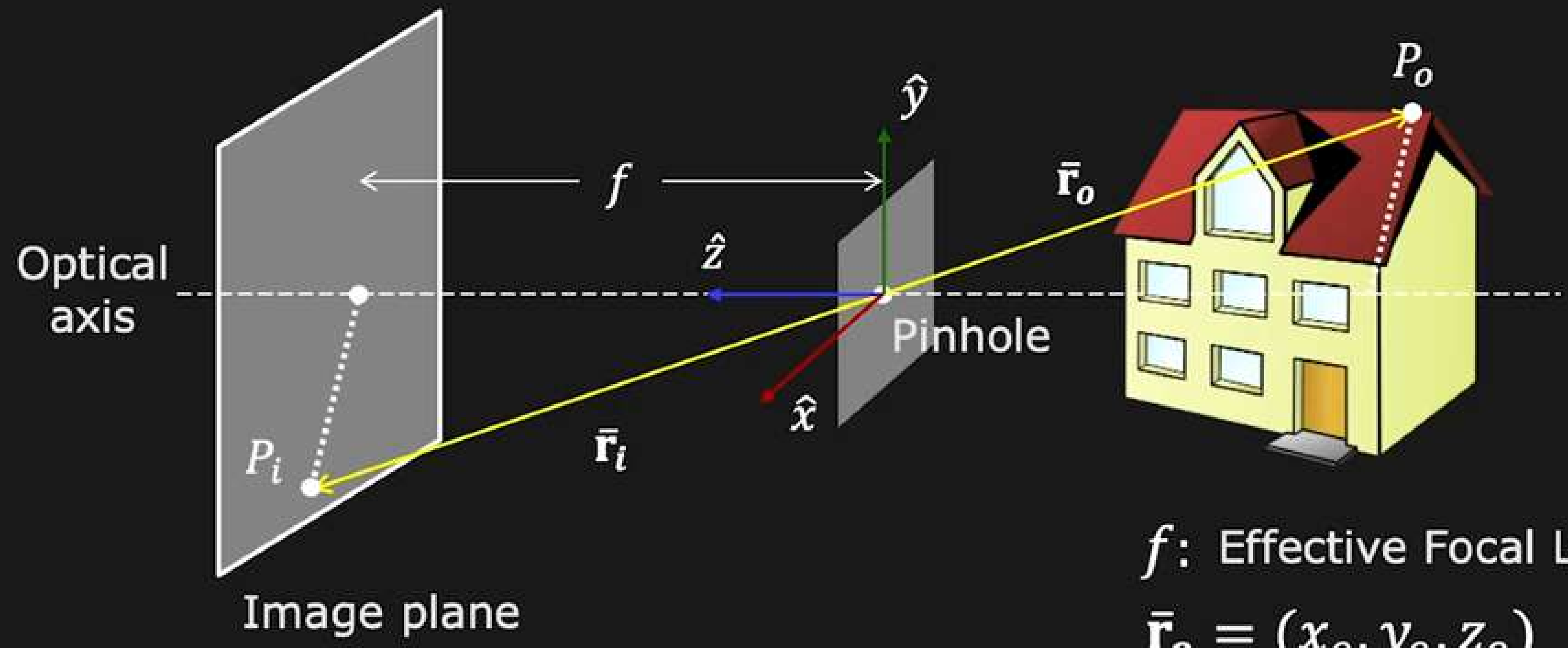
Scene

Is an image being formed on the screen?

Yes! But not a "clear" one.

© 2020 Shree K. Nayar

Perspective Imaging with Pinhole



f: Effective Focal Length

$\bar{\mathbf{r}}_o = (x_o, y_o, z_o)$

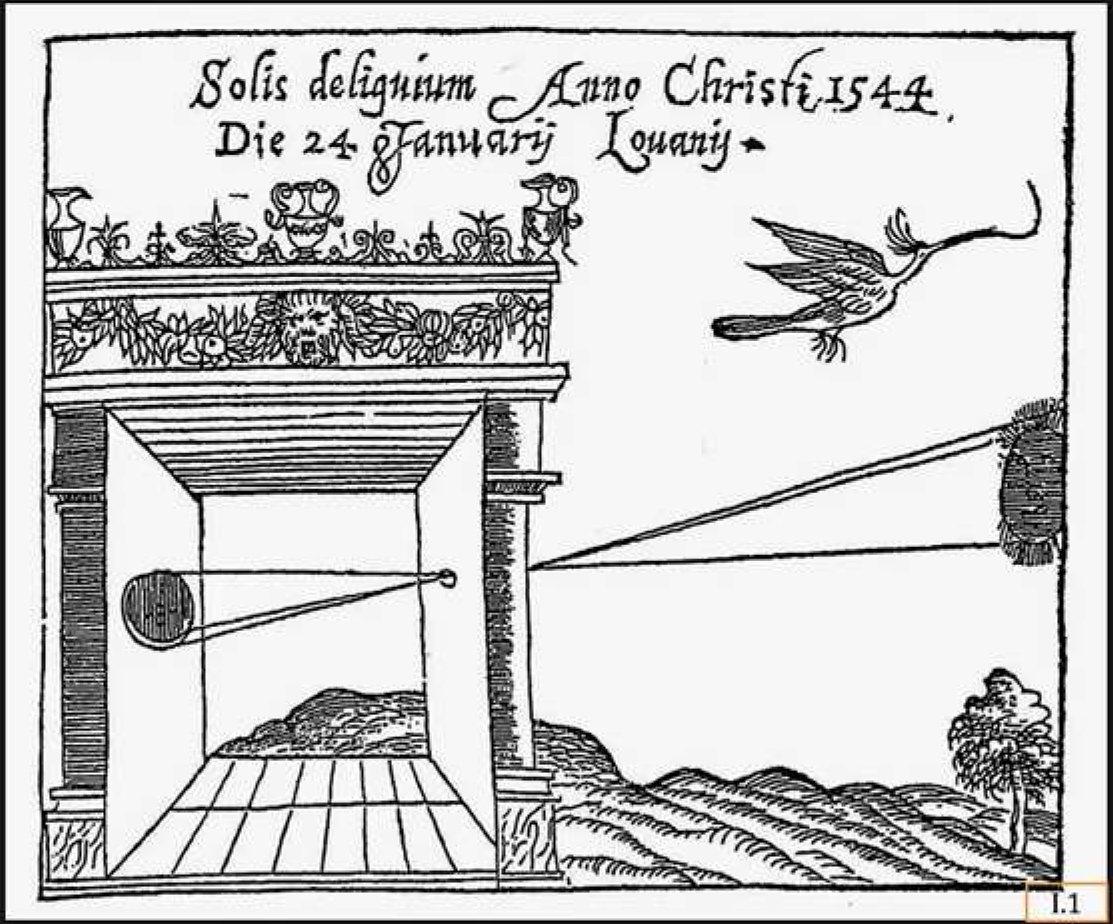
$\bar{\mathbf{r}}_i = (x_i, y_i, f)$

Using similar triangles:

$$\boxed{\frac{\bar{\mathbf{r}}_i}{f} = \frac{\bar{\mathbf{r}}_o}{z_o}} \rightarrow \boxed{\frac{x_i}{f} = \frac{x_o}{z_o}, \quad \frac{y_i}{f} = \frac{y_o}{z_o}}$$

© 2020 Shree K. Nayar

Camera Obscura



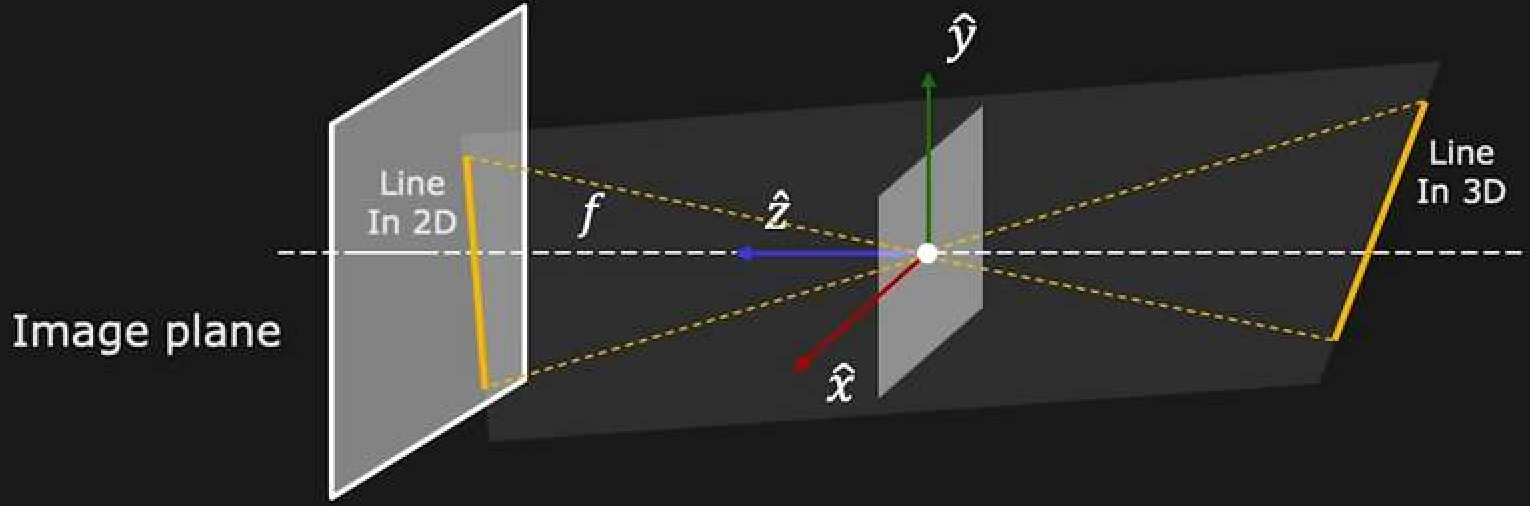
"Dark Chamber"

Pinhole Eye of *Nautilus pompilius*

I.2

I.3

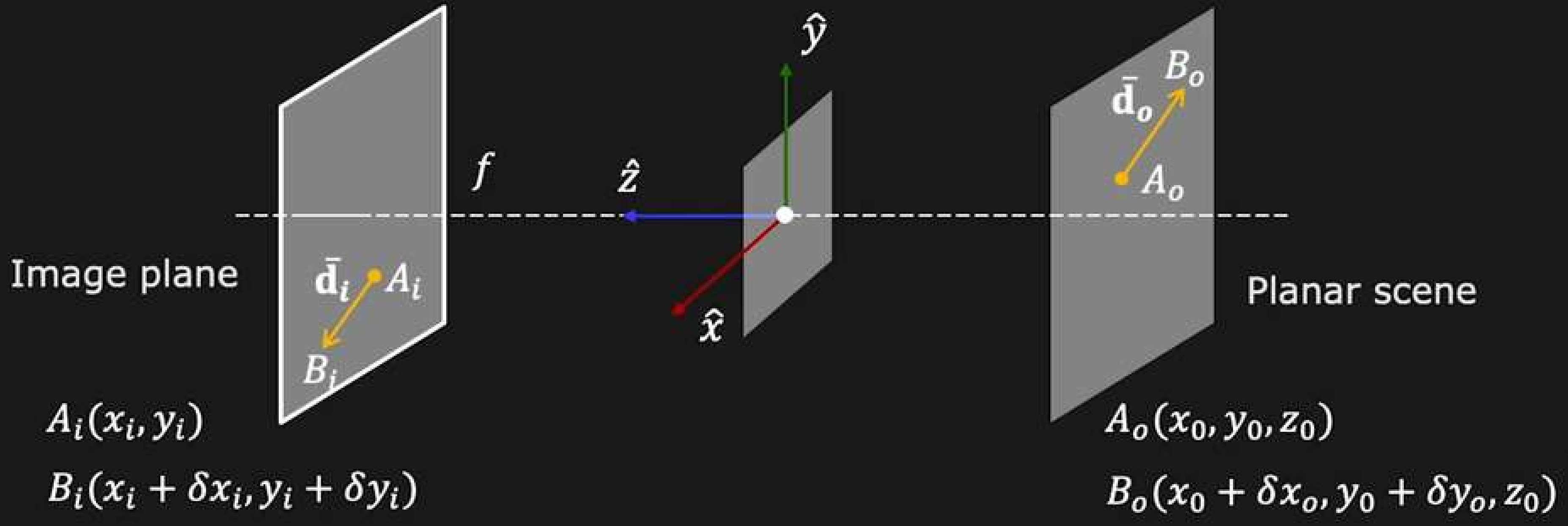
Perspective Projection of a Line



Straight line in scene remains straight in image

© 2020 Shree K. Nayar

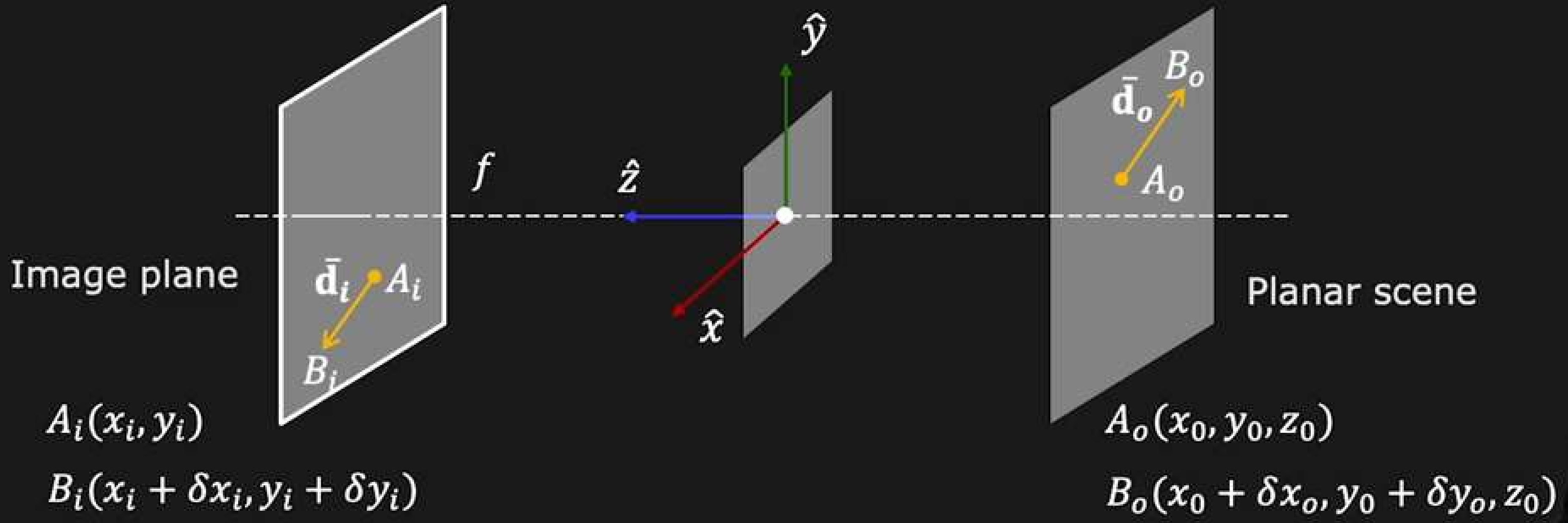
Image Magnification



Magnification: $|m| = \dfrac{\|\bar{\mathbf{d}}_i\|}{\|\bar{\mathbf{d}}_o\|} = \sqrt{\delta x_i^{\,2} + \delta y_i^{\,2}} \Big/ \sqrt{\delta x_o^{\,2} + \delta y_o^{\,2}}$ [redacted]

© 2020 Shree K. Nayar

Image Magnification



From Perspective Projection:

$$\frac{x_i}{f} = \frac{x_o}{z_o} \quad \text{and} \quad \frac{y_i}{f} = \frac{y_o}{z_o} \quad \text{..........} \quad \text{(A)}$$

$$\frac{x_i + \delta x_i}{f} = \frac{x_o + \delta x_o}{z_o} \quad \text{and} \quad \frac{y_i + \delta y_i}{f} = \frac{y_o + \delta y_o}{z_o} \quad \text{..........} \quad \text{(B)}$$

© 2020 Shree K. Nayar

Image Magnification

From (A) and (B) we get:

$$\frac{\delta x_i}{f} = \frac{\delta x_o}{z_o} \quad \text{and} \quad \frac{\delta y_i}{f} = \frac{\delta y_o}{z_o}$$

Magnification:

$$|m| = \frac{\|\bar{\mathbf{d}}_i\|}{\|\bar{\mathbf{d}}_o\|} = \sqrt{{\delta x_i}^2 + {\delta y_i}^2} \Big/ \sqrt{{\delta x_o}^2 + {\delta y_o}^2} = \left|\frac{f}{z_o}\right|$$

$$\boxed{m = \frac{f}{z_o}}$$

m is negative when image is inverted

© 2020 Shree K. Nayar

Image Magnification

$$\boxed{m = \frac{f}{z_o}}$$

Image size inversely proportional to depth

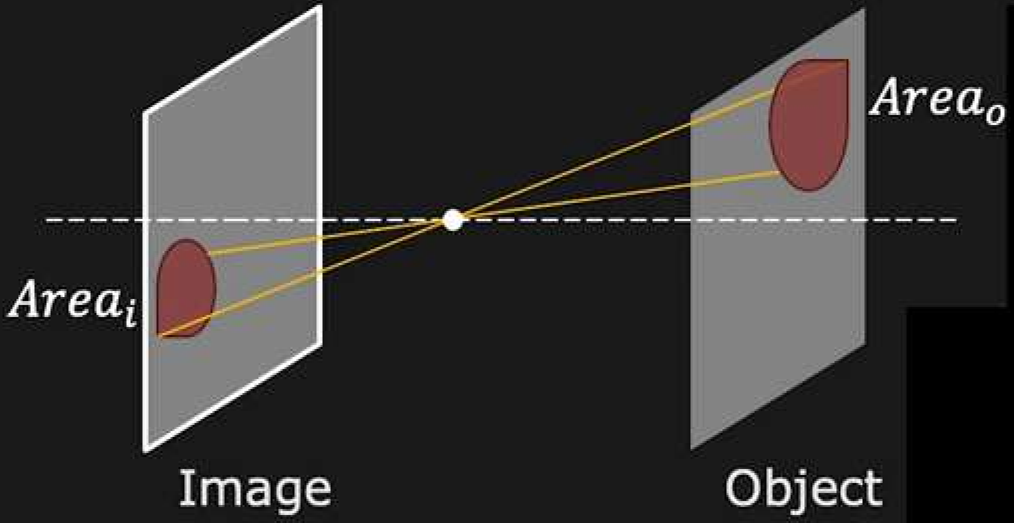
© 2020 Shree K. Nayar

Image Magnification

Remarks:

- m can be assumed to be constant if the range of scene depth Δz is much smaller than the average scene depth $\tilde{z}$

- $$\frac{Area_i}{Area_o} = m^2$$



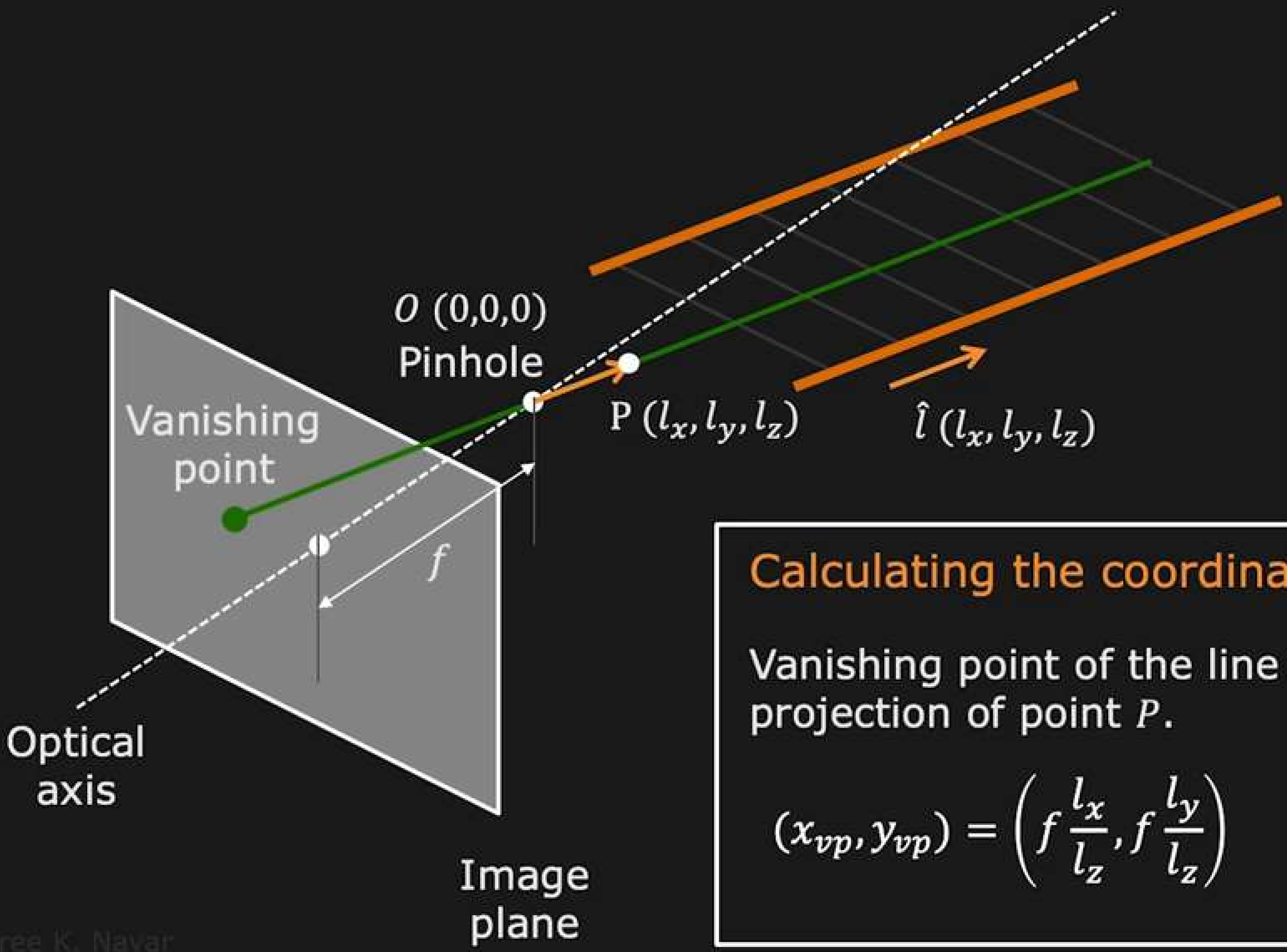
© 2020 Shree K. Nayar

Vanishing Point

Location of Vanishing Point depends on the orientation of parallel straight lines.

© 2020 Shree K. Nayar

Finding Vanishing Point



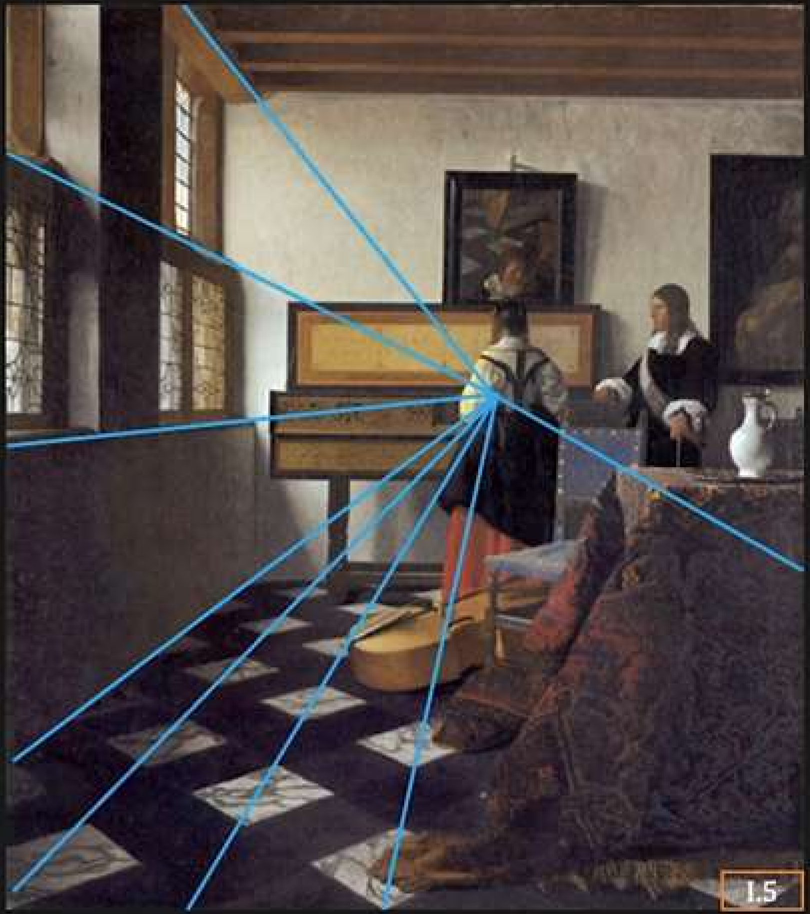
Calculating the coordinates:

Vanishing point of the line is the projection of point P.

$$(x_{vp}, y_{vp}) = \left(f\,\frac{l_x}{l_z}, f\,\frac{l_y}{l_z} \right)$$

© 2020 Shree K. Nayar

Use of Vanishing Point in Art

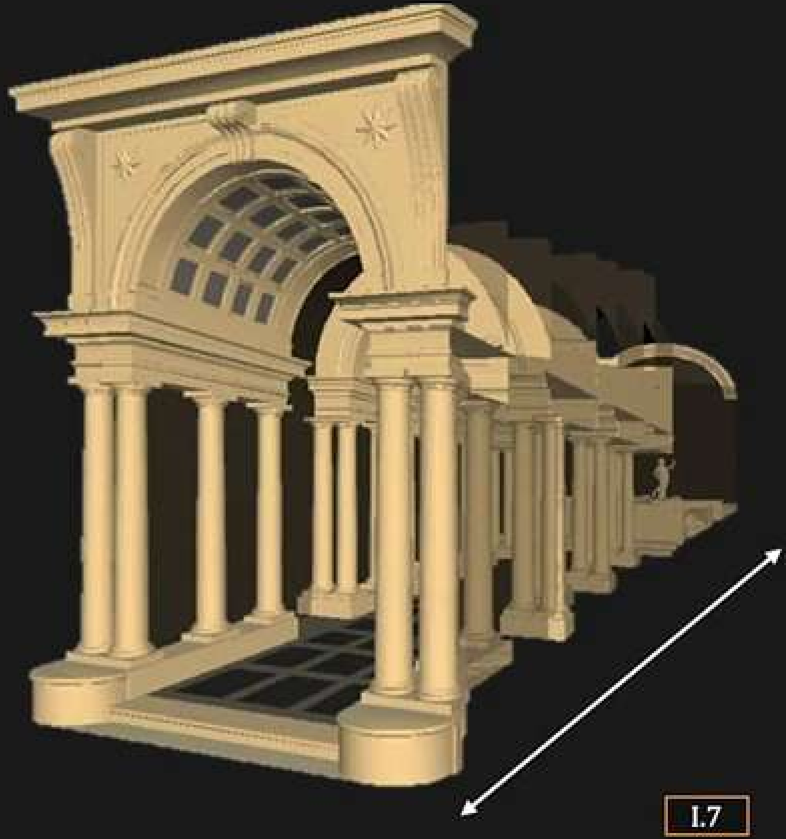


The Music Lesson, Johannes Vermeer, c. 1662-1664

2020 Shree K. Nayar

False Perspective

Depth appears to be ~155 feet



Depth is actually ~30 feet

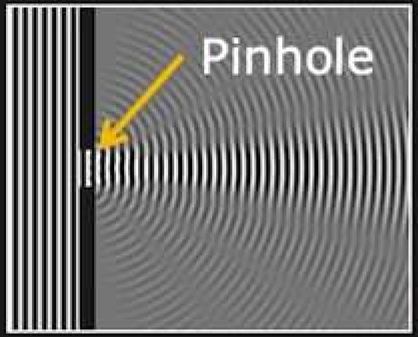
Galleria Spada, Francesco Borromini, 1652

2020 Shree K. Nayar

What is the Ideal Pinhole Size?

2 mm | 1 mm | 0.6 mm | 0.35 mm | 0.15 mm | 0.07 mm

The pinhole must be tiny,
but if it’s too tiny it will cause diffraction.



Diffraction

Ideal pinhole diameter: $\boxed{d \approx 2\sqrt{f\lambda}}$

f: effective focal length
λ: wavelength

© 2020 Shree K. Nayar

What about Exposure Time?

Pinholes pass less light and hence require long exposures to capture bright images.

$f = 73\ mm,\ d = 0.2\ mm,$
Exposure, $T = 12\ s$

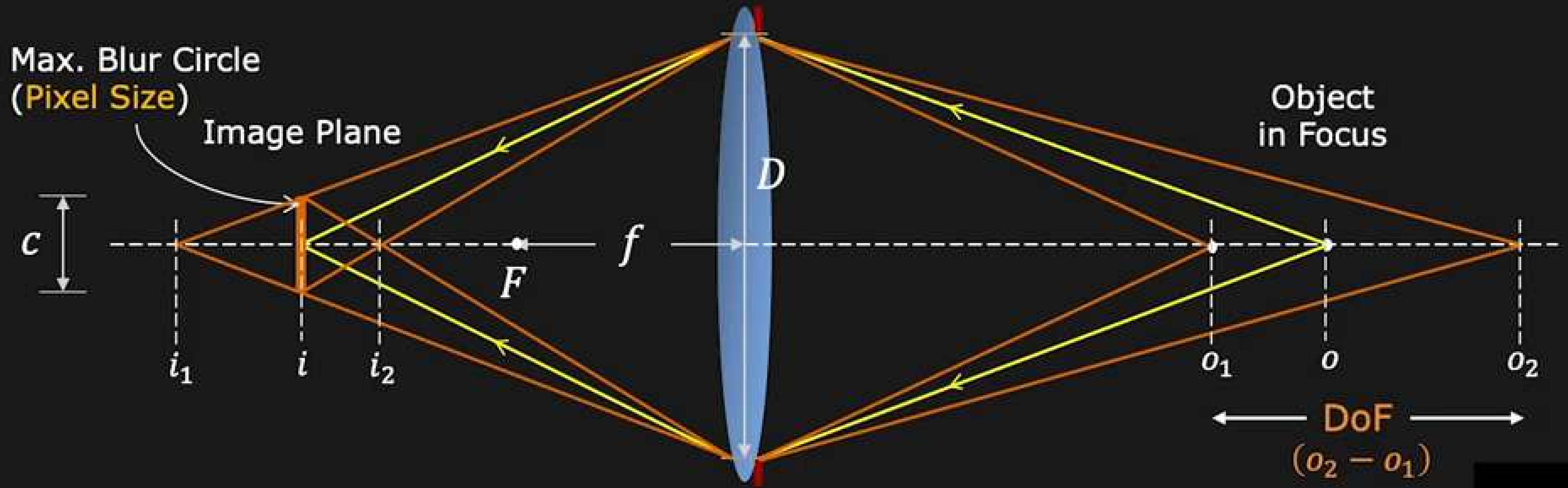
© 2020 Shree K. Nayar

Depth of Field (DoF)

Range of object distances over which the image is "sufficiently well" focused, i.e., range over which blur b is less than pixel size.

© 2020 Shree K. Nayar

Depth of Field (DoF)



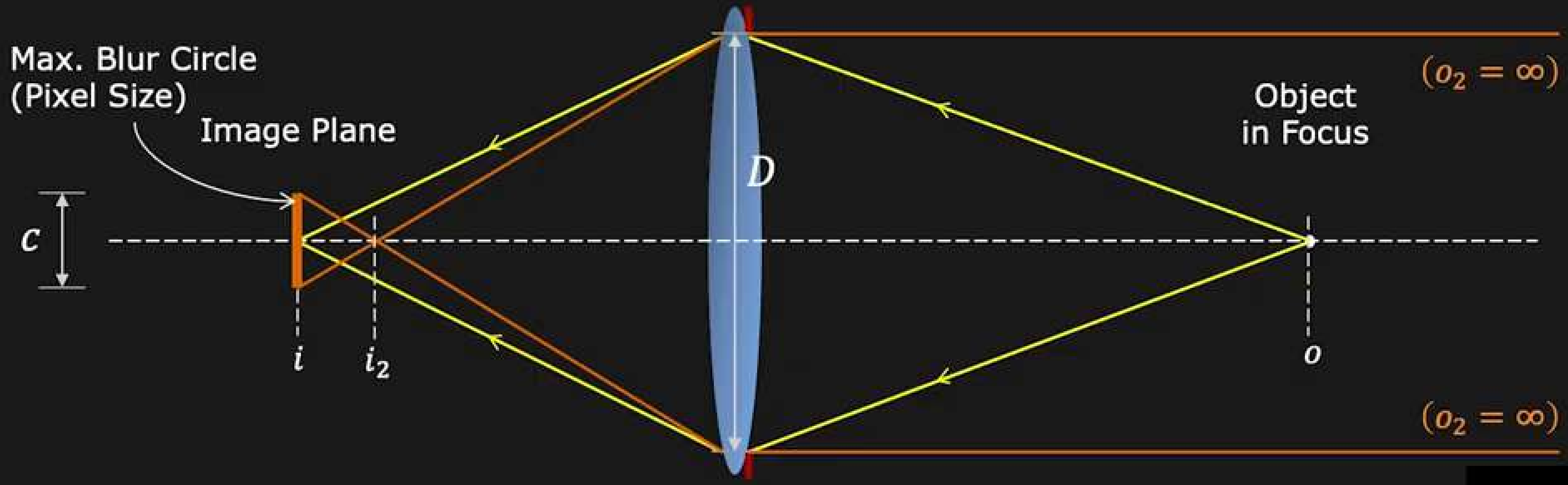
If o_1 and o_2 are the nearest and farthest distances respectively for which blur circle is maximum c, then:

$$c = \frac{f^2(o - o_1)}{N o_1 (o - f)} \qquad c = \frac{f^2(o_2 - o)}{N o_2 (o - f)}$$

Depth of Field: $\boxed{o_2 - o_1 = \dfrac{2 o f^2 c N (o - f)}{f^4 - c^2 N^2 (o - f)^2}}$

© 2020 Shree K. Nayar

Hyperfocal Distance

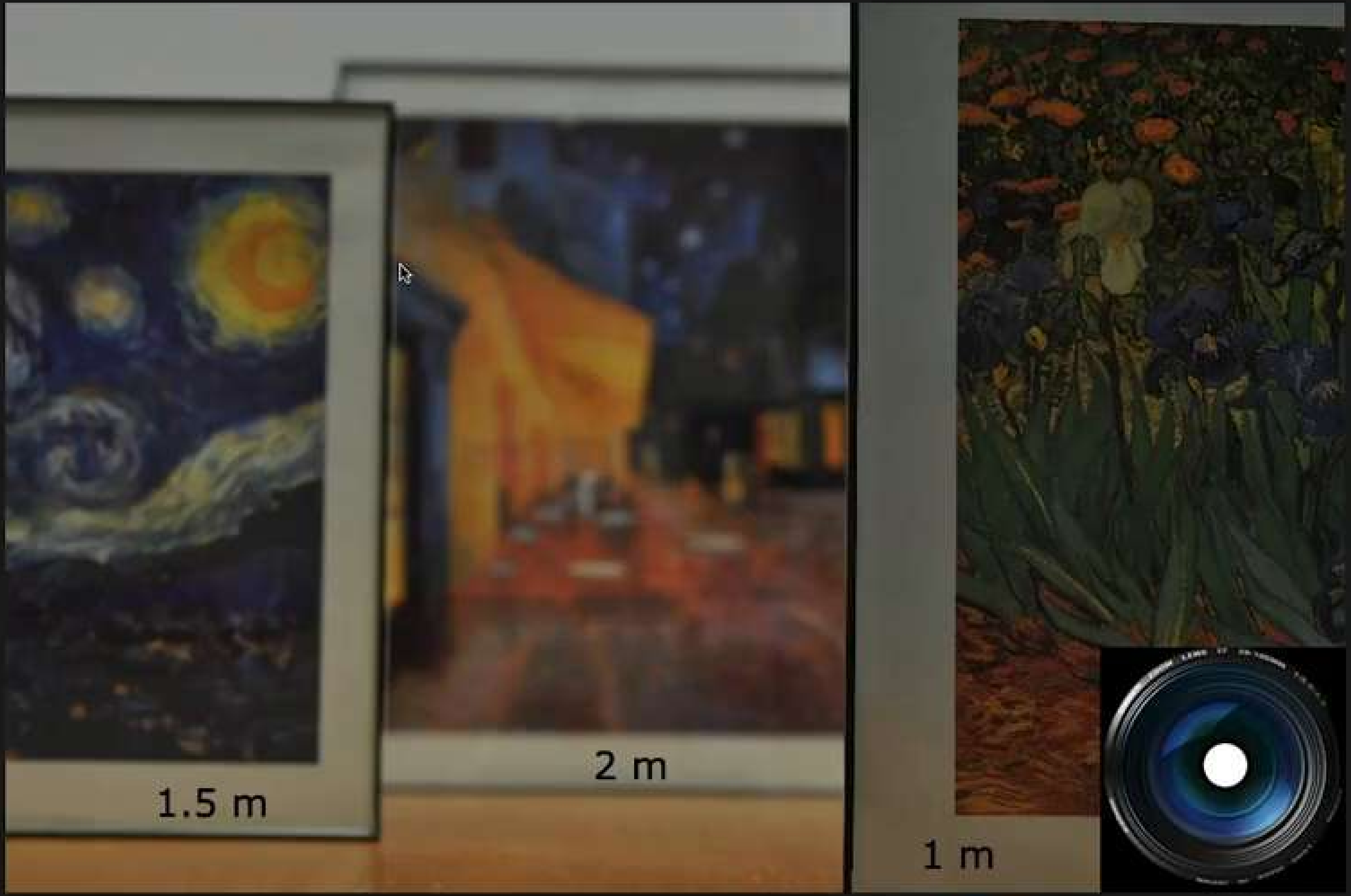


The closest distance $o = h$ the lens must be focused to keep objects at infinity $(o_2 = \infty)$ acceptably sharp (blur circle $\leq c$).

Hyperfocal Distance: $\boxed{h = \frac{f^2}{Nc} + f}$

© 2020 Shree K. Nayar

Aperture Size: DOF vs. Brightness



Focal Length 50 mm, Focus = 1 m, Aperture D = 12.5 mm, f-Number N [redacted]

© 2020 Shree K. Nayar

Aperture Size: DOF vs. Brightness

Large Aperture (Small f-Number)

- Bright Image or Short Exposure Time
- Shallow Depth of Field

Small Aperture (Large f-Number)

- Dark Image or Long Exposure Time
- Large Depth of Field

© 2020 Shree K. Nayar

Tissue Box Camera

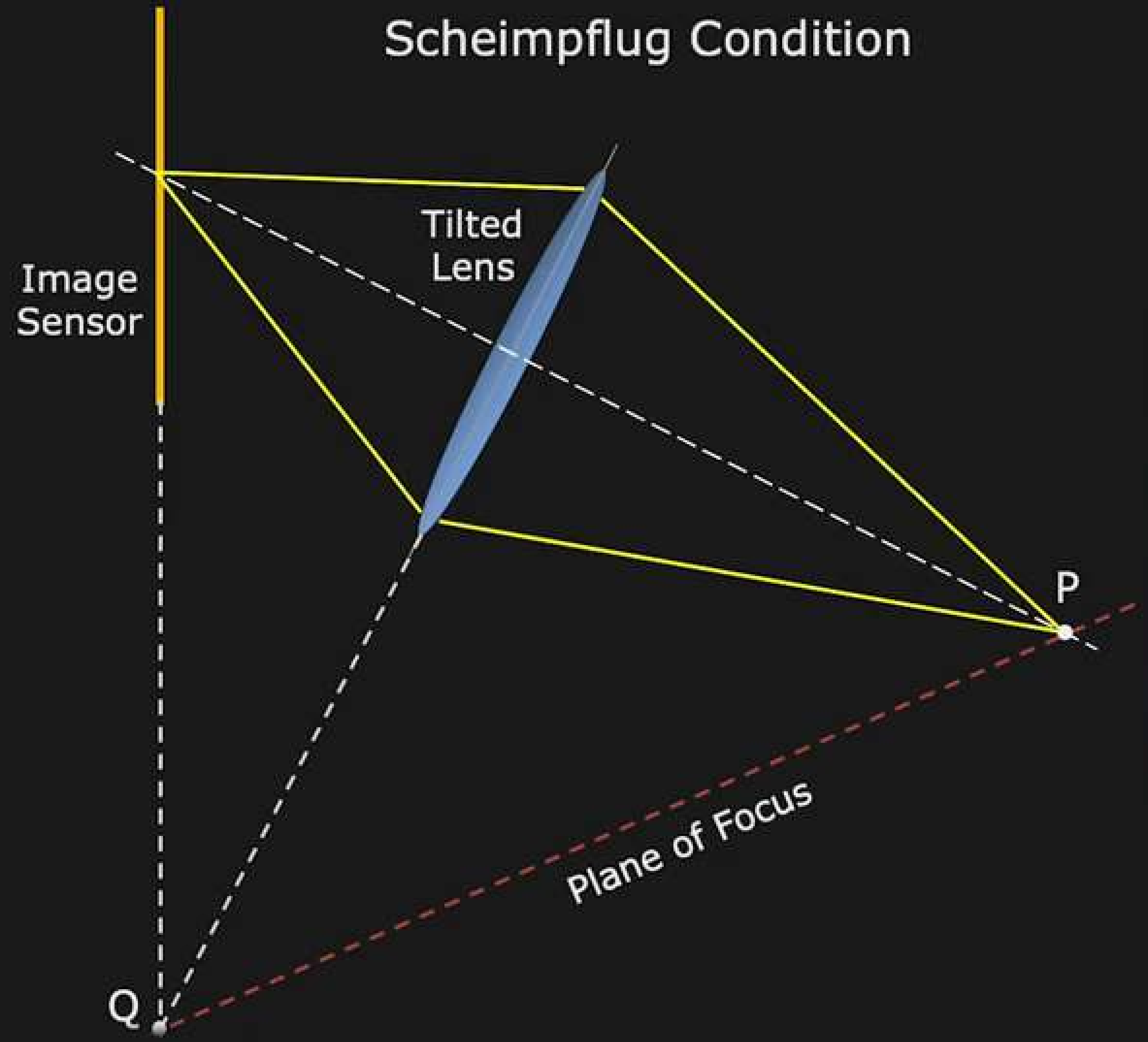
Camera

Image

© 2020 Shree K. Nayar

Tilting the Lens

Scheimpflug Condition

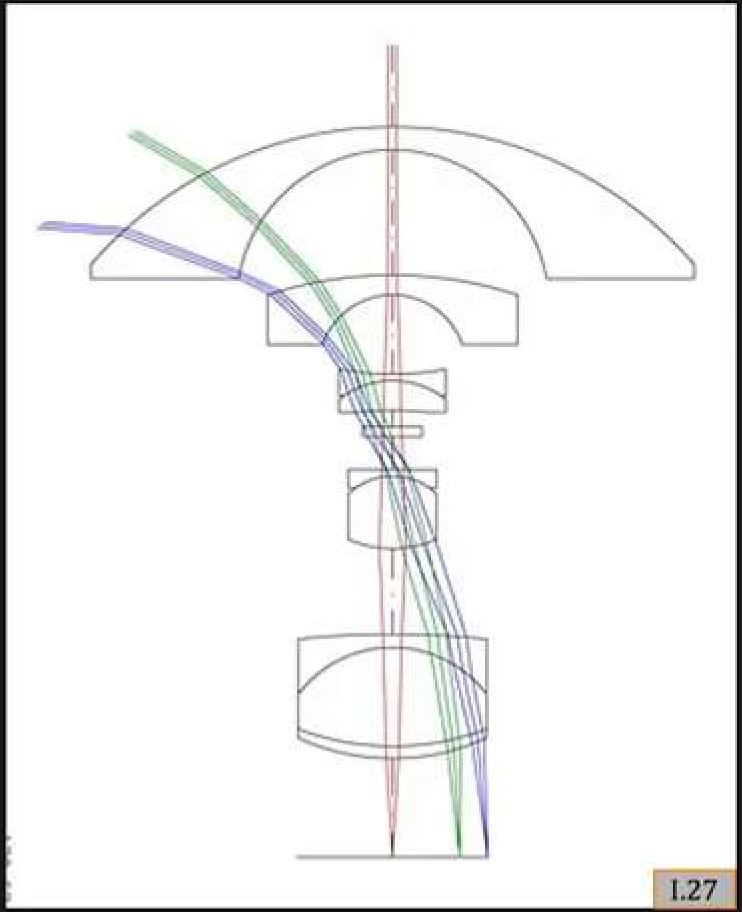


Tilt Camera

stenopeika.com

© 2020 Shree K. Nayar

Fisheye Lens Camera



170^o Fisheye Lens

[Miyamo[redacted]

© 2020 Shree K. Nayar

Fisheye Image

Fisheye Lens

Hemispherical Field of View

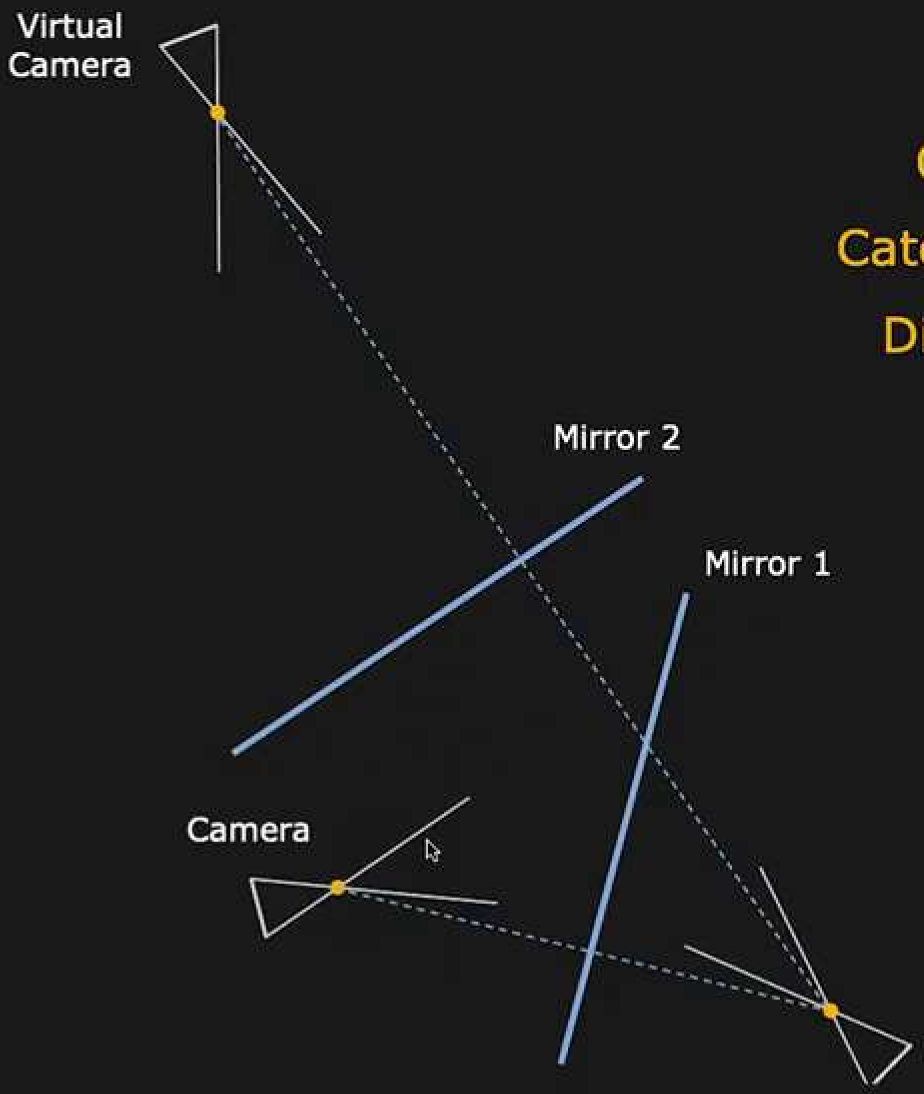
© 2020 Shree K. Nayar

Capturing the Complete Sphere

Ricoh Theta

© 2020 Shree K. Nayar

Planar Mirrors and Reflected Cameras



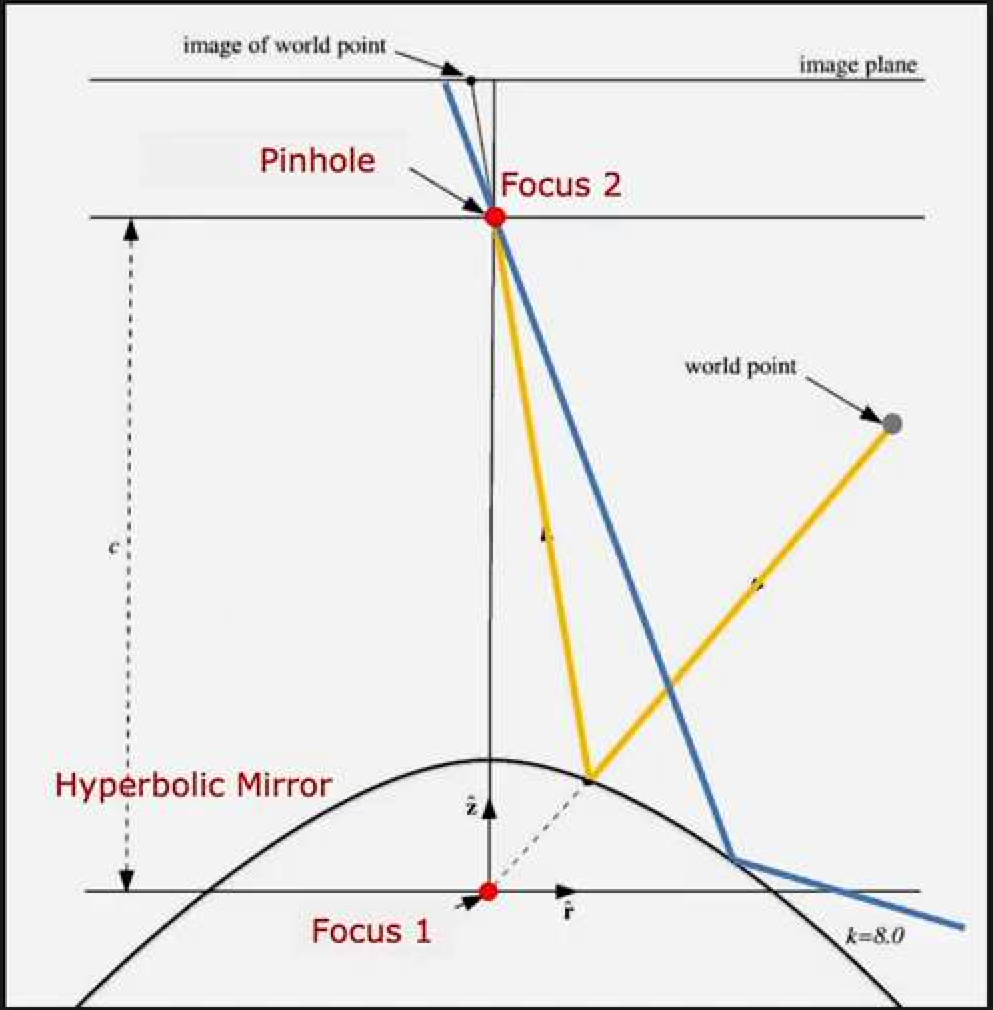
Catadioptrics =

Catoptrics (Mirrors) +

Dioptrics (Lenses)

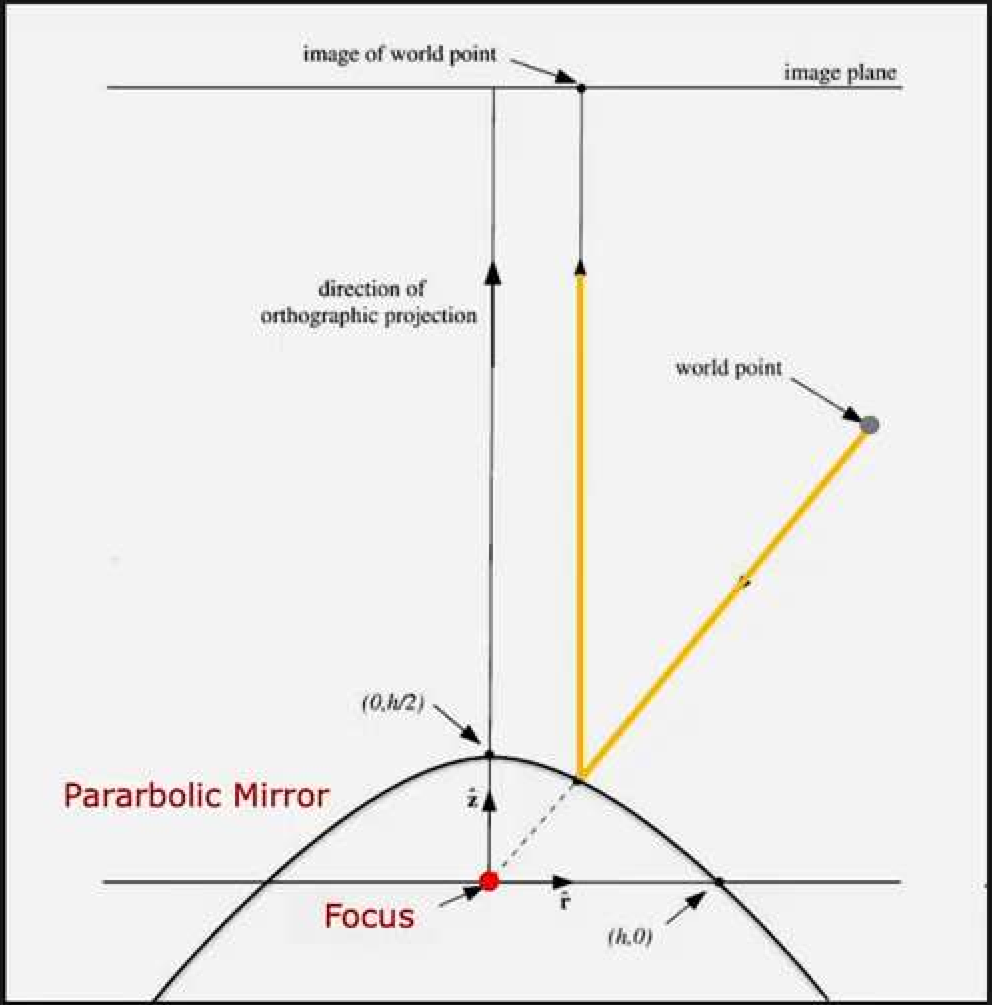
© 2020 Shree K. Nayar

Hyperbolic Mirror Camera



© 2020 Shree K. Nayar

Parabolic Mirror Camera



© 2020 Shree K. Nayar

Parabolic Mirror Image

© 2020 Shree K. Nayar

Sony “Bloggie”

Kogeto “Dot” for iPhone

© 2020 Shree K. Nayar

Concave Mirrors and Telescopes

James Webb Space Telescope

© 2020 Shree K. Nayar

Like Scallops?

© 2020 Shree K. Nayar

Scallop Eyes

Telescopic Eyes with Parabolic Mirrors

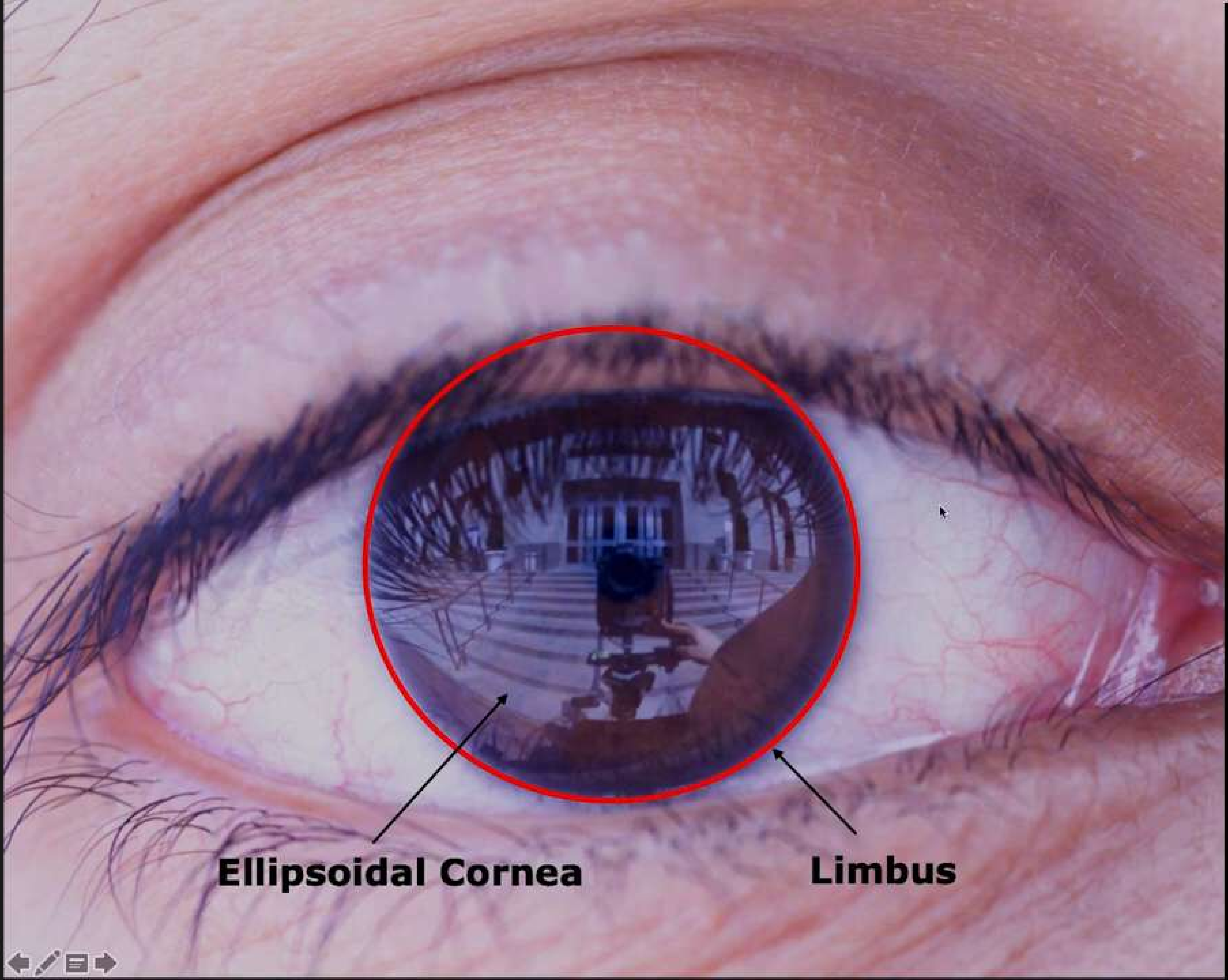
© 2020 Shree K. Nayar

The World in an Eye

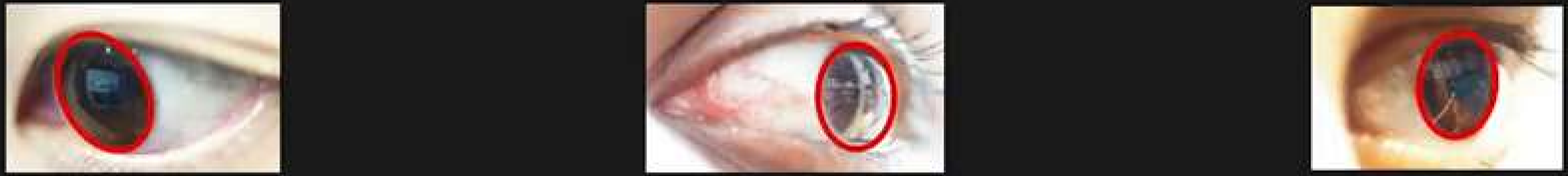
© 2020 Shree K. Nayar

The World in an Eye

© 2020 Shree K. Nayar


Ellipsoidal Cornea
Limbus

Eye Images

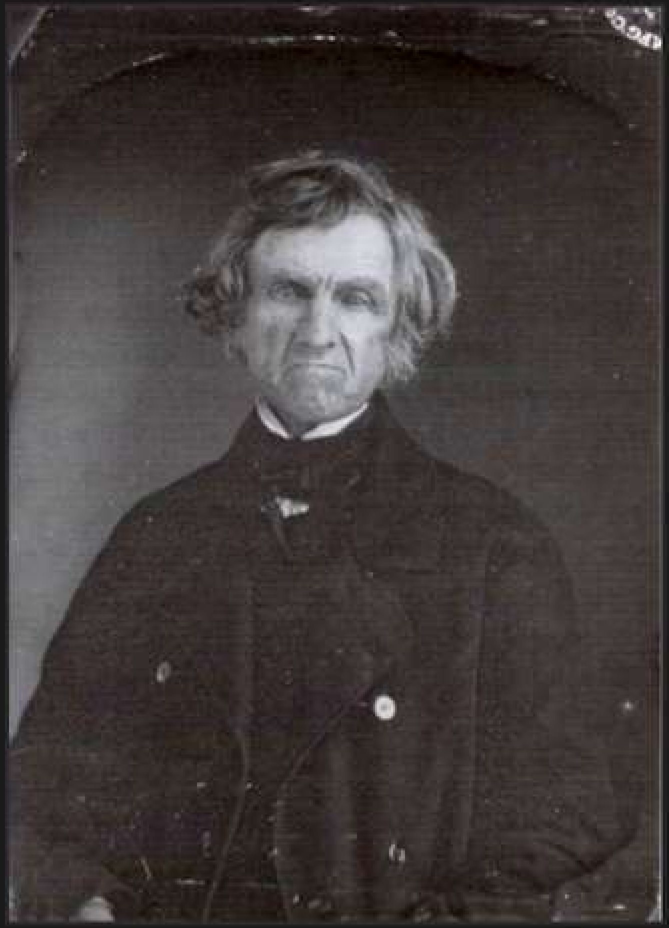


Environment Images

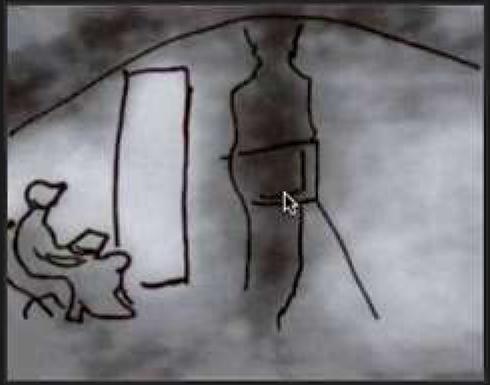
Retinal Images

© 2020 Shree K. Nayar

Going Back in Time



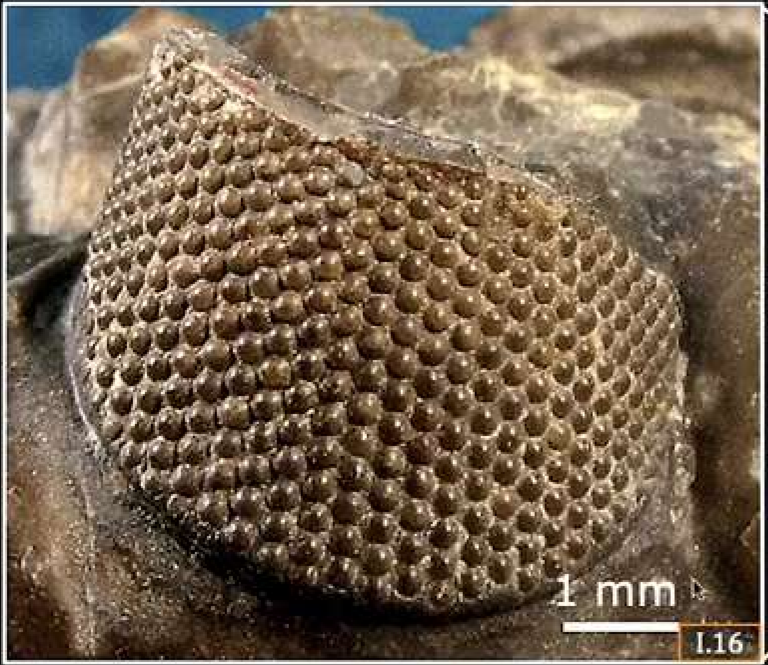
"Grumpy Grandpa", c. 1845

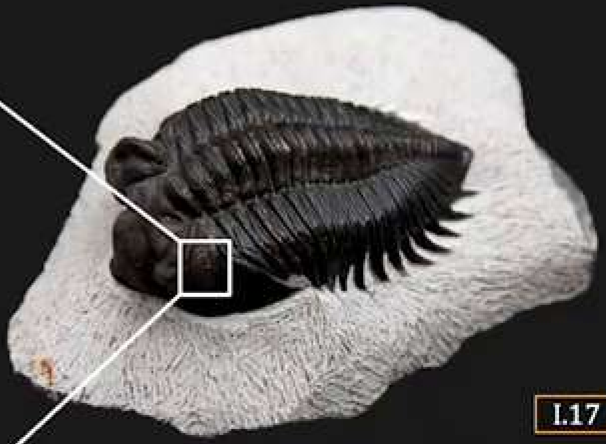


[redacted]

© 2020 Shree K. Nayar

Fossilized Eye of Trilobite

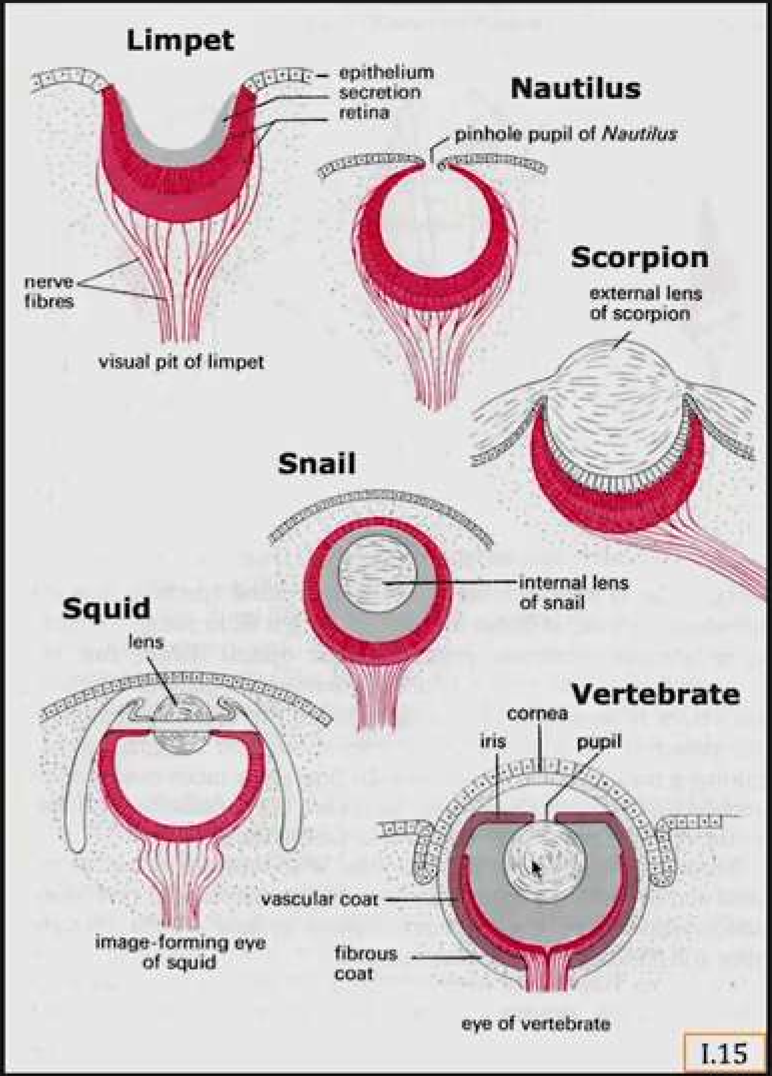




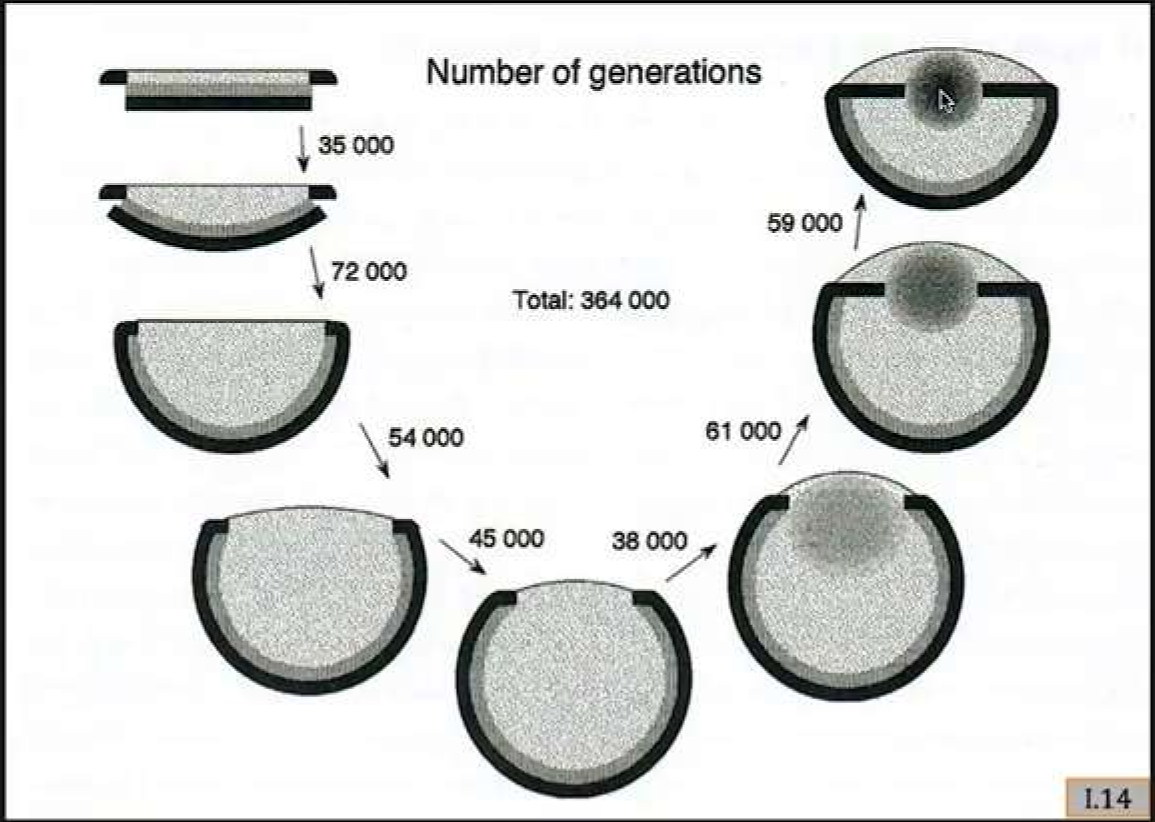
Coltraneia oufatenensis
Devonian Period (416 - 356 Million Years Ago)

Earliest kind of eye preserved as a fossil. The facets are the corneal lenses made from transparent Calcite ($CaCO_3$).

Primitive Eyes

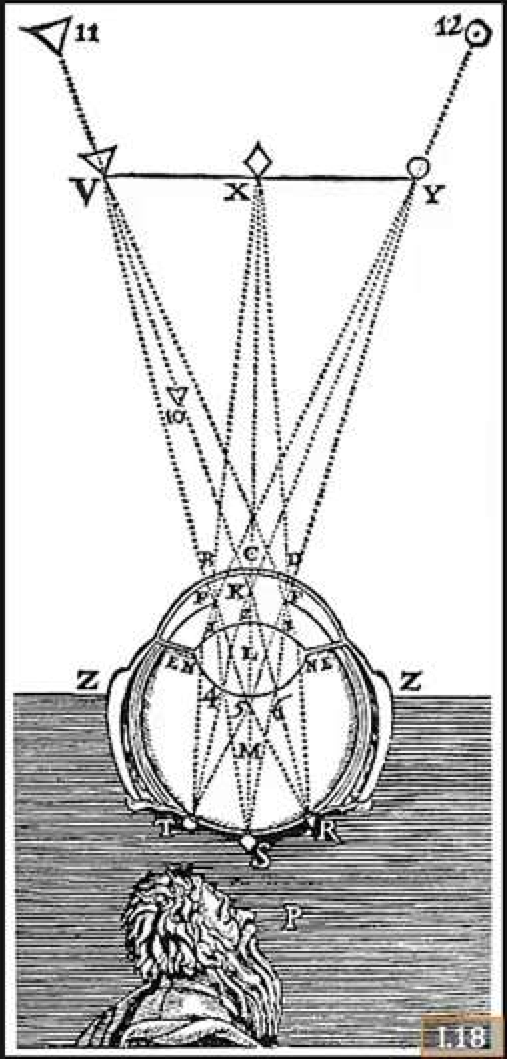


Evolution of Eye: A Simulation



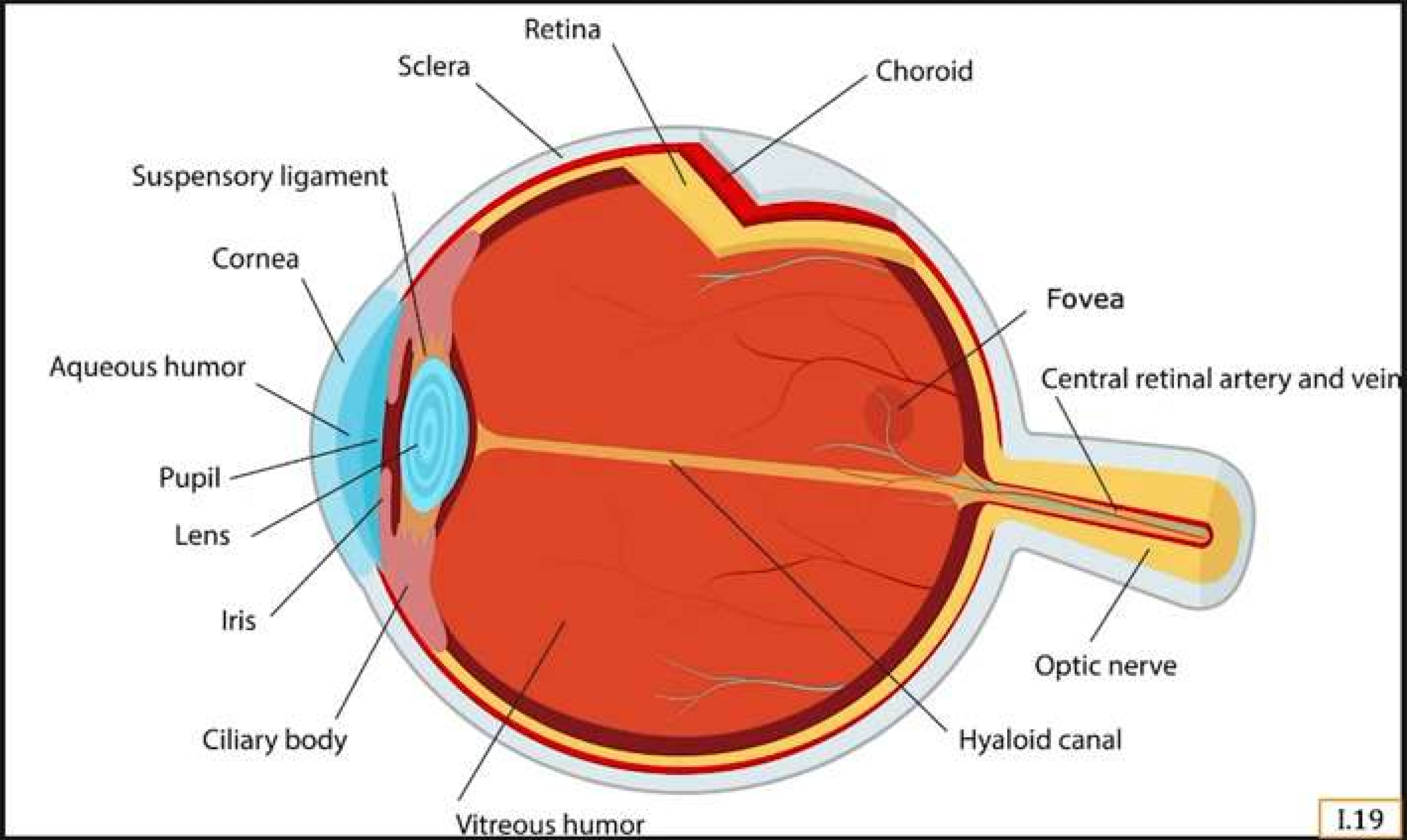
A patch of light sensitive epithelium can be gradually turned into a perfectly focused camera-type eye, if there is a continuous natural selection for improved spatial vision. This simulation reveals that the complete evolution can be accomplished in about 400,000 generations. First a pigment cup eye evolves, and then a lens.

Image Formation in the Eye

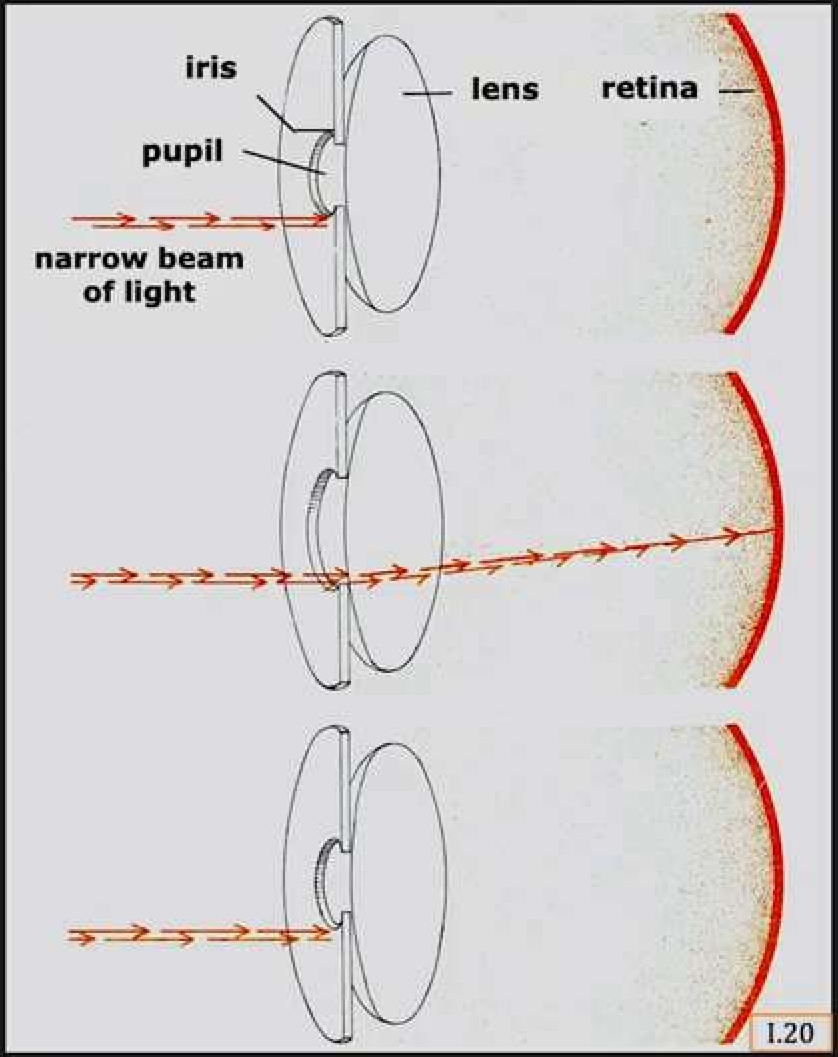


Descartes removed the eye of an ox, scraped its back to make it transparent, and then observed on it from a darkened room "not perhaps without wonder and pleasure" the inverted image of the scene.

Optics in Human Eye

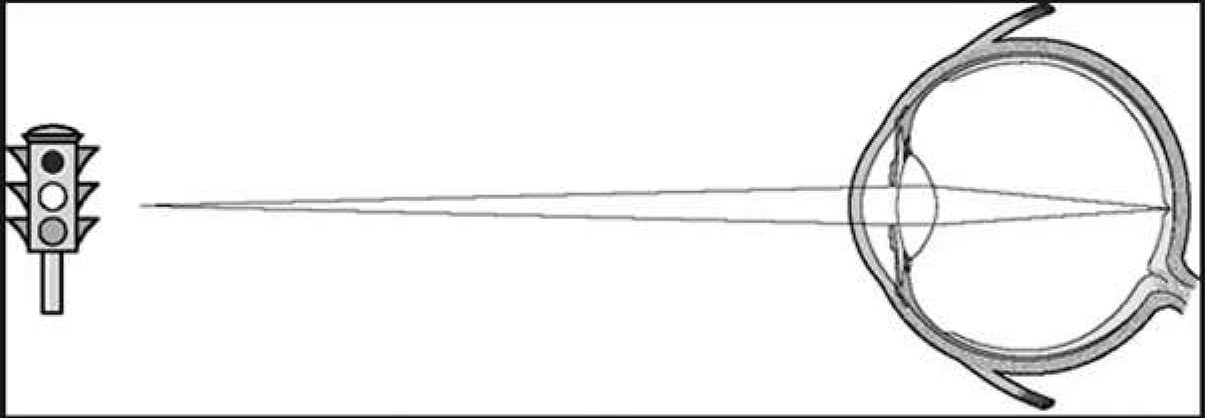


Human Eye: Iris Control System

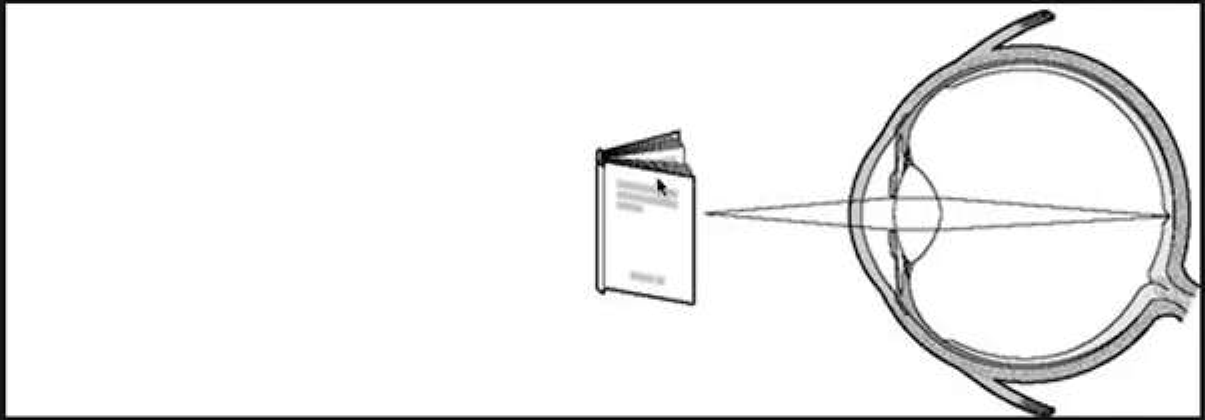


Making the iris oscillate with a narrow beam of light. When the iris opens up, strong light reaches the retina, which causes the retina to close. When it closes, no light is received by the retina and the iris opens again. The frequency and amplitude of this oscillation of the iris reveals the response of its control system.

Accommodation (Focusing) in the Eye

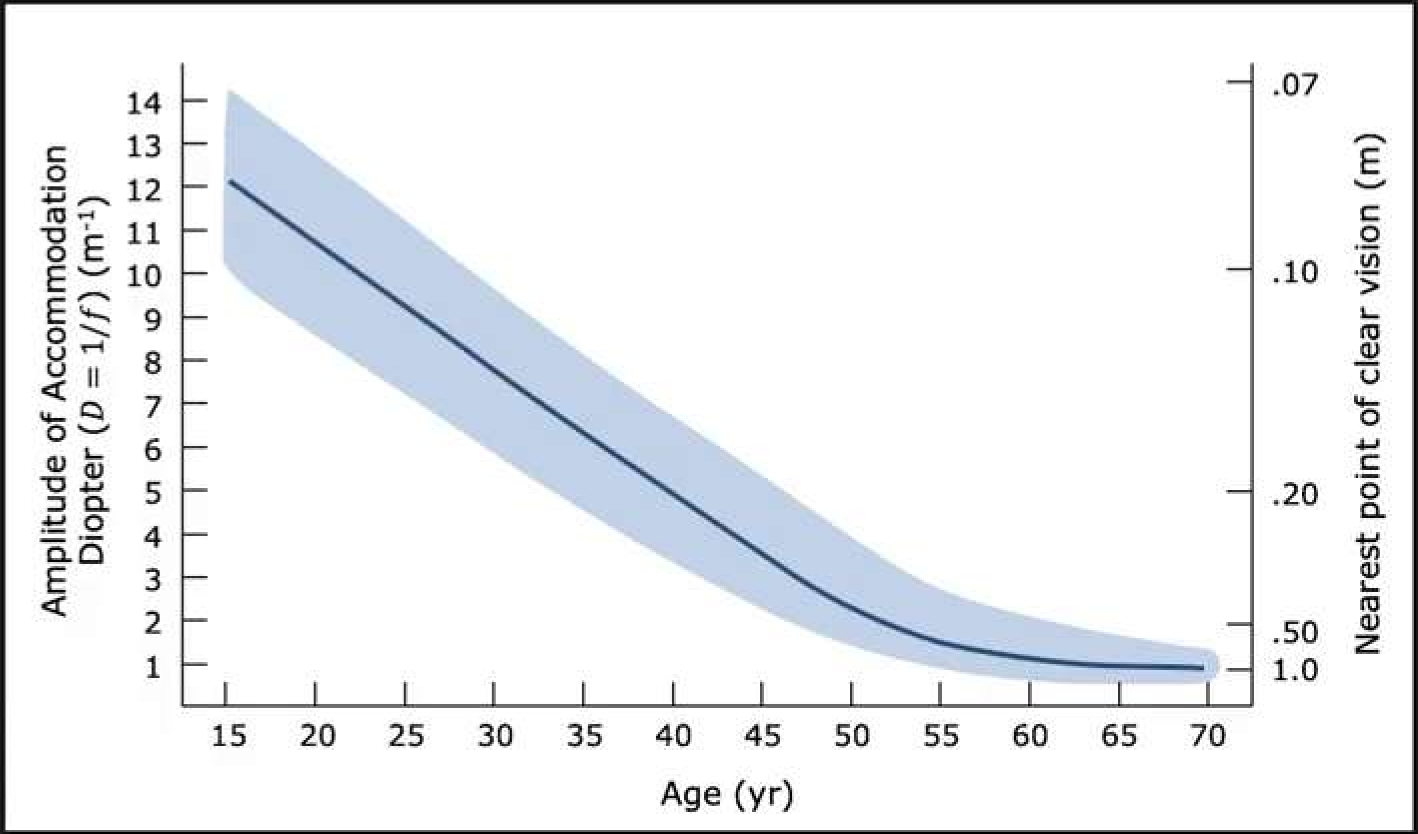


Focusing on distant objects: Lens is relaxed



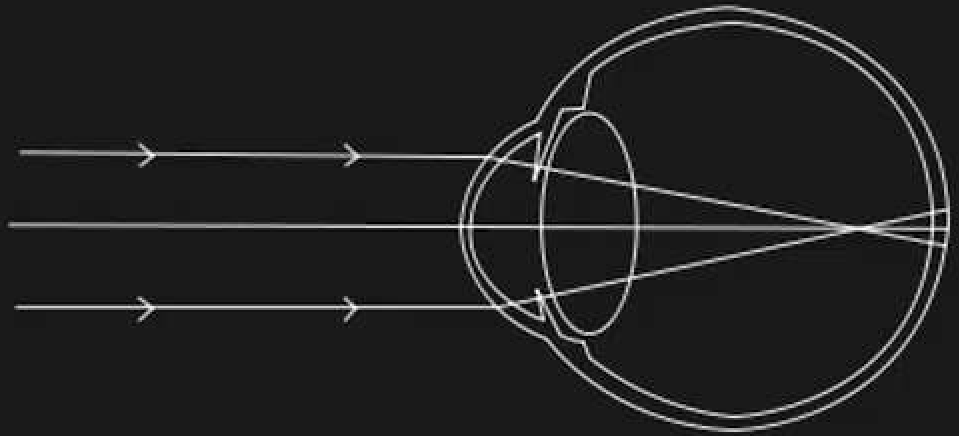
Focusing on nearby objects: Lens is squished

Change in Accommodation with Age

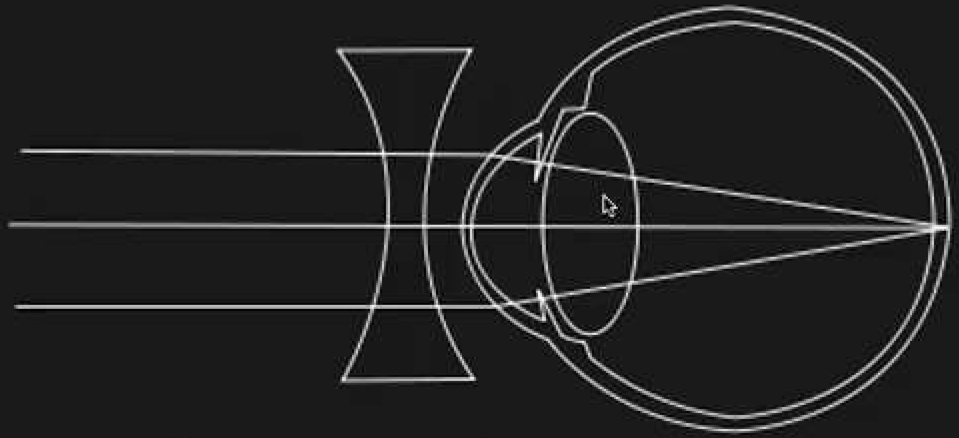


Accommodation decreases with age

Myopia (Near-Sightedness)

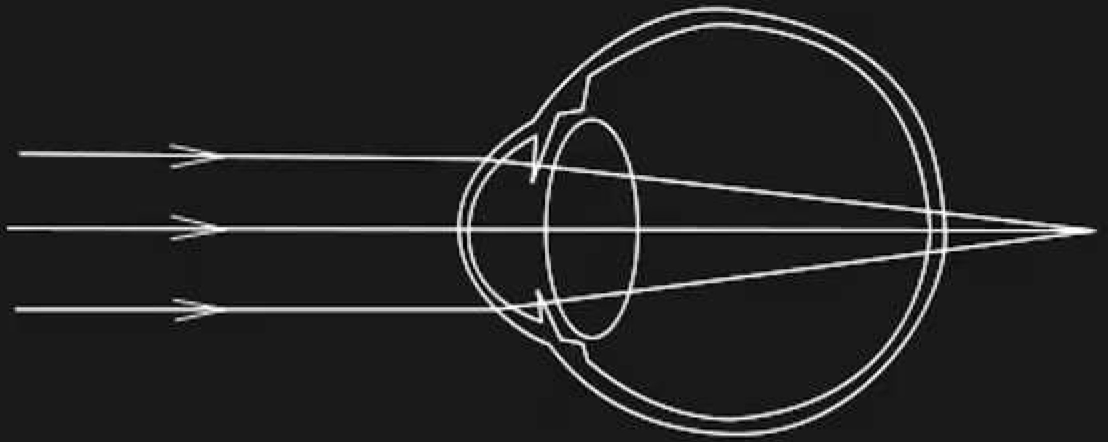


Inability to focus on objects far away

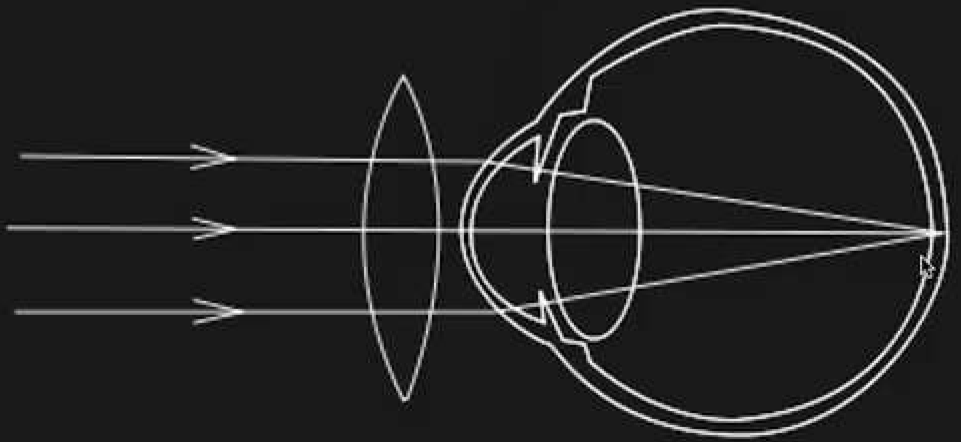


Can be fixed by using a diverging (concave) lens

Hyperopia (Far-Sightedness)

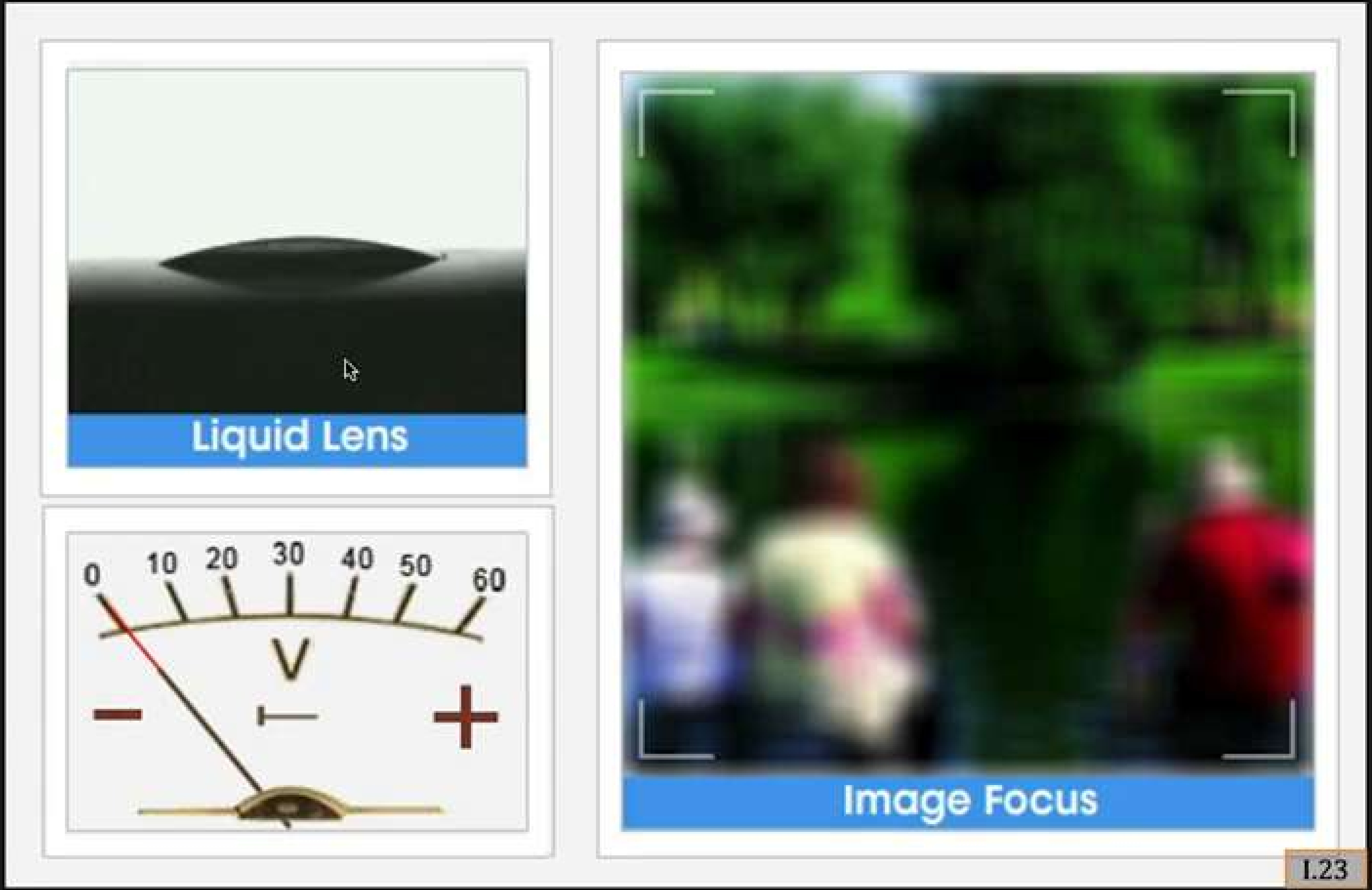


Inability to focus on nearby objects



Can be fixed by using a converging (convex) lens

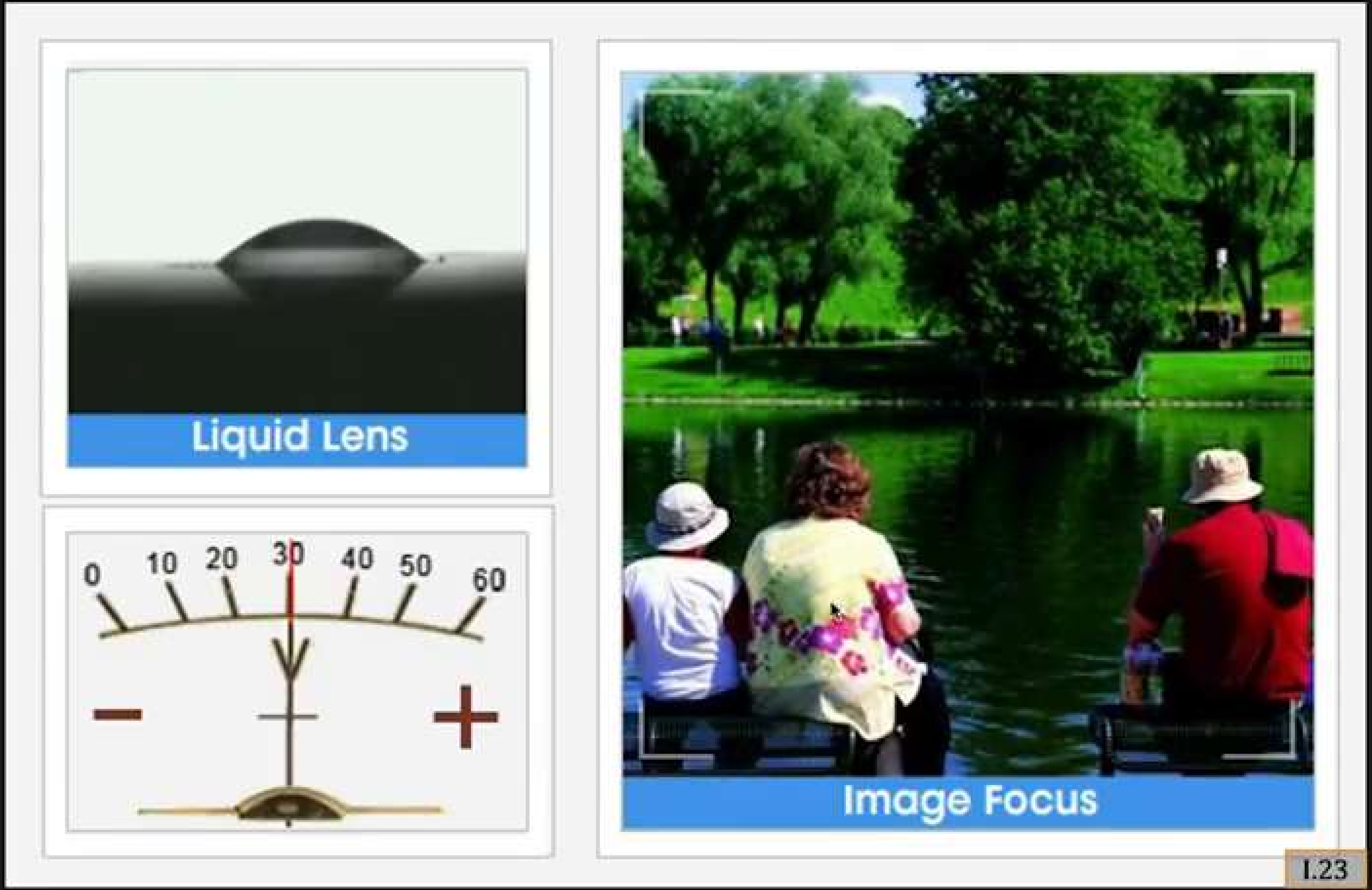
Liquid Lens



The shape and hence the focal length of the liquid lens can be precisely controlled by applying a voltage.

© 2020 Shree K. Nayar

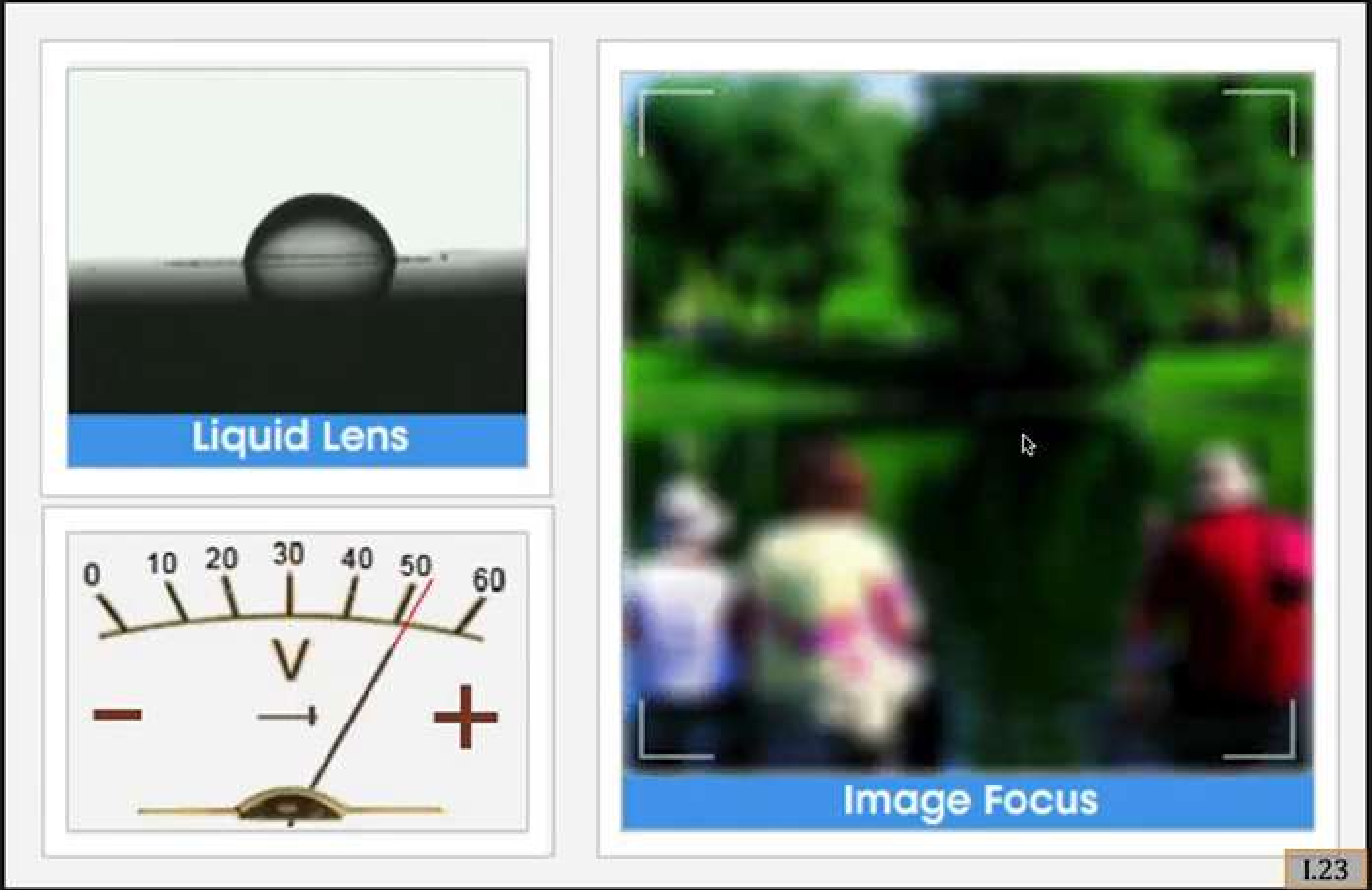
Liquid Lens



The shape and hence the focal length of the liquid lens can be precisely controlled by applying a voltage.

© 2020 Shree K. Nayar

Liquid Lens



The shape and hence the focal length of the liquid lens can be precisely controlled by applying a voltage.

© 2020 Shree K. Nayar

Image Sensing

Need to convert Optical Images to Digital Images (*numbers*) for computer representation and use.

Topics:

(1) A Brief History of Imaging

(2) Types of Image Sensors

(3) Resolution, Noise, Dynamic Range

(4) Sensing Color

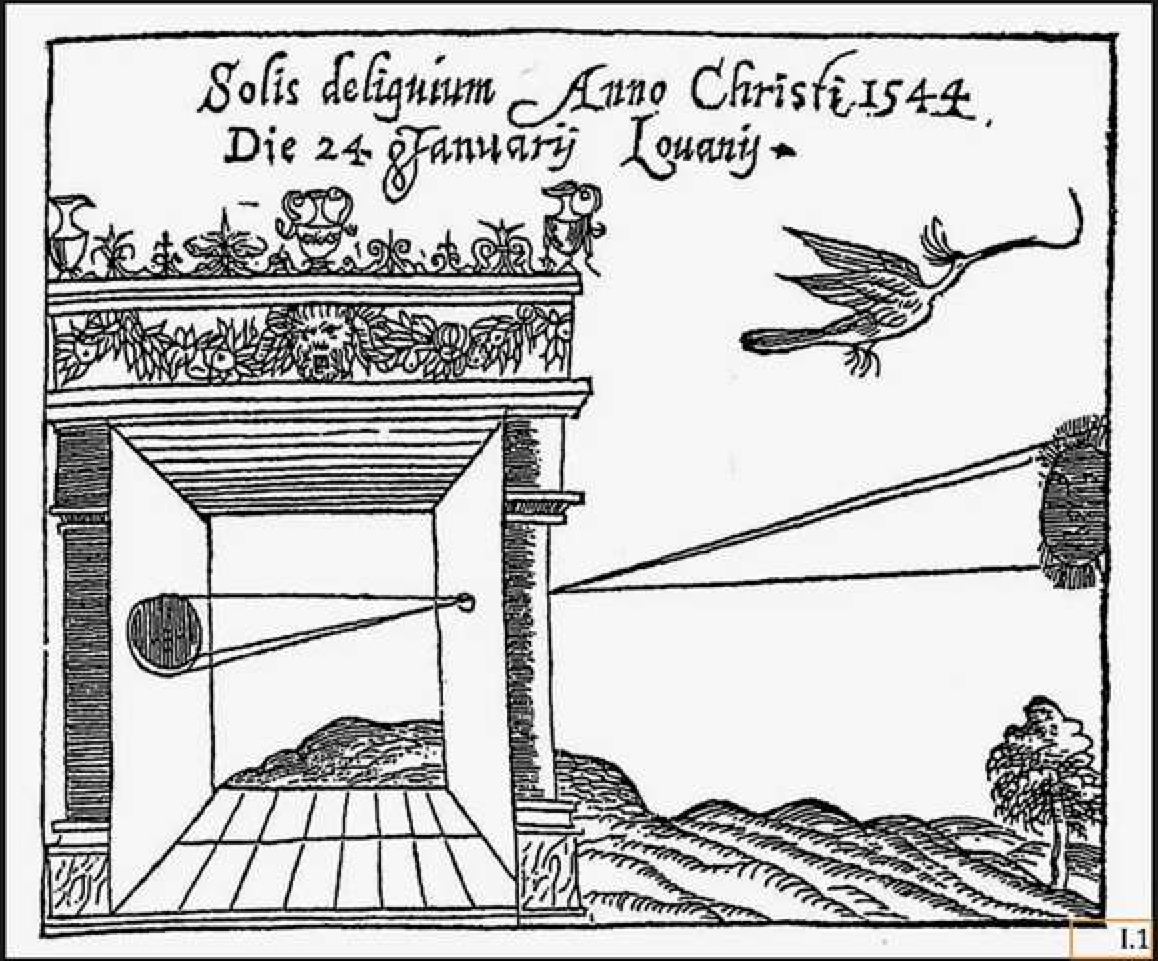
(5) Camera Response and HDR Imaging

(6) Nature’s Image Sensors

© 2020 Shree K. Nayar

Pinhole Camera

1558



Camera Obscura

ee K. Nayar

Lens Based Camera Obscura

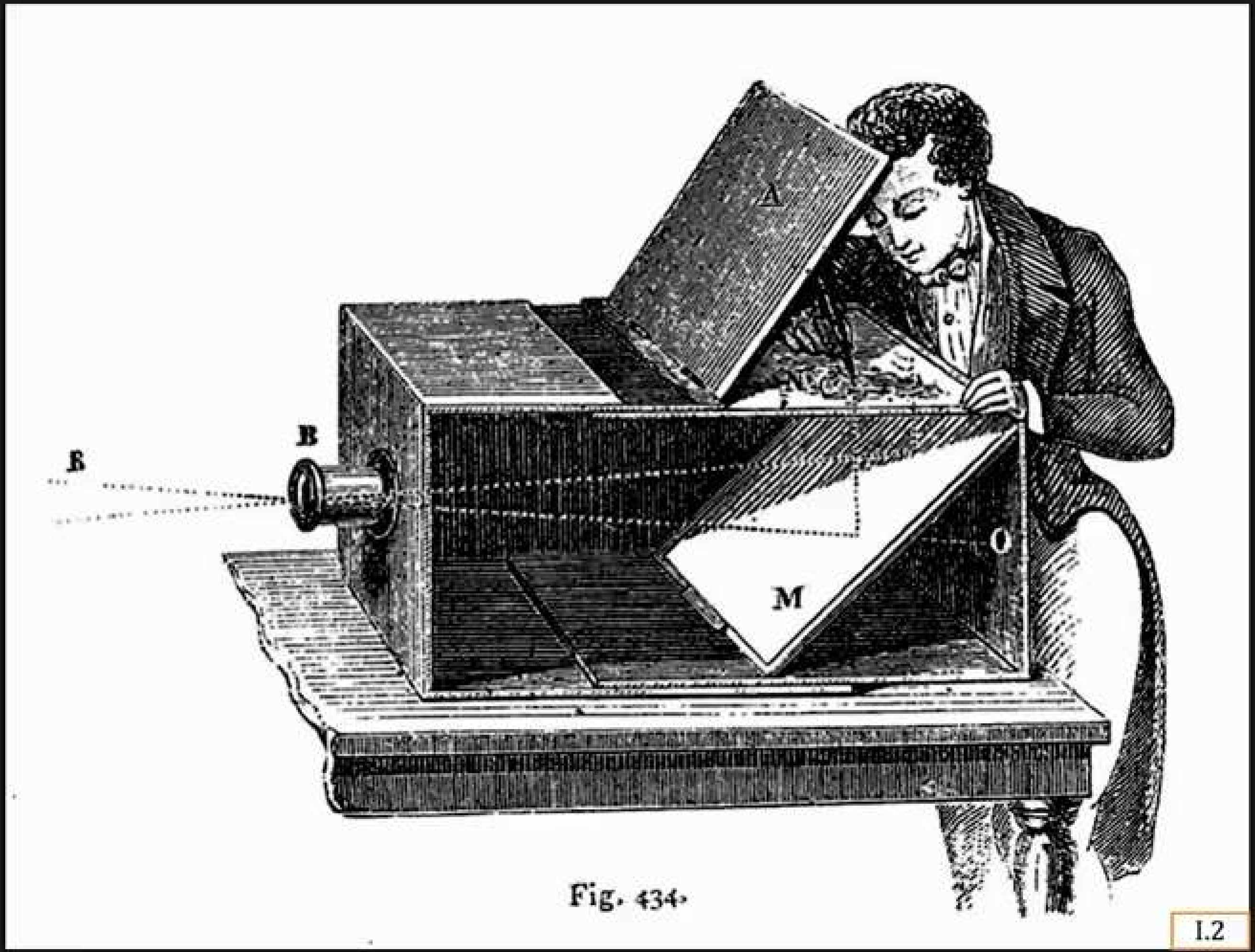


Fig. 434.

1558

1568

ee K. Nayar

Invention of Film

Still Life, Louis Jaques Mande Daguerre, 1837

- 1558
- 1568
- 1837

[redacted]

Color Film

Louis Ducos du Hauron, 1887

1558

1568

1837

1877

HISTORY

ee K. Nayar

Ernemann Camera

- 1558
- 1568
- 1837
- 1887
- 1928

ee K. Nayar

Silicon Image Detector

- 1558
- 1568
- 1837
- 1887
- 1928
- 197[redacted]

ee K. Nayar

Digital Cameras

- 1558
- 1568
- 1837
- 1887
- 1928
- 1970
- 1994

[redacted]

HISTORY

ee K. Nayar

Phones with Cameras

iPhone 1

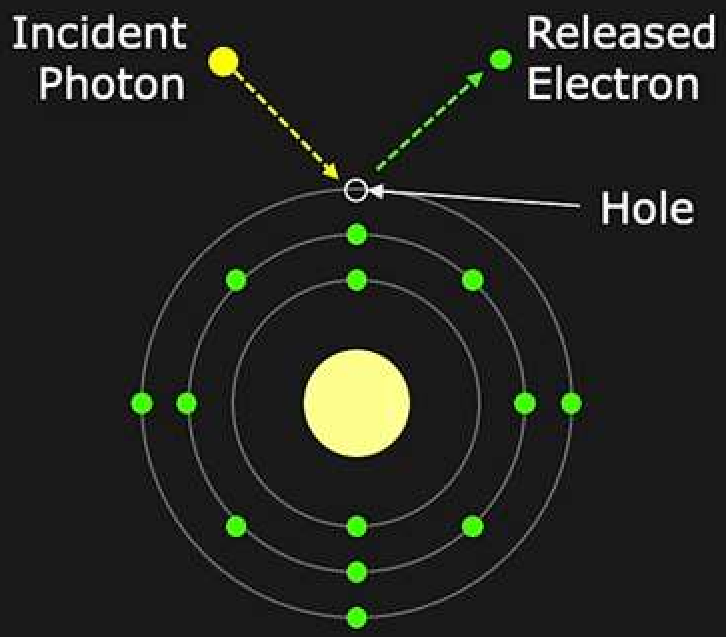
- 1558
- 1568
- 1837
- 1887
- 1928
- 1970
- 1994
- 2007

[redacted]

ee K. Nayar


BROWNIE
TARGET
SIX-20
MADE IN U.S.A. BY
EASTMAN KODAK COMPANY

Converting Light into Electric Charge

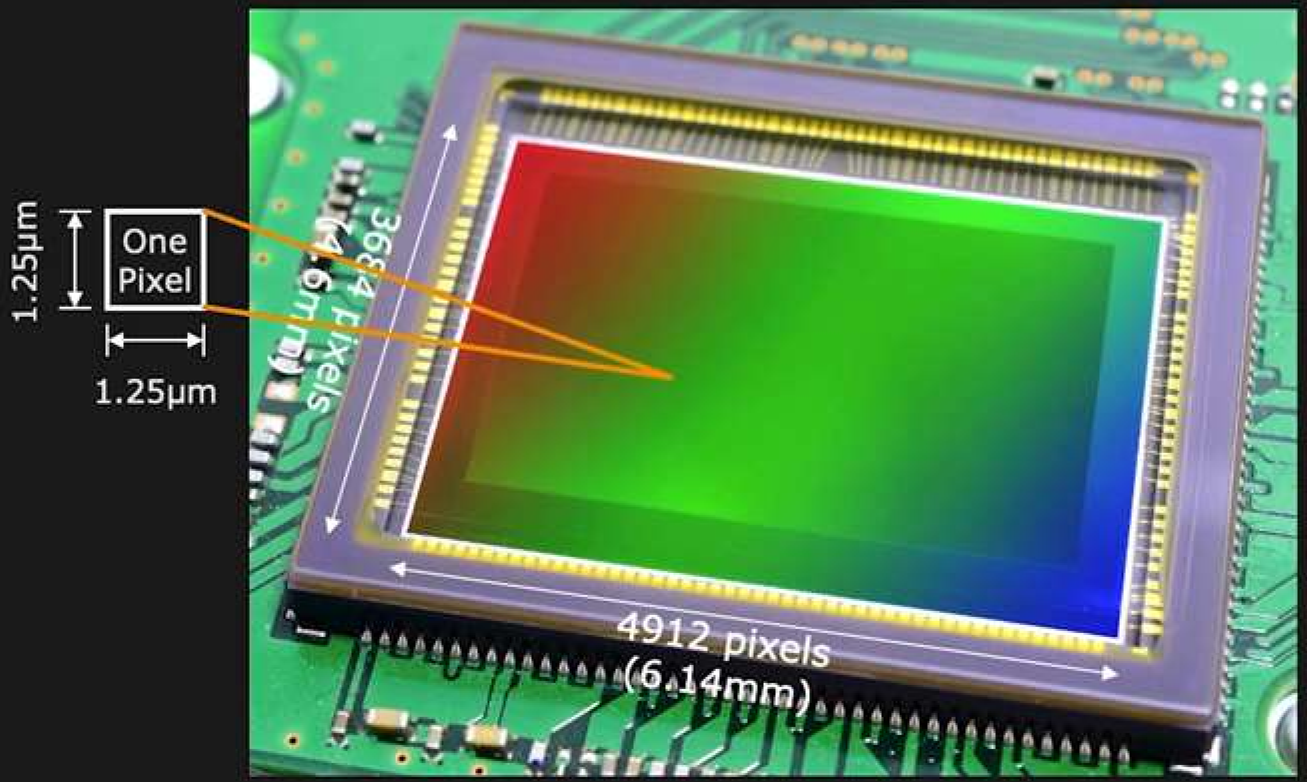


Silicon Atom

Photon with sufficient energy incident on a Si atom creates an electron-hole pair.

© 2020 Shree K. Nayar

Image Sensor: A Closer Look

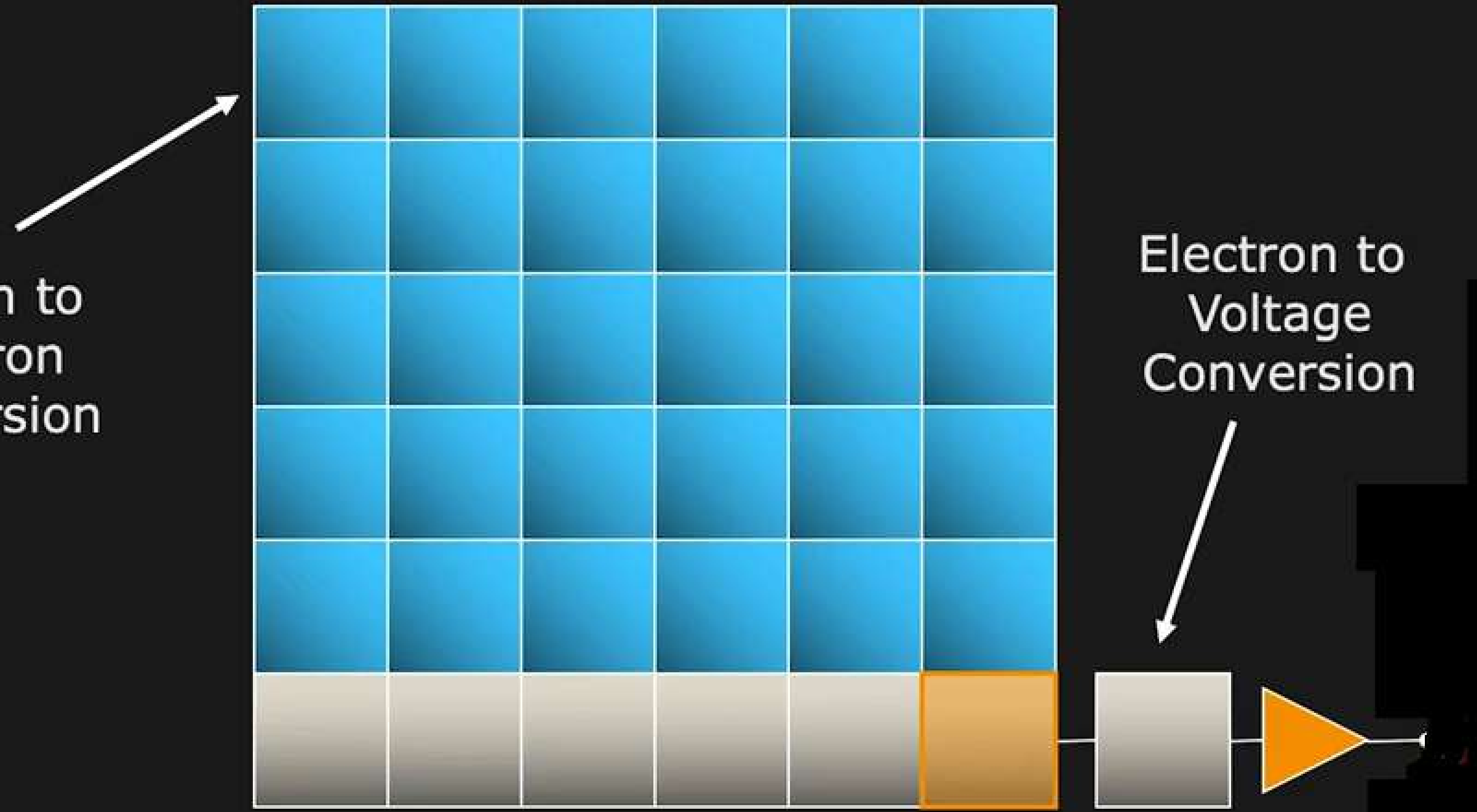


18 Megapixels

© 2020 Shree K. Nayar

Types of Image Sensors: CCD

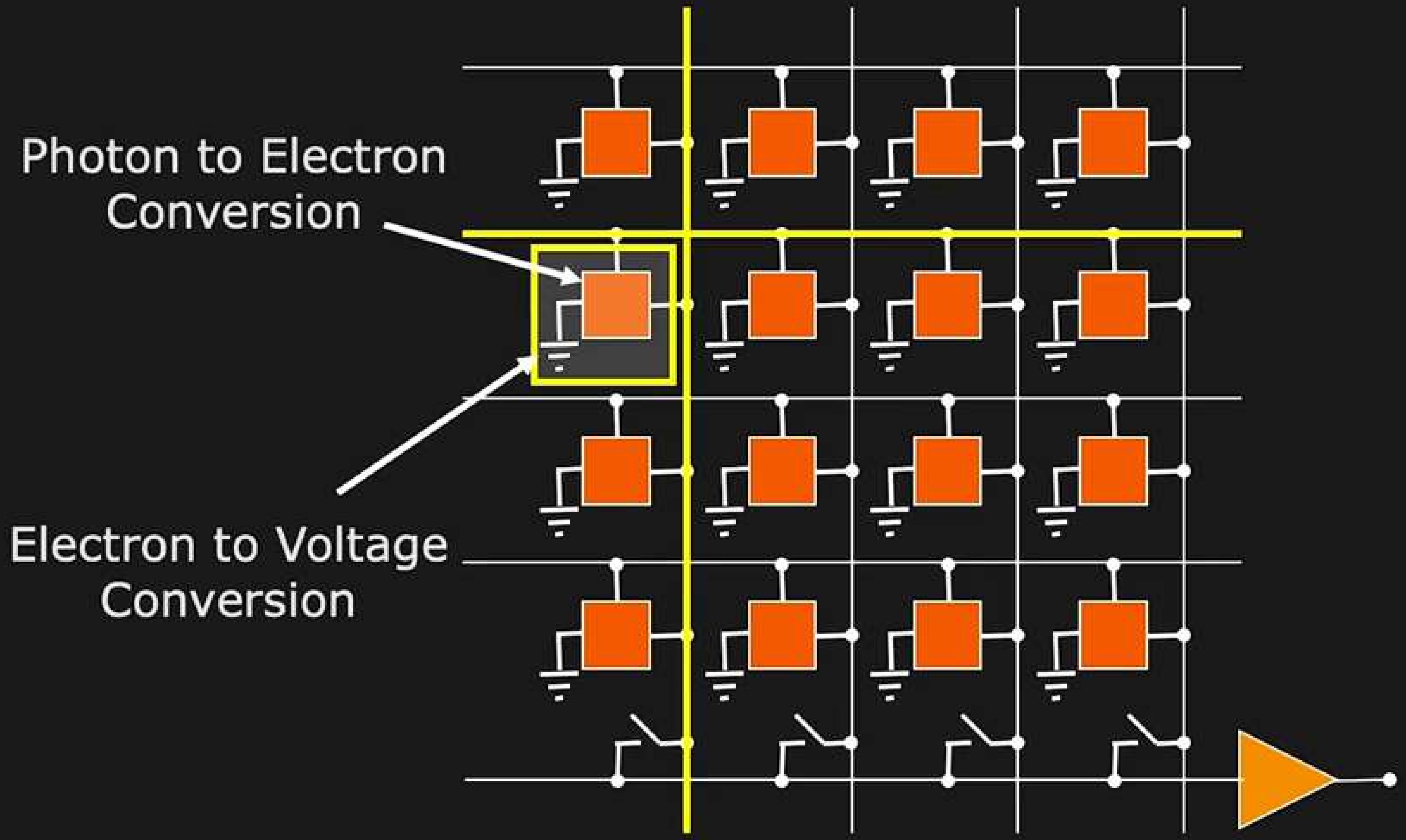
CCD: Charge Coupled Device



© 2020 Shree K. Nayar

Types of Image Sensors: CMOS

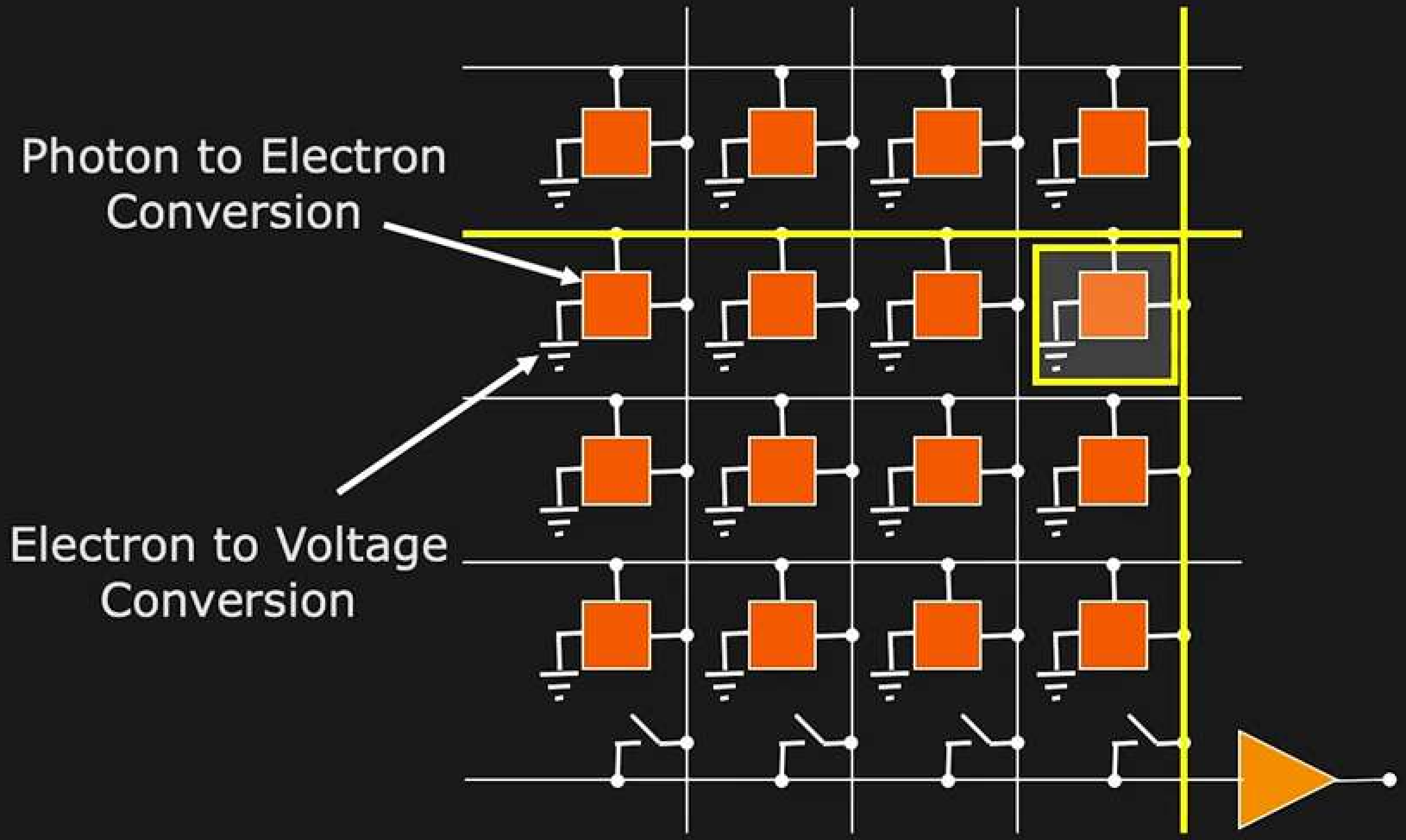
CMOS: Complimentary Metal-Oxide Semiconductor



© 2020 Shree K. Nayar

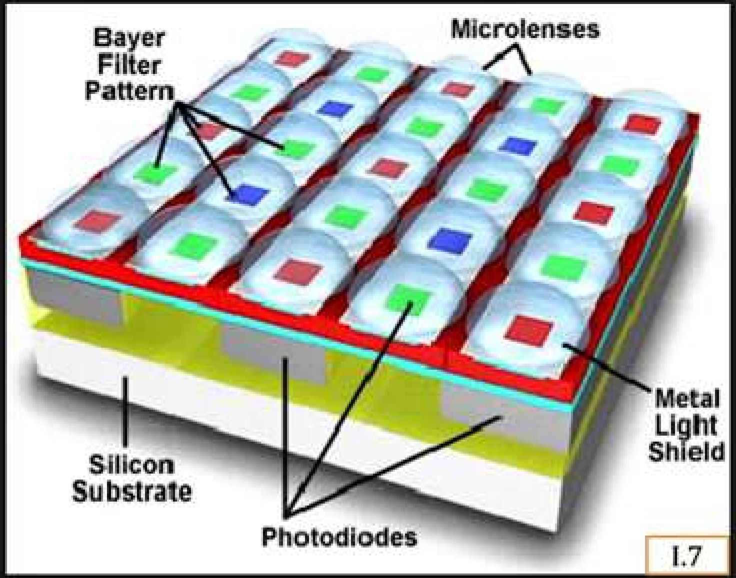
Types of Image Sensors: CMOS

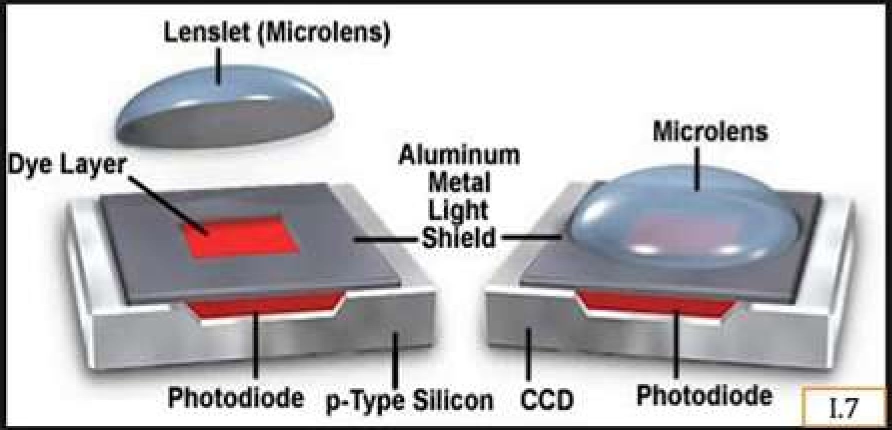
CMOS: Complimentary Metal-Oxide Semiconductor



© 2020 Shree K. Nayar

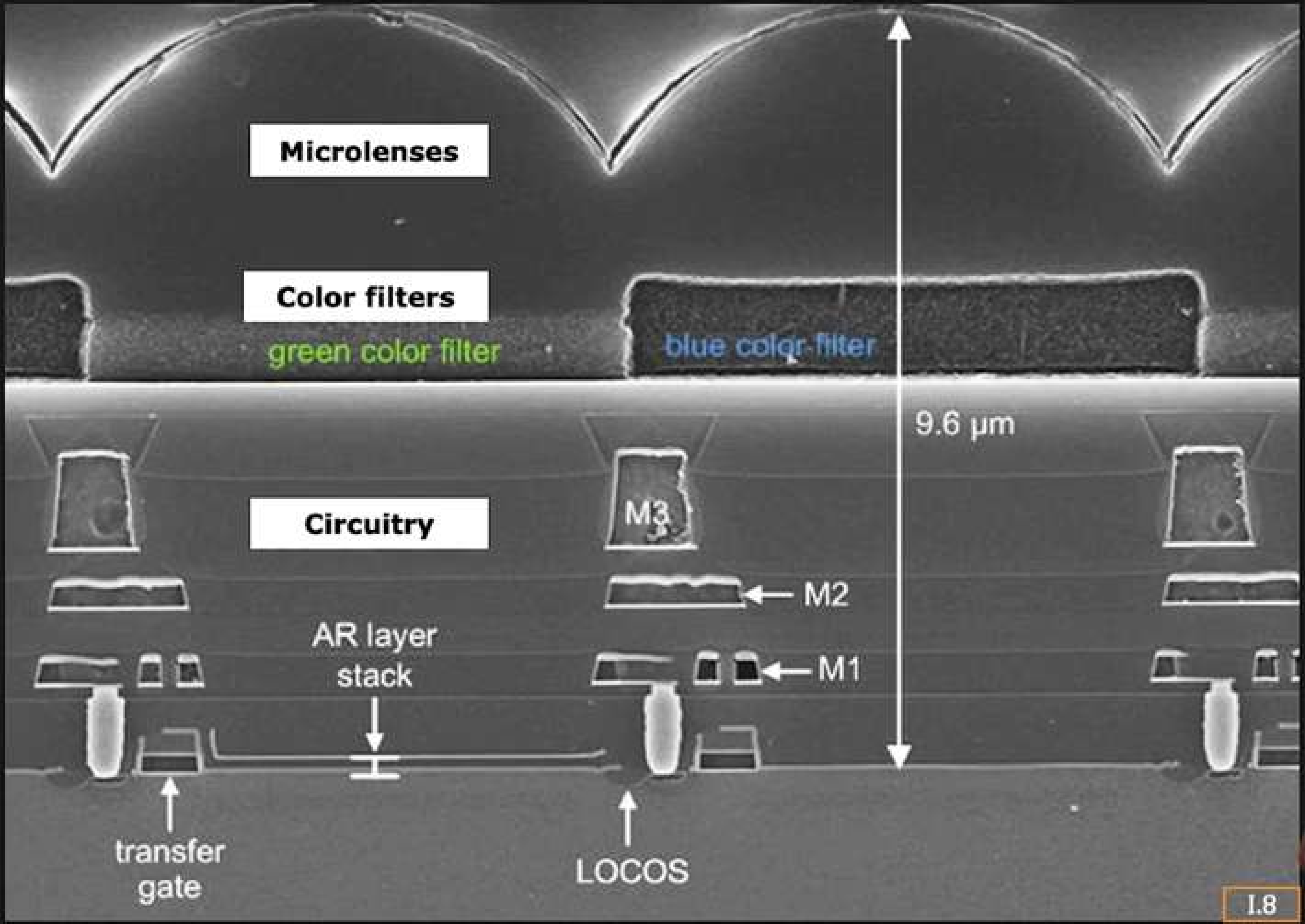
Image Sensor: A Closer Look





© 2020 Shree K. Nayar

Image Sensor: A Closer Look

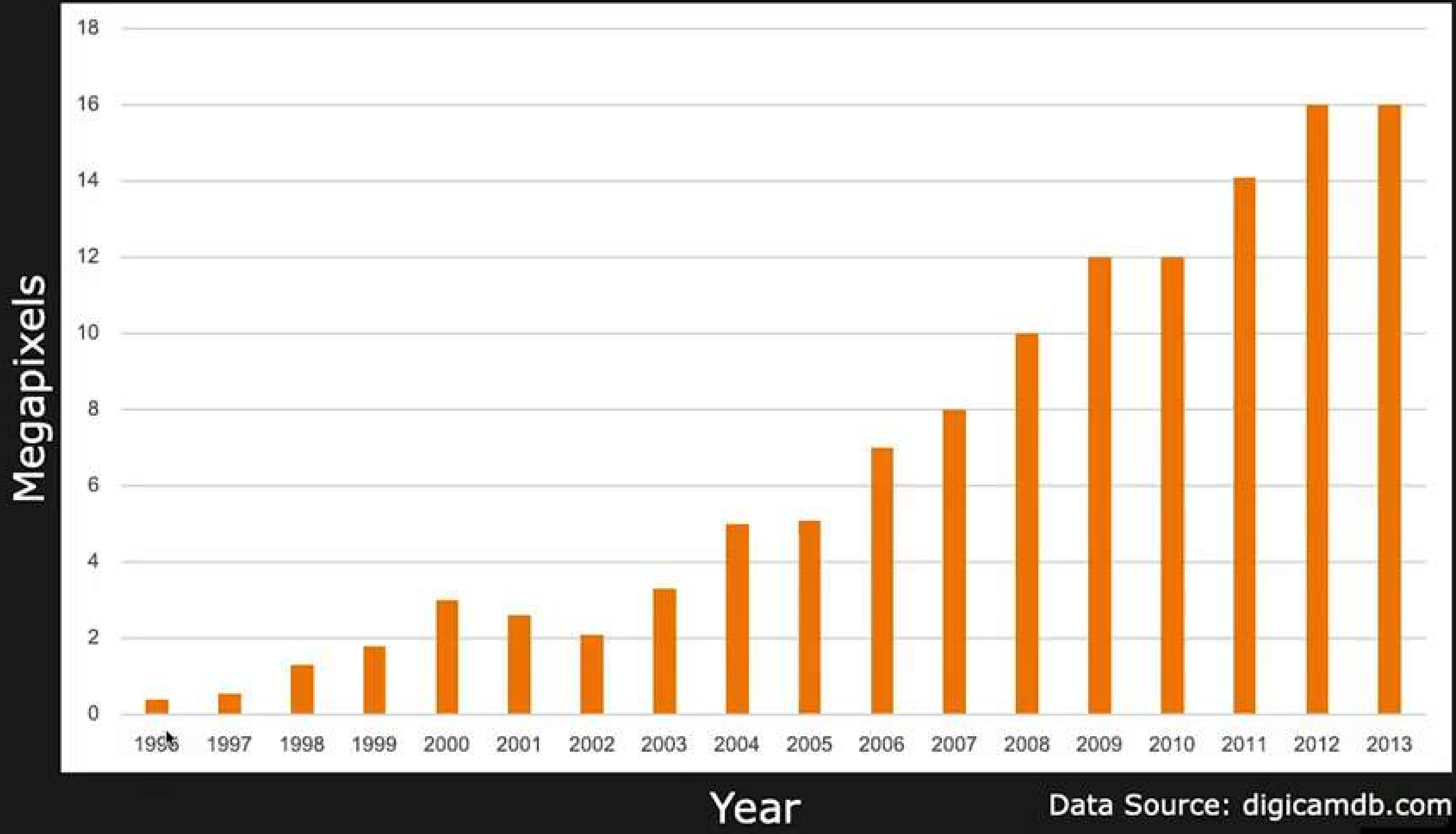


Cross Section of an Image Sensor
(Scanning Electron Microscope Image)

© 2020 Shree K. Nayar

Image Sensor Resolution

Median Sensor Resolution in Consumer Cameras



Data Source: digicamdb.com

© 2020 Shree K. Nayar

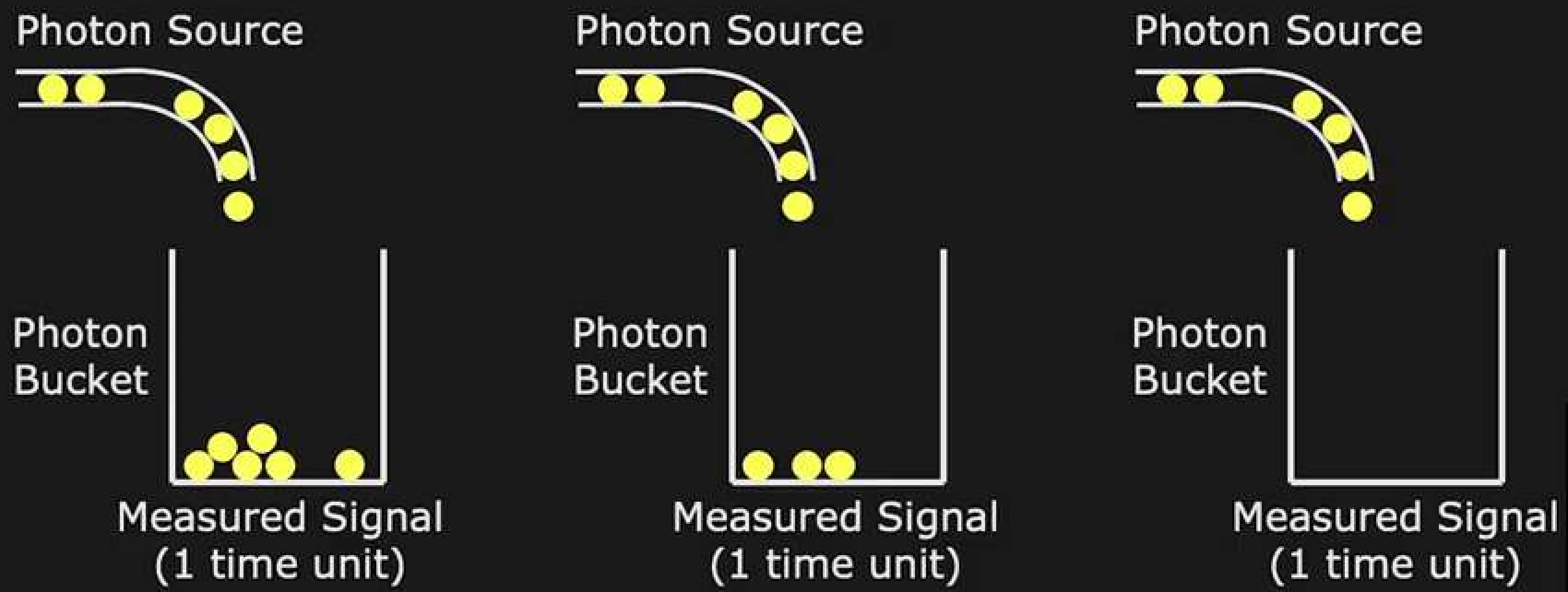
Noise in Image Sensors

Noise: Unwanted modification of signal during capture, conversion, transmission, processing.

- Photon Shot Noise (Scene Dependent)
  - Quantum nature of light
  - Random arrival of photons
- Readout Noise (Scene Independent)
  - Electronic Noise: Pre analog-to-digital conversion
  - Quantization Noise: Post analog-to-digital conversio[redacted]
- Other Sources (Scene Independent)
  - Dark Current Noise: Thermally generated electrons
  - Fixed Pattern Noise: Defective pixels

© 2020 Shree K. Nayar

Photon Shot Noise

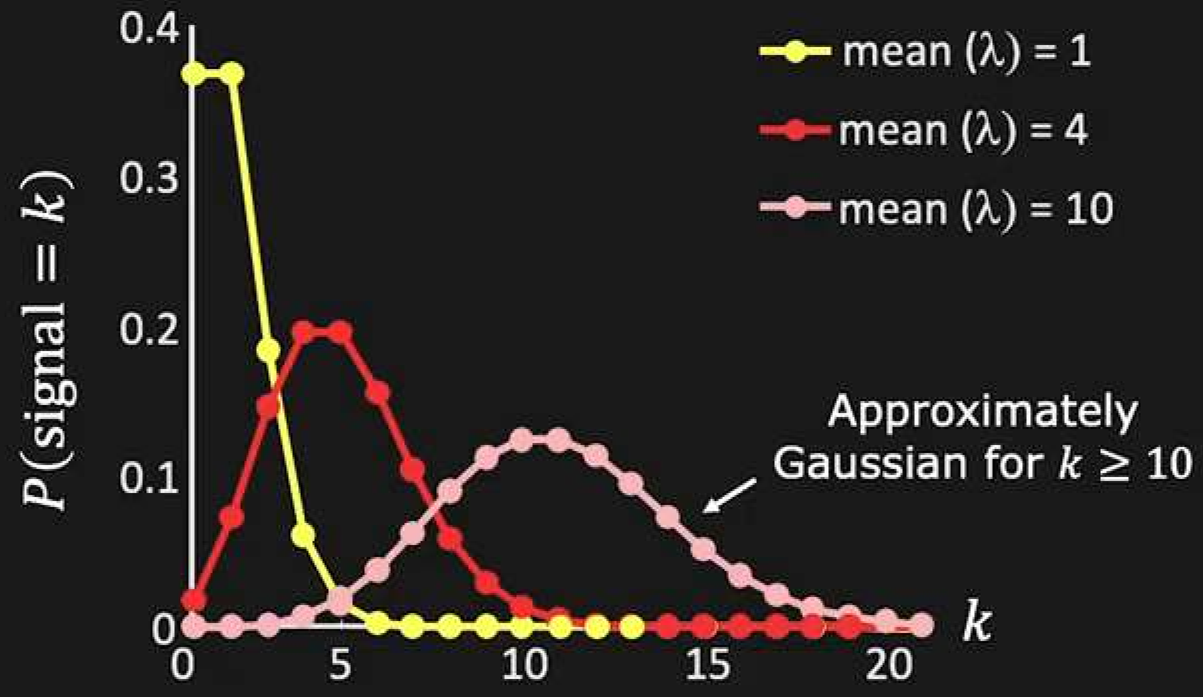


Average Photon Flux (Per Unit Time) = 3 Photons

Variation Due to Random Generation of Photons

© 2020 Shree K. Nayar

Photon Noise: Poisson Distribution

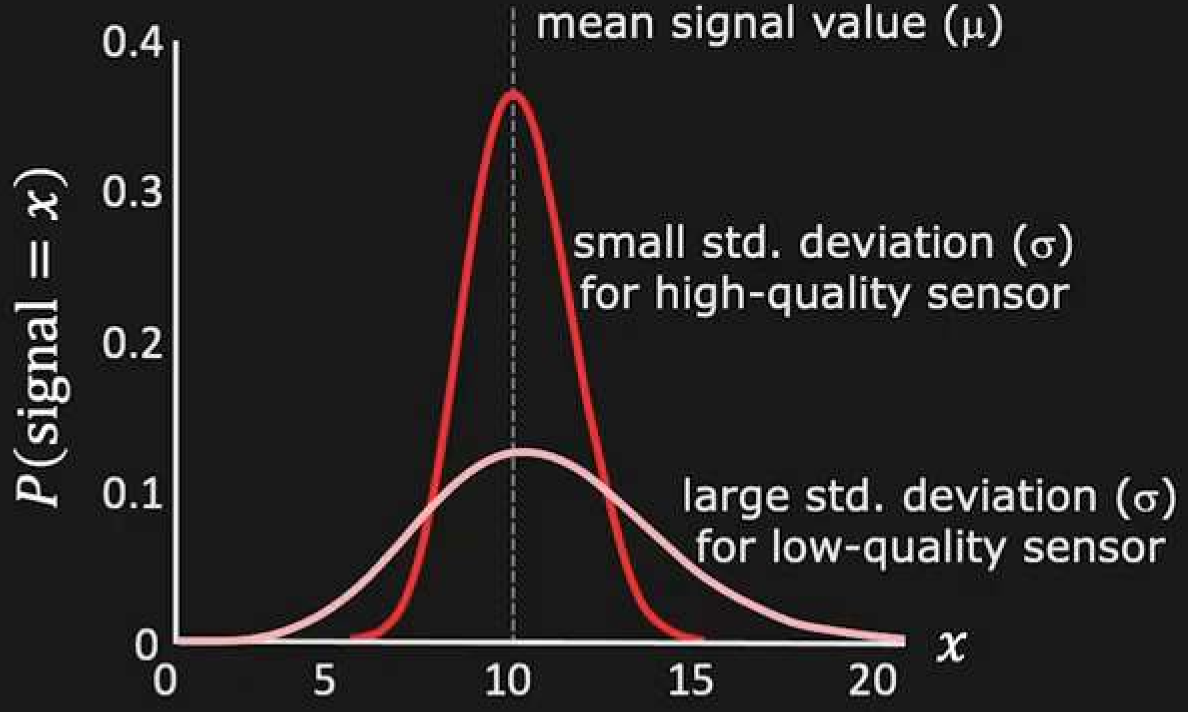


$$P(\text{signal} = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

$Var\,[\text{signal}] = Mean\,[\text{signal}] \Rightarrow$ Scene Dependent Noise

© 2020 Shree K. Nayar

Read Noise: Gaussian Distribution

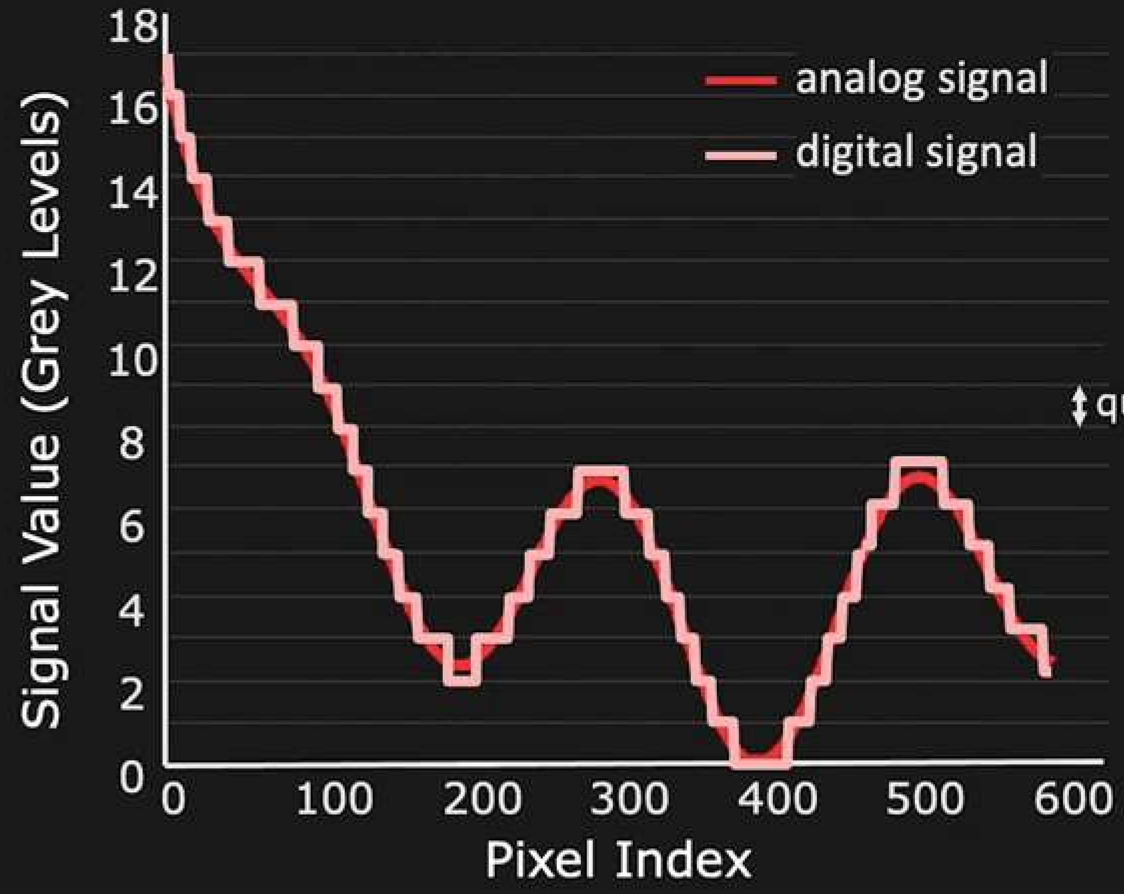


$$P(\text{signal} = x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Depends on Sensor Quality (Scene Independent)

© 2020 Shree K. Nayar

Quantization Noise



quantization step (Δ)

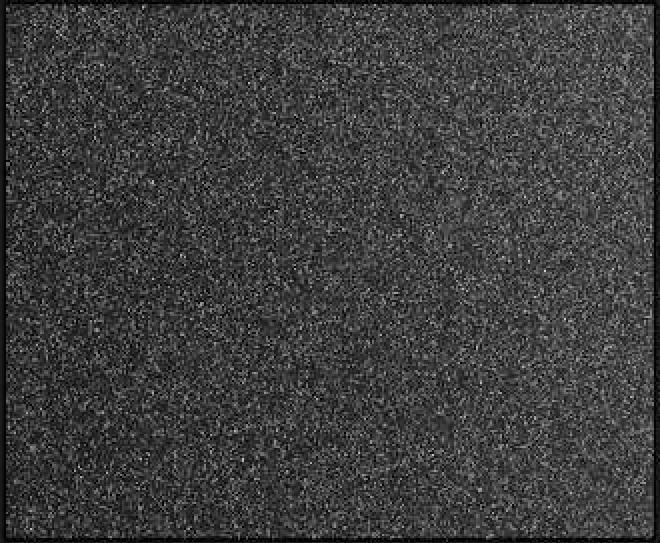
$$Var = \frac{\Delta^2}{12}$$

Negligible in Modern Sensors
Due to High Intensity Resolution (12-14 bits)

© 2020 Shree K. Nayar

Other Noise Sources

Dark Current Noise (Thermal)

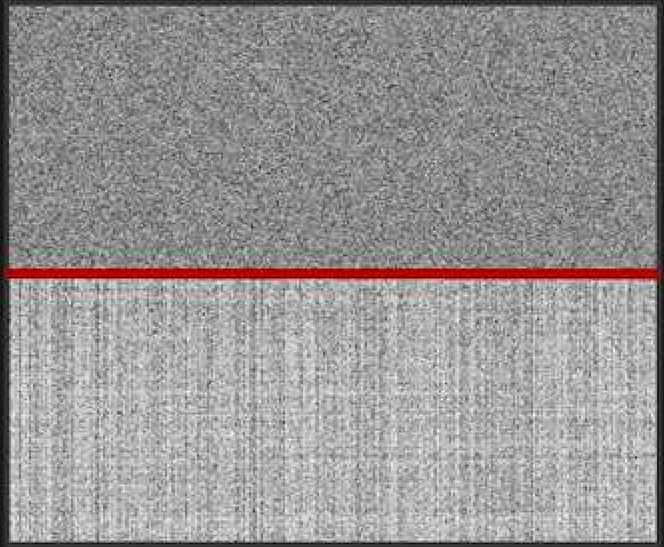


Follows Poisson Distribution

Significant Only for Long (>2 min) Exposures (Astronomy)

Fixed Pattern Noise (Defective Pixels)

Random Noise



Fixed Pattern Noise

Can be Reduced by Dark Frame Subtraction

© 2020 Shree K. Nayar

Sensor Dynamic Range

$$\text{Dynamic Range} = 20 \log \left(\frac{B_{max}}{B_{min}} \right) \quad \text{decibels (dB)}$$

B_{max}: The maximum possible photon energy (full potential well)

B_{min}: The minimum detectable photon energy (in the presence of noise)

Sensor	B_{max}: B_{min}	dB
Human Eye	1,000,000:1	120
HDR Display	200,000:1	106
Digital Camera	4096:1	72.2
Film Camera	2948:1	66.2
Digital Video	45:1	33.1

© 2020 Shree K. Nayar

Quantum Efficiency

Incoming light can vary in Wavelength (λ)

Pixel Intensity (Electron Flux) | Incoming Light (Photon Flux)

$$I \longleftarrow \boxed{q(\lambda)} \longleftarrow p(\lambda)$$

Silicon Pixel

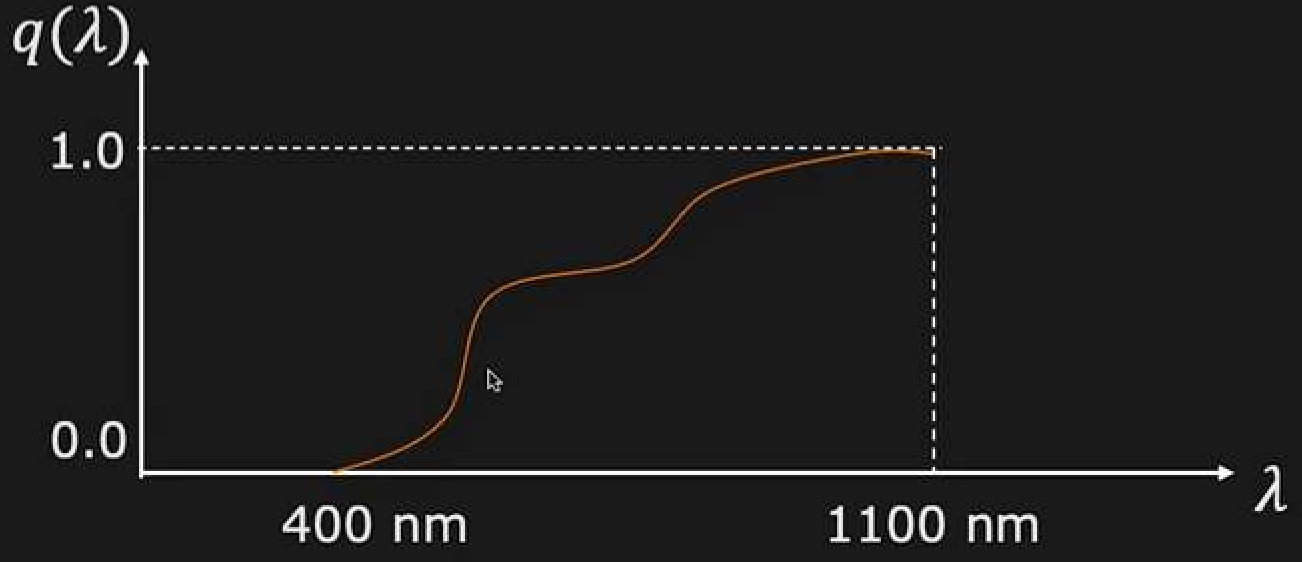
Quantum Efficiency:

$$q(\lambda) = \frac{\text{Electron Flux Generated}}{\text{Photon Flux of wavelength } \lambda}$$

© 2020 Shree K. Nayar

Quantum Efficiency of Silicon

Silicon is: $\begin{cases} \text{Transparent} & \text{for } \lambda > 1100\,nm \\ \text{Opaque} & \text{for } \lambda < 400\,nm \end{cases}$

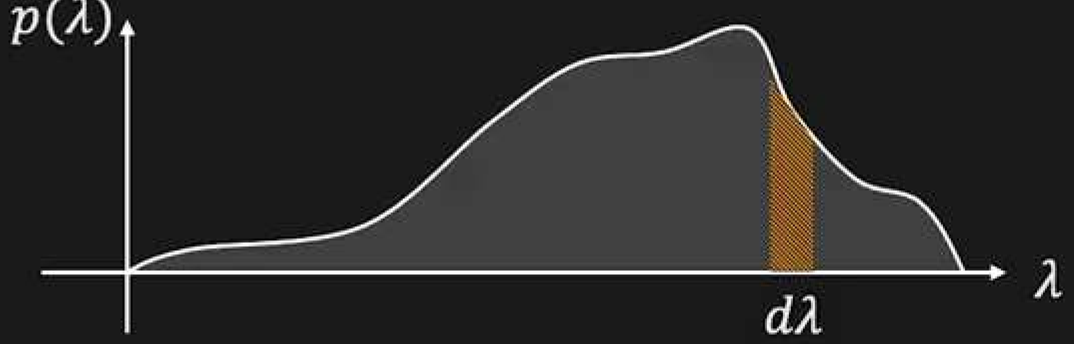


© 2020 Shree K. Nayar

Assume Monochromatic Light $\lambda = \lambda_i$ with flux $p(\lambda_i)$:

$$\boxed{I = q(\lambda_i)p(\lambda_i)}$$

What if incoming light has Spectral Distribution $p(\lambda)$?



Flux (Energy) of Light with wavelength between λ and $\lambda + d\lambda$ is $p(\lambda)d\lambda$. Therefore:

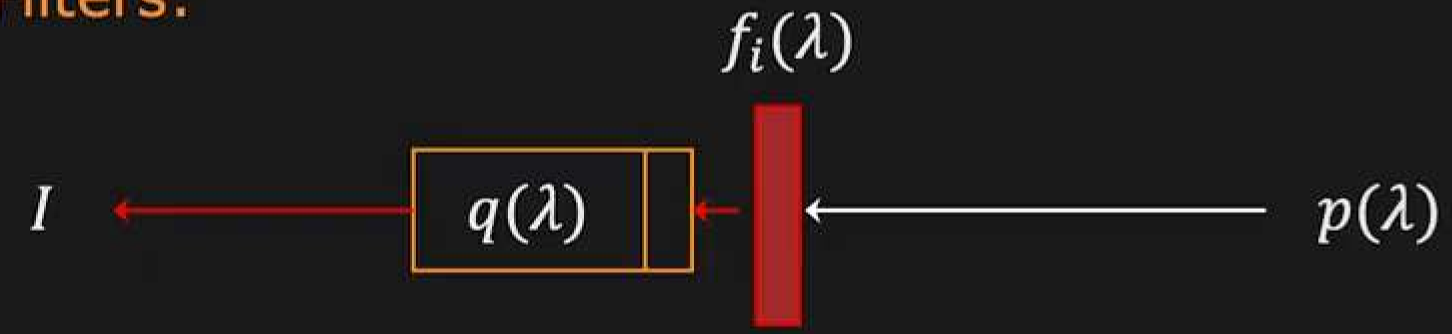
$$\boxed{I = \int_0^\infty q(\lambda)p(\lambda)d\lambda}$$

© 2020 Shree K. Nayar

$$I = \int_0^\infty q(\lambda)p(\lambda)d\lambda$$

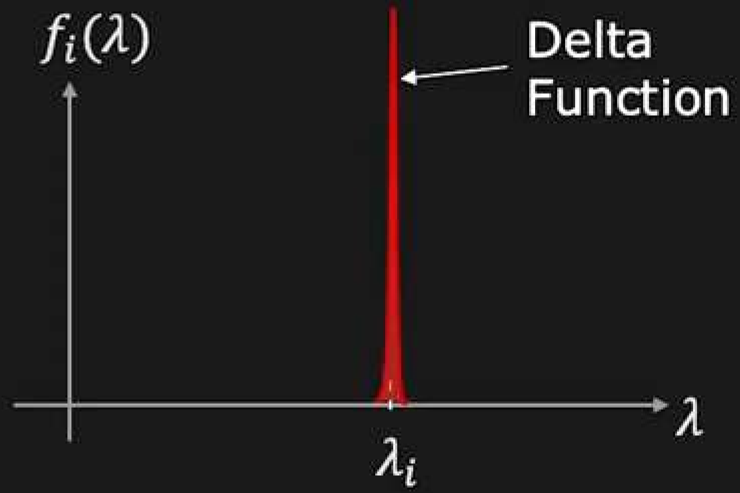
If we know I and $q(\lambda)$, can we find $p(\lambda)$?

Use Filters:



Let $f_i(\lambda) = \delta(\lambda - \lambda_i)$

Note: $\int_0^\infty \delta(\lambda - \lambda_i)d\lambda = 1$

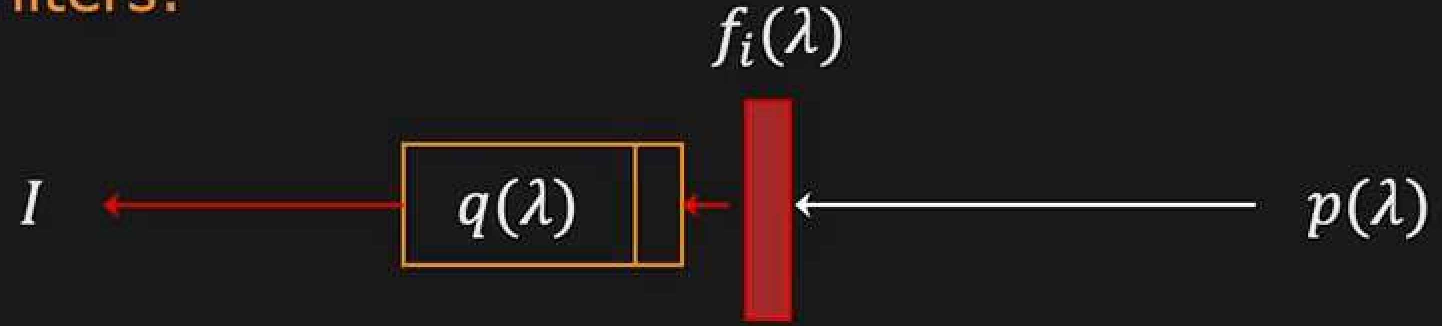


© 2020 Shree K. Nayar

$$\boxed{I = \int_0^\infty q(\lambda)p(\lambda)d\lambda}$$

If we know I and $q(\lambda)$, can we find $p(\lambda)$?

Use Filters:



$$I = \int_0^\infty q(\lambda)p(\lambda)f_i(\lambda)d\lambda \quad = \int_0^\infty q(\lambda)p(\lambda)\delta(\lambda - \lambda_i)d\lambda$$

$$\boxed{I = q(\lambda_i)p(\lambda_i)}$$

How many filters do we need to recover $p(\lambda)$? Infinite?

© 2020 Shree K. Nayar

What is "Color"?

Human Response to different wavelengths

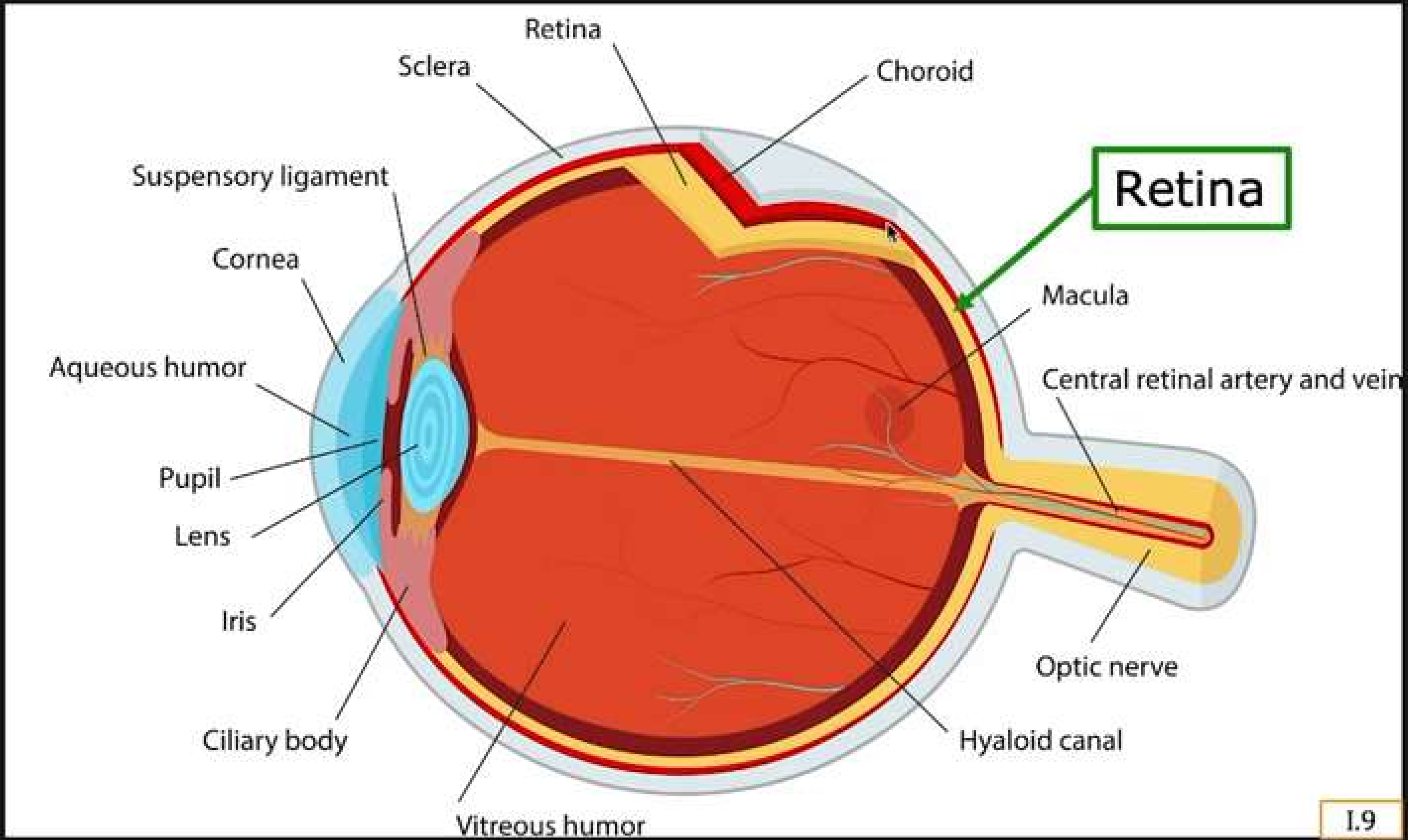
Visible light:

Do We recover spectral distribution $p(\lambda)$?

Sensors in the human eye: Rods & Cones
Neurochemical Sensors (3 types)

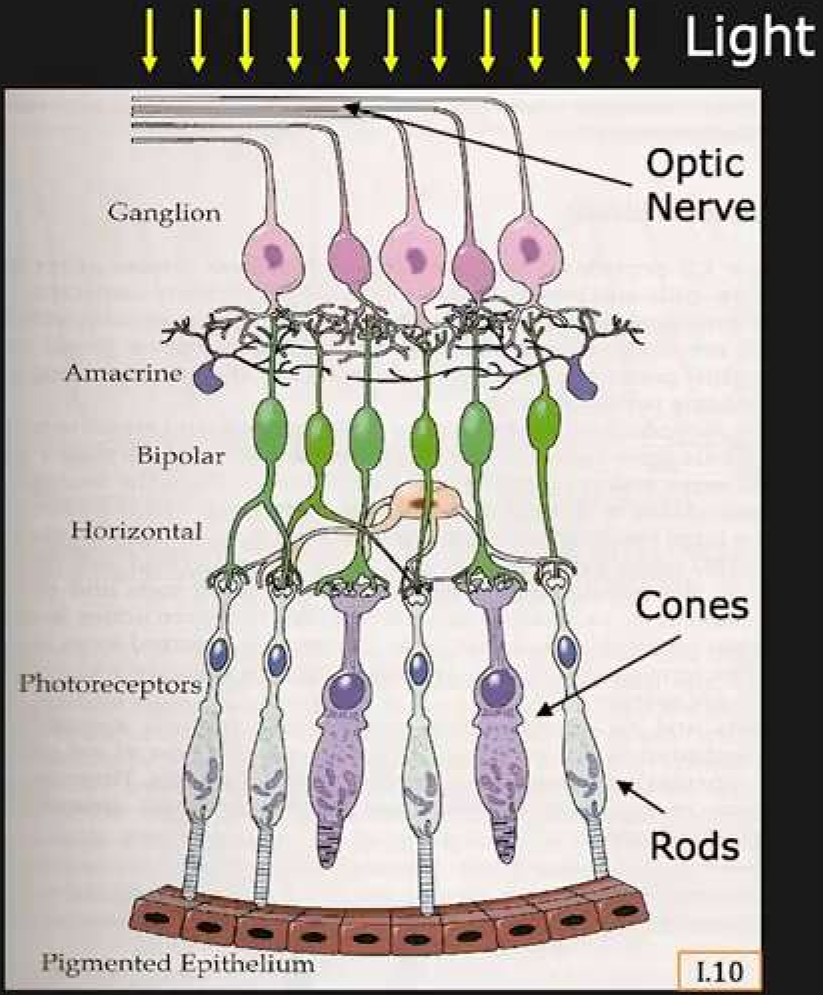
© 2020 Shree K. Nayar

The Human Eye



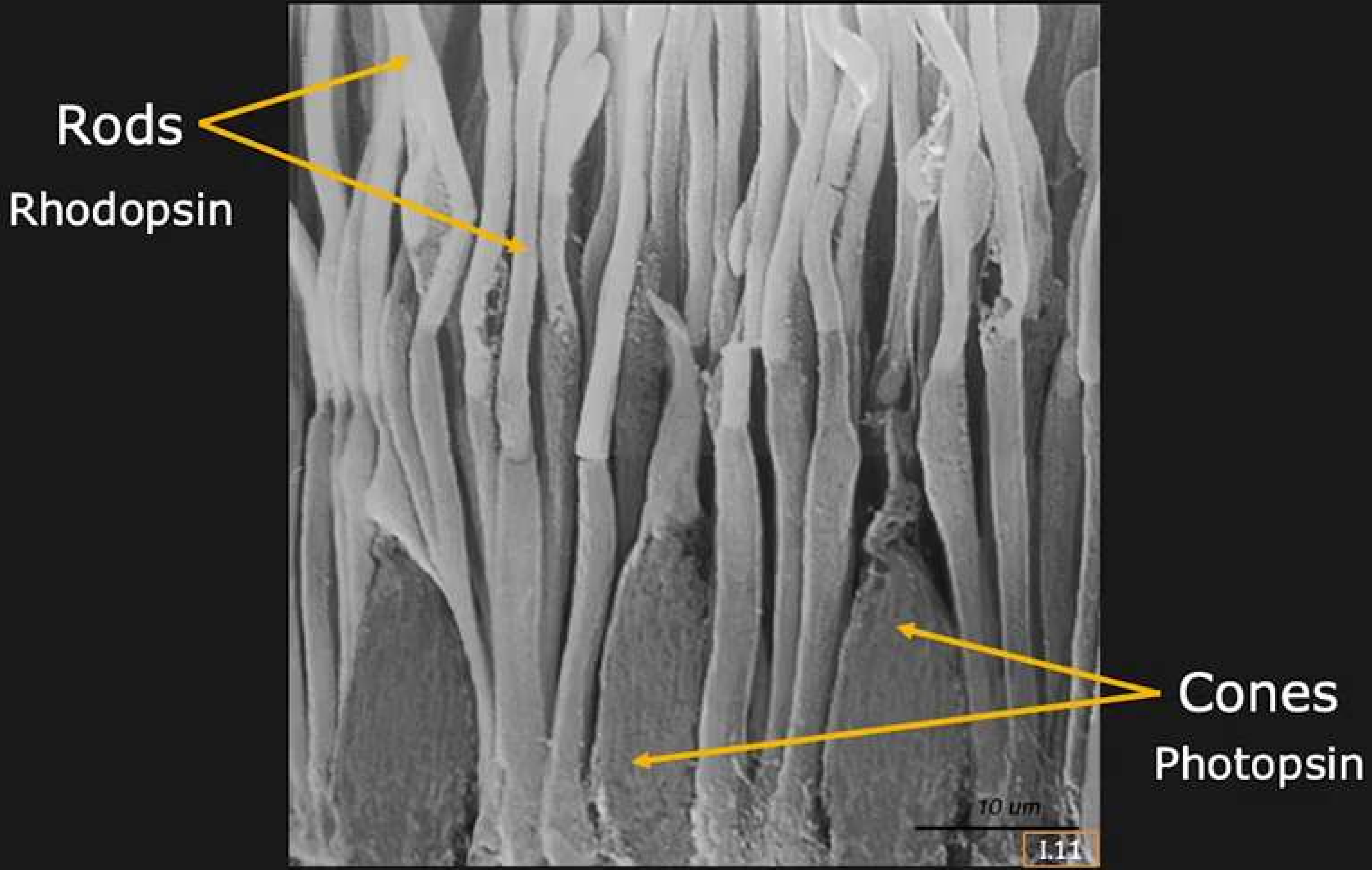
© 2020 Shree K. Nayar

A Cross-Section of the Retina



© 2020 Shree K. Nayar

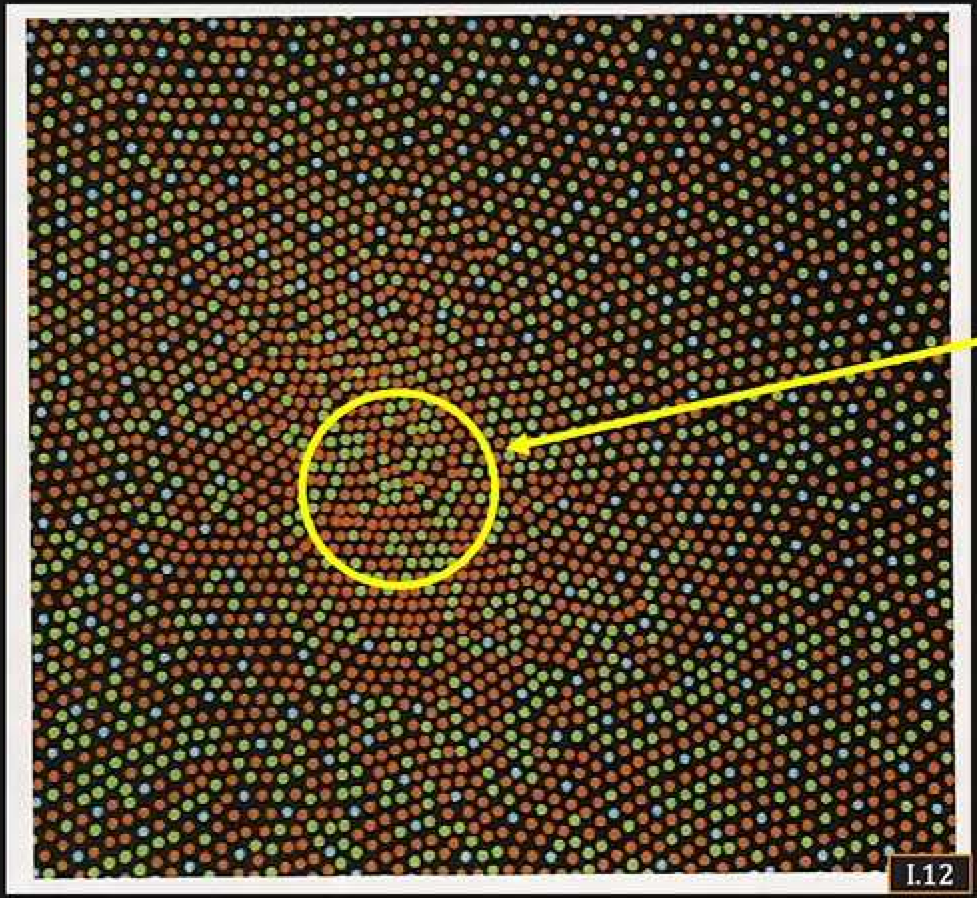
The Eye's Pixels



© 2020 Shree K. Nayar

Distribution of Cones in Human Retina

Three types of cones for sensing red, green, blue

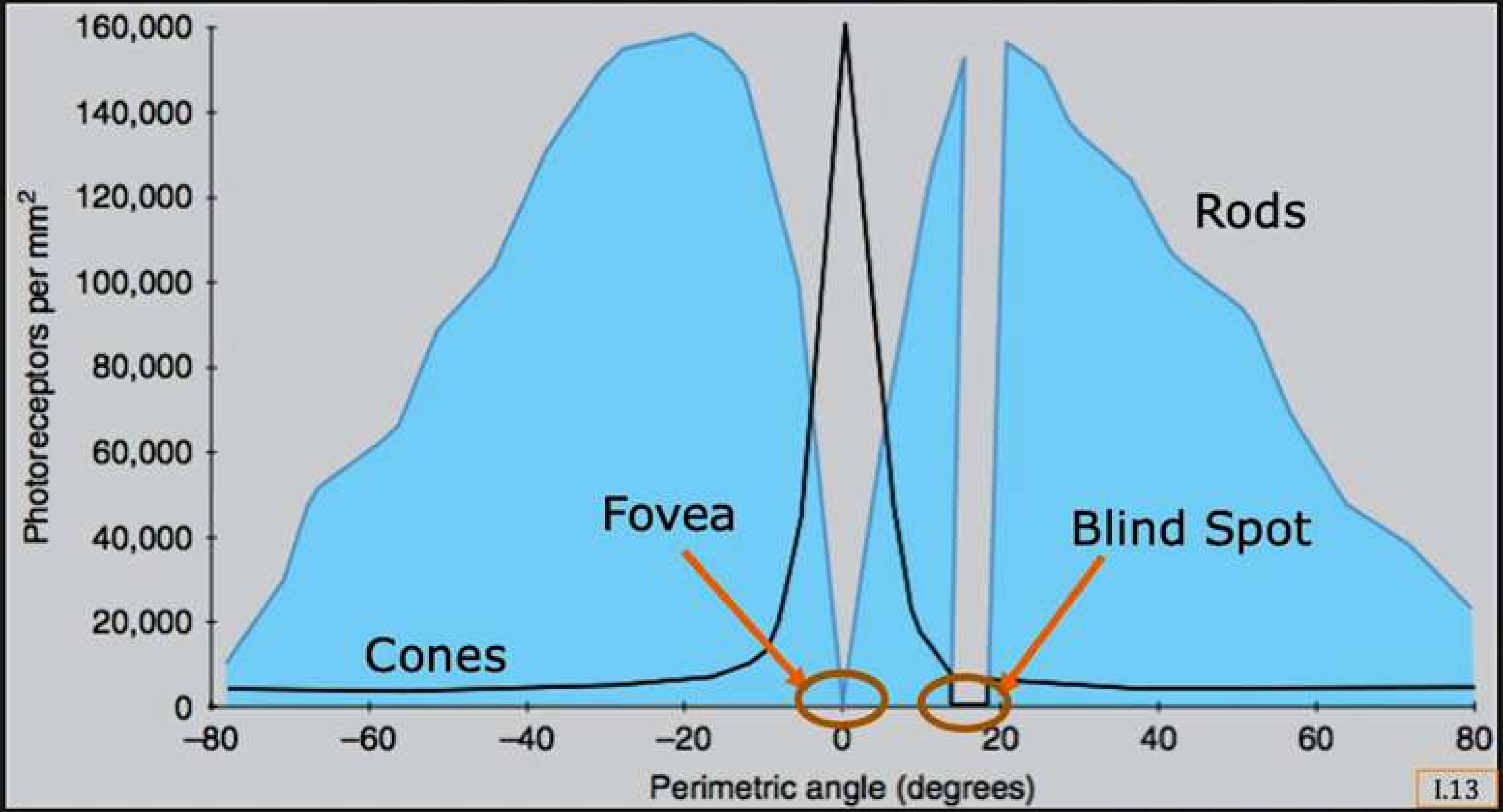


Fovea

© 2020 Shree K. Nayar

Resolution of Rods and Cones

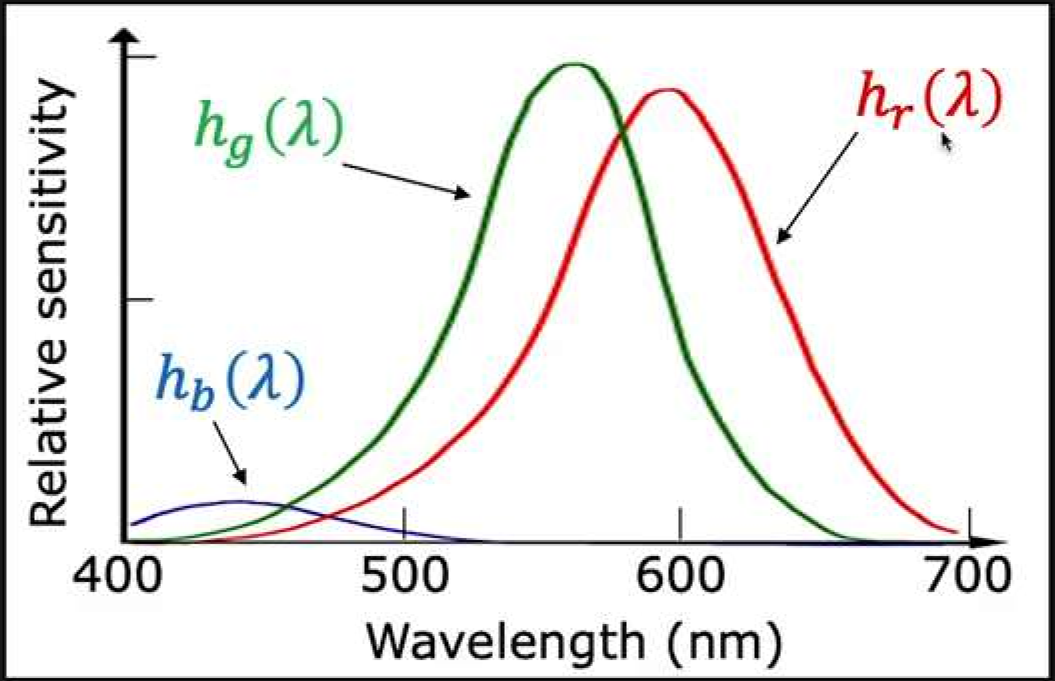
On average, ~120M rods and ~7M cones per retina.



© 2020 Shree K. Nayar

Spectral Responses of Cones

Three Types: Different Spectral Responses



Tristimulus Curves

$h_r(\lambda)$, $h_g(\lambda)$, $h_b(\lambda)$ are near
Red, Green, Blue regions of spectrum.

© 2020 Shree K. Nayar

Tristimulus Values

Three Intensities (R, G, B):

$$R = \int_{-\infty}^{\infty} h_r(\lambda) p(\lambda) d\lambda$$

$$G = \int_{-\infty}^{\infty} h_g(\lambda) p(\lambda) d\lambda$$

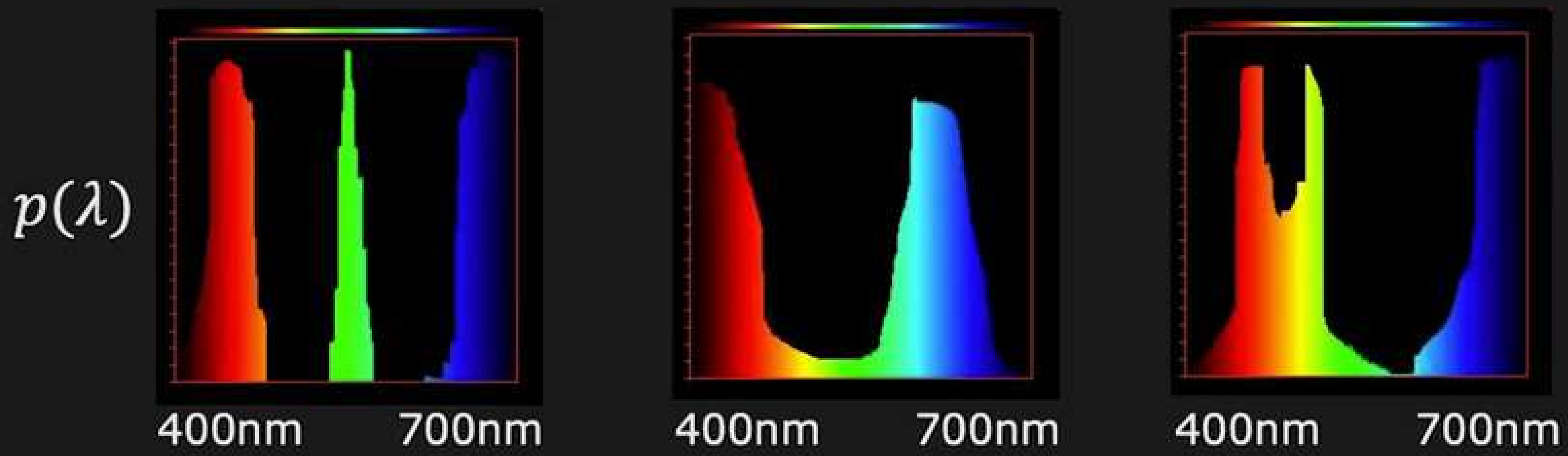
$$B = \int_{-\infty}^{\infty} h_b(\lambda) p(\lambda) d\lambda$$

© 2020 Shree K. Nayar

Metamers

Metamers: Different $p(\lambda)$ that produce same (R, G, B)

For example, different spectra:



Same perceived color:

$(R, G, B) = (115, 60, 108) =$ 

© 2020 Shree K. Nayar

The Mixing of Colors

Young's Experiment on Color Mixture

Human Sensation of nearly all colors can be produced using 3 wavelengths!

$$(\lambda_r, \lambda_g, \lambda_b) = (650, 530, 410)nm$$

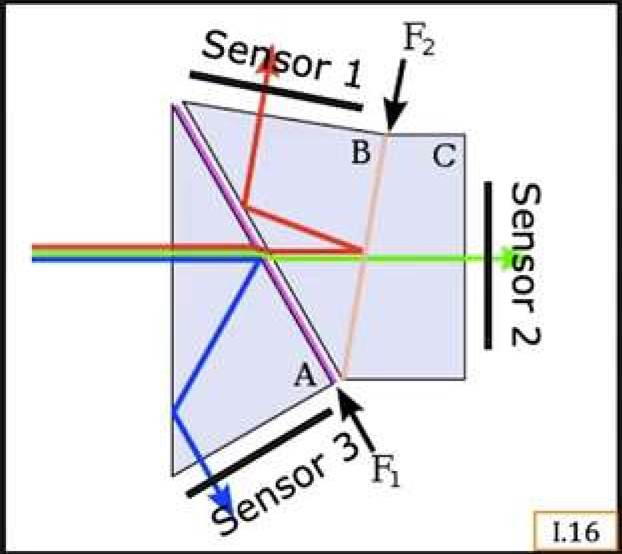
Hence, cameras and displays often use 3 filters:

(red, green, blue)

© 2020 Shree K. Nayar

Sensing Color using Dichroic Prism

Dichroic Prism

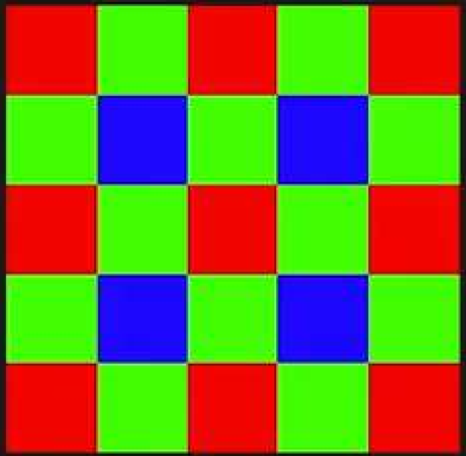


3-CCD Camera using Dichroic Prism

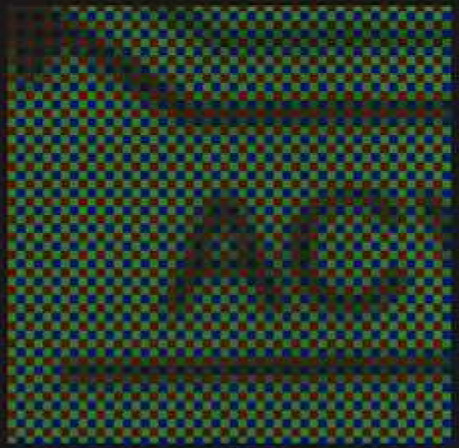
Each Sensor Detects One Color

© 2020 Shree K. Nayar

Sensing Color Using Color Mosaic



Bayer Pattern
(Color Filter Mosaic)



Raw Image

Interpolated Image

Color Filled in by Interpolation (Demosaicing)

© 2020 Shree K. Nayar

Camera Response Function $f(\cdot)$

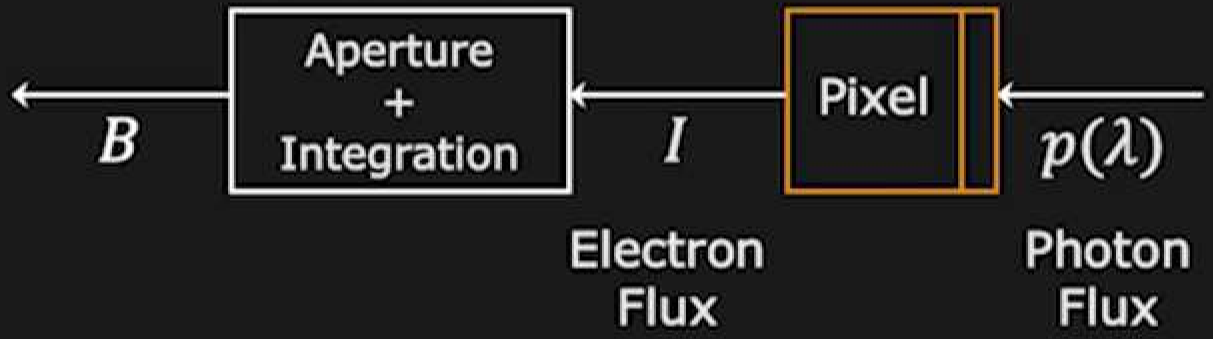
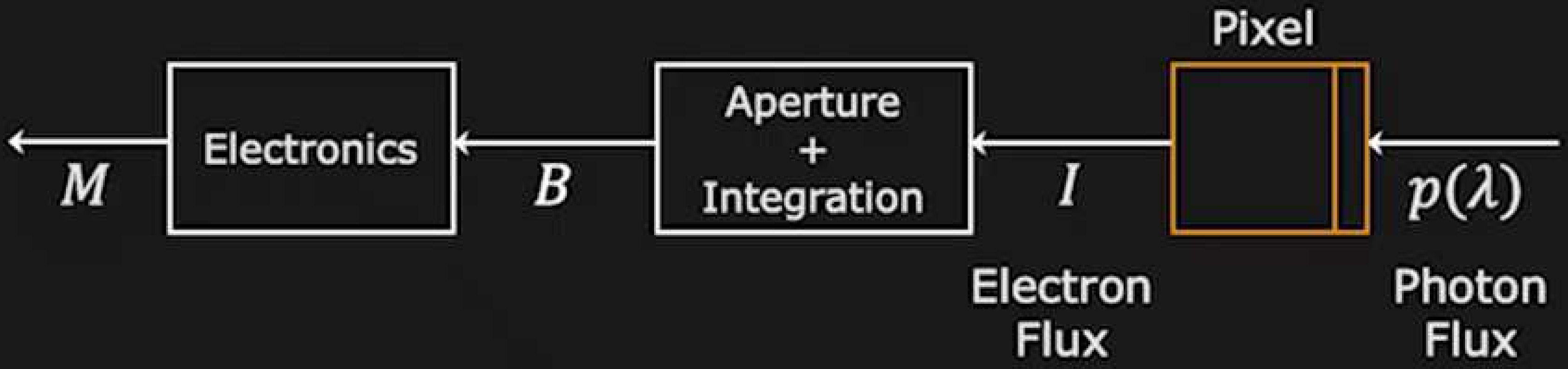


Image Brightness: $$\boxed{B = I \cdot \underbrace{\left(\frac{\pi d^2}{4}\right)}_{\text{Aperture Area (diameter } d\text{)}} \cdot \underbrace{t}_{\text{Integration Time}}}$$

© 2020 Shree K. Nayar

Camera Response Function $f(\cdot)$

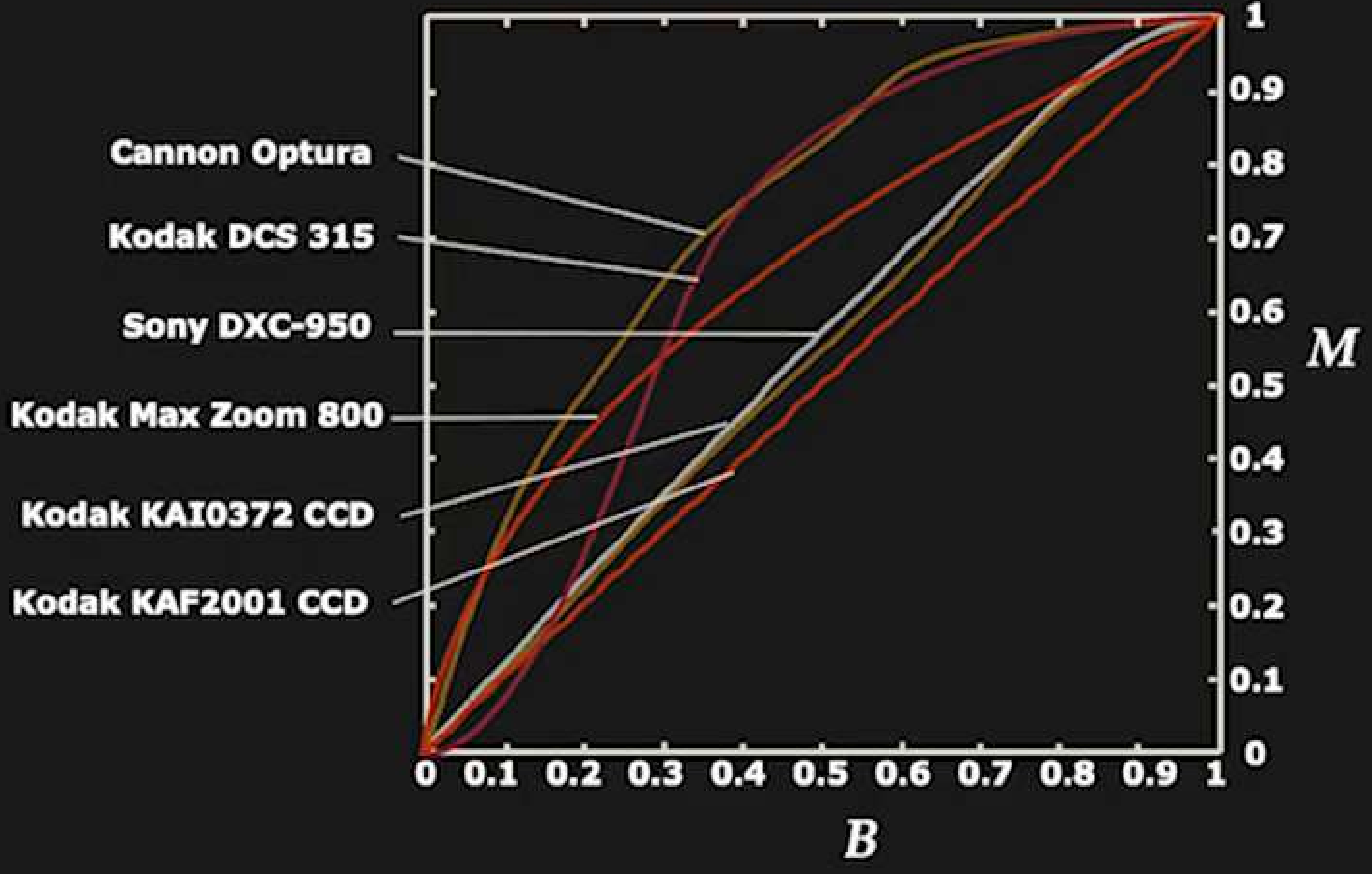


Measured Brightness: $\boxed{M = f(B) = f(I \cdot e)}$

where Exposure, $e = \left(\frac{\pi d^2}{4}\right) \cdot t$

© 2020 Shree K. Nayar

Camera Response Function $f(\cdot)$



"Gamma Curves"

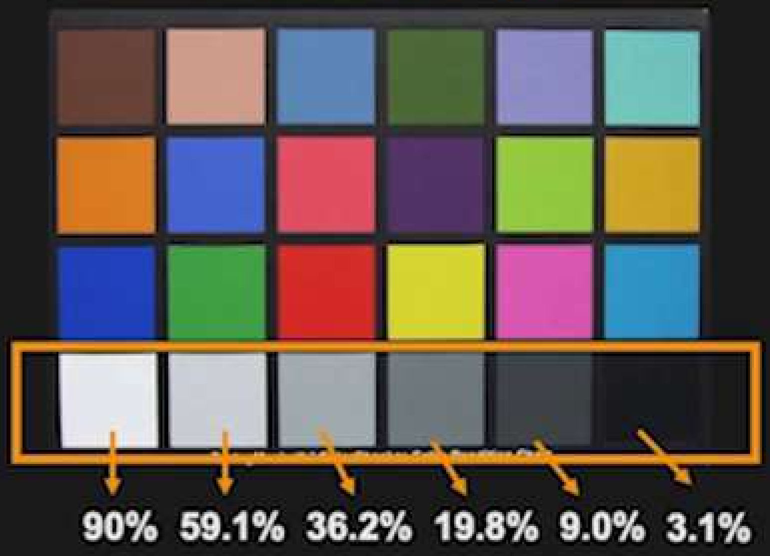
© 2020 Shree K. Nayar

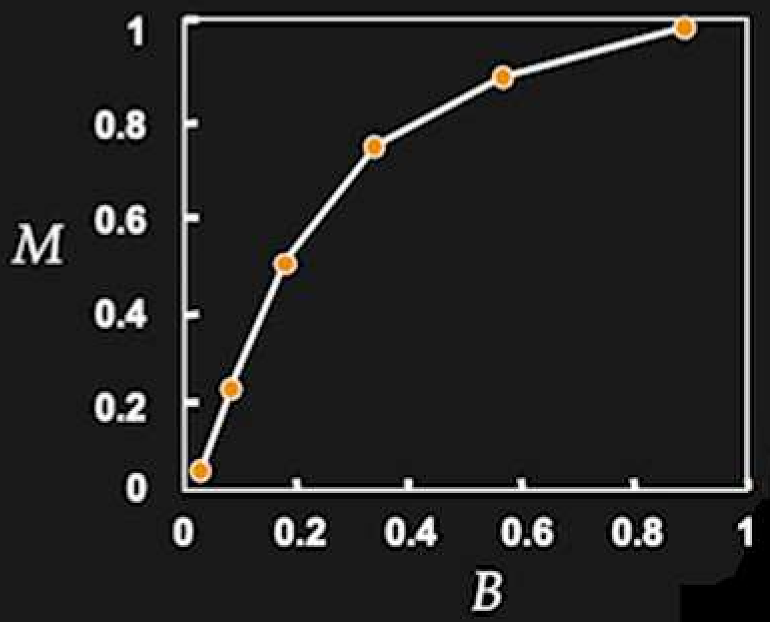
Radiometric Calibration: Finding $f(\cdot)$

Calibration using a chart:

1. Patches with known reflectance (when uniformly lit)
2. Fit linear segments or curve

Macbeth Chart





© 2020 Shree K. Nayar

High Dynamic Range: Multiple Exposures

Assume Camera Response $f(\cdot)$ is Linear

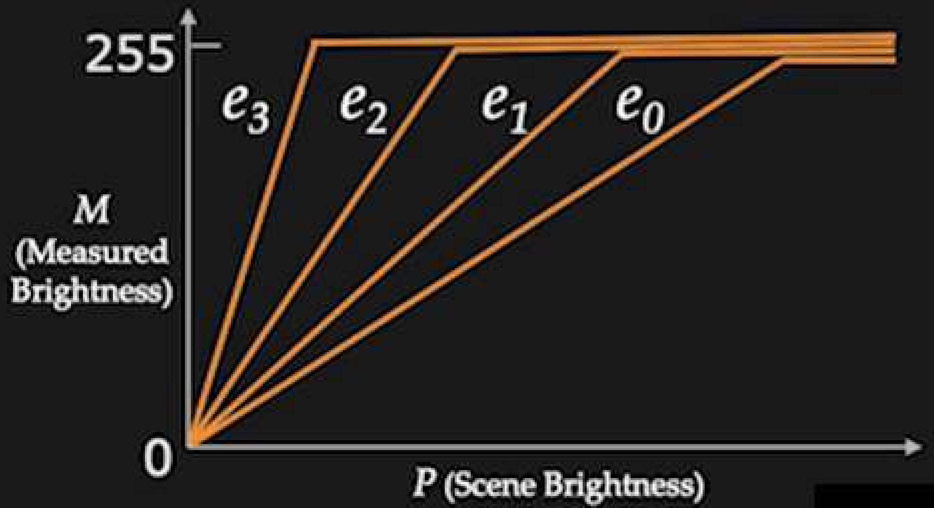
with e_0 $\quad e_1$ $\quad e_2$ $\quad e_3$

$$M_0 = \min(e_0 \cdot P, 255)$$

$$M_1 = \min(e_1 \cdot P, 255)$$

$$M_2 = \min(e_2 \cdot P, 255)$$

$$M_3 = \min(e_3 \cdot P, 255)$$

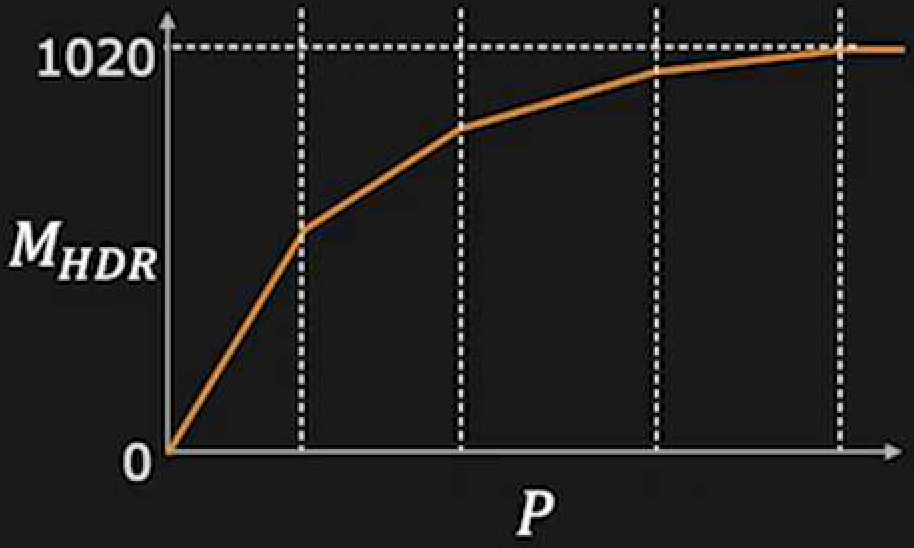


© 2020 Shree K. Nayar

High Dynamic Range: Multiple Exposures

Aggregate Image: $M_{HDR} = M_0 + M_1 + M_2 + M_3$

Camera Response $f(\cdot)$ for Aggregate Image:



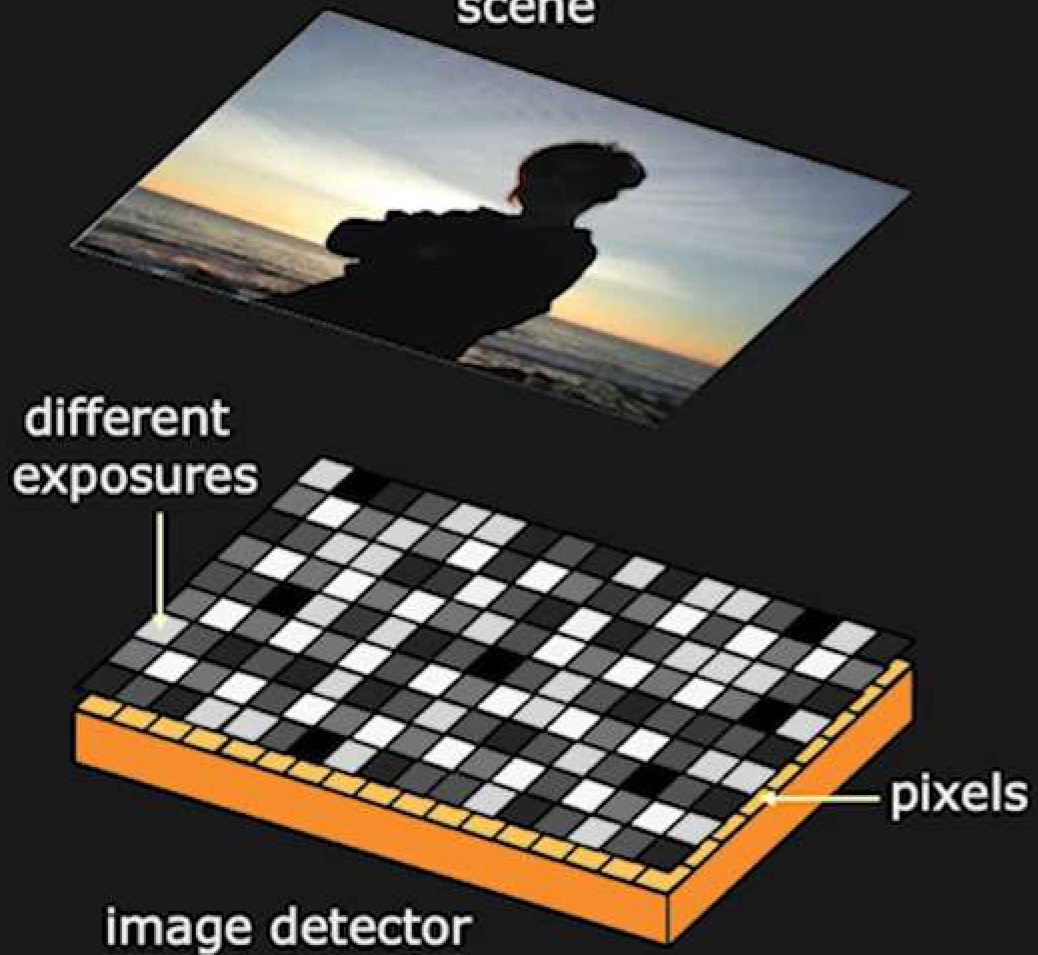
© 2020 Shree K. Nayar

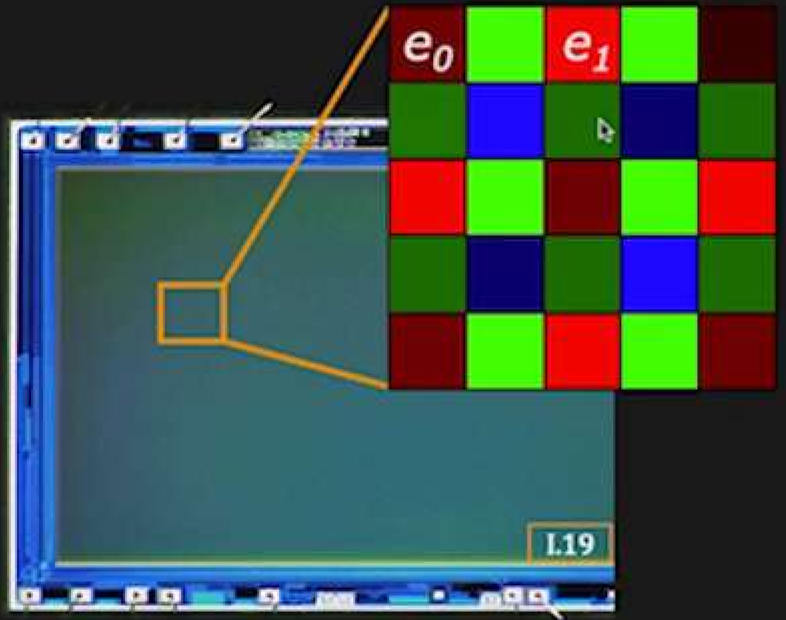
The Motion Problem

iPhone 4 HDR Mode

© 2020 Shree K. Nayar

Single Shot HDR Imaging





Assorted Pixels

© 2020 Shree K. Nayar

Good Old Camera

New HDR Camera

© 2020 Shree K. Nayar

© 2020 Shree K. Nayar

Binary Images

Binary Image: Can have only two values (0 or 1).

Simple to process and analyze.

© 2020 Shree K. Nayar

Making Binary Images

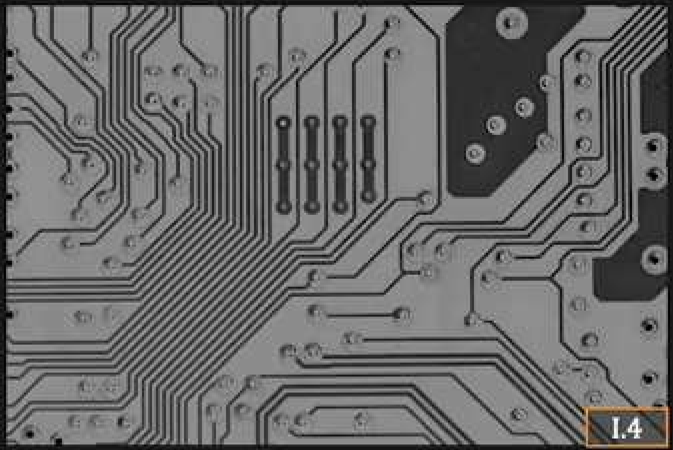
Binary Image $b(x,y)$: Usually obtained from Gray-level image $g(x,y)$ by Thresholding.

Characteristic Function:

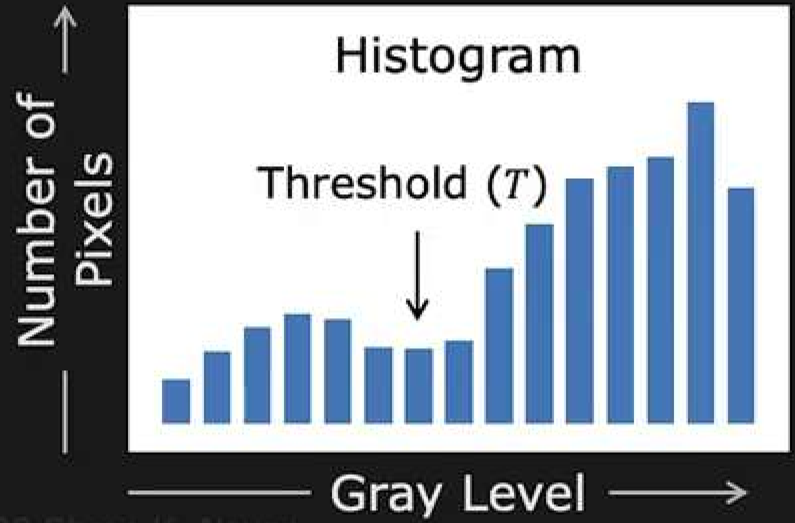
$$b(x,y) = \begin{cases} 0, & g(x,y) < T \\ 1, & g(x,y) \geq T \end{cases}$$

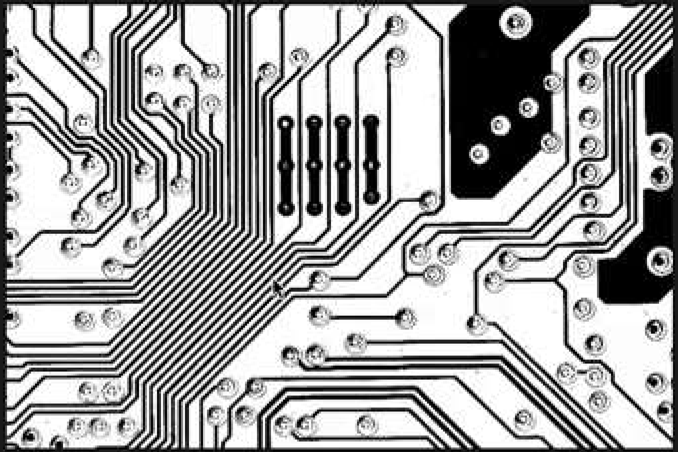
© 2020 Shree K. Nayar

Selecting a Threshold (T)



Gray Image $g(x, y)$

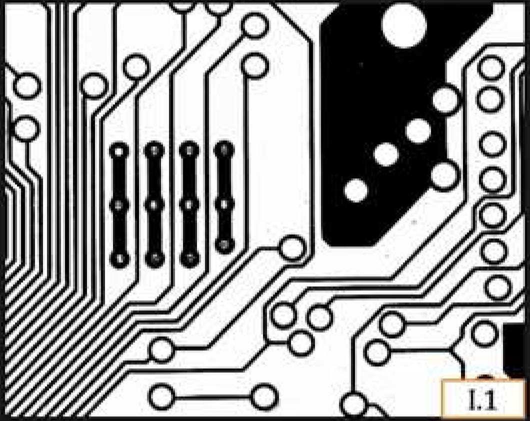




Binary Image $b(x, y)$

© 2020 Shree K. Nayar

Examples of Binary Images



© 2020 Shree K. Nayar

Capturing a Binary Image

Backlighting

© 2020 Shree K. Nayar

Binary Images

Binary Image: Can have only two values (0 or 1).
Simple to process and analyze.

Topics:

(1) Geometric Properties

(2) Segmenting Binary Images

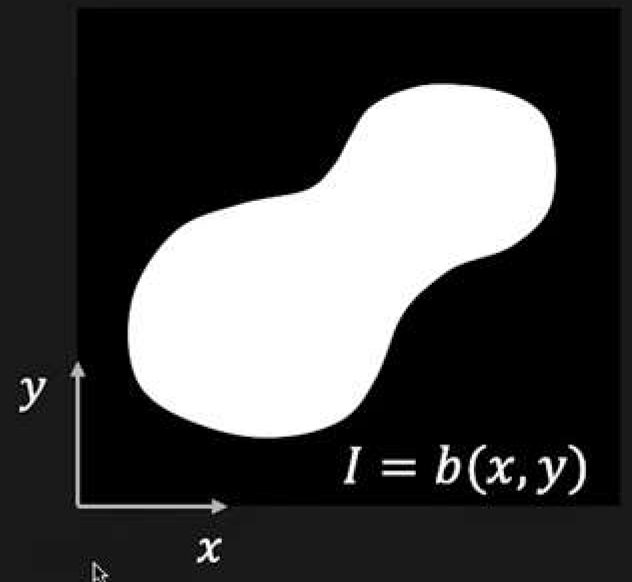
(3) Iterative Modification

© 2020 Shree K. Nayar

Geometric Properties of Binary Images

Assume:

- $b(x, y)$ is continuous
- Only one object

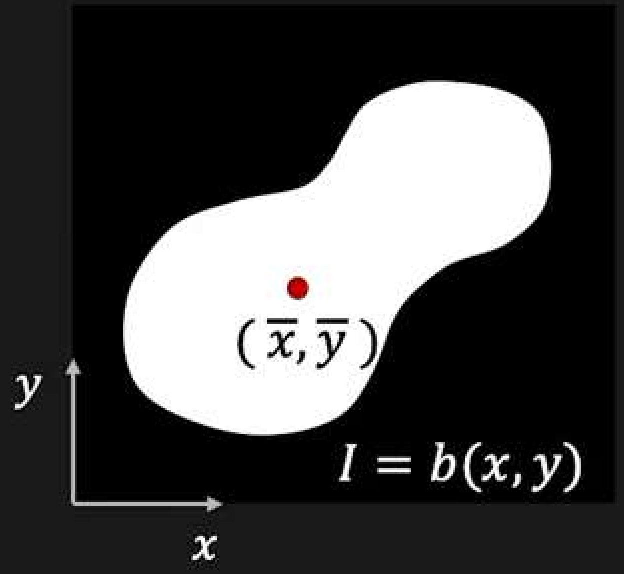


© 2020 Shree K. Nayar

Area and Position

Area: (Zeroth Moment)

$$A = \iint_I b(x,y)\,dx\,dy$$



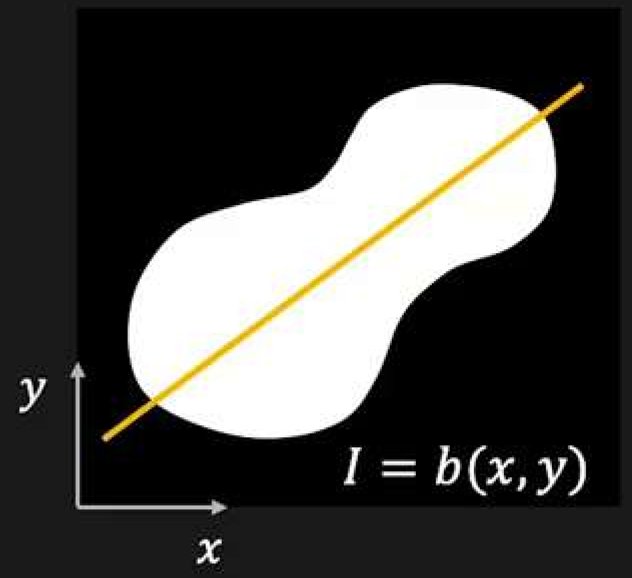
Position: Center of Area (First Moment)

$$\bar{x} = \frac{1}{A}\iint_I x\,b(x,y)\,dx\,dy \quad , \quad \bar{y} = \frac{1}{A}\iint_I y\,b(x,y)\,dx\,dy$$

© 2020 Shree K. Nayar

Orientation

Difficult to define!



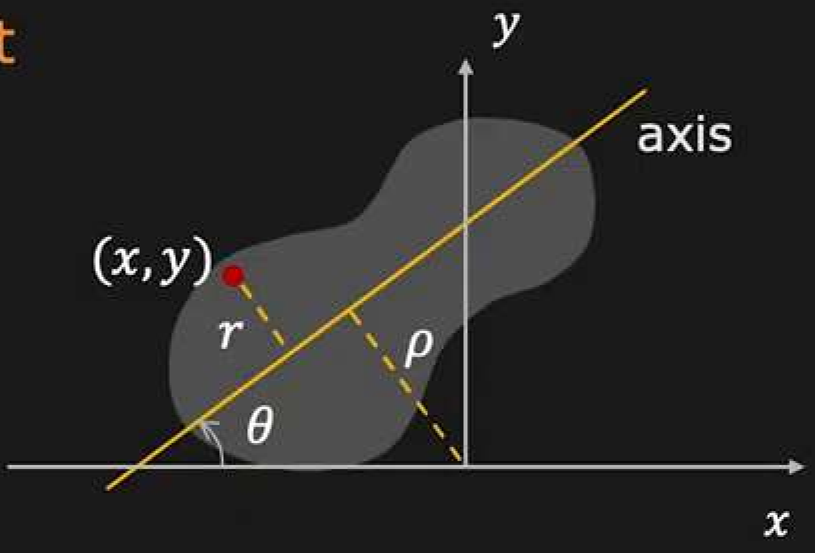
Use: Axis of Least Second Moment

© 2020 Shree K. Nayar

Orientation

Axis of Least Second Moment minimizes:

$$E = \iint_I r^2 \, b(x,y) \, dx \, dy$$



Which equation to use for axis?

$$y = mx + b \ ? \qquad -\infty \le m \le \infty$$

Use: $\boxed{x \sin\theta - y \cos\theta + \rho = 0}$ $\quad \rho$, θ are finite

Find ρ and θ that minimize E for given $b(x,y)$

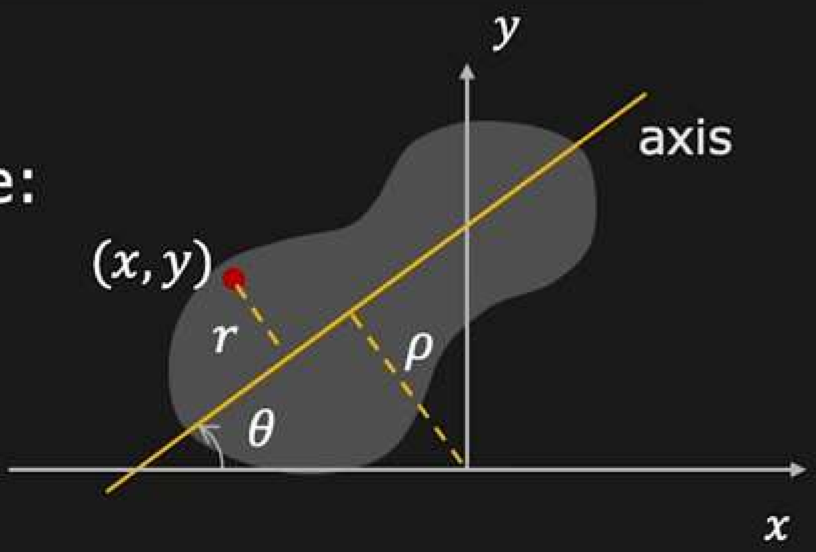
© 2020 Shree K. Nayar

Distance Between Point and Line

Given a line $ax + by + c = 0$

Distance of point (x, y) from line:

$$r = \left|\frac{ax + by + c}{\sqrt{a^2 + b^2}}\right|$$



Similarly, given axis $x \sin\theta - y\cos\theta + \rho = 0$

Distance of point (x, y) from axis:

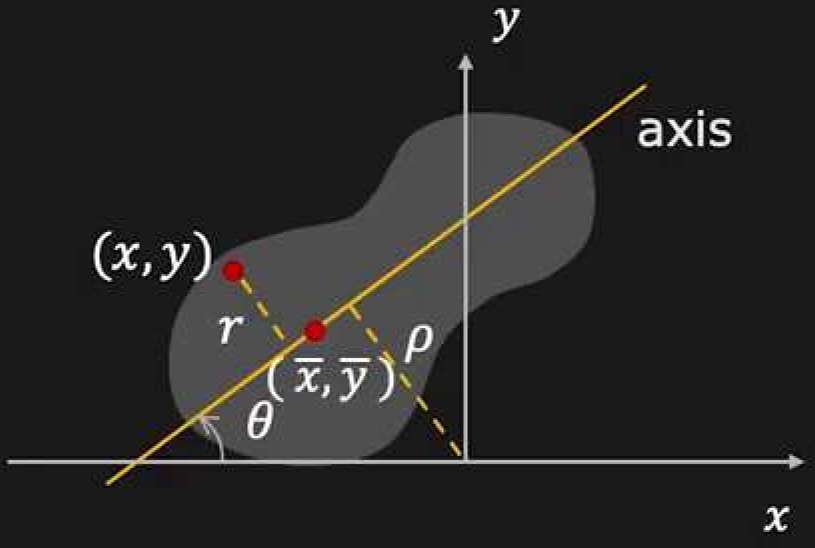
$$r = \left|\frac{x \sin\theta - y\cos\theta + \rho}{\sqrt{\sin^2\theta + \cos^2\theta}}\right|$$

$$\boxed{r = |x \sin\theta - y\cos\theta + \rho|}$$

Minimizing Second Moment

Axis of Least Second Moment minimizes:

$$E = \iint_I r^2 \, b(x,y) \, dx \, dy$$



So, minimize:

$$E = \iint_I (x \sin\theta - y \cos\theta + \rho)^2 \, b(x,y) \, dx \, dy$$

Using $\dfrac{\partial E}{\partial \rho} = 0$ we get: $A(\overline{x} \sin\theta - \overline{y} \cos\theta + \rho) = 0$

Axis passes through center $(\overline{x},\overline{y})$!

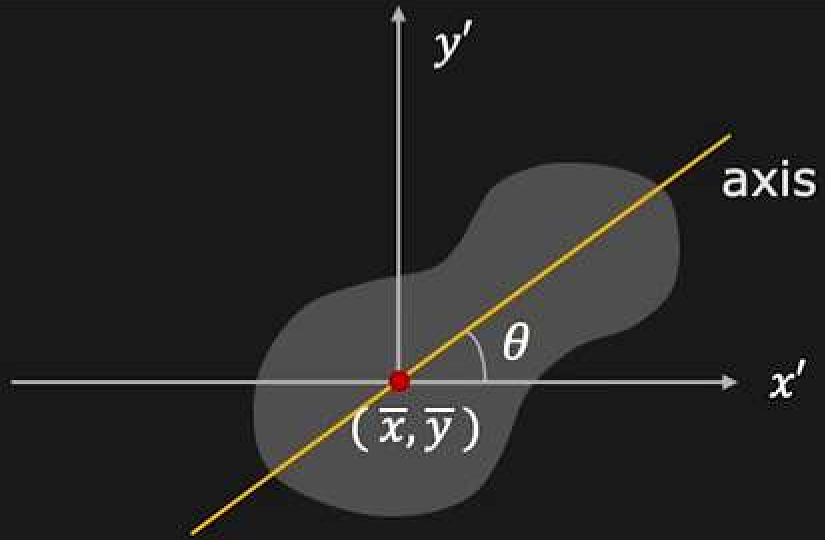
© 2020 Shree K. Nayar

Shift the Coordinate System

Change coordinates:

$$x' = x - \bar{x},\ y' = y - \bar{y}$$

$$x \sin\theta - y\cos\theta + \rho$$
$$= x' \sin\theta - y' \cos\theta$$



Therefore, we can rewrite E as:

$$\boxed{E = a\sin^2\theta - b\sin\theta\cos\theta + c\cos^2\theta}$$

where:
$$\begin{cases} a = \iint_{I'} (x')^2\, b(x,y)\, dx'\, dy' \\ c = \iint_{I'} (y')^2\, b(x,y)\, dx'\, dy' \end{cases}$$

$$b = 2\iint_{I'} (x'y')\, b(x,y)\, dx'\, dy'$$

(a, b, c are easy to compute)

© 2020 Shree K. Nayar

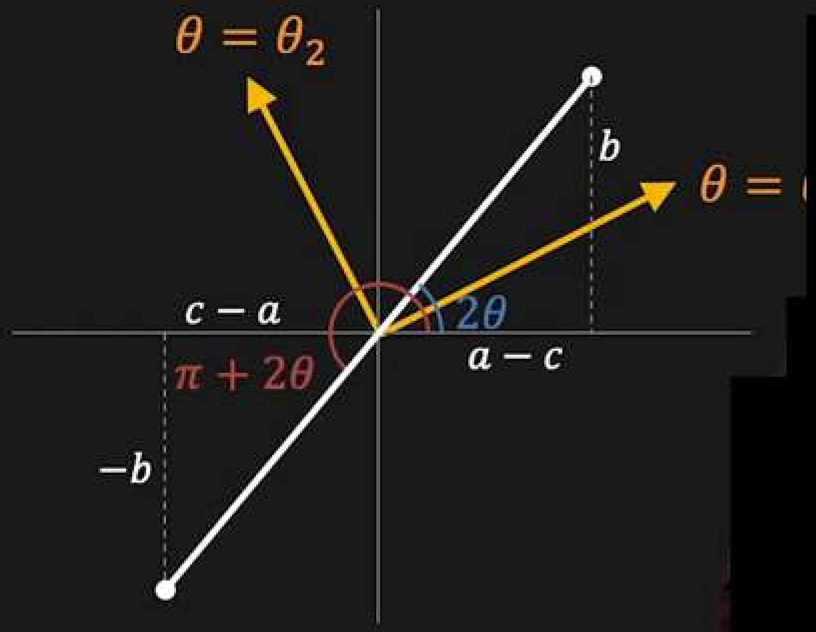
Finally, Minimize E

Using $\frac{dE}{d\theta} = (a-c)\sin 2\theta - b\cos 2\theta = 0$ we get: $\boxed{\tan 2\theta = \frac{b}{a-c}}$

We know that: $\tan 2\theta = \tan(2\theta + \pi) = \frac{-b}{c-a}$

θ has two solutions.

1. $\theta = \theta_1$
2. $\theta = \theta_2 = \theta_1 + \frac{\pi}{2}$



One gives Minimum of E
and the other Maximum of E

© 2020 Shree K. Nayar

Which One To Use?

Using second derivative test:

If $\dfrac{d^2E}{d\theta^2} = (a-c)\cos 2\theta + b\sin 2\theta \begin{cases} > 0 \text{ then Minimum} \\ < 0 \text{ then Maximum} \end{cases}$

Substituting $\cos 2\theta_1$, $\sin 2\theta_1$, $\cos 2\theta_2$ and $\sin 2\theta_2$:

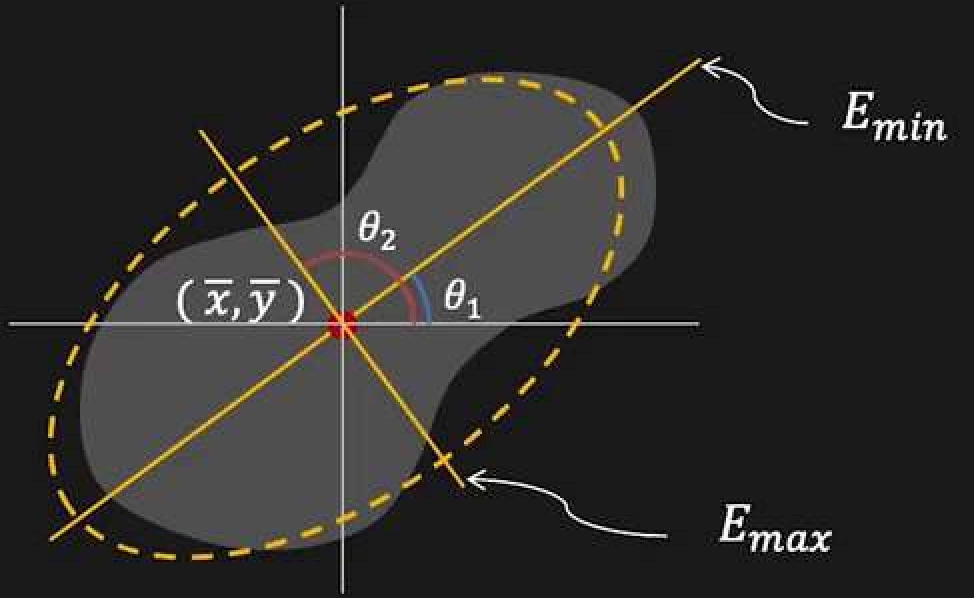
$$\frac{d^2E}{d\theta^2}(\theta_1) > 0 \quad \text{and} \quad \frac{d^2E}{d\theta^2}(\theta_2) < 0$$

Therefore,

Orientation: $\boxed{\theta = \theta_1 = \dfrac{atan2(b, a-c)}{2}}$

© 2020 Shree K. Nayar

Roundedness

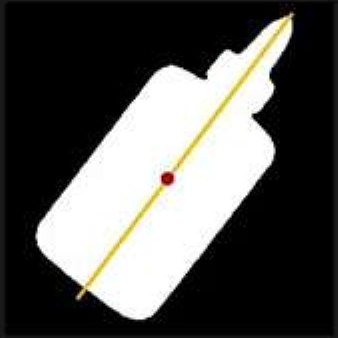
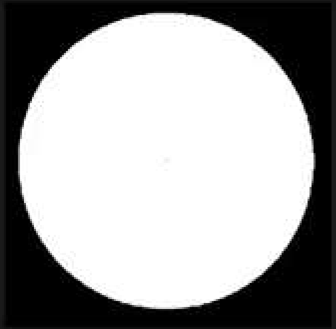
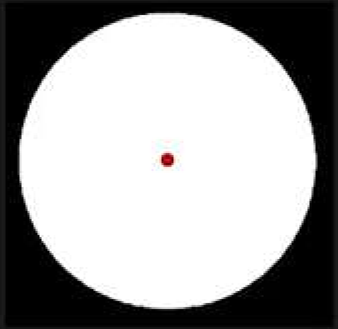


$$\boxed{\text{Roundedness} = \frac{E_{min}}{E_{max}}}$$

where: $E_{min} = E(\theta_1)$ and $E_{max} = E(\theta_2)$

© 2020 Shree K. Nayar

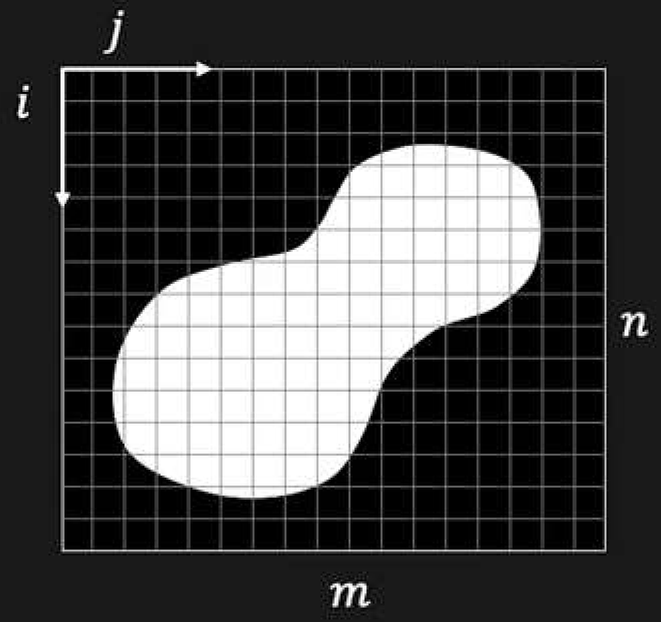
Examples

Gray Image	Binary Image	Orientation	Roundedness
			0.19
			0.49
			1.0

© 2020 Shree K. Nayar

Discrete Binary Images

b_{ij}: Value at cell (pixel) in row i and column j.
Assume pixel area = 1.



Area:

$$A = \sum_{i=1}^{n} \sum_{j=1}^{m} b_{ij}$$

Position: Center of Area (First Moment)

$$\bar{x} = \frac{1}{A} \sum_{i=1}^{n} \sum_{j=1}^{m} i b_{ij} \qquad \bar{y} = \frac{1}{A} \sum_{i=1}^{n} \sum_{j=1}^{m} j b_{ij}$$

© 2020 Shree K. Nayar

Discrete Binary Images

Second Moments:

$$a' = \sum_{i=1}^{n}\sum_{j=1}^{m} i^2 b_{ij} \qquad b' = 2\sum_{i=1}^{n}\sum_{j=1}^{m} ij b_{ij} \qquad c' = \sum_{i=1}^{n}\sum_{j=1}^{m} j^2 b_{ij}$$

Note: a', b', c' are second moments w.r.t origin.
a, b, c (w.r.t. center) can be found from a', b', c', $\bar{x}$, $\bar{y}$, A

Hint: Expand $a = \sum_{i=1}^{n}\sum_{j=1}^{m}(i - \bar{x})^2 b_{ij}$ and represent in terms of a', $\bar{x}$, A.

© 2020 Shree K. Nayar

Multiple Objects

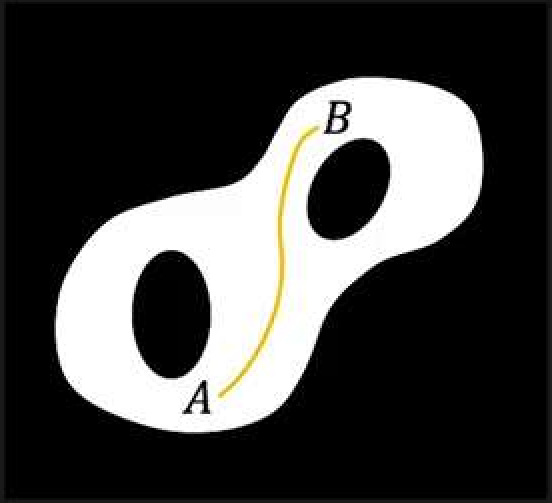
Need to Segment image into separate Components

Non-Trivial!

© 2020 Shree K. Nayar

Connected Component

Maximal Set of Connected Points



A and B are connected if path exists between A and B along which $b(x,y)$ is constant.

© 2020 Shree K. Nayar

Connected Component Labeling

Region Growing Algorithm

(a) Find Unlabeled "Seed" point with $b = 1$.
If not found, Terminate.

(b) Assign New Label to seed point

(c) Assign Same Label to its Neighbors with $b = 1$

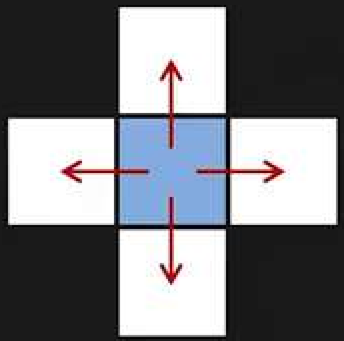
(d) Assign Same Label to Neighbors of Neighbors with $b = 1$. Repeat until no more Unlabeled Neighbors with b=1.

(e) Go to (a)

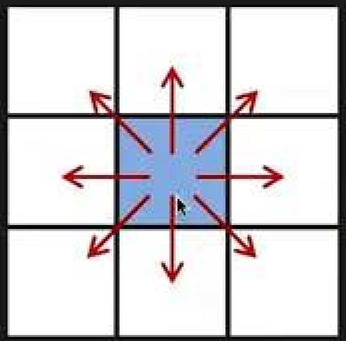
© 2020 Shree K. Nayar

What do we mean by Neighbors?

Connectedness



4-Connectedness
4-C



8-Connectedness
8-C

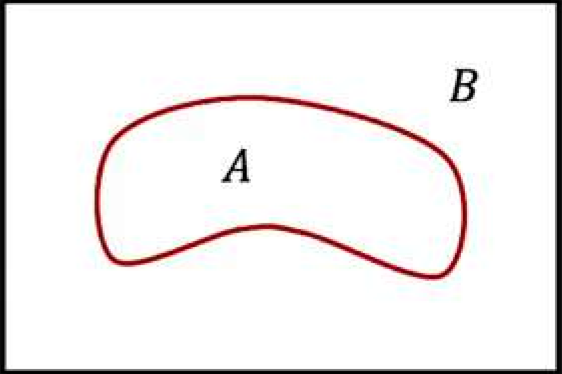
Neither is Perfect!

© 2020 Shree K. Nayar

Connectedness

Jordan's Curve Theorem

Closed curve
→ 2 Connected Regions



Consider

0	1	0
1	0	1
0	1	0

B1	O1	B1
O4	B2	O2
B1	O3	B1

4-C
Hole without a closed loop!

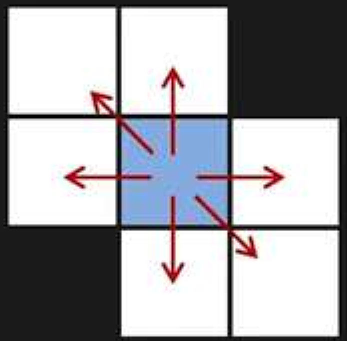
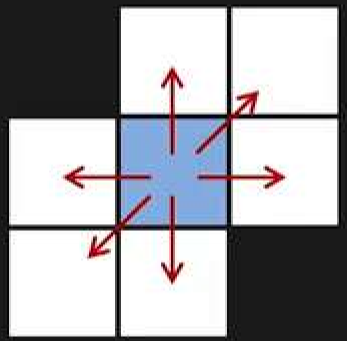
B	O	B
O	B	O
B	O	B

8-C
Connected background with a closed loop!

© 2020 Shree K. Nayar

Solution to Neighborhood Problem

Introduce Asymmetry

(a)  or (b) 

6-Connectedness
6-C

6-Connectedness
6-C

Using (a):

0	1	0
1	0	1
0	1	0

$\Rightarrow$

B	O2	B
O1	B	O2
B	O1	B

Two Separate
Line Segment[redacted]

© 2020 Shree K. Nayar

Hexagonal Tessellation

Above asymmetry makes a Square Grid behave like a Hexagonal Grid

© 2020 Shree K. Nayar

Sequential Labeling Algorithm

Raster Scanning

We want to label A.
B, C, D are already labeled.

© 2020 Shree K. Nayar

Sequential Labeling Algorithm

X	X
X	0

→ label(A) = "background"

0	0
0	1

→ label(A) = new label

D	X
X	1

→ label(A) = label(D)

0	0
C	1

→ label(A) = label(C)

0	B
0	1

→ label(A) = label(B)

0	B
C	1

→ If
label(B) = label(C)
then,
label(A) = label(C)

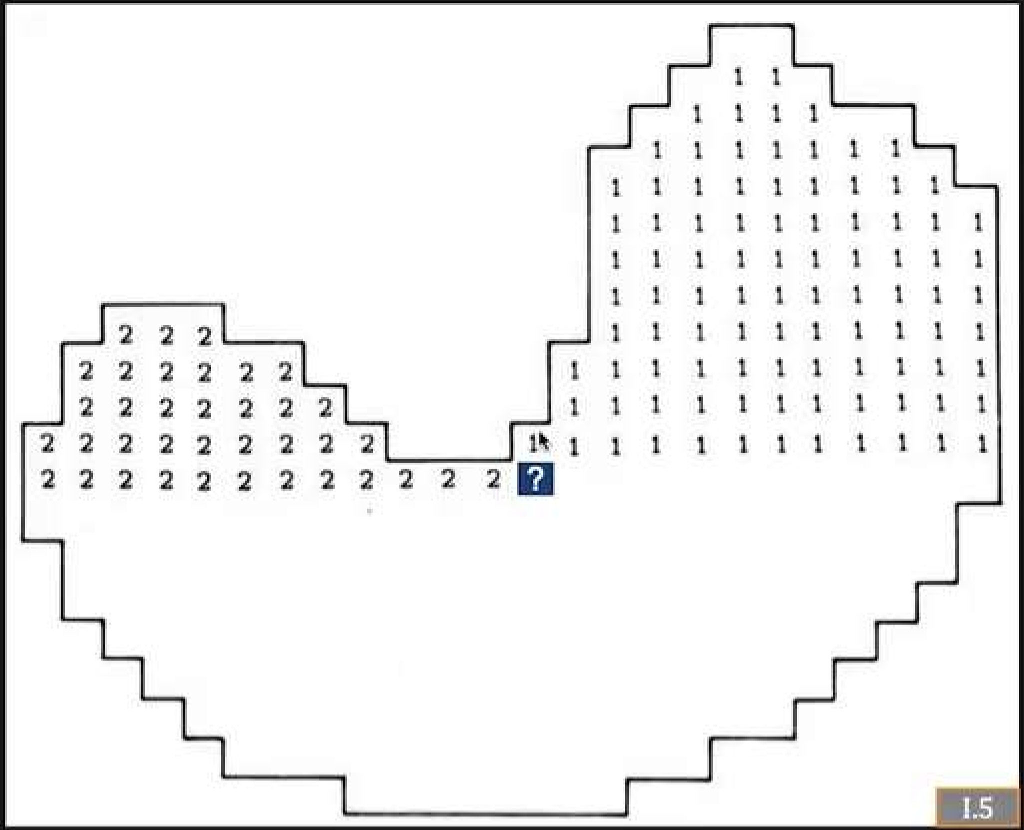
X: Value does not matter (Can be 0 or 1)

© 2020

Sequential Labeling Algorithm

0	B
C	1

→ What if label(B) not equal to label(C)?



© 2020 Shree K. Nayar

Sequential Labeling Algorithm

0	B
C	1

→ What if label(B) not equal to label(C)?

Solution: Create Equivalence Table

- Note down that label(B) ≡ label(C)
- Assign label(A) = label(B)

$2 \equiv 1$
$7 \equiv 3, 6, 4$
$8 \equiv 5$
$\vdots$

© 2020 Shree K. Nayar

Euler Number (E)

No. of Bodies (B) – No. of Holes (H)

Letter B: $E_b = -1$

Letter i: $E_i = 2$

Letter n: $E_n = 1$

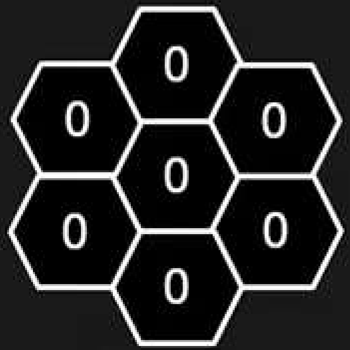
Image : $E = 4$

$$E_{image} = \sum E_{non-overlapping\ regions}$$

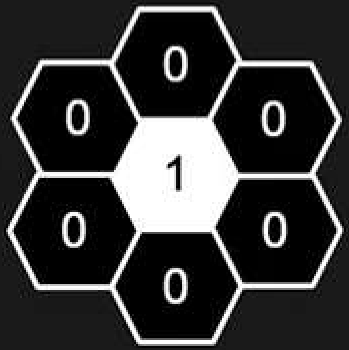
© 2020 Shree K. Nayar

Euler Differential (E^*)

Change in the Euler number of the image.



$E = 0$



$E = 1$

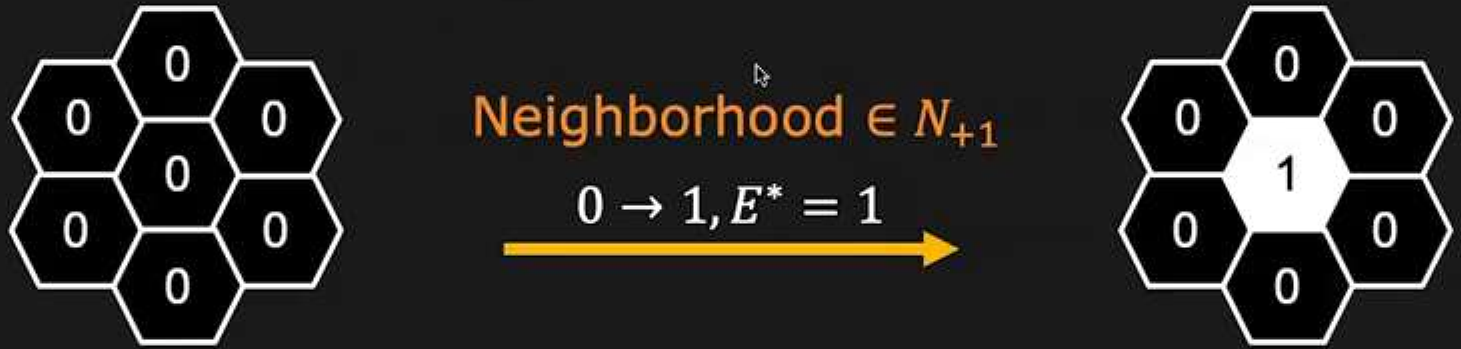
Euler Differential: $E^* = 1$

© 2020 Shree K. Nayar

Neighborhood Sets Based on E^*

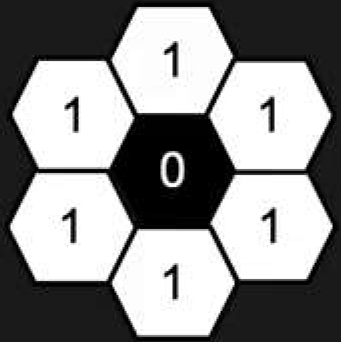
Each pixel has $2^6 = 64$ possible neighborhoods

Neighborhood patterns are classified based on the Euler Differential they generate, assuming the center pixel goes from 0 to 1.



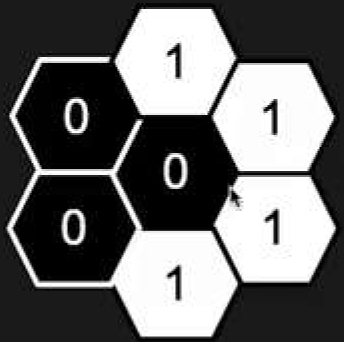
© 2020 Shree K. Nayar

Neighborhood Sets Based on E^*



Neighborhood $\in N_{+1}$

$0 \rightarrow 1, E^* = 1$

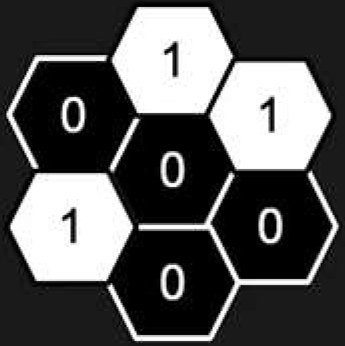


Neighborhood $\in N_0$

$0 \rightarrow 1, E^* = 0$

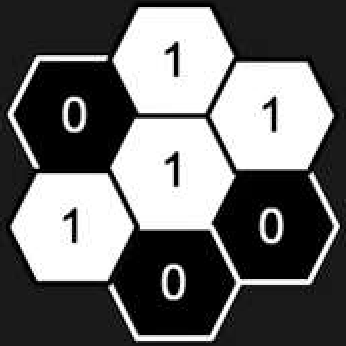
© 2020 Shree K. Nayar

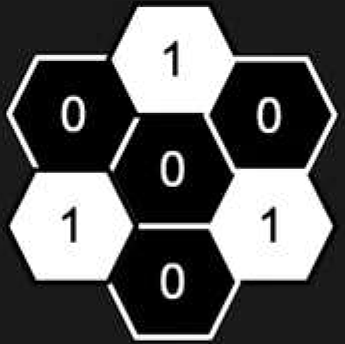
Neighborhood Sets Based on E^*



Neighborhood $\in N_{-1}$

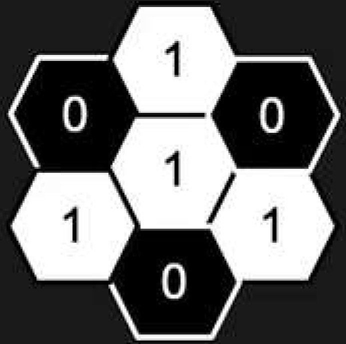
$0 \rightarrow 1, E^* = -1$





Neighborhood $\in N_{-2}$

$0 \rightarrow 1, E^* = -2$

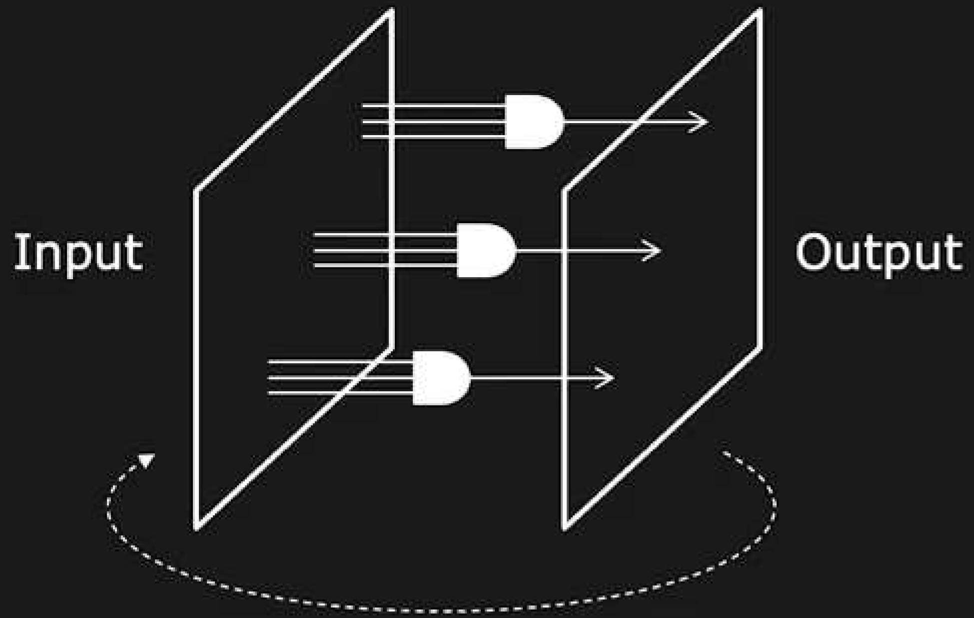


Only 4 possible neighborhood types: N_{+1}, N_0, N_{-1}, N_{-2}

© 2020 Shree K. Naya

Iterative Neighborhood Operations

Incrementally apply neighborhood operations on images



Conservative Operations do not change the Euler number of the image.

© 2020 Shree K. Nayar

Notation for Iterative Modification

Specify Neighborhood Set, S. Ex: S can be N_{+1}, N_0, N_{-1} or N_{-2} or a combination of these.

Consider pixel (i, j). Let:

- $a_{ij} = 1$ if Neighborhood of $(i, j) \in S$ else 0
- b_{ij} = current value of pixel (i, j)
- c_{ij} = new value of pixel (i, j)

a_{ij}	b_{ij}	$\Rightarrow$	c_{ij}
0	0		?
0	1		?
1	0		?
1	1		?

$2^4 = 16$ algorithms

© 2020 Shree K. Nayar

Iterative Modification Algorithms

Specify Neighborhood Set S and apply one of the 16 algorithms to each pixel.

← 16 algorithms →

a_{ij}	b_{ij}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Growing Objects: $S \in N_0$ and $Algorithm\ 7$

Thinning Objects: $S \in N_0$ and $Algorithm\ 4$

© 2020 Shree K. Nayar

Finding Skeletons

Thinning without changing the Euler number

$S \in N_0$ and $Algorithm$ 4

© 2020 Shree K. Nayar

Finding Skeletons

Thinning without changing the Euler number

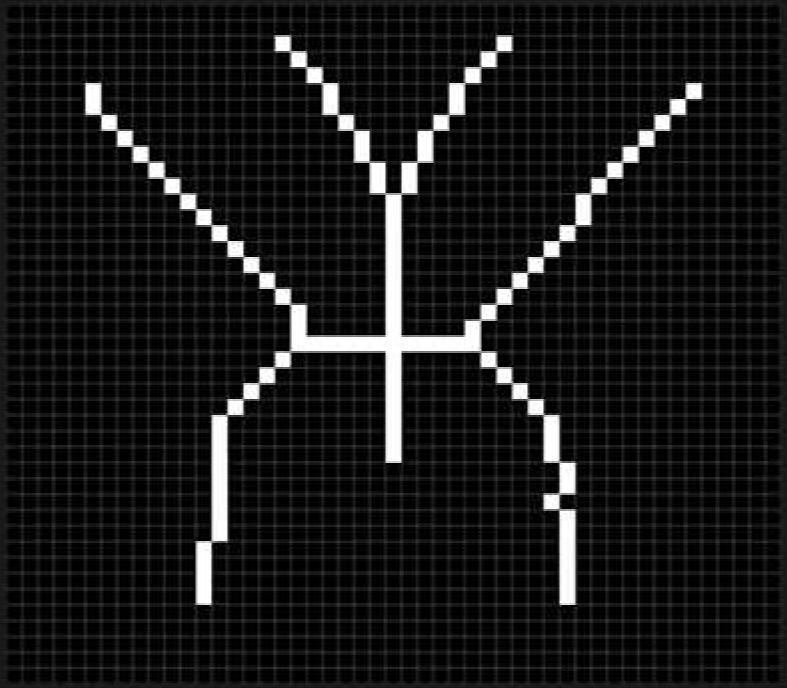
$S \in N_0$ and $Algorithm\ 4$

© 2020 Shree K. Nayar

Finding Skeletons

Thinning without changing the Euler number

$S \in N_0$ and $Algorithm\ 4$

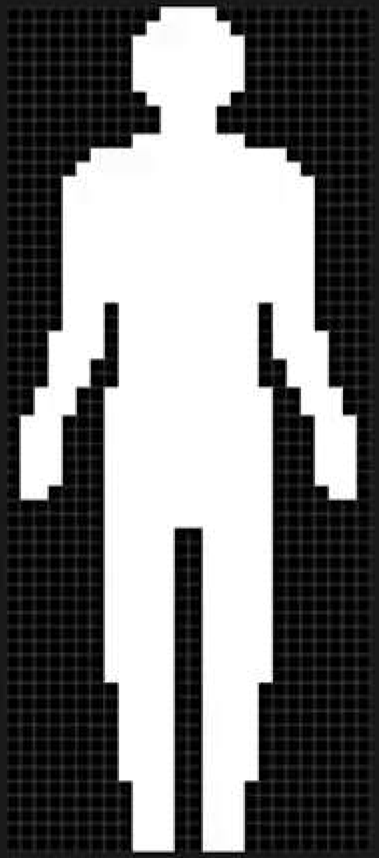


© 2020 Shree K. Nayar

Finding Skeletons

Thinning without changing the Euler number

$S \in N_0$ and $Algorithm\ 4$



© 2020 Shree K. Nayar

Finding Skeletons

Thinning without changing the Euler number

$S \in N_0$ and $Algorithm\ 4$

© 2020 Shree K. Nayar

Image Processing I

Transform image to new one that is clearer or easier to analyze.

Topics:

(1) Pixel Processing

(2) LSIS and Convolution

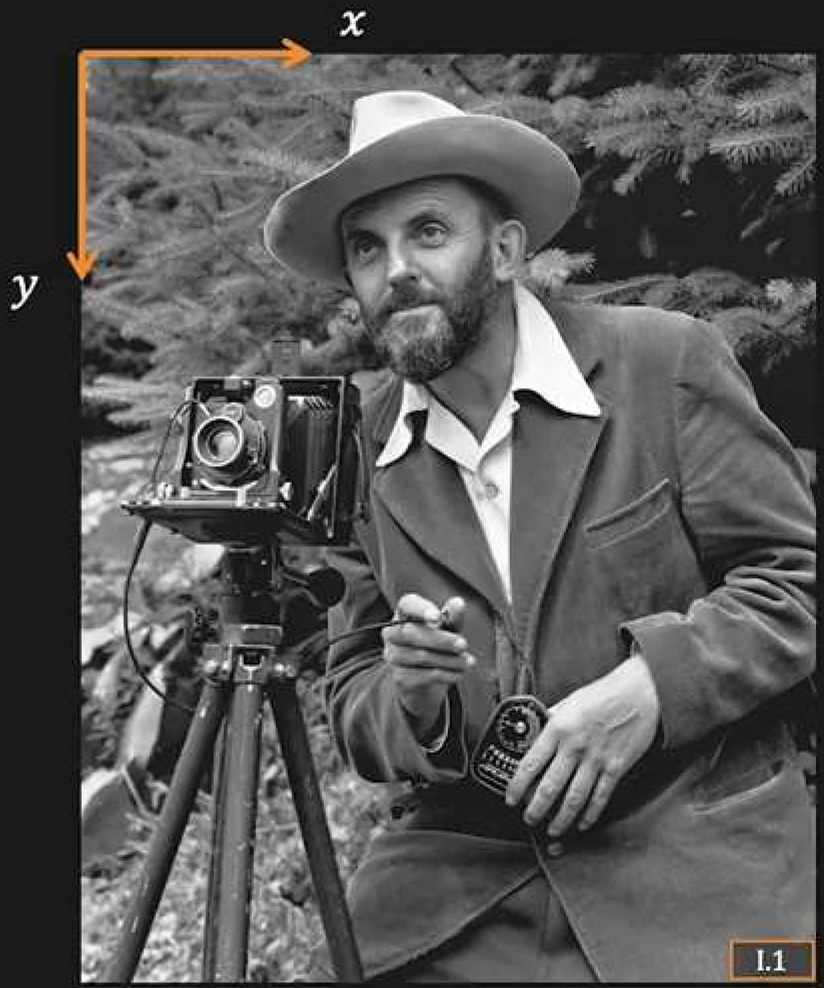
(3) Linear Image Filters

(4) Non-Linear Image Filters

(5) Template Matching by Correlation

© 2020 Shree K. Nayar

Image as a Function



$f(x, y)$ is the image intensity at position (x, y)

© 2020 Shree K. Nayar

Pixel (Point) Processing

Transformation T of intensity f at each pixel to intensity g:

$$g(x, y) = T\,(f\,(x, y))$$

© 2020 Shree K. Nayar

Point Processing

Darken $(f - 128)$

Original (f)

Lighten $(f + 128)$

Invert $(255 - f)$

© 2020 Shree K. Nayar

Pixel Processing

Low Contrast $(f/2)$

Original (f)

High Contrast $(f * 2)$

Gray $(0.3f_R + 0.6f_G + 0.1f_B)$

© 2020 Shree K. Nayar

Linear Shift Invariant System (LSIS)

$$f(x) \longrightarrow \boxed{\text{LSIS}} \longrightarrow g(x)$$

Study of Linear Shift Invariant Systems (LSIS) leads to useful image processing algorithms.

© 2020 Shree K. Nayar

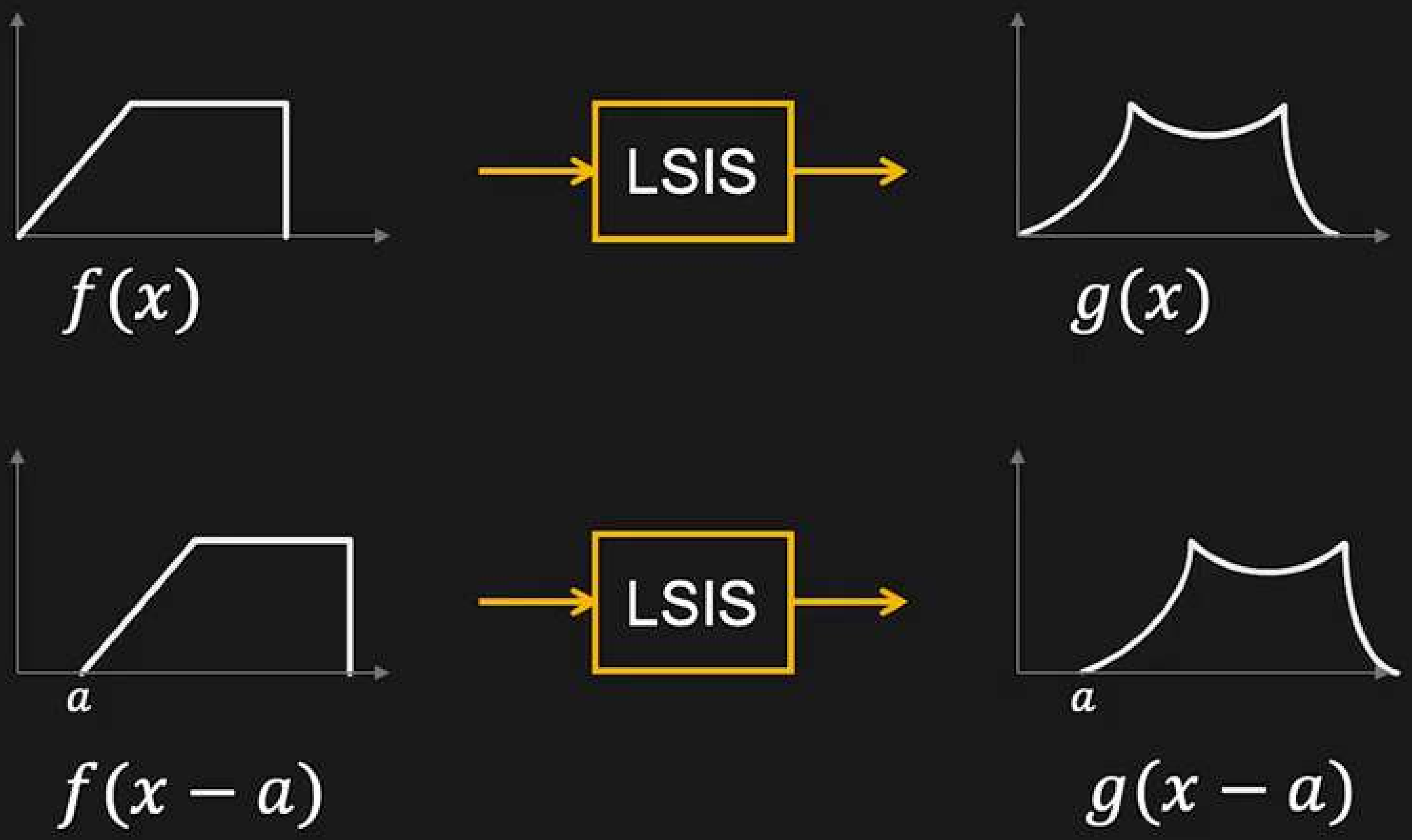
LSIS: Linearity

$$f_1 \longrightarrow \boxed{\text{LSIS}} \longrightarrow g_1 \qquad\qquad f_2 \longrightarrow \boxed{\text{LSIS}} \longrightarrow g_2$$

$$\alpha f_1 + \beta f_2 \longrightarrow \boxed{\text{LSIS}} \longrightarrow \alpha g_1 + \beta g_2$$

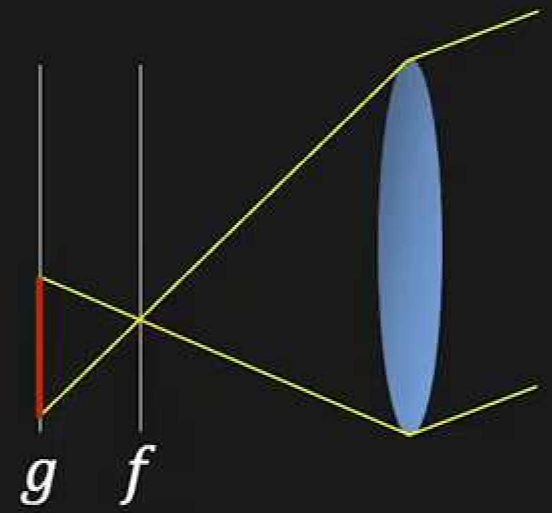
© 2020 Shree K. Nayar

LSIS: Shift Invariance



© 2020 Shree K. Nayar

Ideal Lens is an LSIS



Defocused Image (g): Processed version of Focused Image (f)

Linearity: Brightness variation

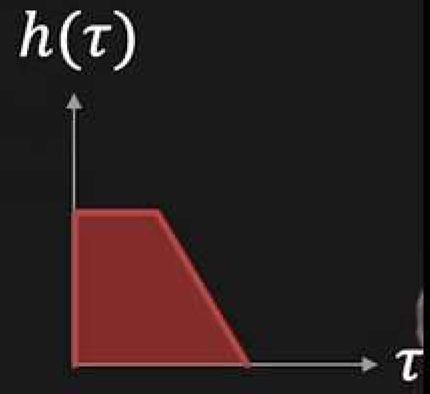
Shift invariance: Scene movement

© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau}$$

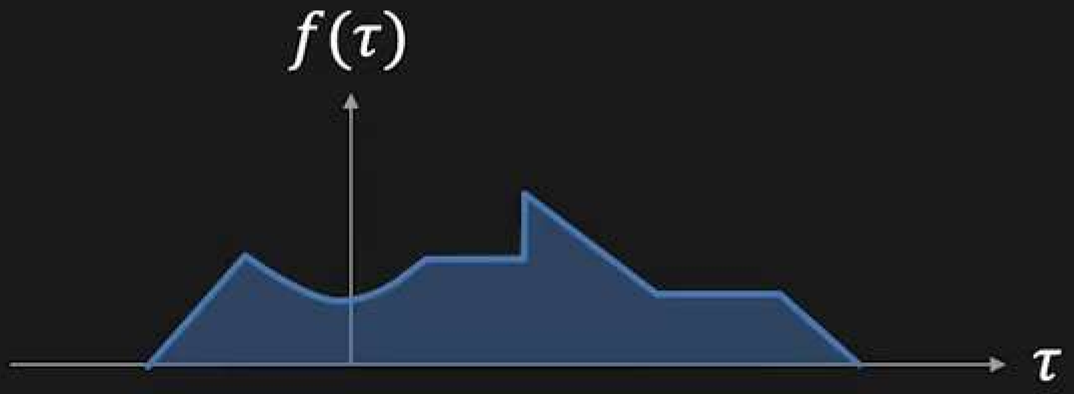


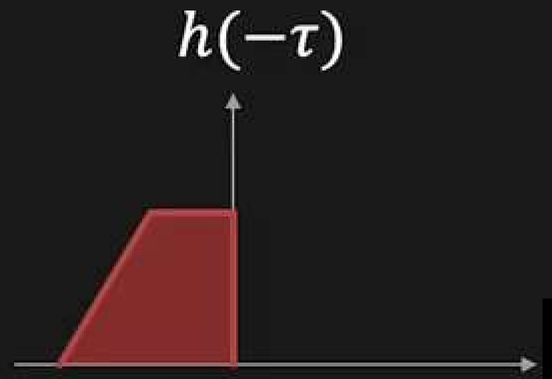
© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau) h(x-\tau)\, d\tau}$$



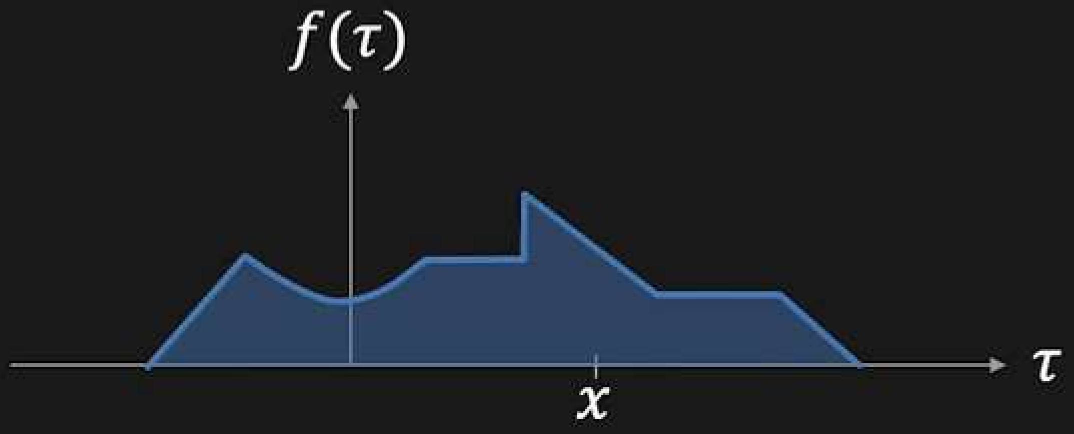


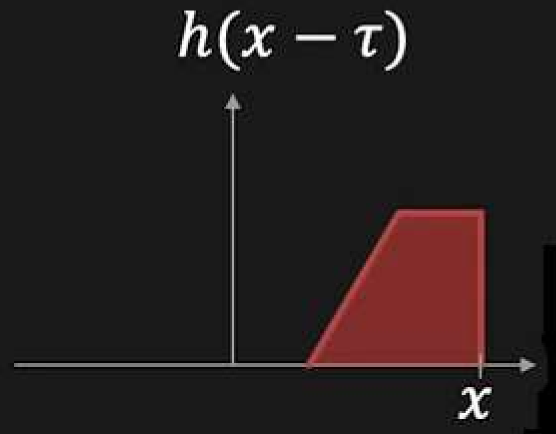
© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau}$$





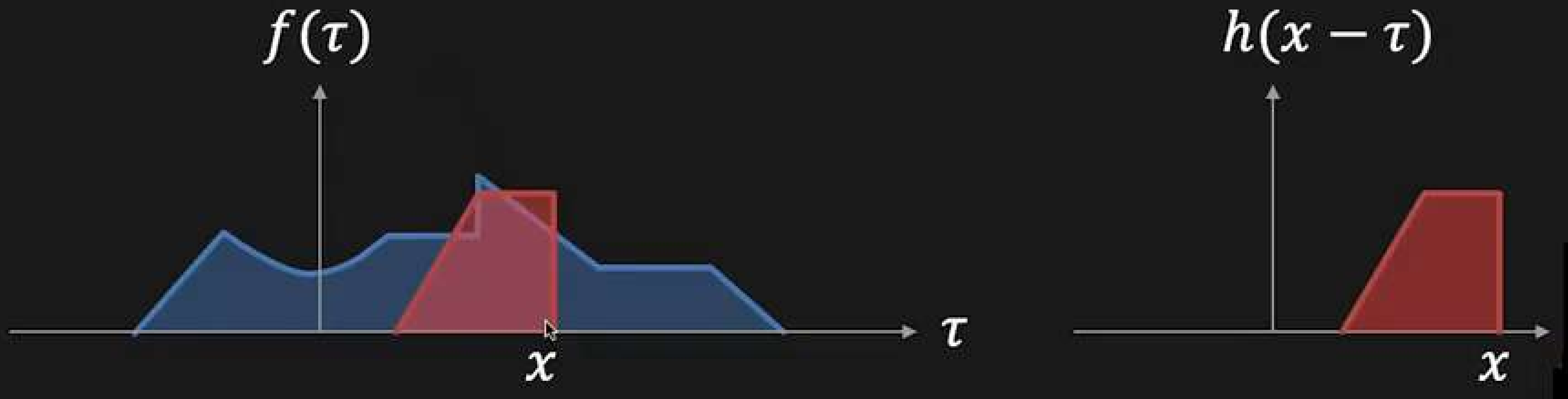
[redacted]

© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau$$

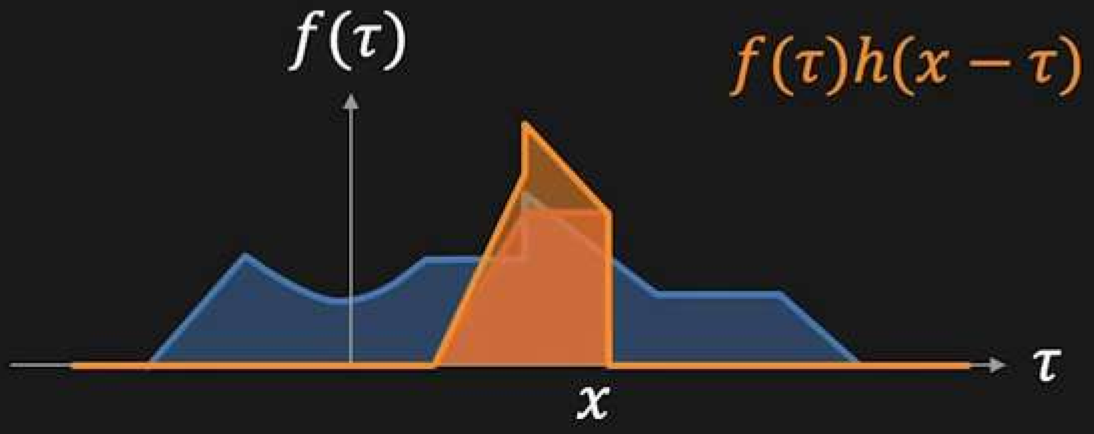


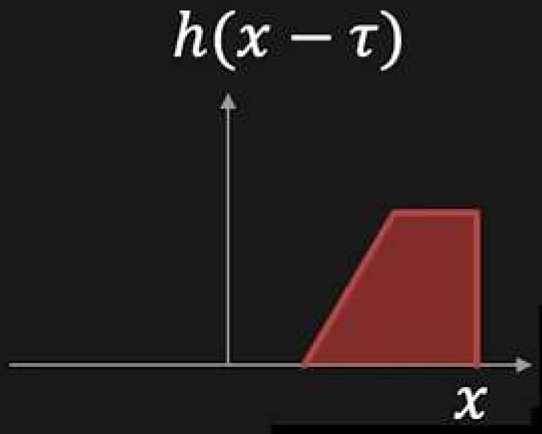
© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau}$$



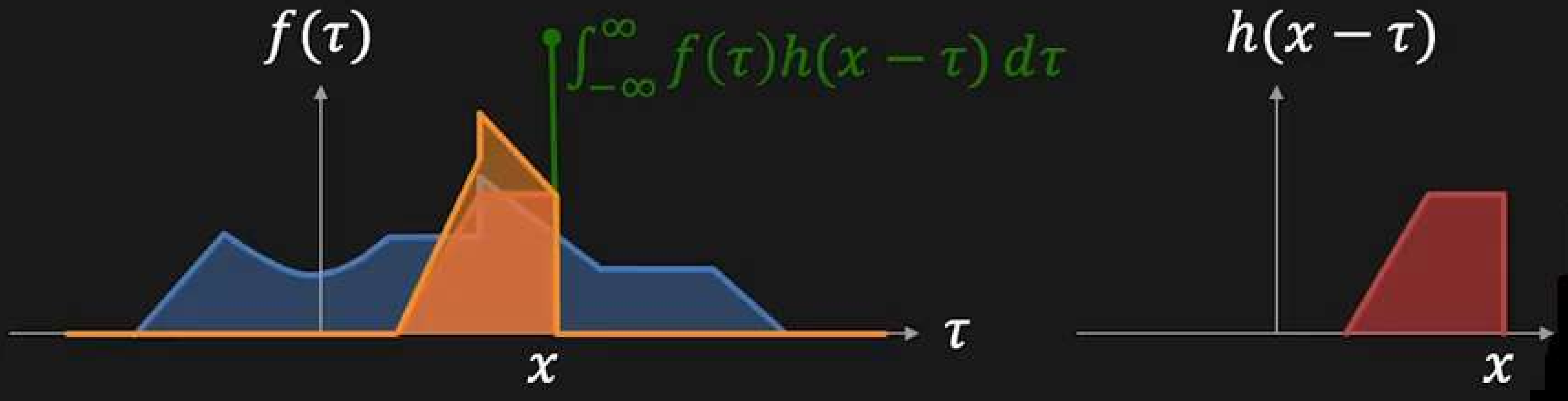


© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau}$$

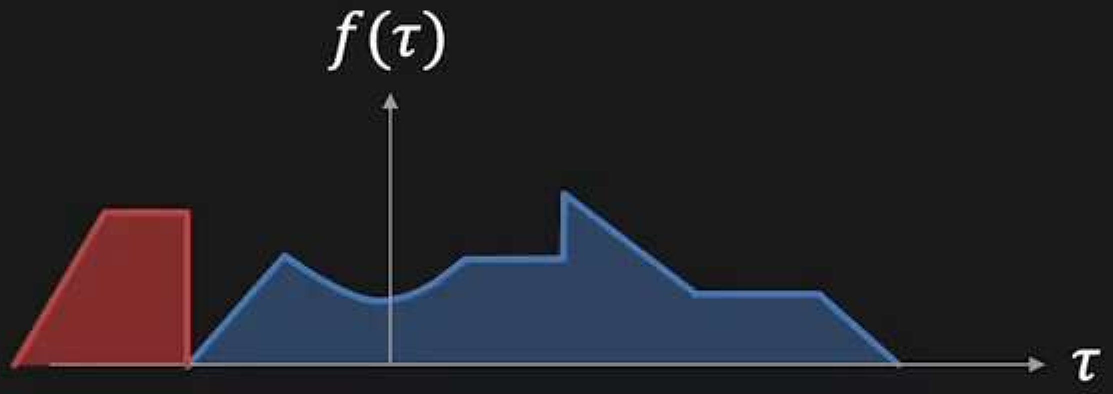


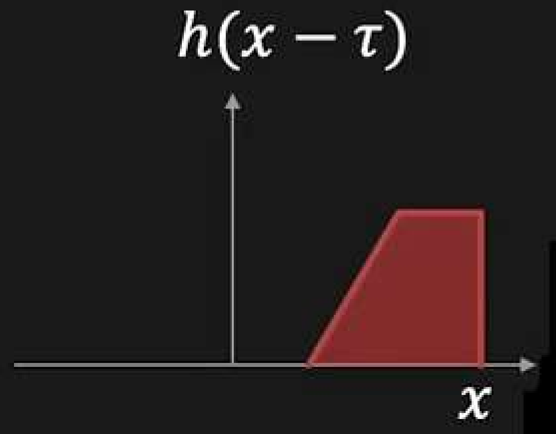
© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau}$$



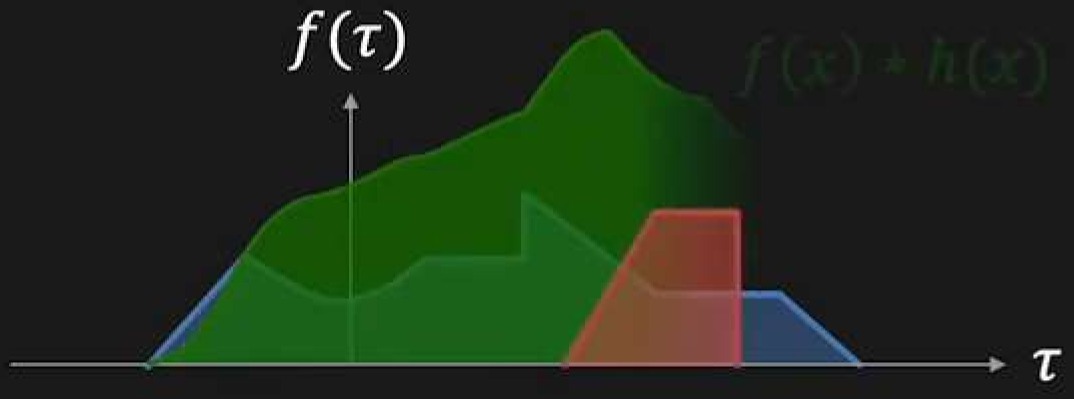


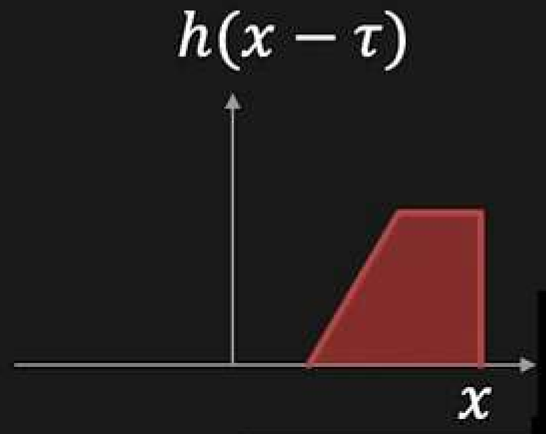
© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau}$$



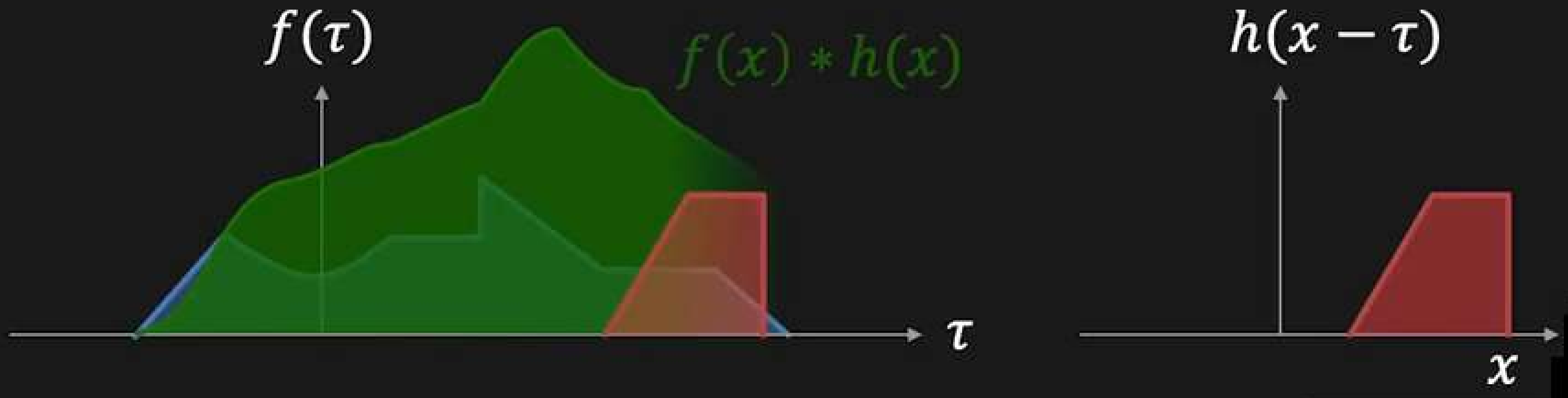


© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau$$

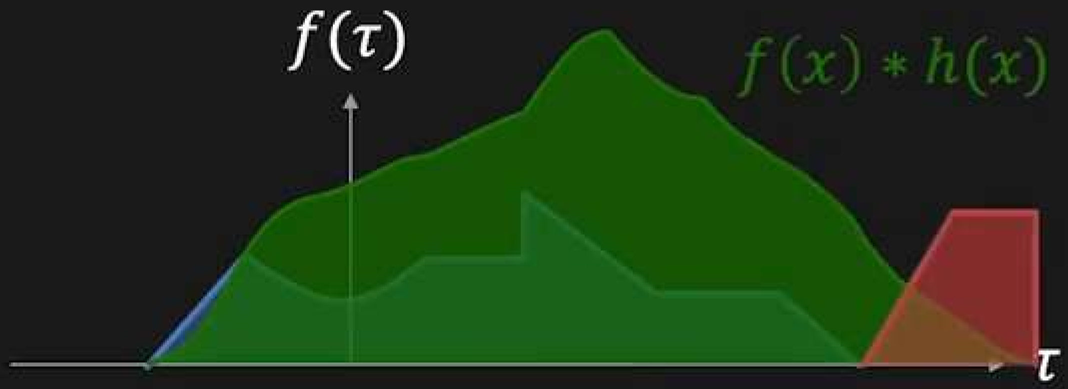


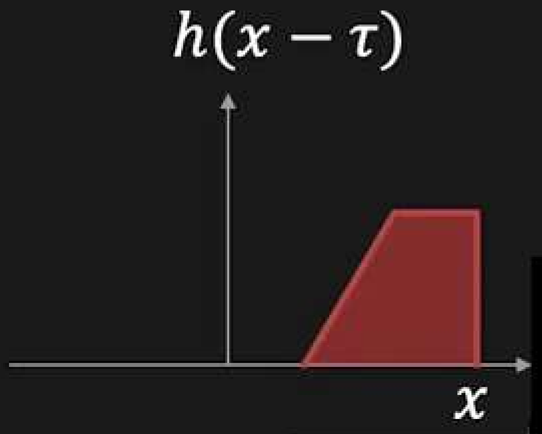
© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau}$$



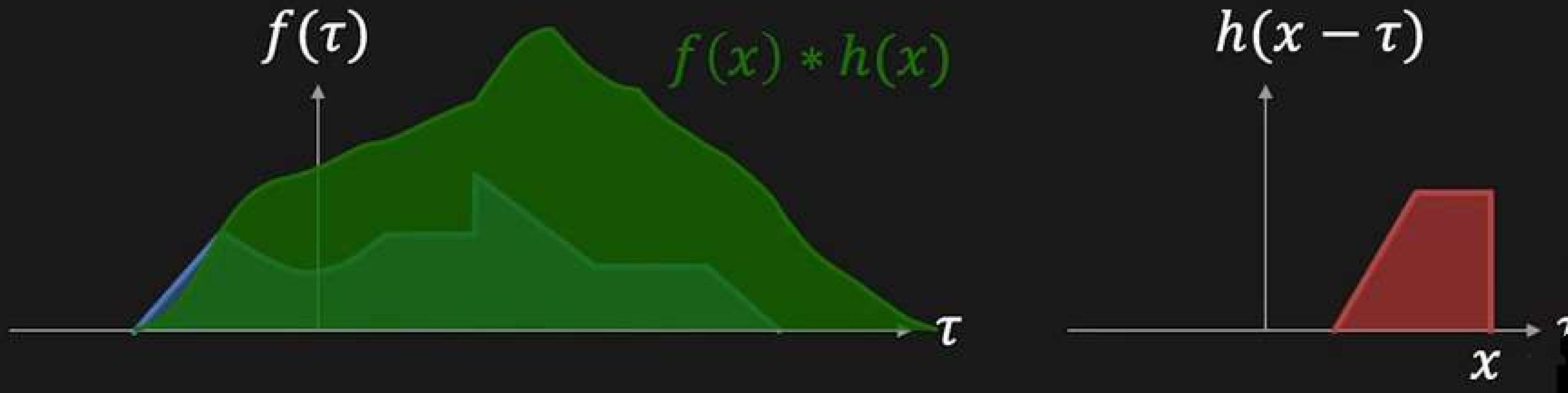


© 2020 Shree K. Nayar

Convolution

Convolution of two functions $f(x)$ and $h(x)$

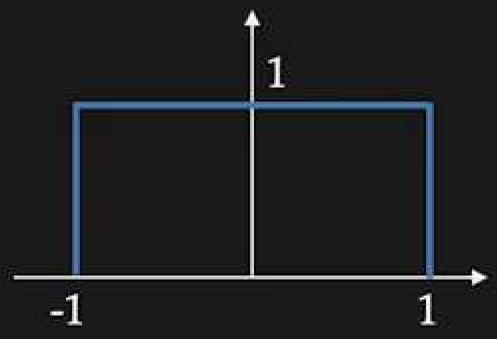
$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\, d\tau}$$



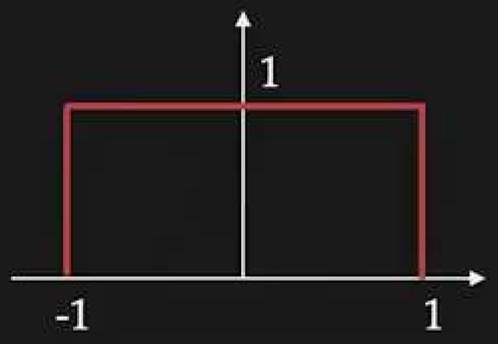
LSIS implies Convolution and Convolution implies LSIS [redacted]

© 2020 Shree K. Nayar

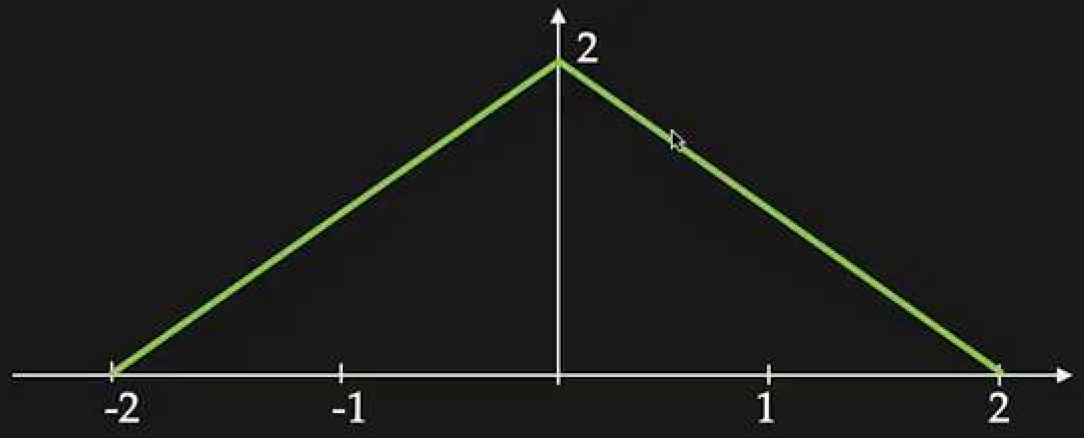
Convolution: Example



$f(x)$



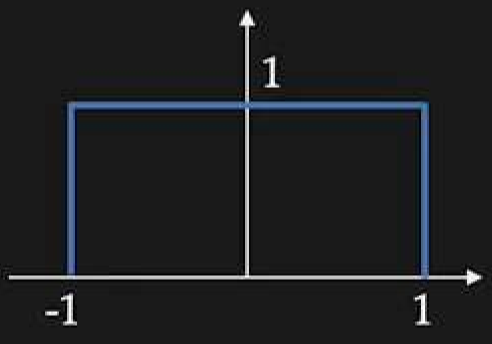
$h(x)$



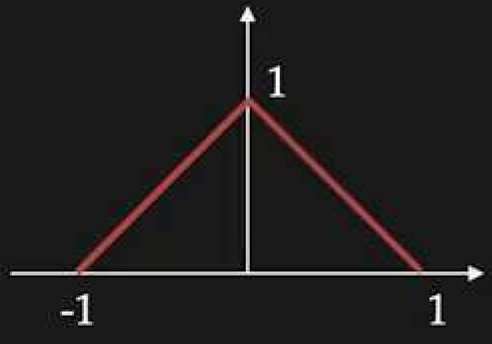
$f(x) * h(x)$

© 2020 Shree K. Nayar

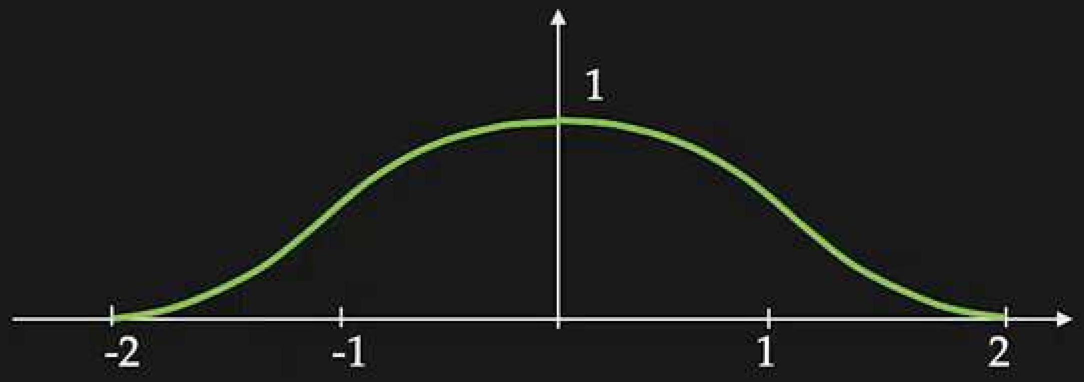
Convolution: Example



$f(x)$



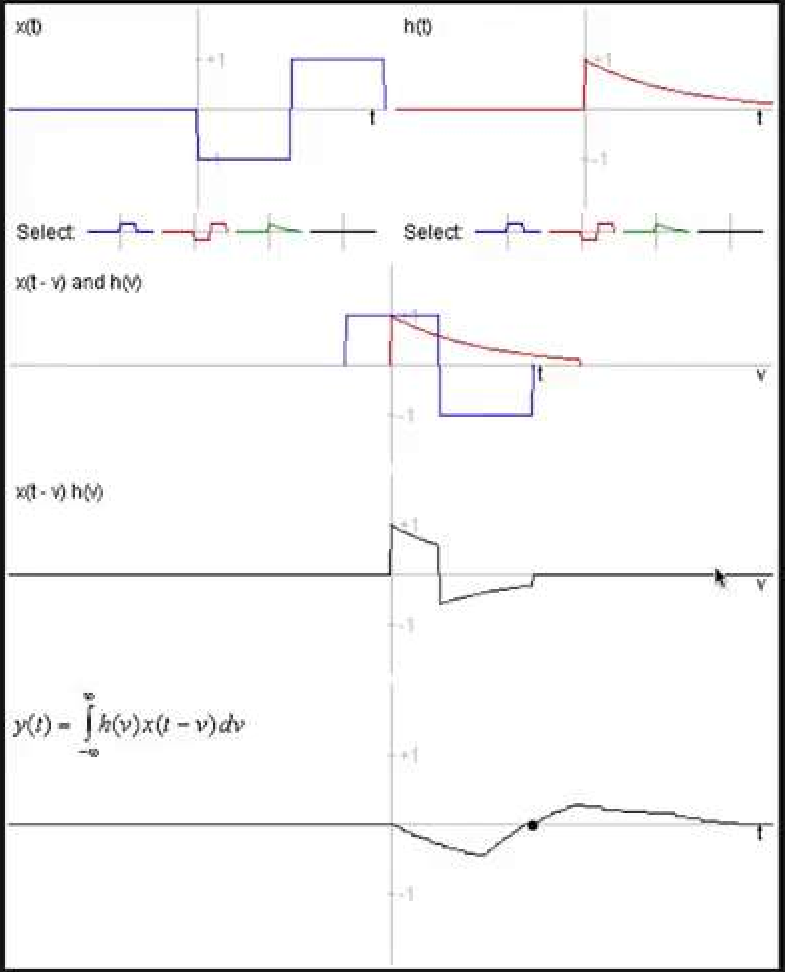
$h(x)$



$f(x) * h(x)$

© 2020 Shree K. Nayar

Convolution: Online Demo



http://www.jhu.edu/signals/convolve/

© 2020 Shree K. Nayar

Convolution is LSIS

Linearity:

Let: $g_1(x) = \int_{-\infty}^{\infty} f_1(\tau) h(x-\tau)\, d\tau$ and $g_2(x) = \int_{-\infty}^{\infty} f_2(\tau) h(x-\tau)\, d\tau$

Then:

$$\int_{-\infty}^{\infty} (\alpha f_1(\tau) + \beta f_2(\tau)) h(x-\tau)\, d\tau$$

$$= \alpha \int_{-\infty}^{\infty} f_1(\tau) h(x-\tau)\, d\tau + \beta \int_{-\infty}^{\infty} f_2(\tau) h(x-\tau)\, d\tau$$

$$= \alpha g_1(x) + \beta g_2(x)$$

© 2020 Shree K. Nayar

Convolution is LSIS

Shift Invariance:

Let: $$g(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau$$

Then:

$$\int_{-\infty}^{\infty} f(\tau - a)h(x-\tau)\,d\tau$$

$$= \int_{-\infty}^{\infty} f(\mu)h(x-a-\mu)\,d\mu \qquad \text{(Substituting } \mu = \tau - a\text{)}$$

$$= g(x-a)$$

© 2020 Shree K. Nayar

Can we find h?

$$f \longrightarrow \boxed{h} \longrightarrow g \qquad g(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau$$

What input f will produce output $g = h$?

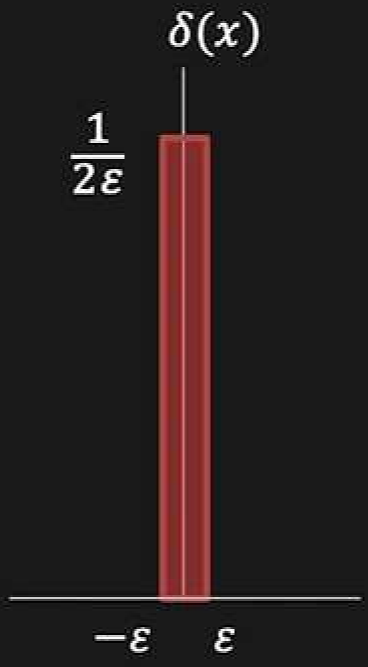
$$h(x) = \int_{-\infty}^{\infty} ?\,(\tau)h(x-\tau)\,d\tau$$

© 2020 Shree K. Nayar

Unit Impulse Function

$$\delta(x) = \begin{cases} 1/2\varepsilon, & |x| \le \varepsilon \\ 0, & |x| > \varepsilon \end{cases}$$

$$\varepsilon \to 0$$



$$\int_{-\infty}^{\infty} \delta(\tau)\, d\tau = \frac{1}{2\varepsilon}.\, 2\varepsilon = 1$$

$$\boxed{\int_{-\infty}^{\infty} \delta(\tau) b(x-\tau)\, d\tau = b(x)}$$

Sifting Property

© 2020 Shree K. Nayar

Impulse Response

$$f \longrightarrow \boxed{h} \longrightarrow g$$

$$g(x) = f(x) * h(x)$$

$$g(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau$$

$$\delta \longrightarrow \boxed{h} \longrightarrow h$$

Unit Impulse (δ) — Impulse Response (h)

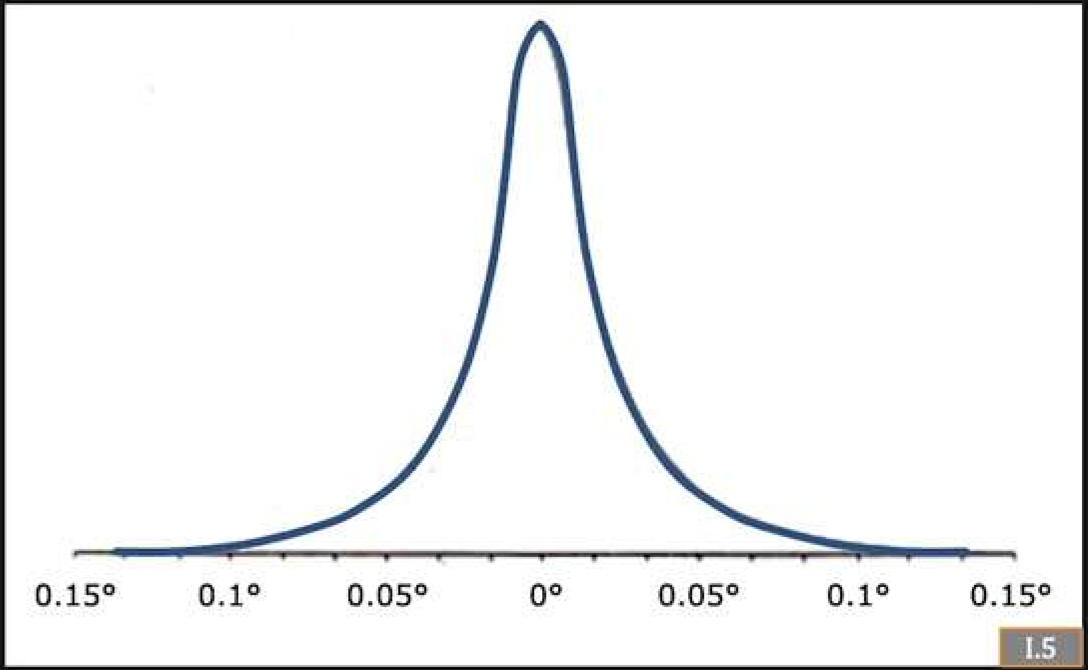
$$h(x) = \delta(x) * h(x)$$

$$h(x) = \int_{-\infty}^{\infty} \delta(\tau)h(x-\tau)\,d\tau$$

© 2020 Shree K. Nayar

Impulse Response of Human Eye

Distant Star $\delta(x,y)$ → **Human Eye** → Point Spread Function (PSF) $h(x,y)$



Human Eye PSF

© 2020 Shree K. Nayar

Properties of Convolution

Commutative $a * b = b * a$

Associative $(a * b) * c = a * (b * c)$

Cascaded System

$$f \rightarrow \boxed{h_1} \rightarrow \boxed{h_2} \rightarrow g$$

$$\equiv \quad f \rightarrow \boxed{h_1 * h_2} \rightarrow g$$

$$\equiv \quad f \rightarrow \boxed{h_2 * h_1} \rightarrow g$$

© 2020 Shree K. Nayar

2D Convolution

LSIS:

$$f(x,y) \longrightarrow \boxed{h(x,y)} \longrightarrow g(x,y)$$

Convolution:

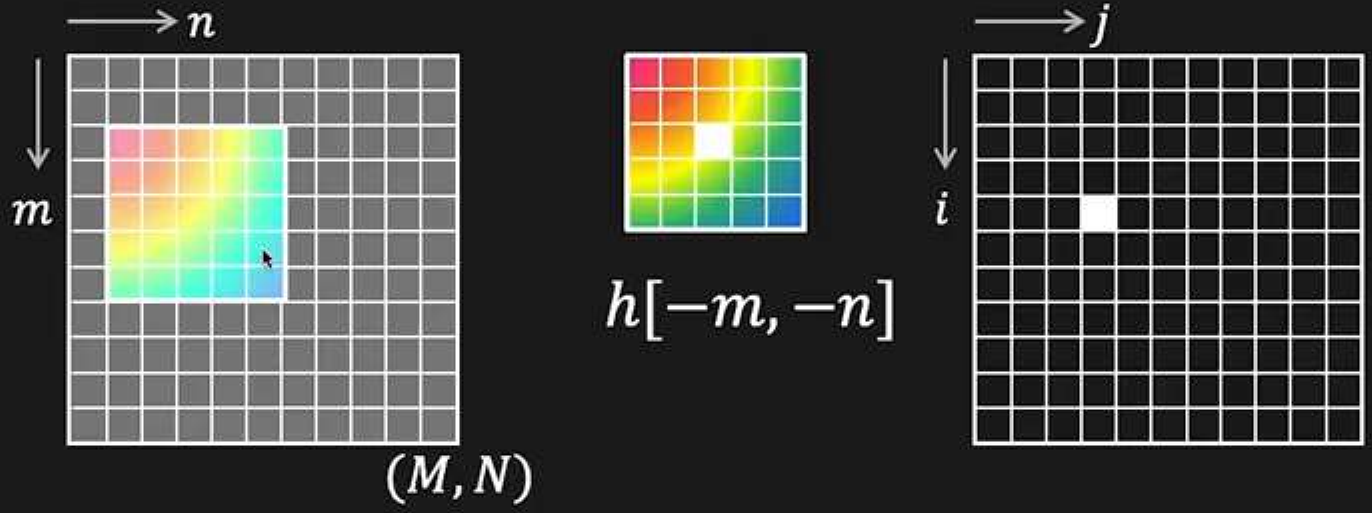
$$g(x,y) = \iint_{-\infty}^{\infty} f(\tau,\mu)h(x-\tau,y-\mu)\,d\tau d\mu$$

© 2020 Shree K. Nayar

Convolution with Discrete Images

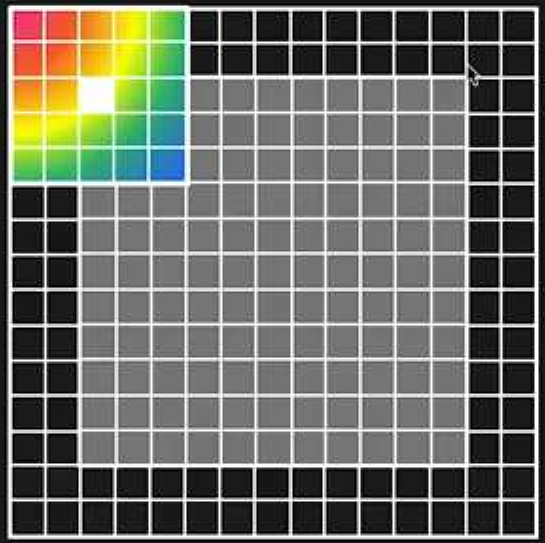
$$f[i,j] \longrightarrow \boxed{h[i,j]} \longrightarrow g[i,j]$$

$$g[i,j] = \sum_{m=1}^{M} \sum_{n=1}^{N} f[m,n] \underbrace{h[i-m, j-n]}_{\text{"Mask," "Kernel," "Filter"}}$$



© 2020 Shree K. Nayar

Border Problem



Solution:

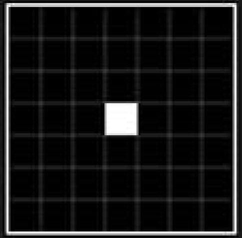
- Ignore border
- Pad with constant value
- Pad with reflection

© 2020 Shree K. Nayar

Example: Impulse Filter

Input

*



=

Output

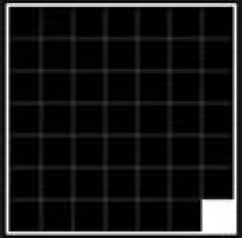
$$f(x,y) * \delta(x,y) = f(x,y)$$

© 2020 Shree K. Nayar

Example: Image Shift

Input

$*$



$=$

Output

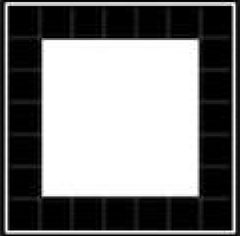
$f(x, y)$ $\delta(x-u, y-v)$ $f(x-u, y-v)$

© 2020 Shree K. Nayar

Example: Averaging

Input

$*$



“Box Filter”
5 x 5

$=$

Output

$f(x,y)$ $a(x,y)$ $g(x,y)$

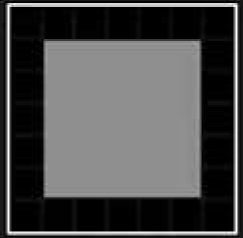
Result Image is saturated. Why?

© 2020 Shree K. Nayar

Example: Averaging

Input

$*$



"Box Filter"
5 x 5

$=$

Output

$f(x,y)$ $a(x,y)$ $g(x,y)$

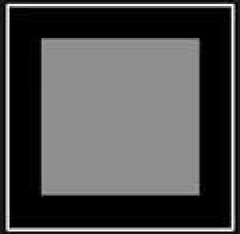
Sum of all the filter (kernel) weights should be 1.

© 2020 Shree K. Nayar

Smoothing With Box Filter

Input

*



“Box Filter”
21 x 21

=

Output

$f(x,y)$ $\quad a(x,y)$ $\quad g(x,y)$

Image smoothed with a box filter does not look “natural.” Has blocky artifacts.

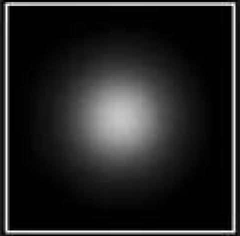
© 2020 Shree K. Nayar

Smoothing With “Fuzzy” Filter

Input

Output

$*$



“Fuzzy Filter”
21 x 21

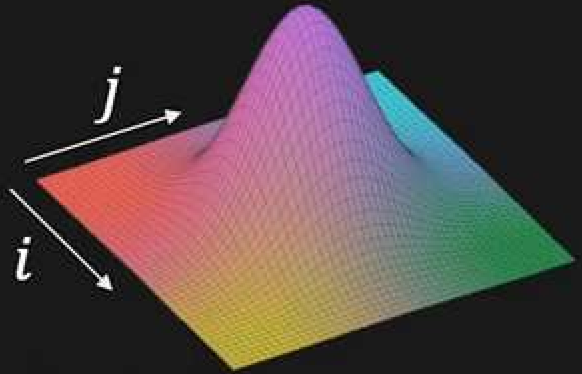
$=$

$f(x,y)$ $\quad$ $b(x,y)$ $\quad$ $g(x,y)$

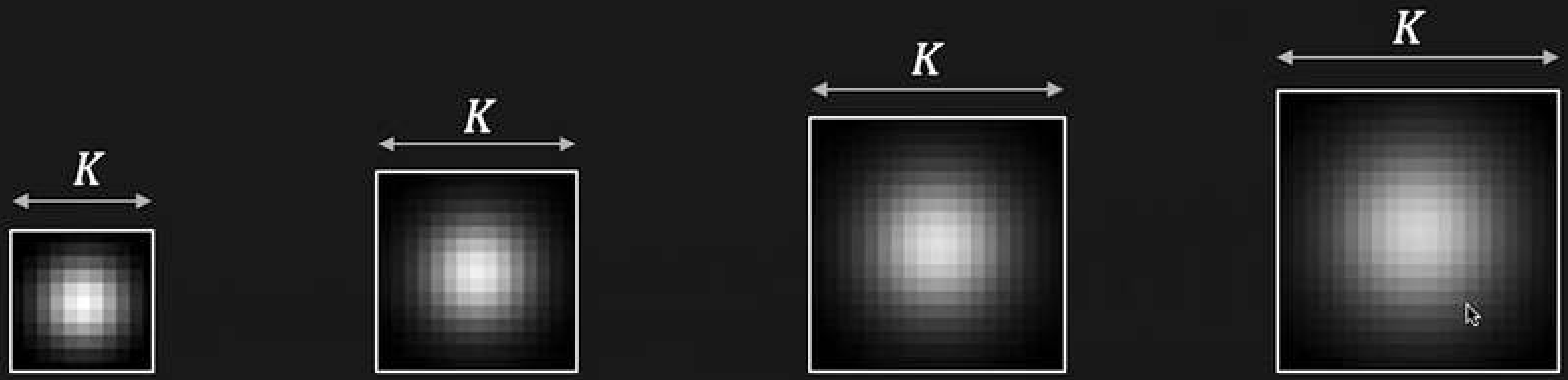
© 2020 Shree K. Nayar

Gaussian Kernel: A Fuzzy Filter

$$n_\sigma[i,j] = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{i^2+j^2}{\sigma^2}\right)}$$



σ^2: Variance



$\sigma = 2$ $\sigma = 3$ $\sigma = 4$ $\sigma = 5$

Rule of thumb: Set kernel size $K \approx 2\pi\sigma$

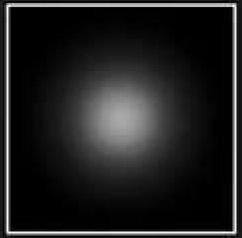
© 2020 Shree K. Nayar

Gaussian Smoothing

Input

Output

$*$



$=$

$\sigma = 16$

$f(x, y)$ $\quad$ $n_{16}(x, y)$ $\quad$ $g(x, y)$

Larger the kernel (or σ), more the blurring

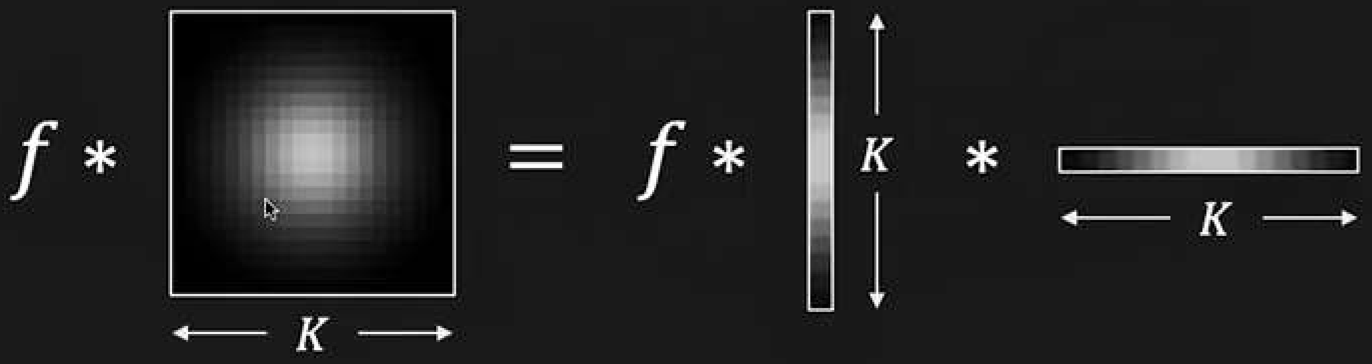
© 2020 Shree K. Nayar

Gaussian Smoothing is Separable

$$g[i,j] = \frac{1}{2\pi\sigma^2} \sum_{m=1}^{K} \sum_{n=1}^{K} e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma^2}\right)} f[i-m, j-n]$$

$$g[i,j] = \frac{1}{2\pi\sigma^2} \sum_{m=1}^{K} e^{-\frac{1}{2}\left(\frac{m^2}{\sigma^2}\right)} . \sum_{n=1}^{K} e^{-\frac{1}{2}\left(\frac{n^2}{\sigma^2}\right)} f[i-m, j-n]$$

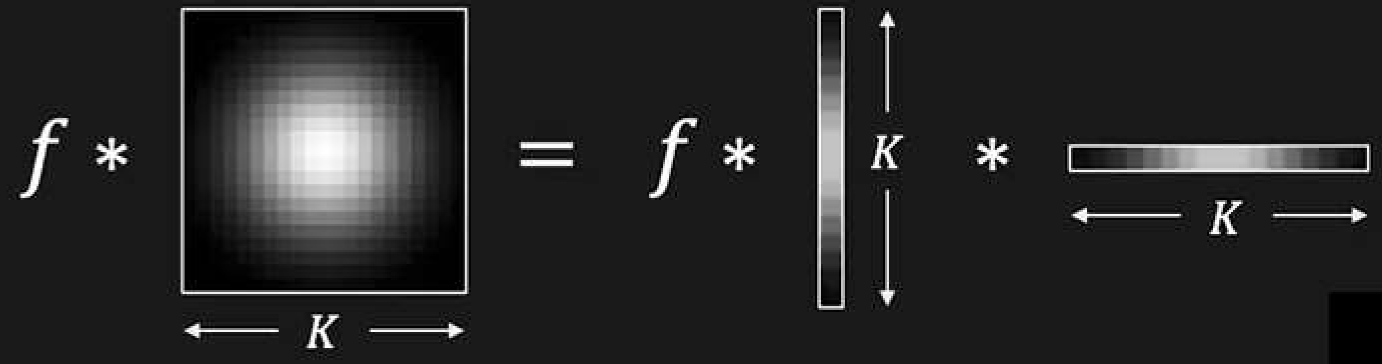
Using One 2D Gaussian Filter ≡ Using Two 1D Gaussian Filter[redacted]



© 2020 Shree K. Nayar

Gaussian Smoothing is Separable

Using One 2D Gaussian Filter ≡ Using Two 1D Gaussian Filters

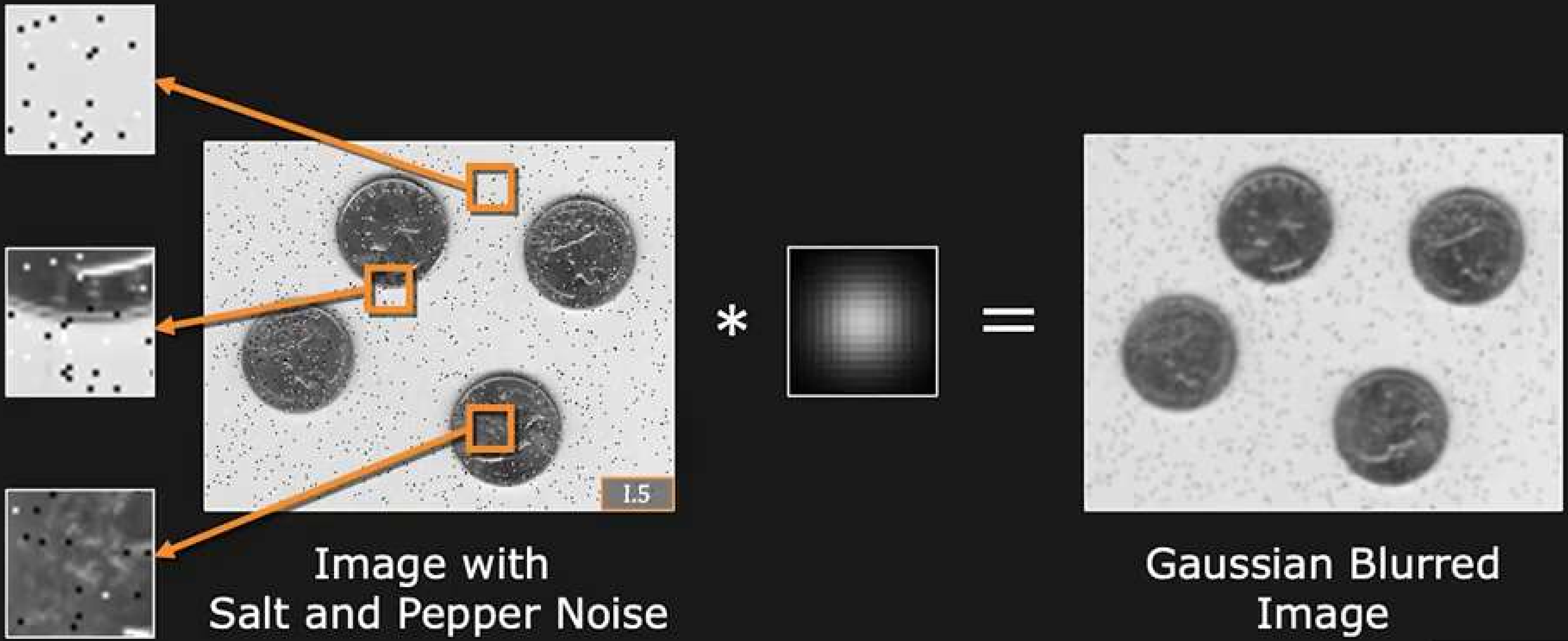


Which one is faster? Why?

K^2 Multiplications	$2K$ Multiplications
$K^2 - 1$ Additions	$2(K-1)$ Additions

© 2020 Shree K. Nayar

Smoothing to Remove Image Noise



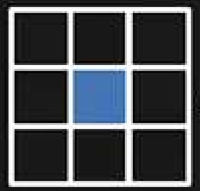
Problem with Smoothing:

- Does not remove outliers (Noise)
- Smooths edges (Blur)

© 2020 Shree K. Nayar

Median Filtering

1. Sort the K^2 values in window centered at the pixel
2. Assign the Middle Value (Median) to pixel



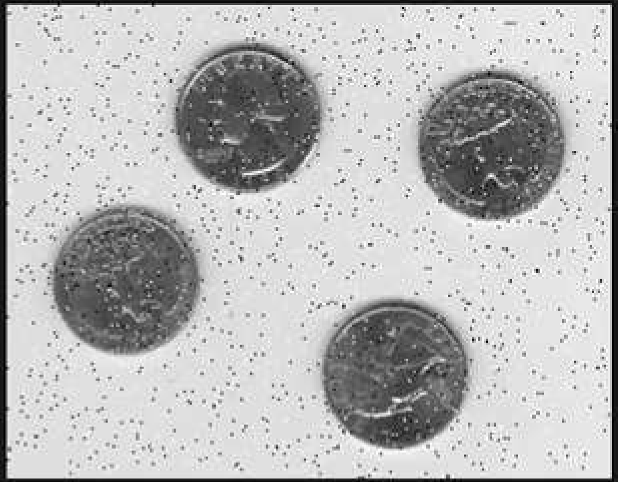
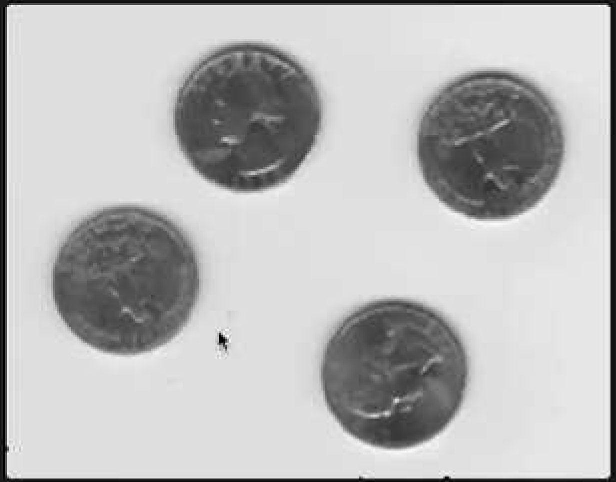


Image with
Salt and Pepper Noise



Median Filtered
Image ($K = 3$)

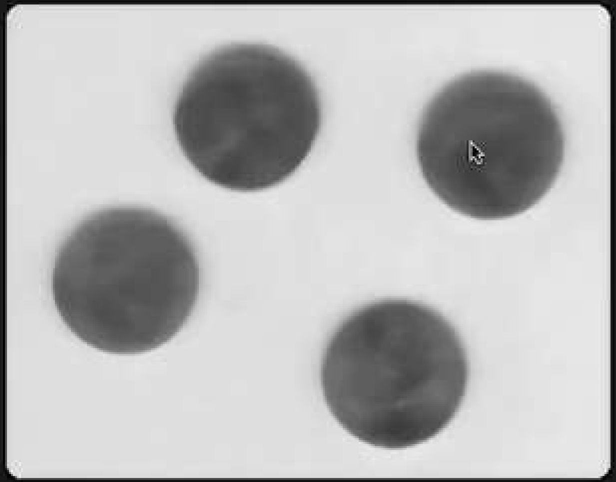
Non-linear Operation
(Cannot be implemented using convolution)

© 2020 Shree K. Nayar

Median Filtering

Not Effective when Image Noise is not a Simple Salt and Pepper Noise.

Image with Noise

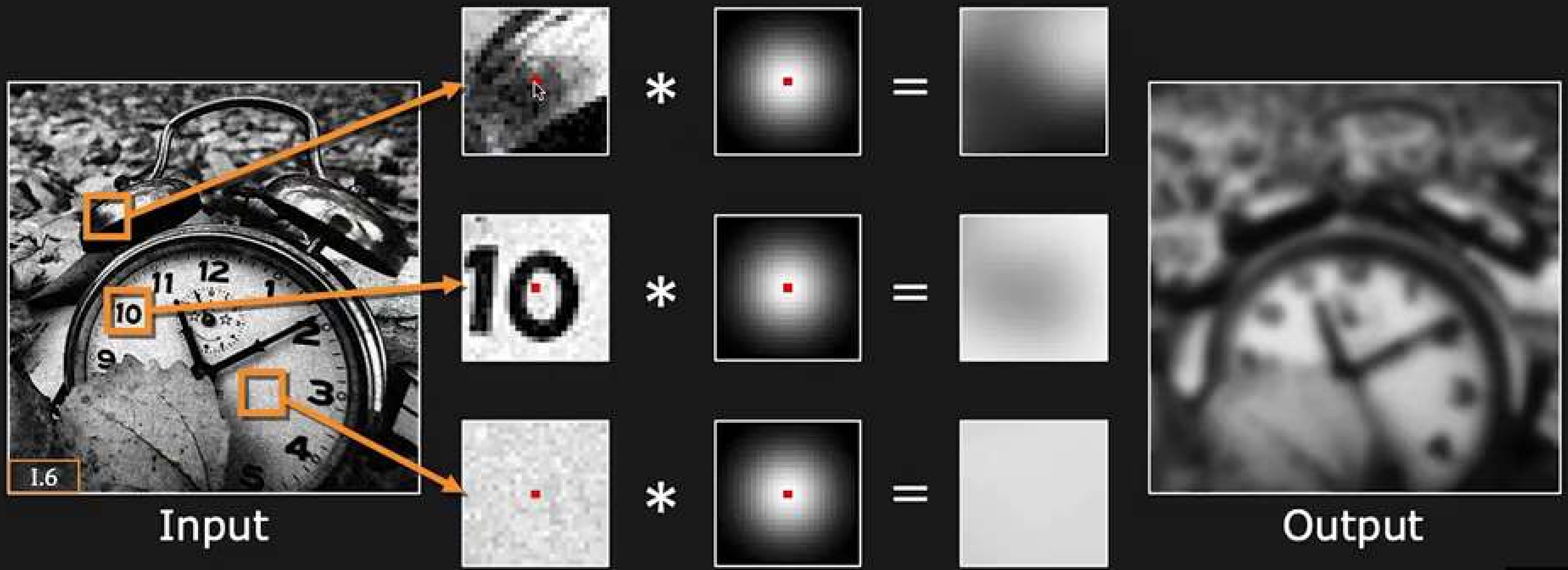


Median Filtered Image ($K = 11$)

Larger K causes blurring of image detail

© 2020 Shree K. Nayar

Revisiting Gaussian Smoothing

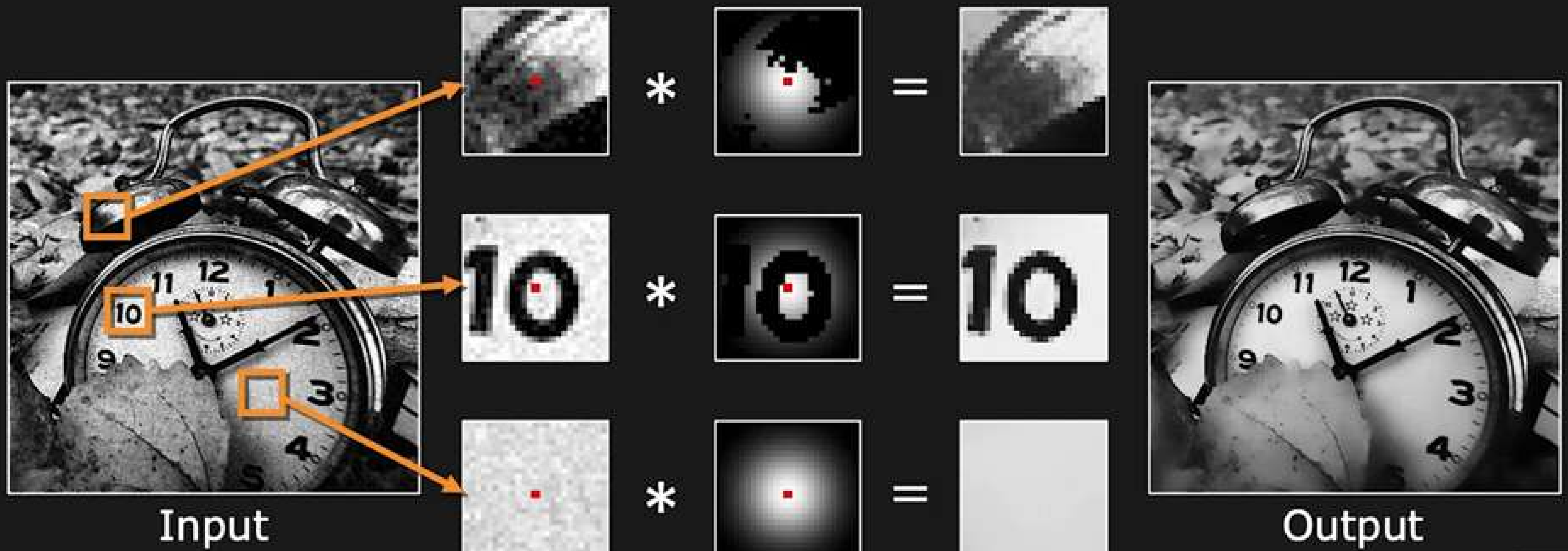


Same Gaussian kernel is used everywhere.

Blurs across edges.

© 2020 Shree K. Nayar

Blur Similar Pixels Only



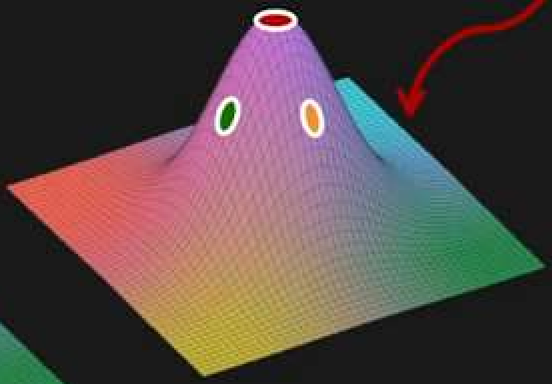
"Bias" Gaussian Kernel such that pixels not similar in intensity to the center pixel receive a lower weight.

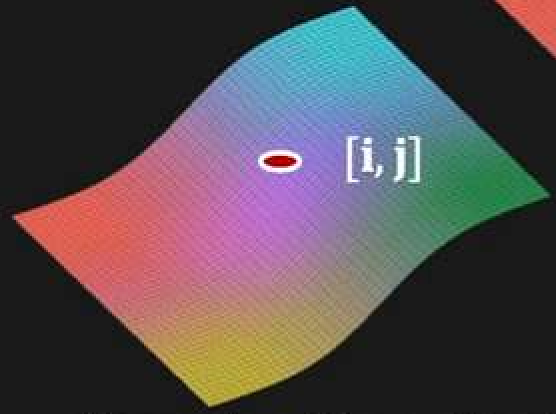
© 2020 Shree K. Nayar

Bilateral Filter: Start With Gaussian

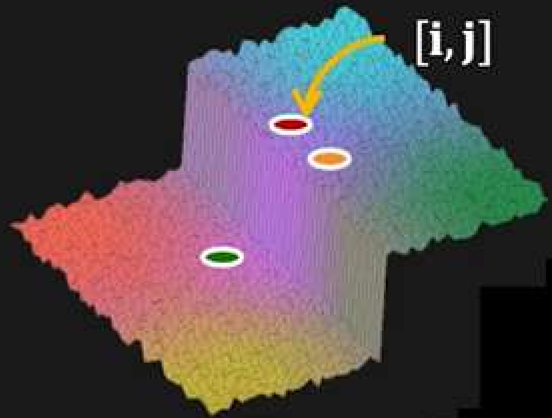
Spatial Gaussian

$$g[i,j] = \frac{1}{W_s}\sum_{m}\sum_{n} f[m,n]\,\underbrace{n_{\sigma_s}[i-m,j-n]}$$





Gaussian Smoothed Output (g)



Input (f)

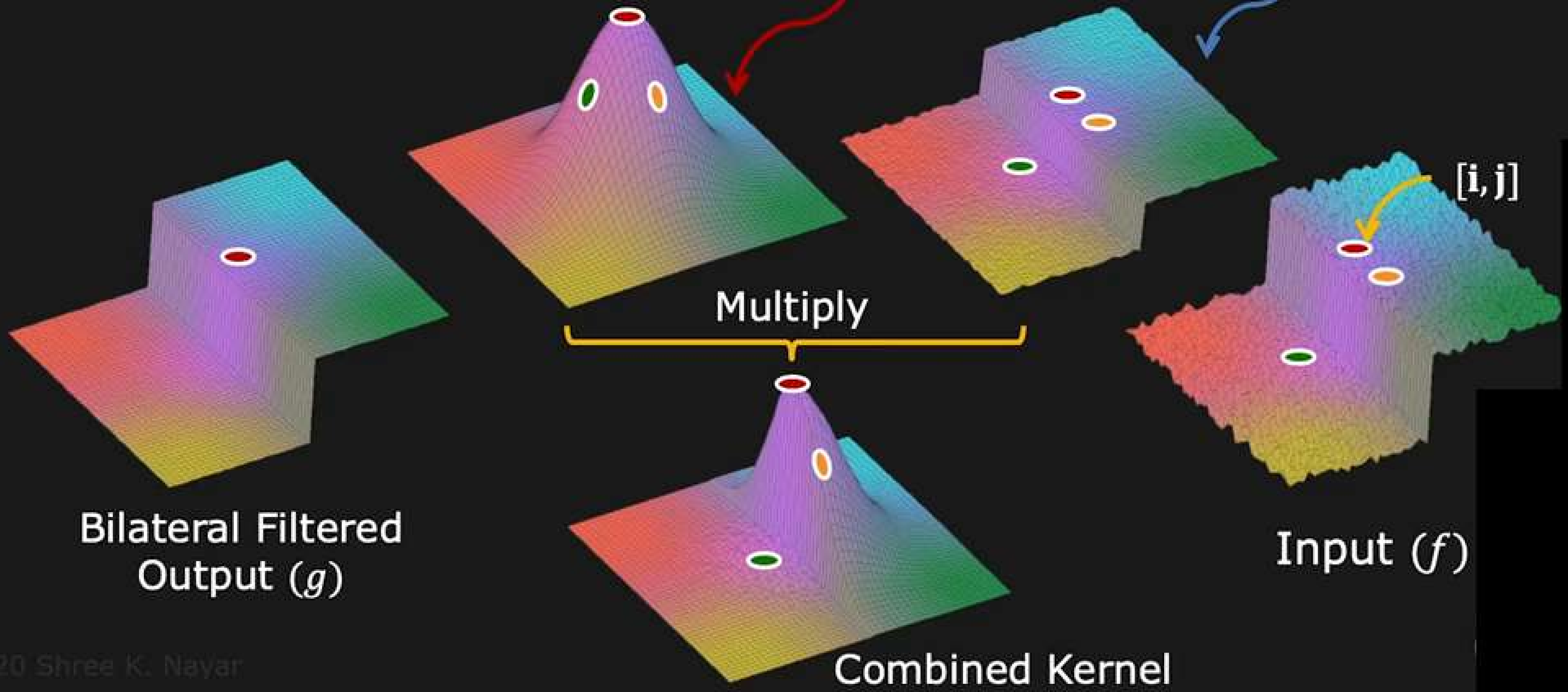
Gaussian blurs across edges

© 2020 Shree K. Nayar

Bilateral Filter: Add Bias to Gaussian

Spatial Gaussian Brightness Gaussian

$$g[i,j] = \frac{1}{W_{sb}} \sum_{m} \sum_{n} f[m,n]\, \underbrace{n_{\sigma_s}[i-m, j-n]}\, \underbrace{n_{\sigma_b}(f[m,n] - f[i,j])}$$



© 2020 Shree K. Nayar

Bilateral Filter: Summary

$$g[i,j] = \frac{1}{W_{sb}} \sum_m \sum_n f[m,n]\, n_{\sigma_s}[i-m, j-n]\, n_{\sigma_b}(f[m,n] - f[i,j])$$

Where:

$$n_{\sigma_s}[m,n] = \frac{1}{2\pi {\sigma_s}^2} e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma_s^2}\right)} \qquad n_{\sigma_b}(k) = \frac{1}{\sqrt{2\pi}\sigma_b} e^{-\frac{1}{2}\left(\frac{k^2}{\sigma_b^2}\right)}$$

$$W_{sb} = \sum_m \sum_n n_{\sigma_s}[i-m, j-n]\, n_{\sigma_b}(f[m,n] - f[i,j])$$

Non-linear Operation

(Cannot be implemented using convolution)

© 2020 Shree K. Nayar

Gaussian vs. Bilateral Filtering: Example

Original

Gaussian
$\sigma_s = 2$

Bilateral
$\sigma_s = 2, \sigma_b = 10$

© 2020 Shree K. Nayar

Gaussian vs. Bilateral Filtering: Example

Original

Gaussian
$\sigma_s = 4$

Bilateral
$\sigma_s = 4, \sigma_b = 10$

© 2020 Shree K. Nayar

Gaussian vs. Bilateral Filtering: Example

Original

Gaussian
$\sigma_s = 8$

Bilateral
$\sigma_s = 8, \sigma_b = 10$

© 2020 Shree K. Nayar

Bilateral Filtering: Changing σ_b

Bilateral
$\sigma_s = 6, \sigma_b = 10$

Bilateral
$\sigma_s = 6, \sigma_b = 20$

Bilateral
$\sigma_s = 6, \sigma_b = \infty$
(Gaussian Smoothing)

© 2020 Shree K. Nayar

Template Matching

Template

How do we locate the template in the image?

Minimize:

$$E[i,j] = \sum_{m}\sum_{n}(f[m,n] - t[m-i,n-j])^2$$

$$E[i,j] = \sum_{m}\sum_{n}(f^2[m,n] + t^2[m-i,n-j] - \underbrace{2f[m,n]t[m-i,n-j]}_{\text{Maximize}})$$

© 2020 Shree K. Nayar

Template Matching

Template

How do we locate the template in the image?

Maximize:

$$\boxed{R_{tf}[i,j] = \sum_{m}\sum_{n} f[m,n]\,t[m-i,n-j] = t \otimes f}$$

(Cross-Correlation)

© 2020 Shree K. Nayar

Convolution vs. Correlation

Convolution:

$$g[i,j] = \sum_{m}\sum_{n} f[m,n]\underline{t[i-m,j-n]} = t * f$$

Correlation:

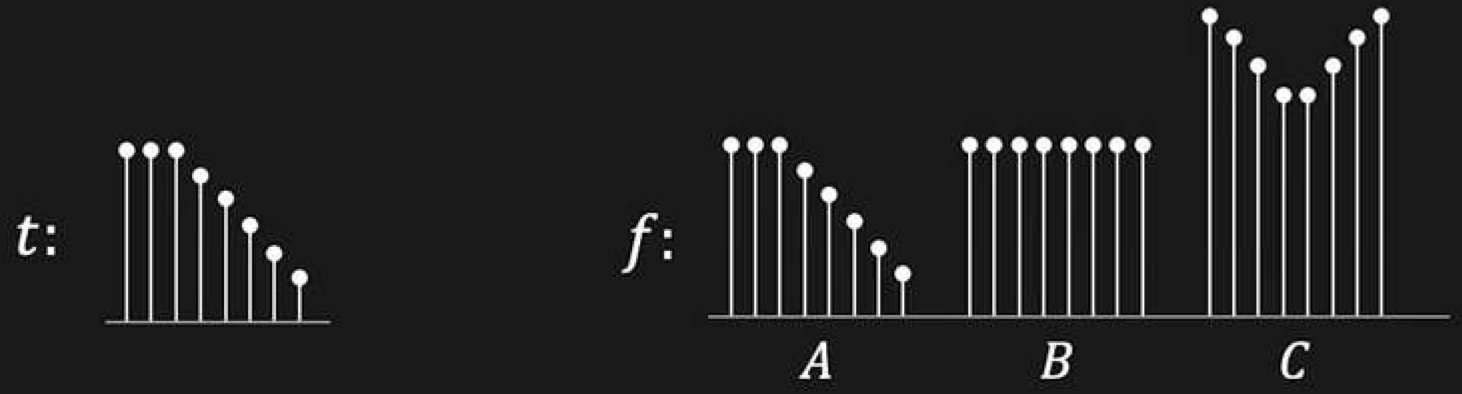
$$R_{tf}[i,j] = \sum_{m}\sum_{n} f[m,n]\underline{t[m-i,n-j]} = t \otimes f$$

No Flipping in Correlation

© 2020 Shree K. Nayar

Problem with Cross-Correlation

$$R_{tf}[i,j] = \sum_{m}\sum_{n} f[m,n]\,t[m-i,n-j] = t \otimes f$$



$$R_{tf}(C) > R_{tf}(B) > R_{tf}(A)$$

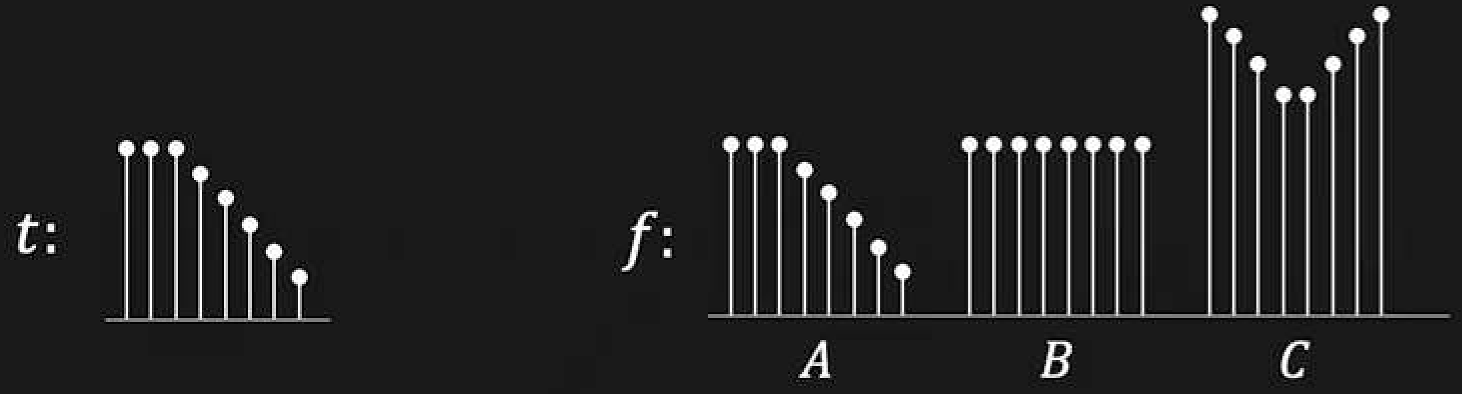
We need $R_{tf}(A)$ to be the maximum!

© 2020 Shree K. Nayar

Normalized Cross-Correlation

Account for energy differences

$$N_{tf}[i,j] = \frac{\sum_m \sum_n f[m,n]\,t[m-i,n-j]}{\sqrt{\sum_m \sum_n f^2[m,n]}\ \sqrt{\sum_m \sum_n t^2[m-i,n-j]}}$$



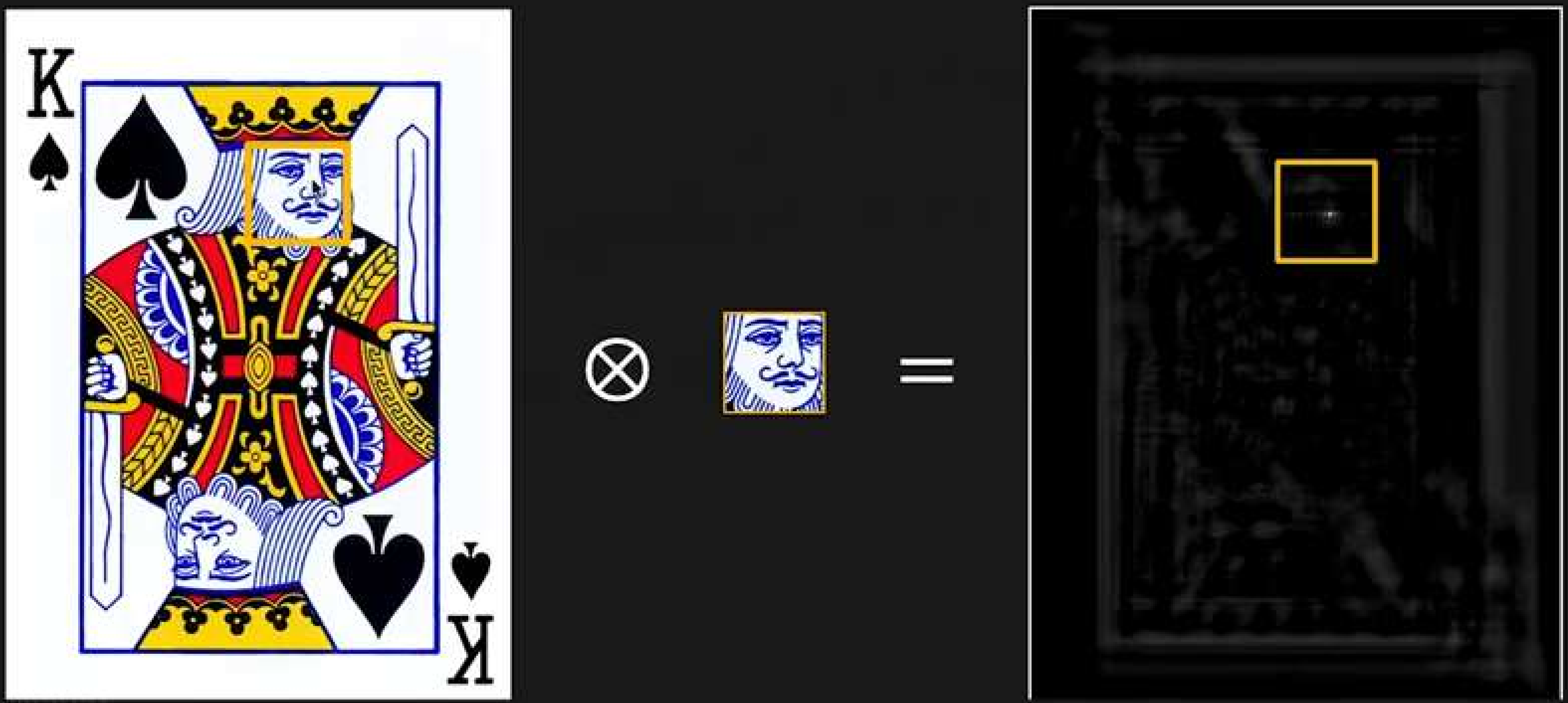
$$N_{tf}(A) > N_{tf}(B) > N_{tf}(C)$$

© 2020 Shree K. Nayar

Normalized Cross-Correlation

Account for energy differences

$$N_{tf}[i,j] = \frac{\sum_m \sum_n f[m,n]\,t[m-i,n-j]}{\sqrt{\sum_m \sum_n f^2[m,n]}\ \sqrt{\sum_m \sum_n t^2[m-i,n-j]}}$$



© 2020 Shree K. Nayar

Image Processing II

Transform image to new one that is clearer or easier to analyze.

Topics:

(1) Fourier Transform

(2) Convolution Theorem

(3) Deconvolution in Frequency Domain

(4) Sampling Theory and Aliasing

© 2020 Shree K. Nayar

Jean Baptiste Joseph Fourier

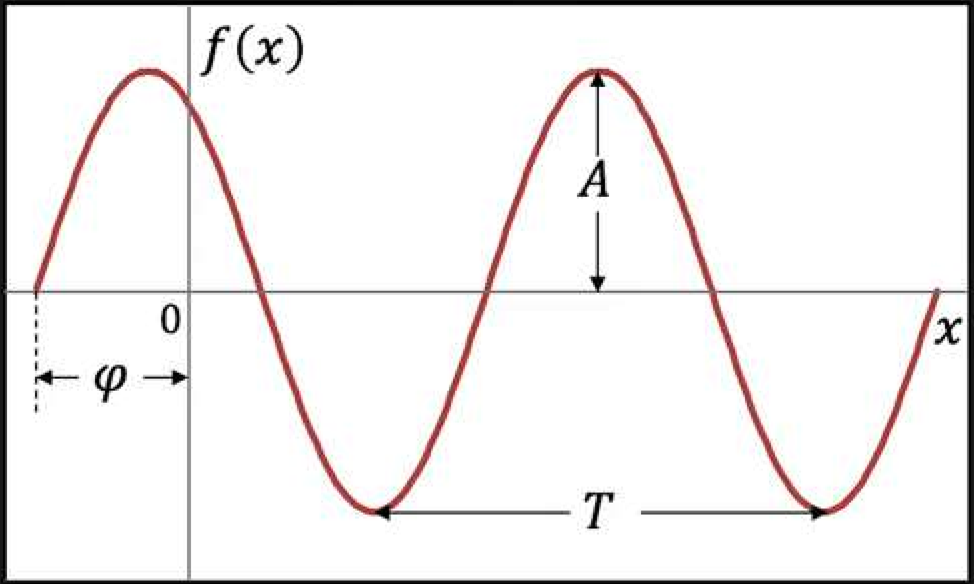
(1768-1830)

Any Periodic Function can be rewritten as a Weighted Sur[redacted] of Infinite Sinusoids of Different Frequencies.

© 2020 Shree K. Nayar

Sinusoid

$$f(x) = A\sin(2\pi u x + \varphi)$$



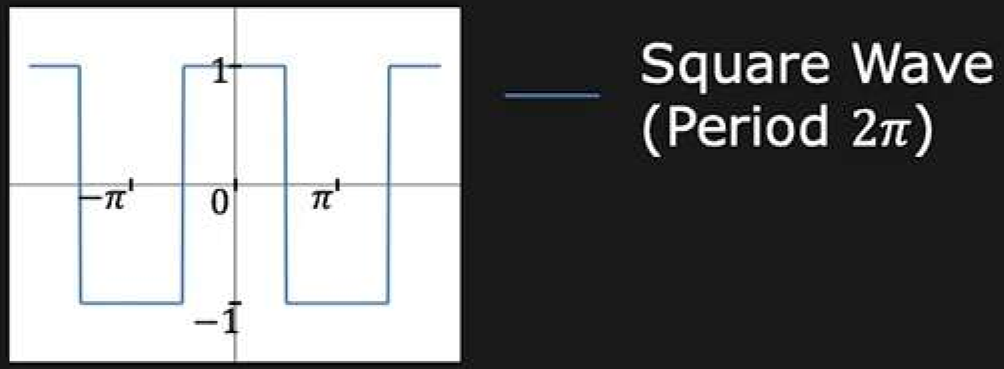
A: Amplitude

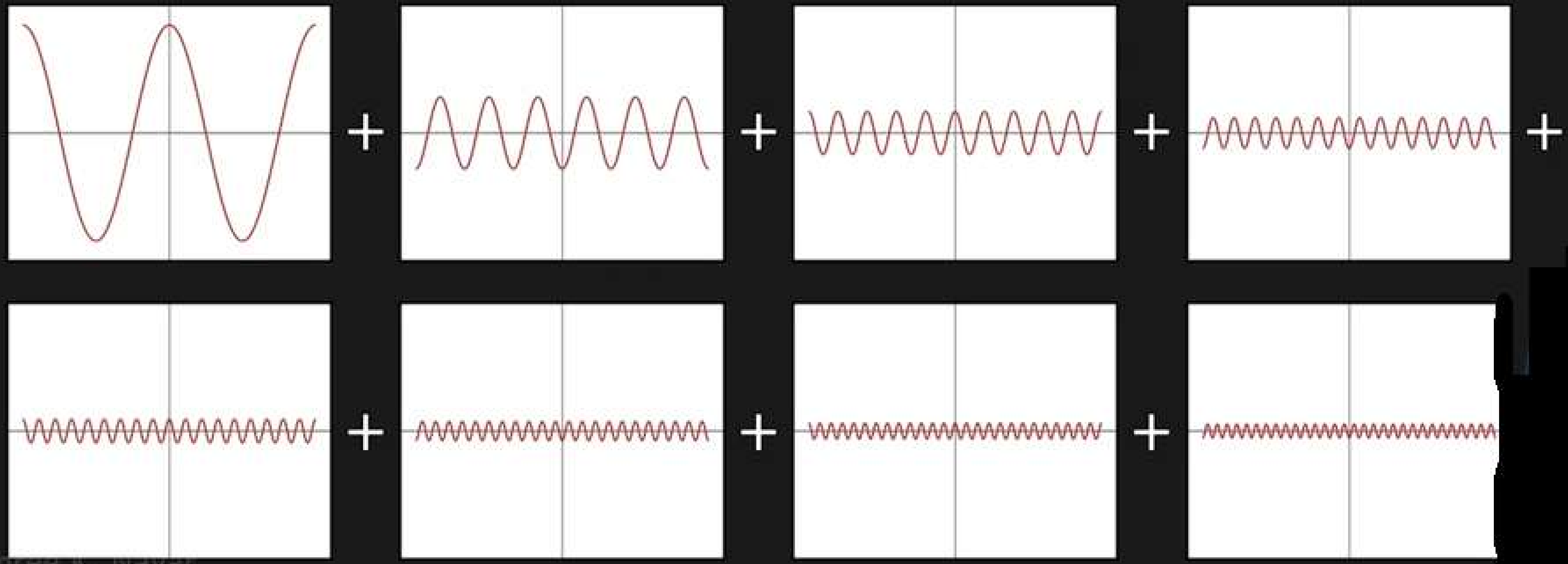
T: Period

φ: Phase

u: Frequency $(1/T)$

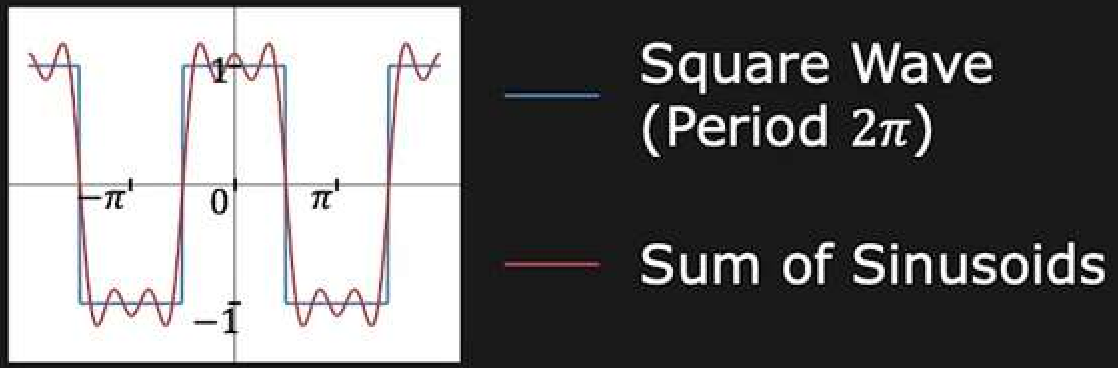
Fourier Series

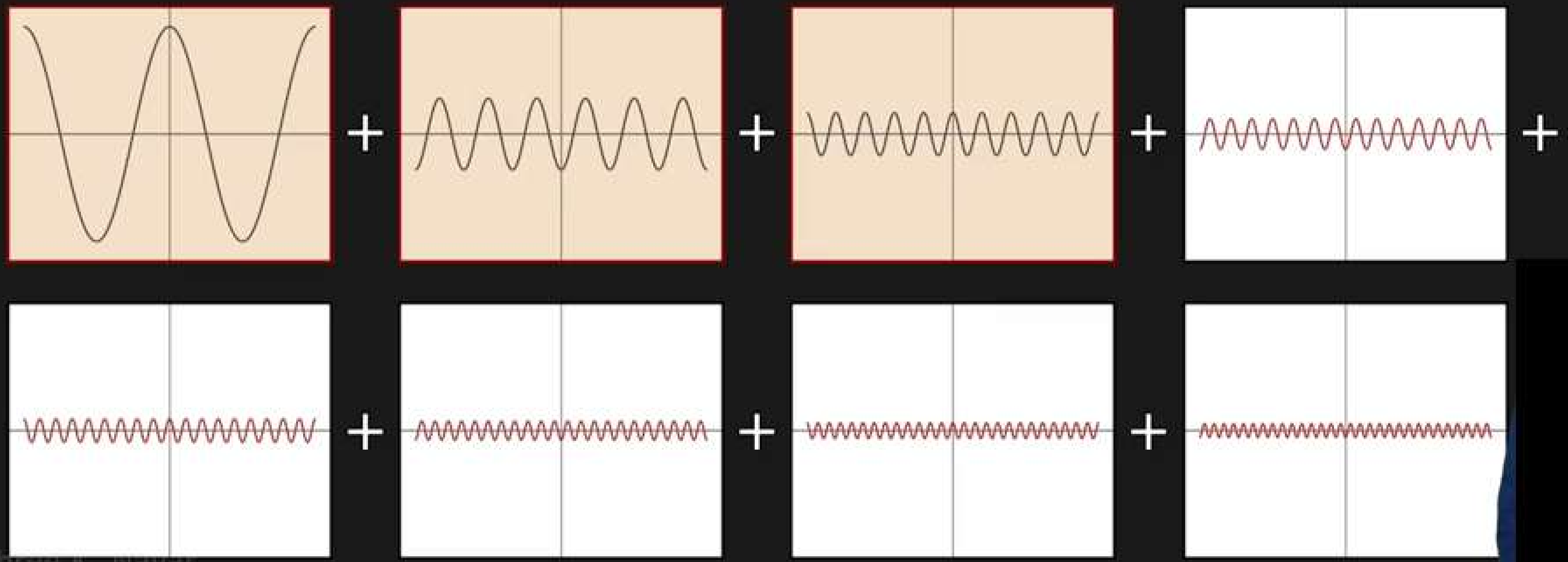




© 2020 Shree K. Nayar

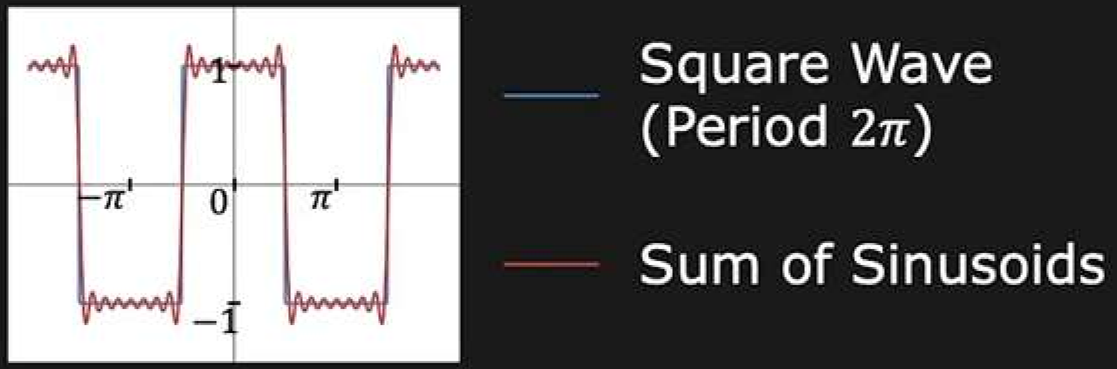
Fourier Series

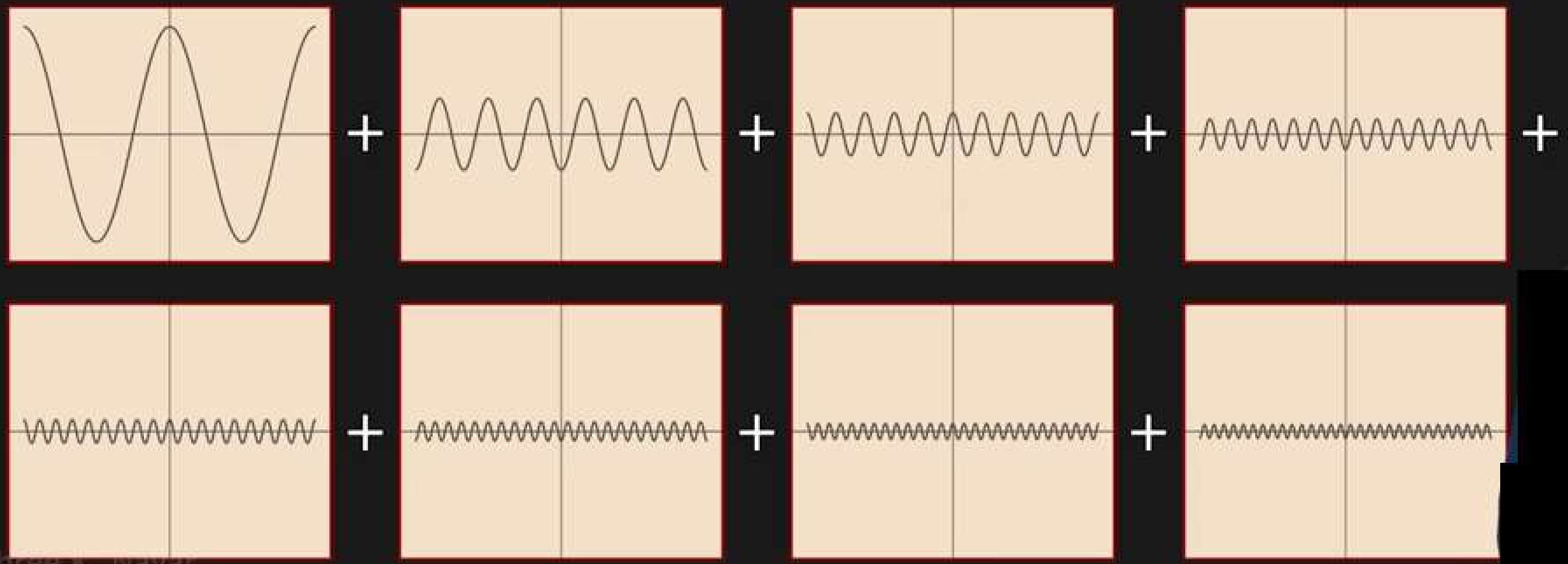




© 2020 Shree K. Nayar

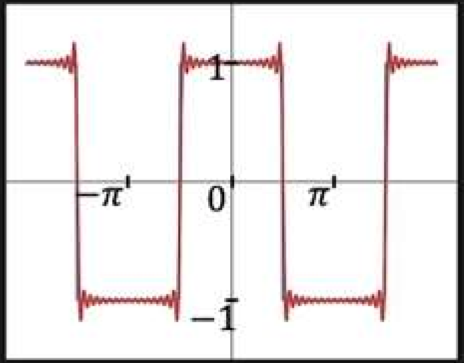
Fourier Series





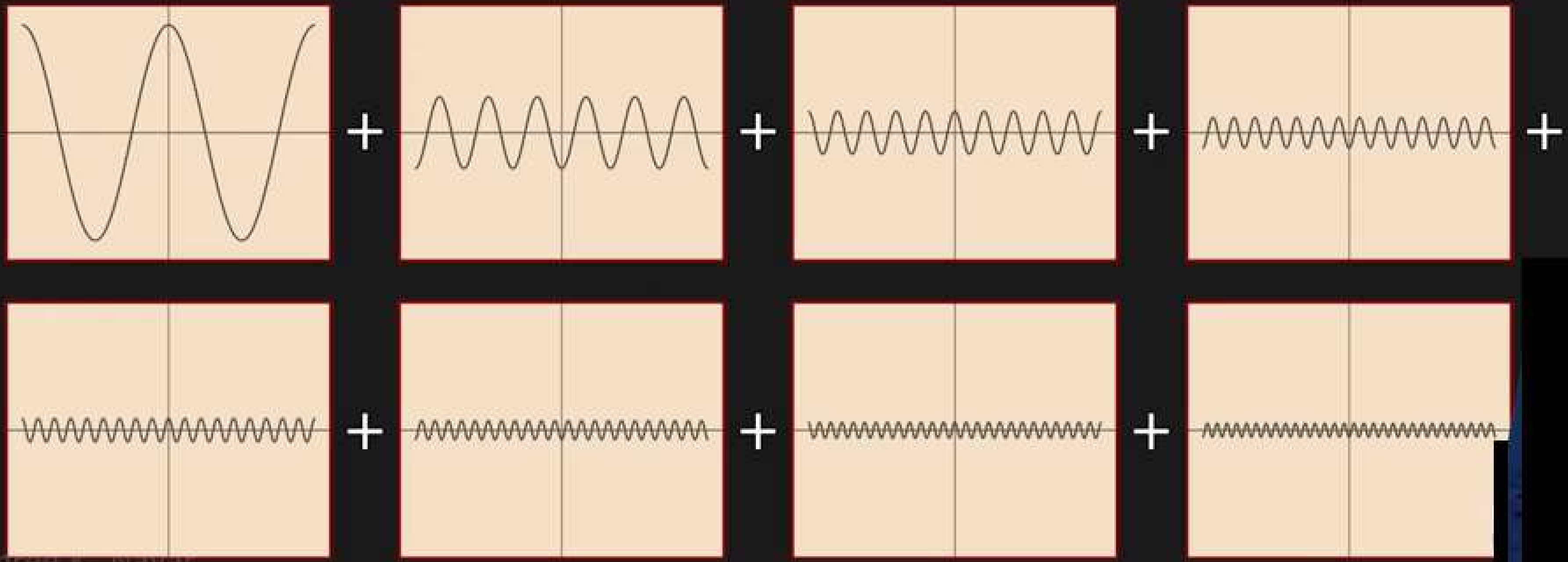
© 2020 Shree K. Nayar

Fourier Series



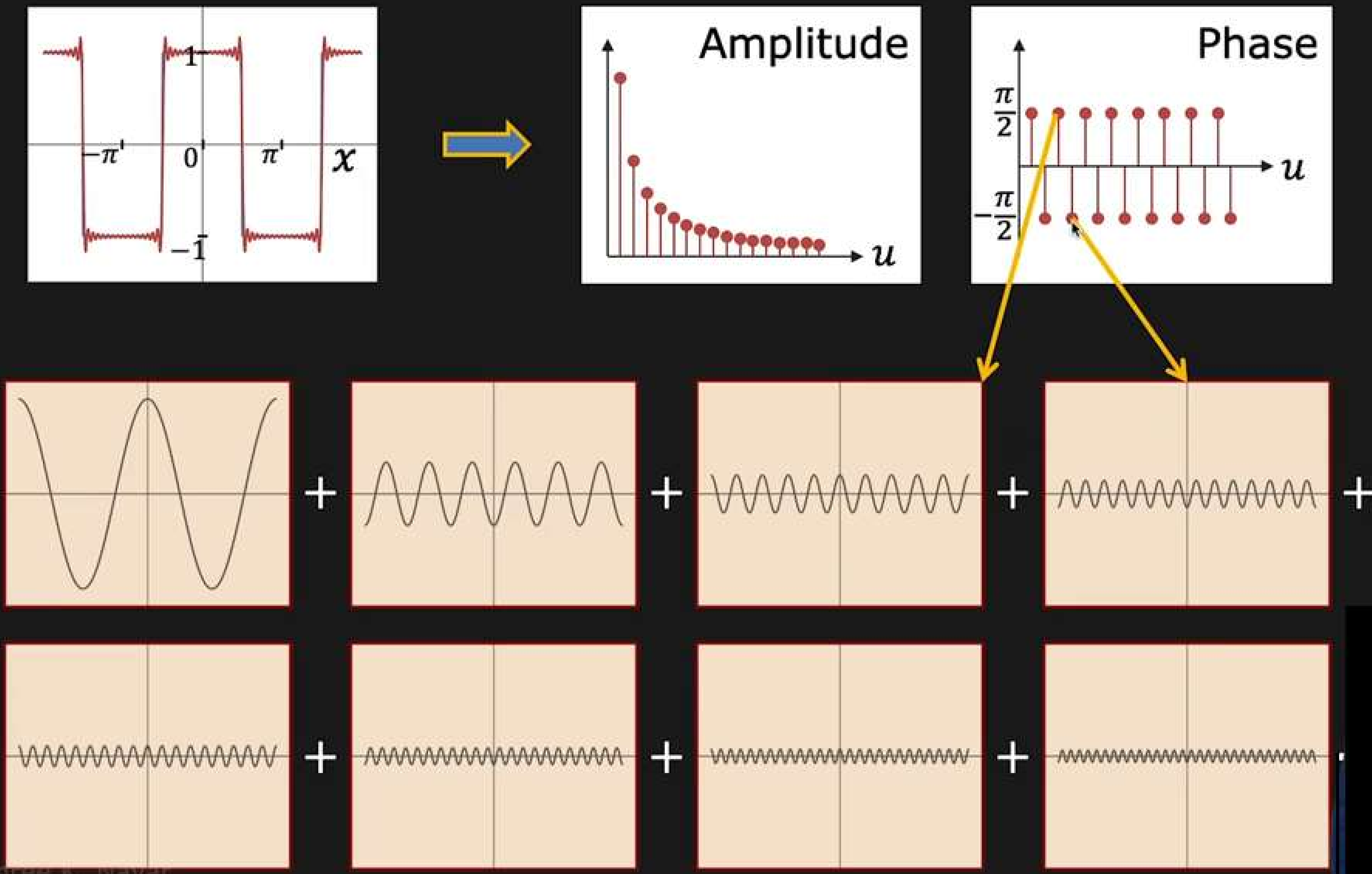
Square Wave (Period 2π)

Sum of Sinusoids



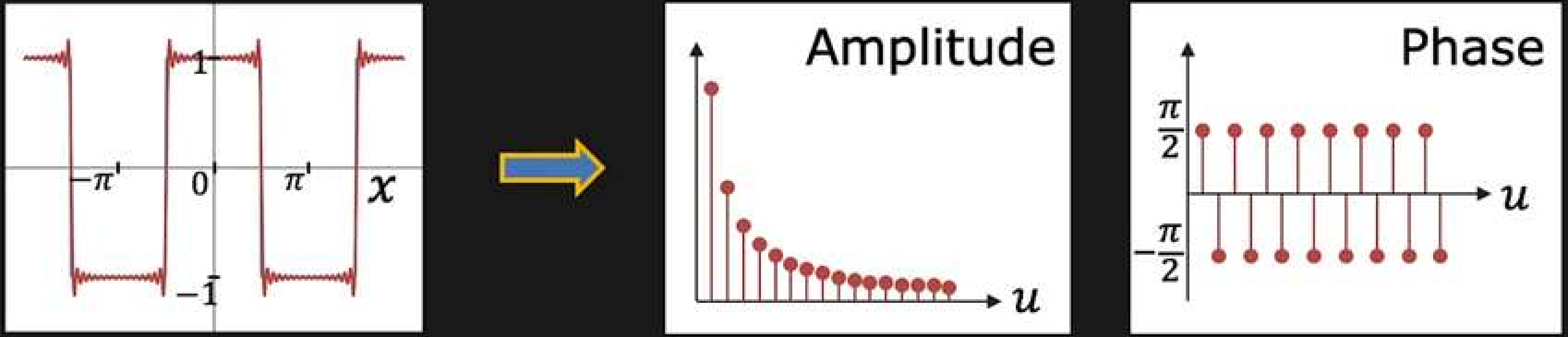
© 2020 Shree K. Nayar

Frequency Representation of Signal



© 2020 Shree K. Nayar

Fourier Transform (FT)

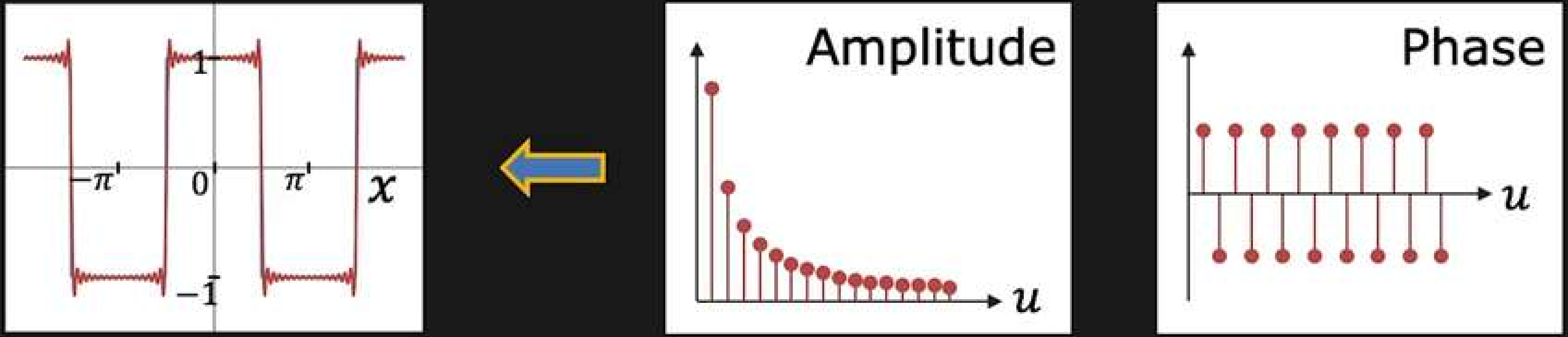


Represents a signal $f(x)$ in terms of Amplitudes and Phases of its Constituent Sinusoids.

$$f(x) \longrightarrow \boxed{\mathrm{FT}} \longrightarrow F(u)$$

© 2020 Shree K. Nayar

Inverse Fourier Transform (IFT)



Computes the signal $f(x)$ from the Amplitudes and Phases of its Constituent Sinusoids.

$$f(x) \longleftarrow \boxed{\text{IFT}} \longleftarrow F(u)$$

© 2020 Shree K. Nayar

Finding FT and IFT

Fourier Transform:

$$\boxed{F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx}$$

x: space

u: frequency

$e^{i\theta} = \cos\theta + i\sin\theta$

$i = \sqrt{-1}$

Inverse Fourier Transform:

$$\boxed{f(x) = \int_{-\infty}^{\infty} F(u) e^{i2\pi ux} du}$$

© 2020 Shree K. Nayar

Complex Exponential (Euler Formula)

$$\boxed{e^{i\theta} = \cos\theta + i\sin\theta} \qquad i = \sqrt{-1}$$

Expand $e^{i\theta}$ using Taylor Series:

$$e^{i\theta} = 1 + i\theta + \frac{(i\theta)^2}{2!} + \frac{(i\theta)^3}{3!} + \frac{(i\theta)^4}{4!} + \frac{(i\theta)^5}{5!} + \frac{(i\theta)^6}{6!} + \ldots$$

$$e^{i\theta} = \underbrace{\left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + \cdots\right)}_{\cos\theta} + i\underbrace{\left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \cdots\right)}_{\sin\theta}$$

Fourier Transform is Complex!

$F(u)$ holds the Amplitude and Phase of the sinusoid of frequency u.

$$\boxed{F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx}$$

$$F(u) = \Re e\{F(u)\} + i\, \Im m\{F(u)\}$$

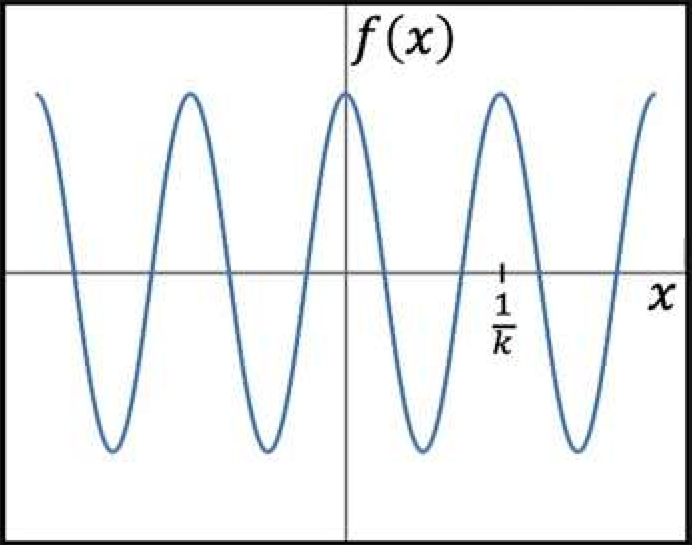
Amplitude: $A(u) = \sqrt{\Re e\{F(u)\}^2 + \Im m\{F(u)\}^2}$

Phase: $\varphi(u) = \text{atan2}(\Im m\{F(u)\}, \Re e\{F(u)\})$

© 2020 Shree K. Nayar

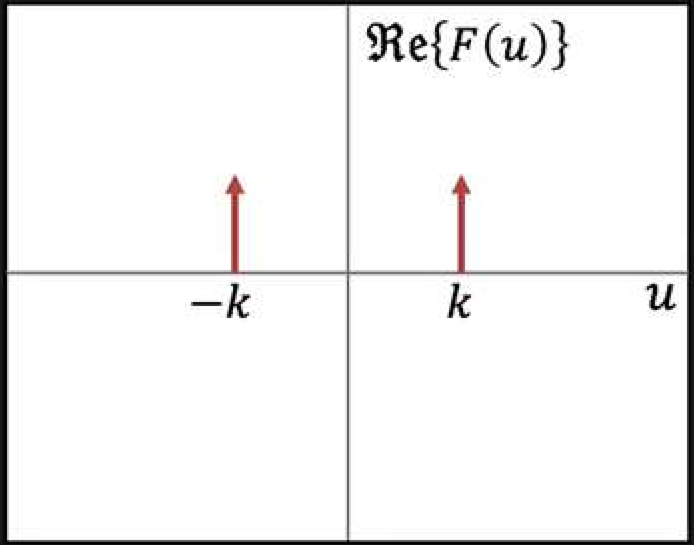
Fourier Transform Examples

Signal $f(x)$



$$f(x) = \cos 2\pi kx$$

Fourier Transform $F(u)$

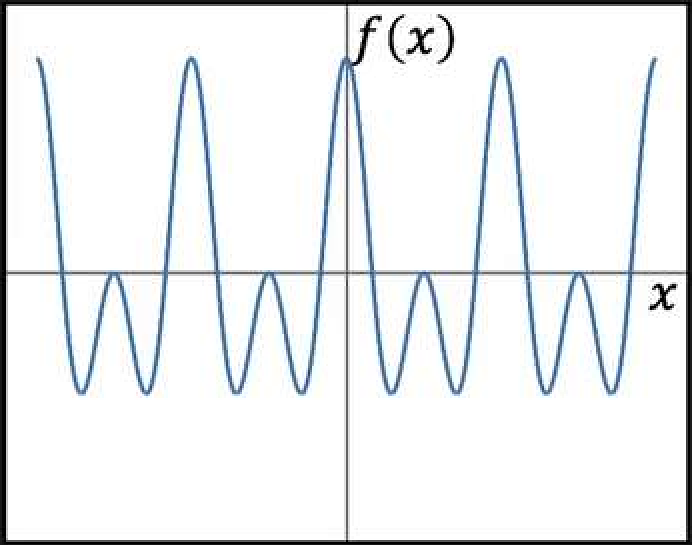


$$F(u) = \frac{1}{2}[\delta(u+k) + \delta(u-k)]$$

© 2020 Shree K. Nayar

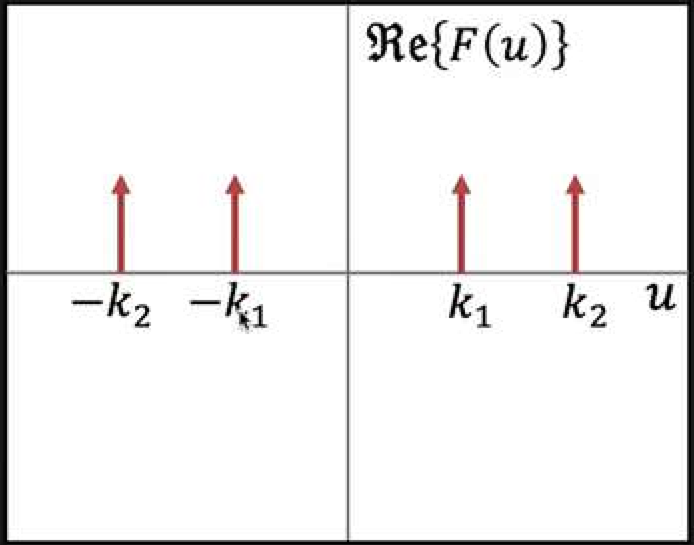
Fourier Transform Examples

Signal $f(x)$



$$f(x) = \cos 2\pi k_1 x \; + \; \cos 2\pi k_2 x$$

Fourier Transform $F(u)$

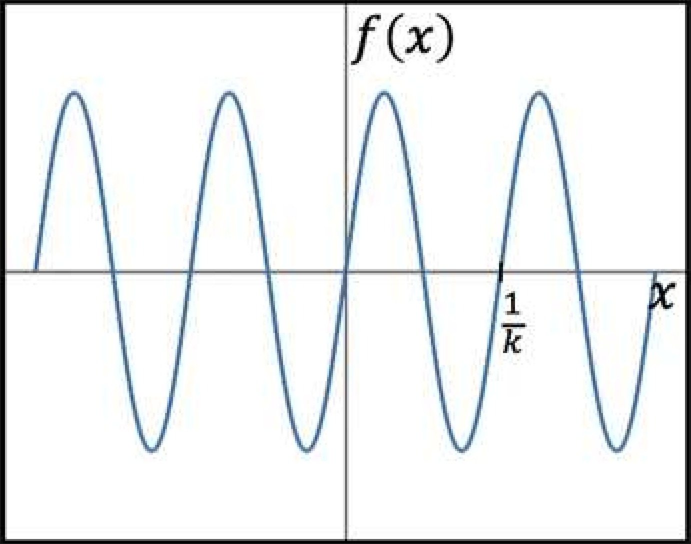


$F(u) = \frac{1}{2}\,[\delta(u + k_1) + \delta(u - k$ [redacted]
$+ \; \delta(u + k_2) + \delta(u - k_2$ [redacted]

© 2020 Shree K. Nayar

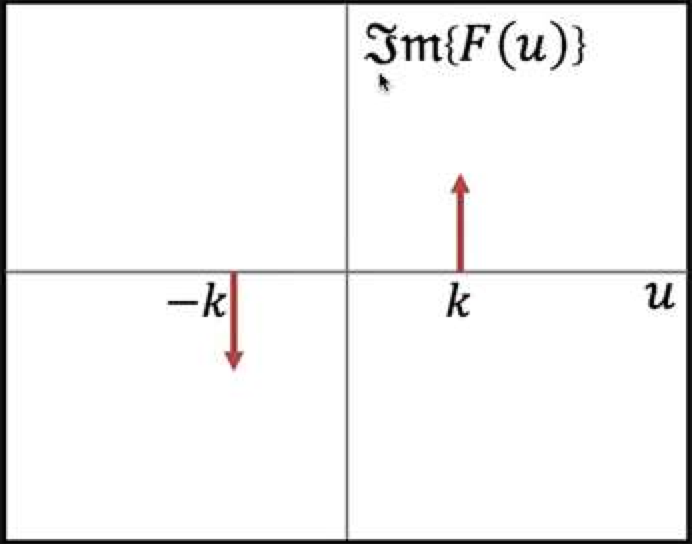
Fourier Transform Examples

Signal $f(x)$



$$f(x) = \sin 2\pi kx$$

Fourier Transform $F(u)$

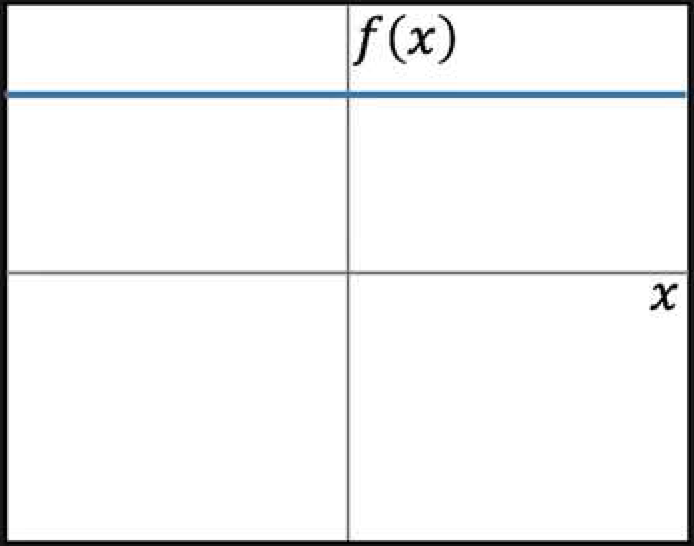


$$F(u) = \frac{1}{2}i[\delta(u+k) - \delta(u-k)$$ [redacted]

© 2020 Shree K. Nayar

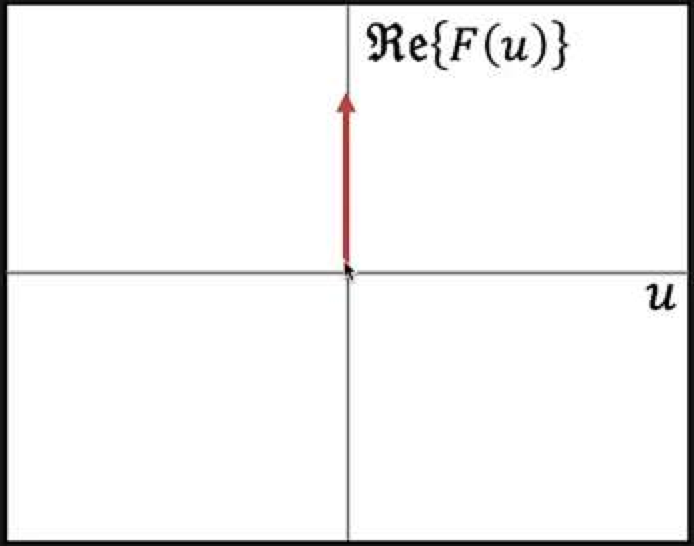
Fourier Transform Examples

Signal $f(x)$



$$f(x) = 1$$

Fourier Transform $F(u)$



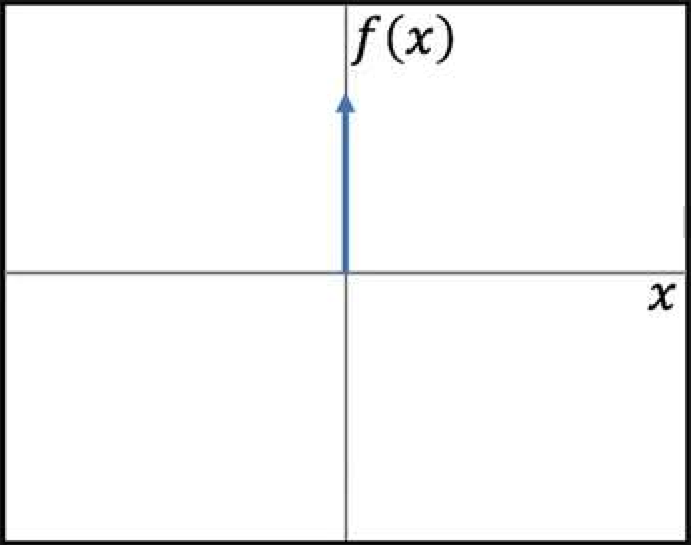
$$F(u) = \delta(u)$$

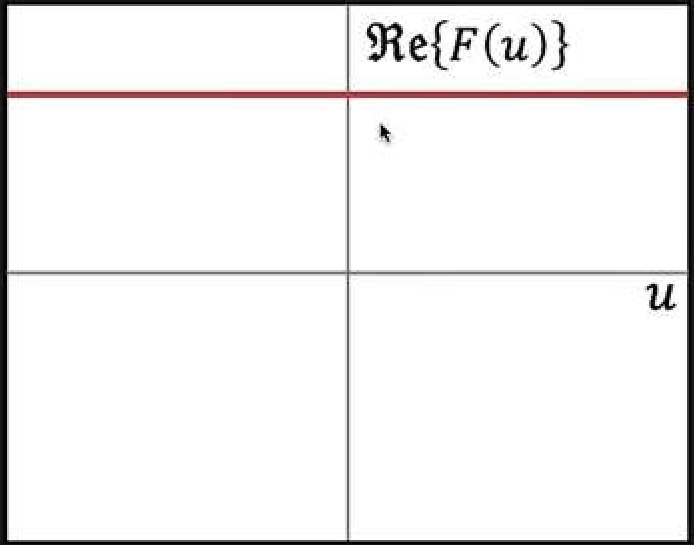
© 2020 Shree K. Nayar

Fourier Transform Examples

Signal $f(x)$

Fourier Transform $F(u)$





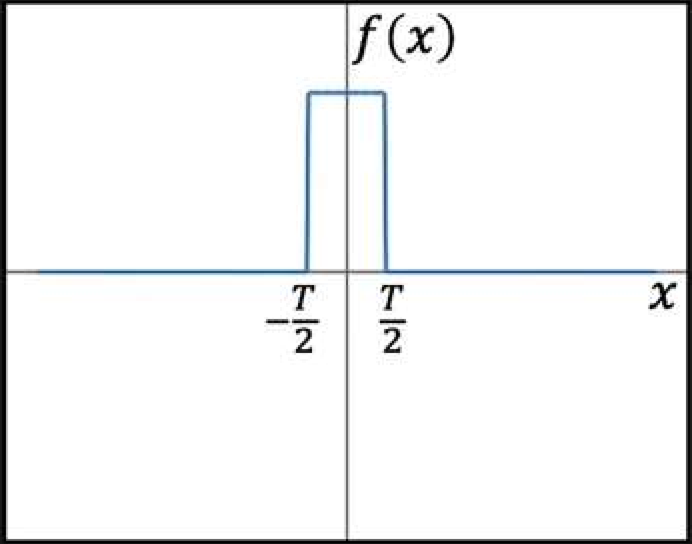
$$f(x) = \delta(x)$$

$$F(u) = 1$$

© 2020 Shree K. Nayar

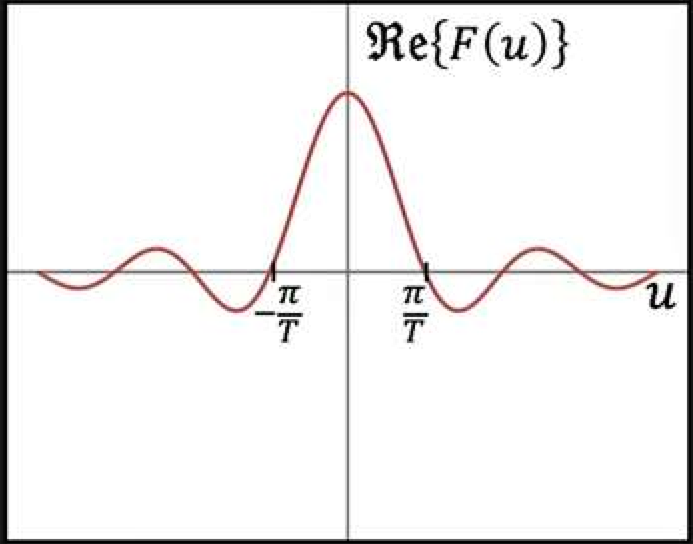
Fourier Transform Examples

Signal $f(x)$



$$f(x) = \mathrm{Rect}\left(\frac{x}{T}\right)$$

Fourier Transform $F(u)$

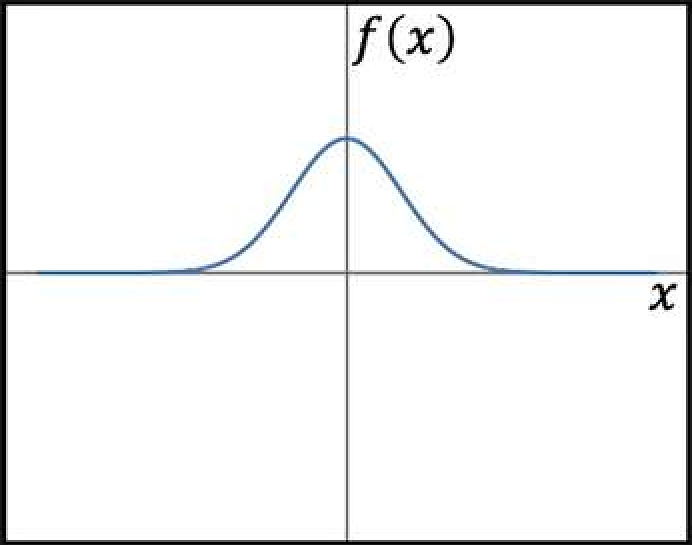


$$F(u) = T \operatorname{sinc} Tu$$

© 2020 Shree K. Nayar

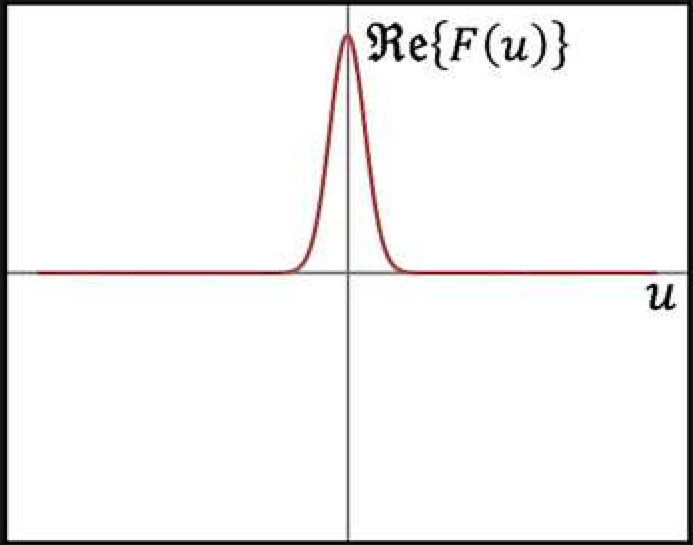
Fourier Transform Examples

Signal $f(x)$



$$f(x) = e^{-ax^2}$$

Fourier Transform $F(u)$



$$F(u) = \sqrt{\pi/a}\ e^{-\pi^2 u^2/a}$$

© 2020 Shree K. Nayar

Properties of Fourier Transform

Property	Spatial Domain	Frequency Domain
Linearity	$\alpha f_1(x) + \beta f_2(x)$	$\alpha F_1(u) + \beta F_2(u)$
Scaling	$f(ax)$	$\frac{1}{\lvert a \rvert} F\left(\frac{u}{a}\right)$
Shifting	$f(x-a)$	$e^{-i2\pi ua} F(u)$
Differentiation	$\frac{d^n}{dx^n}(f(x))$	$(i2\pi u)^n F(u)$

© 2020 Shree K. Nayar

2D Fourier Transform

Fourier Transform:

$$\boxed{F(u,v) = \iint_{-\infty}^{\infty} f(x,y)e^{-i2\pi(ux+vy)}dxdy}$$

u and v are frequencies along x and y, respectively

Inverse Fourier Transform:

$$\boxed{f(x,y) = \iint_{-\infty}^{\infty} F(u,v)e^{i2\pi(xu+yv)}dudv}$$

© 2020 Shree K. Nayar

2D Fourier Transform: Discrete Images

Discrete Fourier Transform (DFT):

$$F[p,q] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m,n] e^{-i2\pi pm/M} e^{-i2\pi qn/N}$$

$p = 0 \ldots M - 1$

$q = 0 \ldots N - 1$

p and q are frequencies along m and n, respectively

Inverse Discrete Fourier Transform (IDFT):

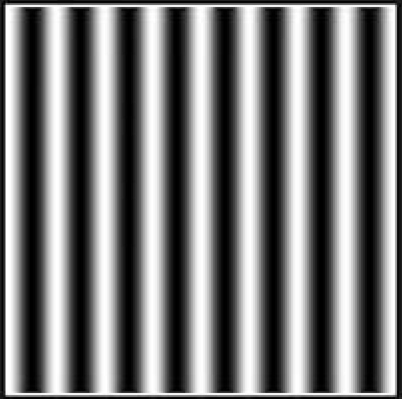
$$f[m,n] = \frac{1}{MN} \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} F[p,q] e^{i2\pi pm/M} e^{i2\pi qn/N}$$

$m = 0 \ldots M$ [redacted]

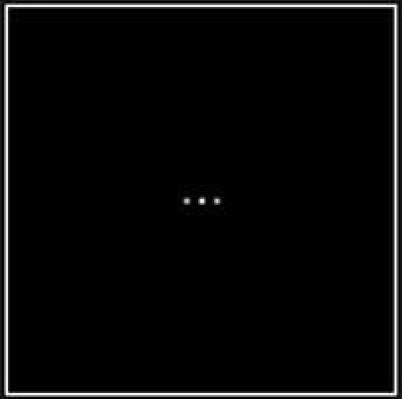
$n = 0 \ldots N -$ [redacted]

© 2020 Shree K. Nayar

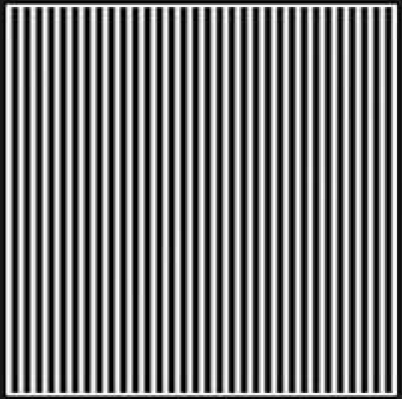
2D Fourier Transform: Example 1



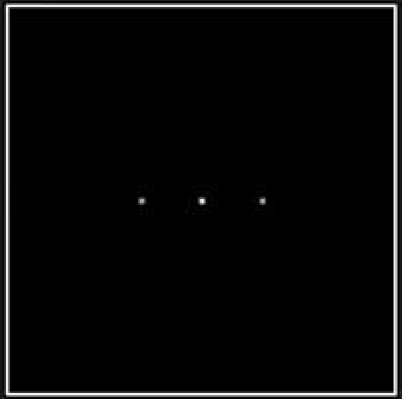
$f(m,n)$



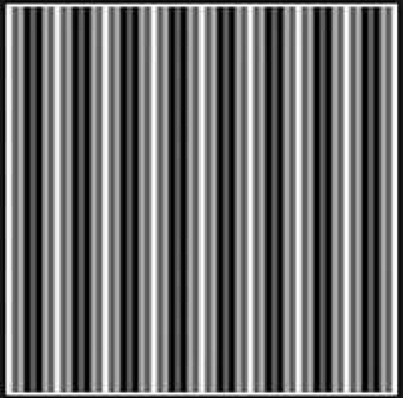
$\log(|F(p,q)|)$



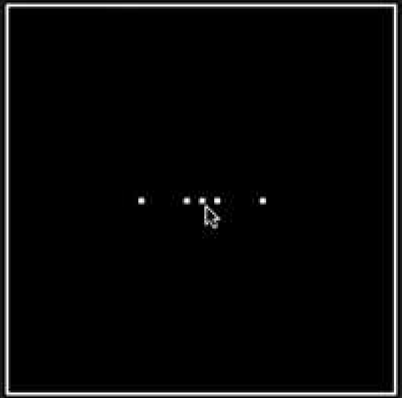
$g(m,n)$



$\log(|G(p,q)|)$



$f(m,n) + g(m,n)$

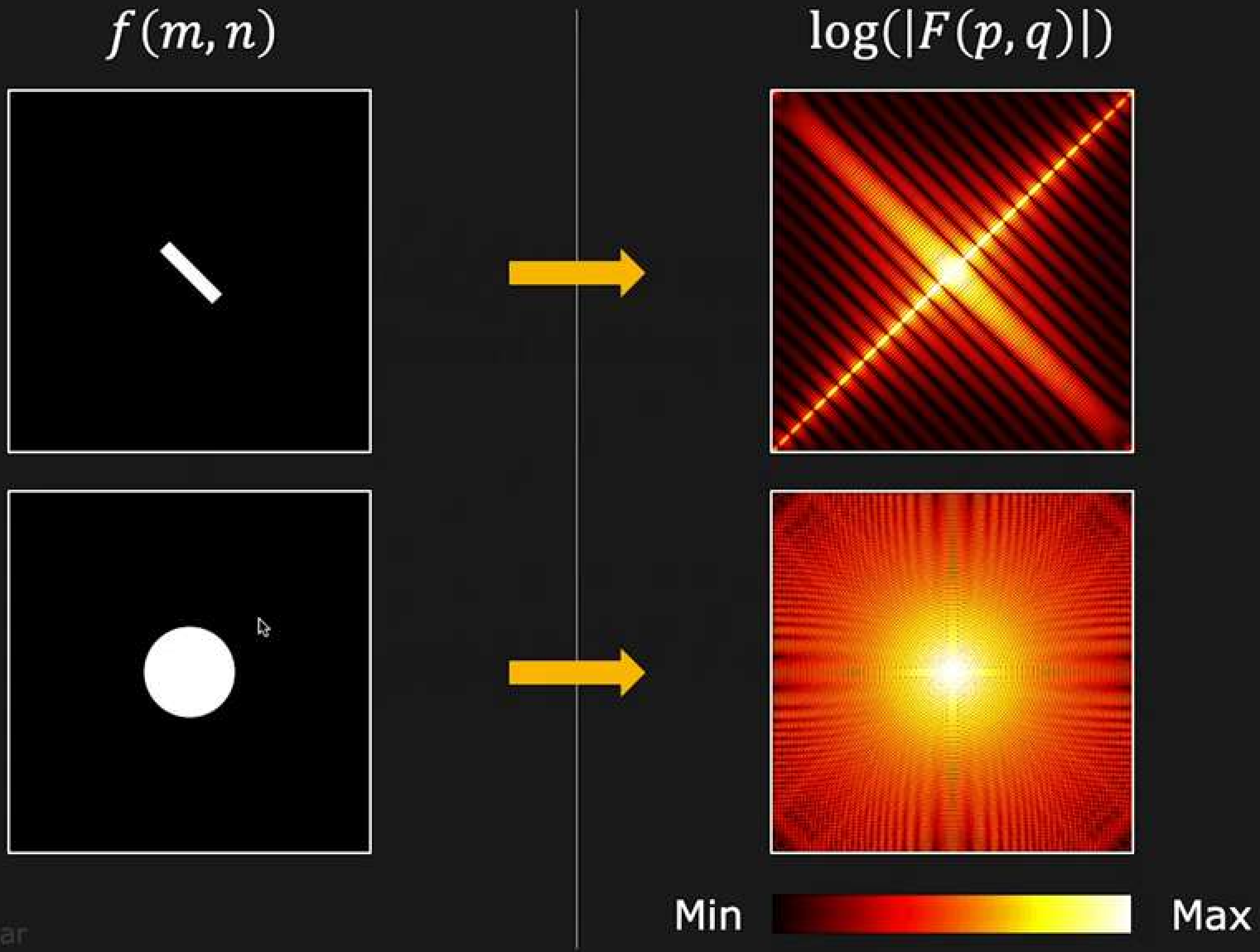


$\log(|F(p,q) + G(p,q)|)$

© 2020 Shree K. Nayar

Note: $\log(|F|)$ is used just for display

2D Fourier Transform: Example 2

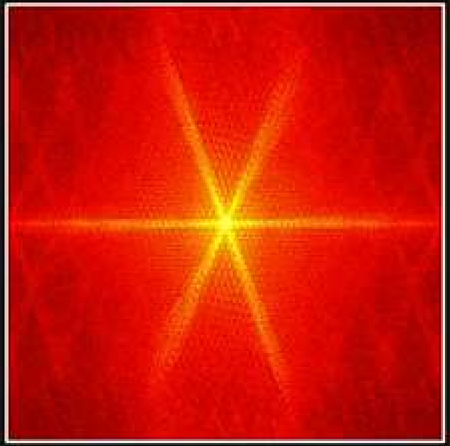


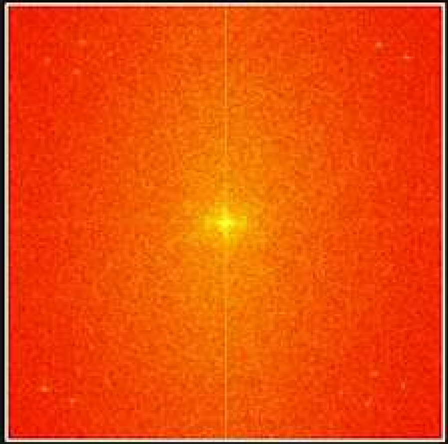
© 2020 Shree K. Nayar

2D Fourier Transform: Example 3

$f(m,n)$

$\log(|F(p,q)|)$



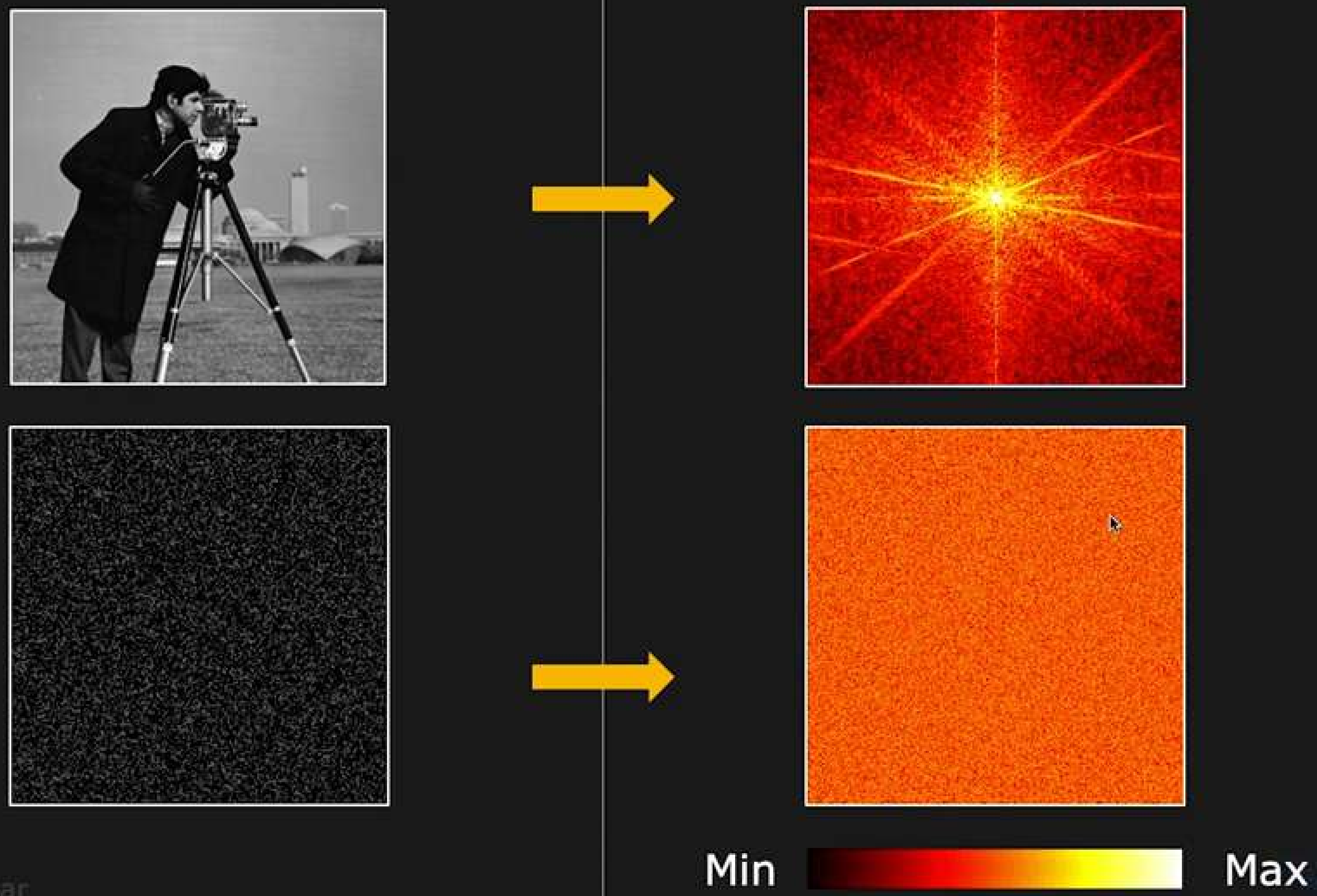


Min Max

© 2020 Shree K. Nayar

2D Fourier Transform: Example 4

$f(m,n)$ $\log(|F(p,q)|)$

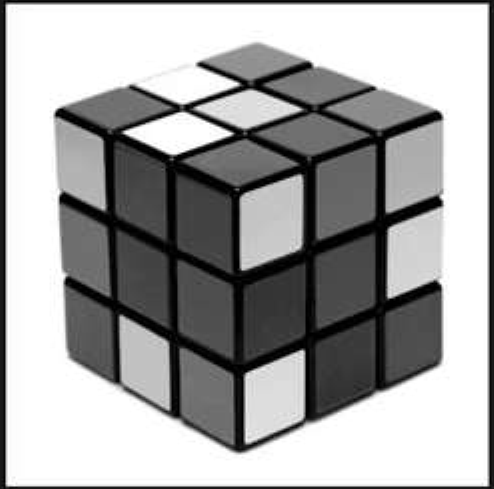


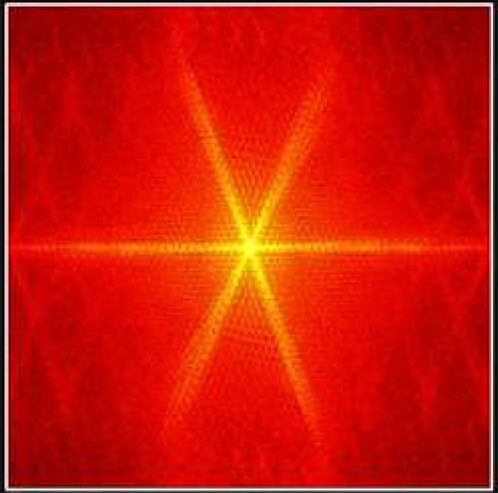
© 2020 Shree K. Nayar

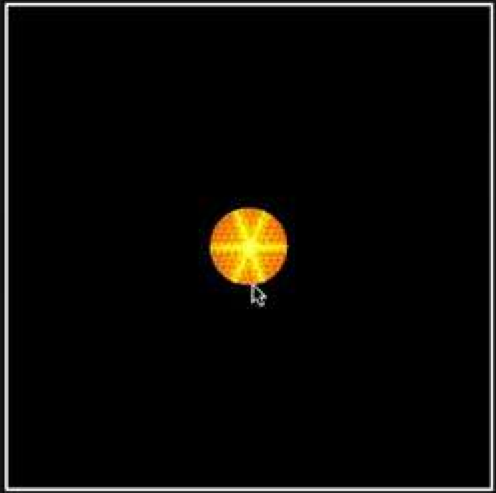
Low Pass Filtering

$f(m,n)$

$\log(|F(p,q)|)$



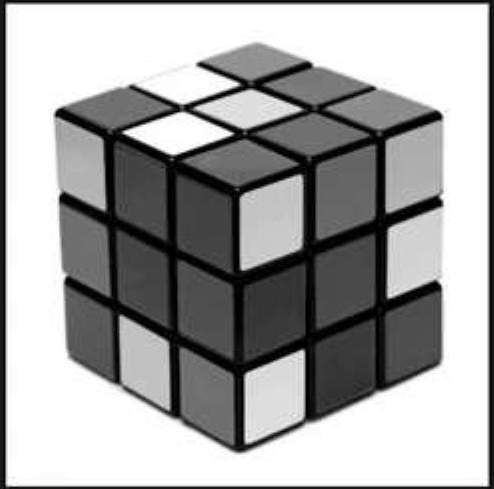




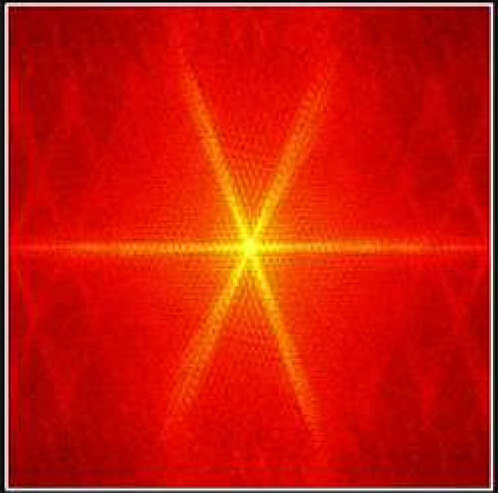
© 2020 Shree K. Nay

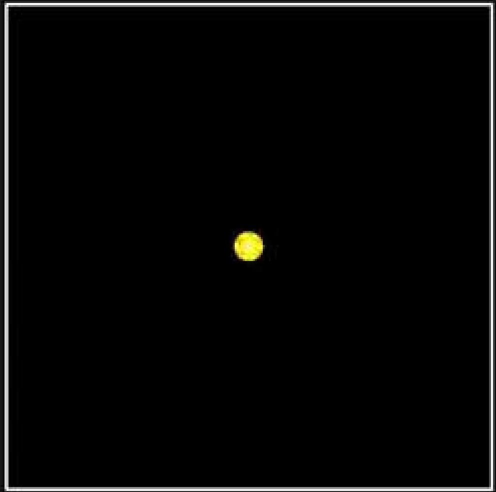
Low Pass Filtering

$f(m,n)$



$\log(|F(p,q)|)$

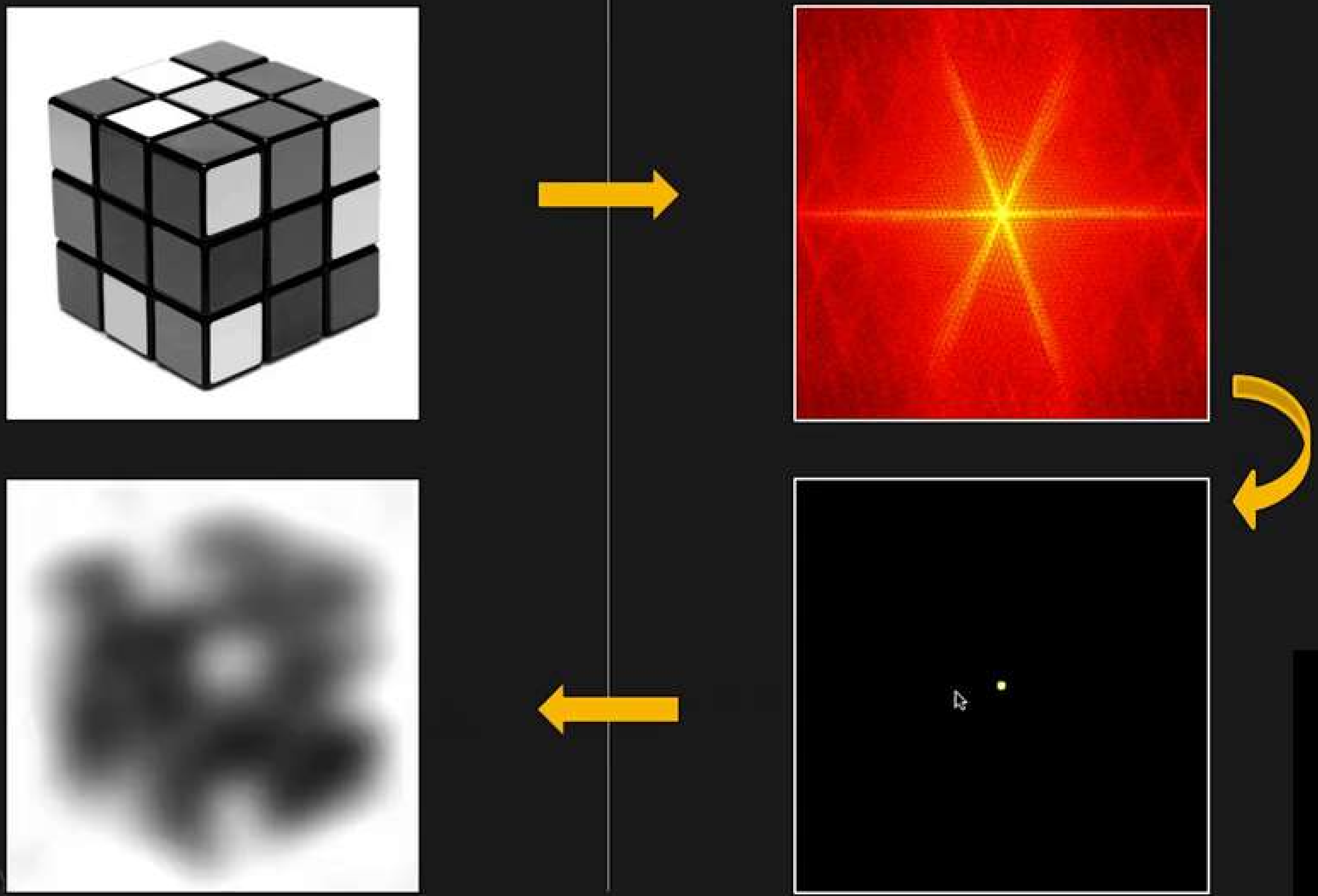




© 2020 Shree K. Nay

Low Pass Filtering

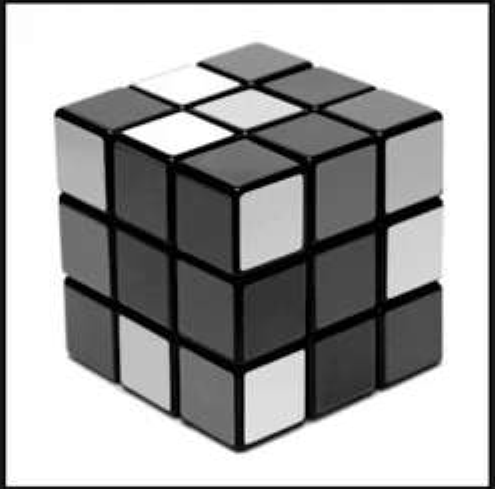
$f(m,n)$ $\log(|F(p,q)|)$



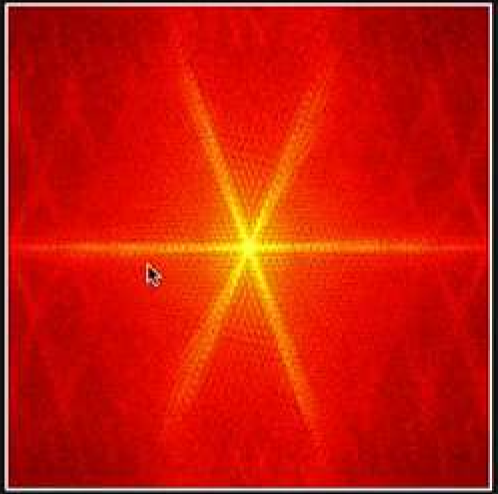
© 2020 Shree K. Nay

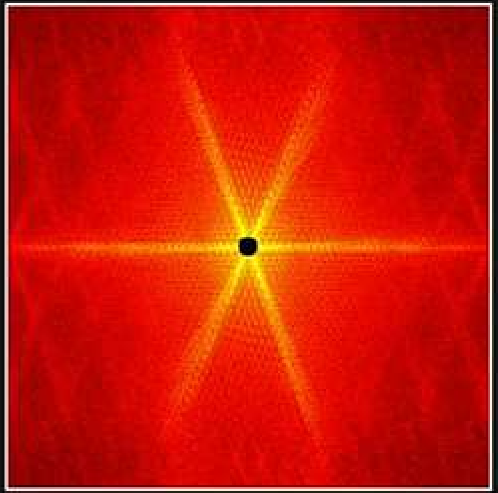
High Pass Filtering

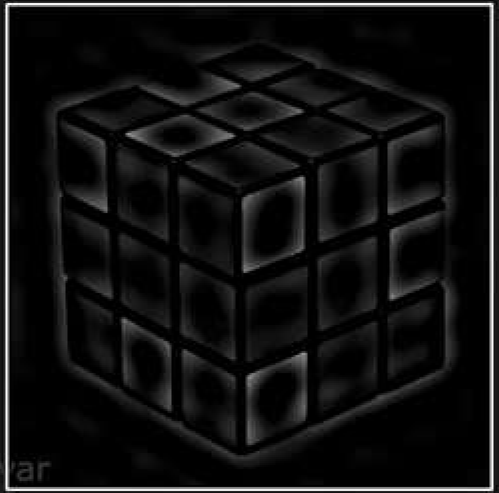
$f(m, n)$



$\log(|F(p, q)|)$



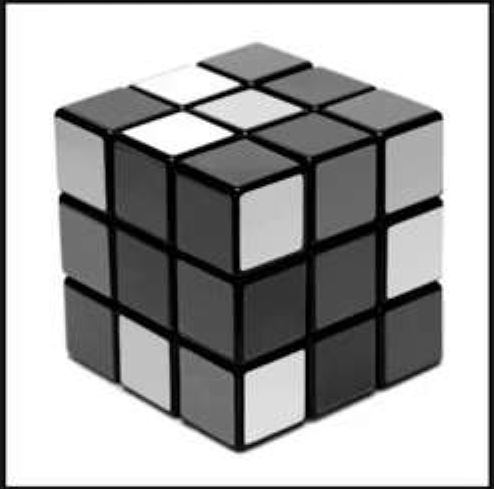




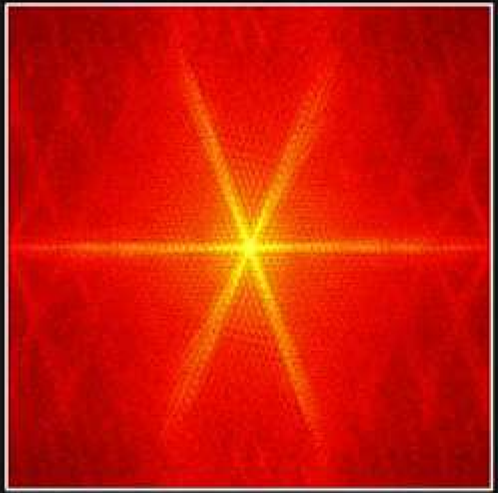
© 2020 Shree K. Nayar

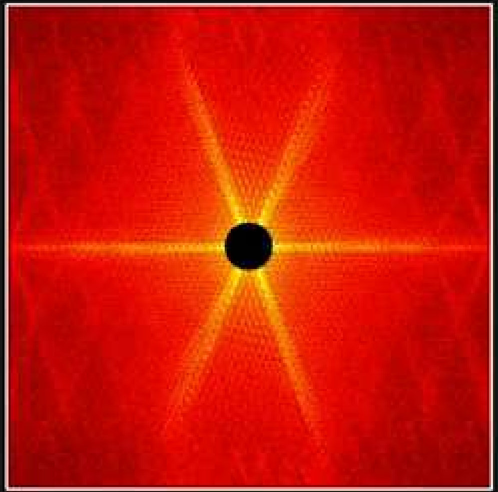
High Pass Filtering

$f(m,n)$



$\log(|F(p,q)|)$

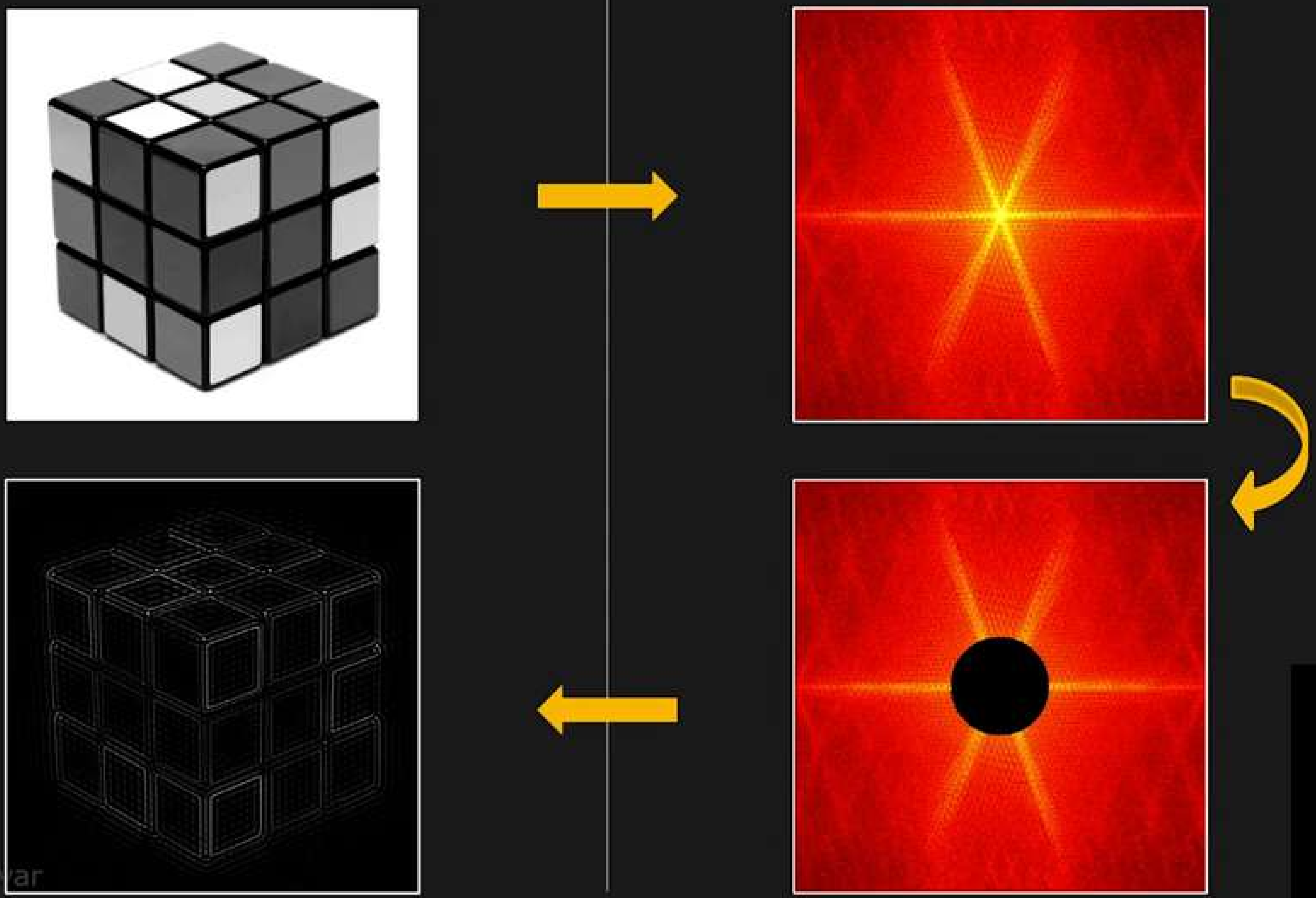




© 2020 Shree K. Nayar

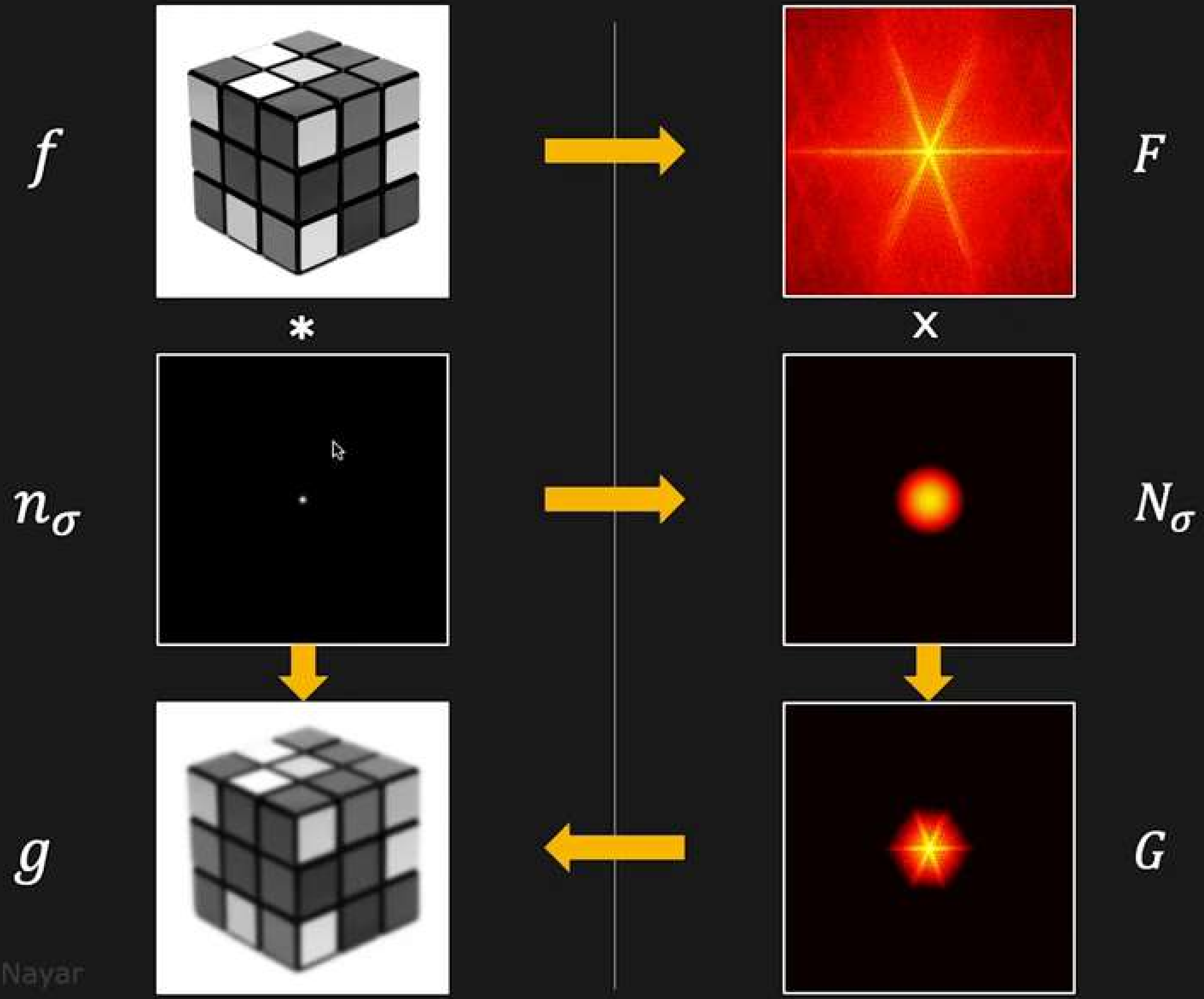
High Pass Filtering

$f(m,n)$ $\log(|F(p,q)|)$



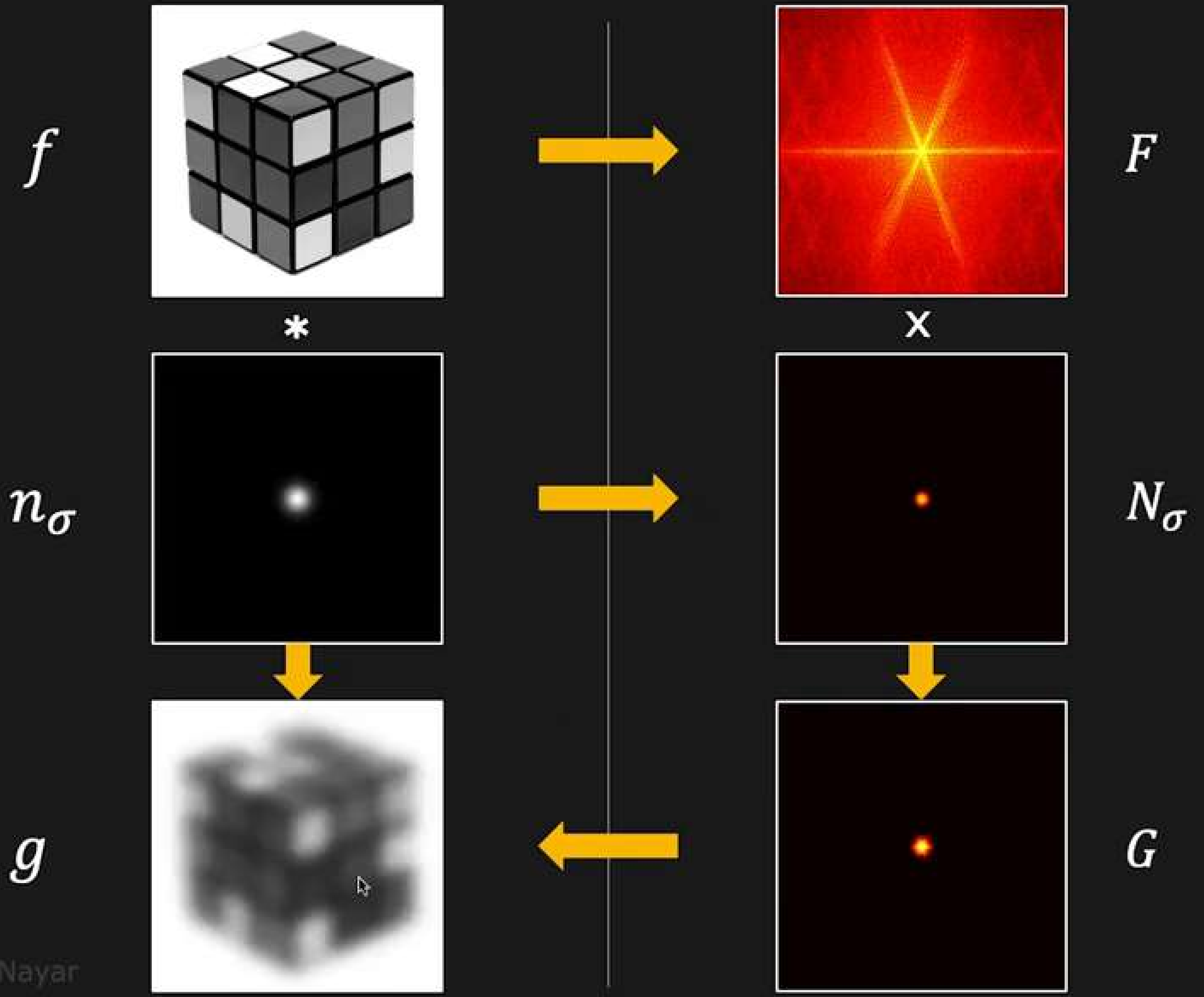
© 2020 Shree K. Nayar

Gaussian Smoothing



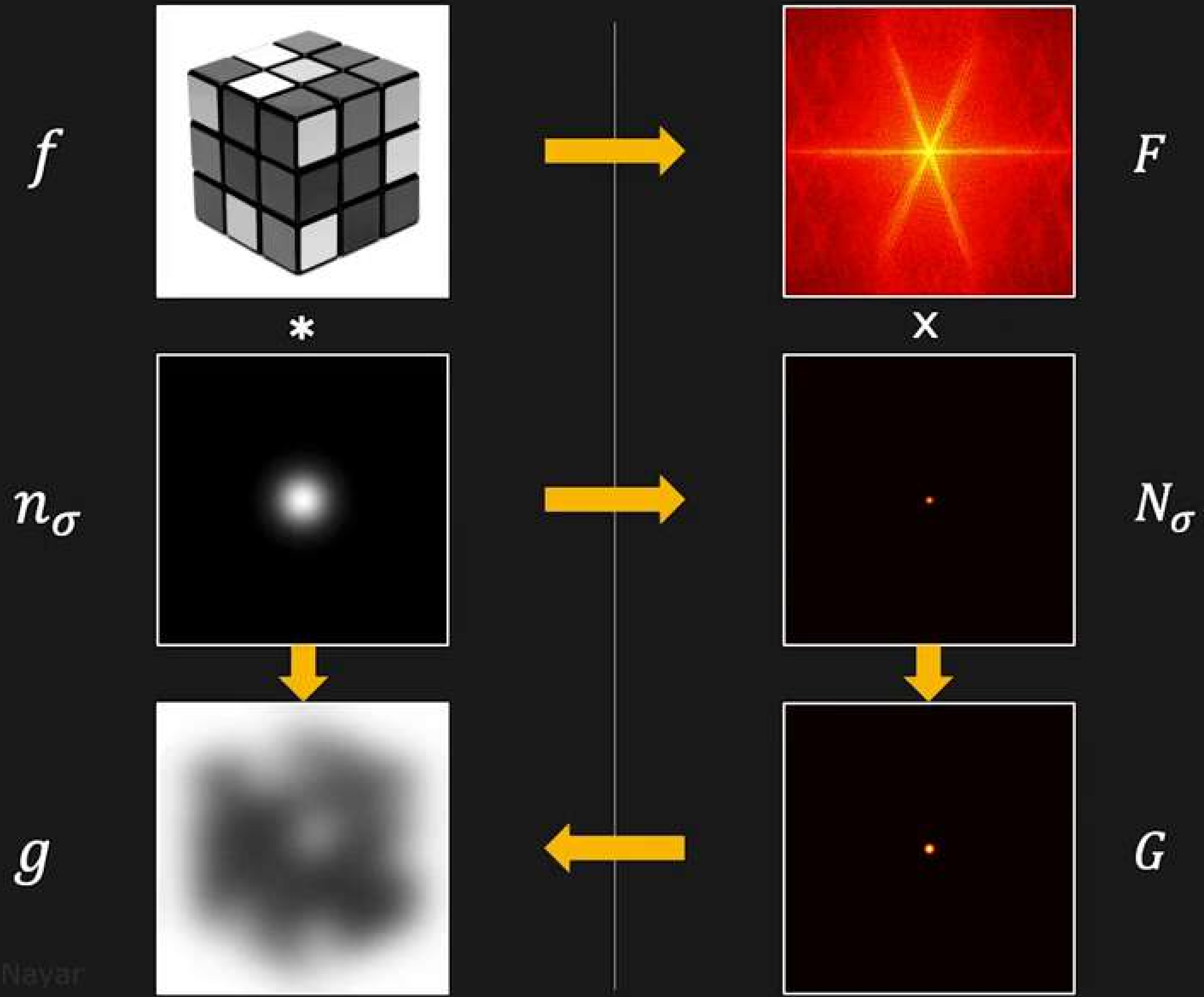
© 2020 Shree K. Nayar

Gaussian Smoothing



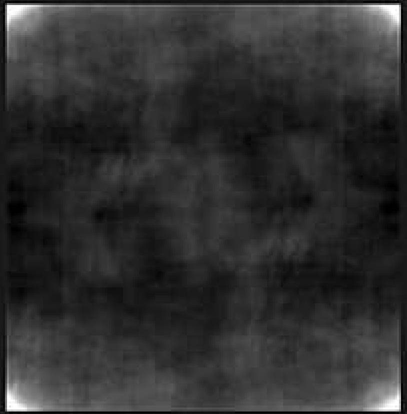
© 2020 Shree K. Nayar

Gaussian Smoothing



© 2020 Shree K. Nayar

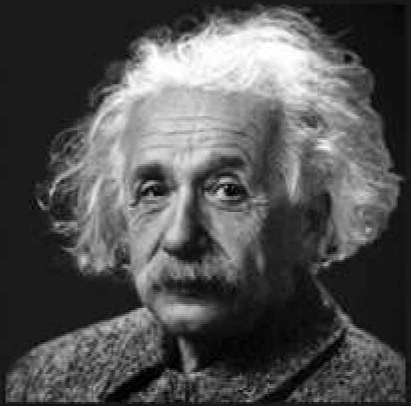
Importance of Phase

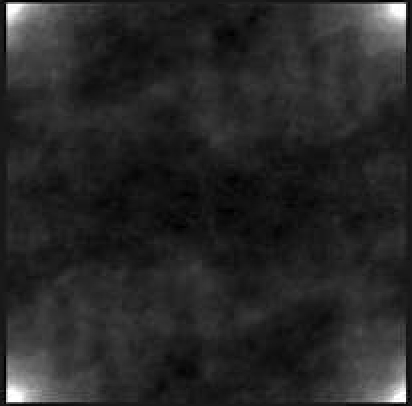


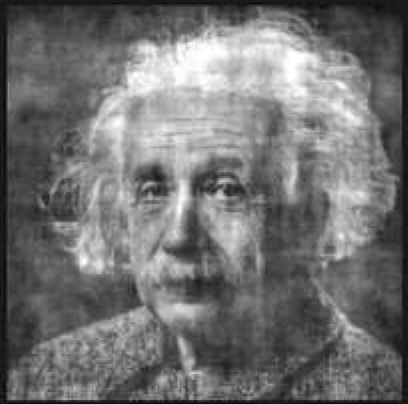
Original Image

Magnitude Preserved,
Phase Set to Zero

Phase Preserved,
Magnitude Set to Average
of Natural Images



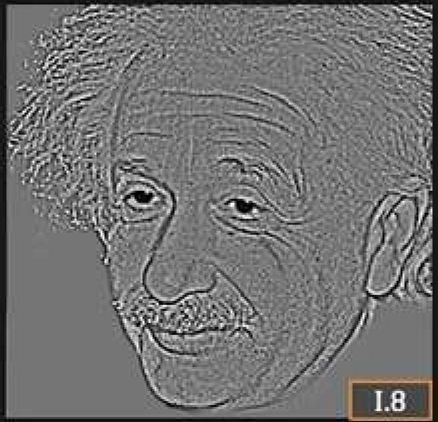




© 2020 Shree K. Nayar

Hybrid Images

Low Freq Only



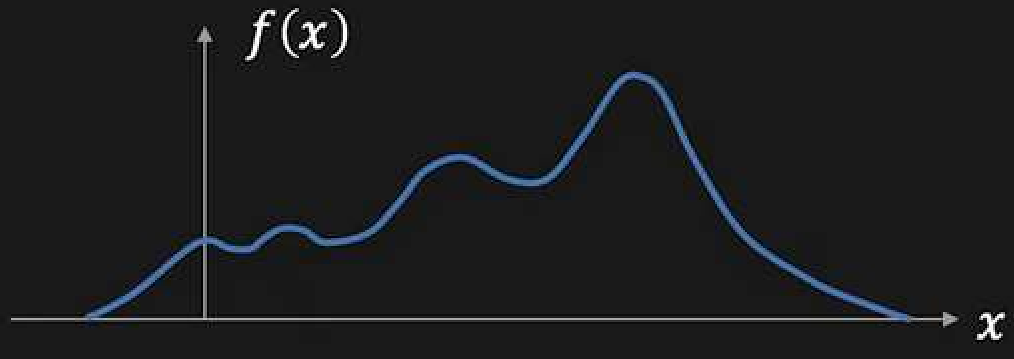
High Freq Only

Hybrid (Sum) Image

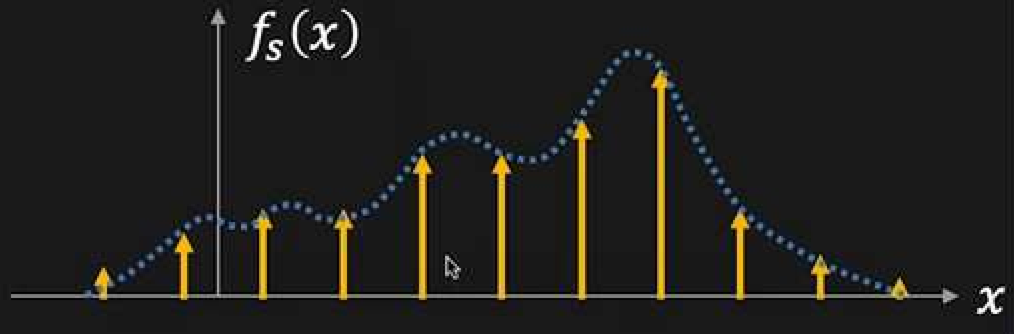
© 2020 Shree K. Nayar

From Continuous to Digital Image

Continuous Signal:



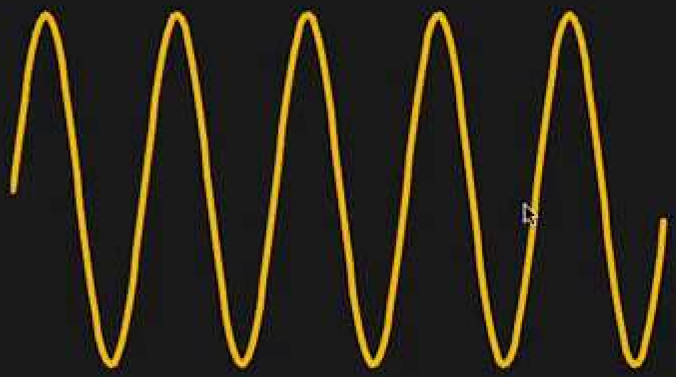
Digital Signal:



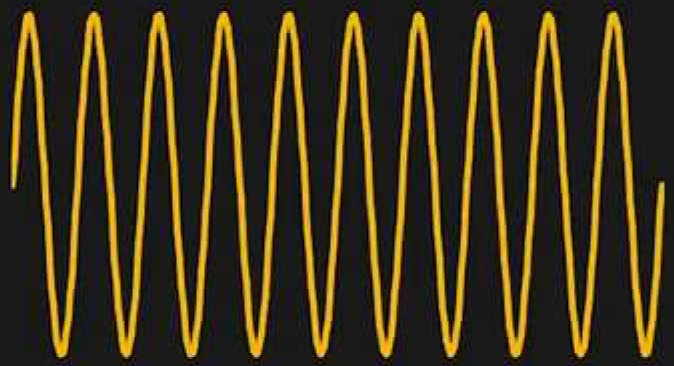
How "dense" should the samples be?

© 2020 Shree K. Nayar

Sampling Problem



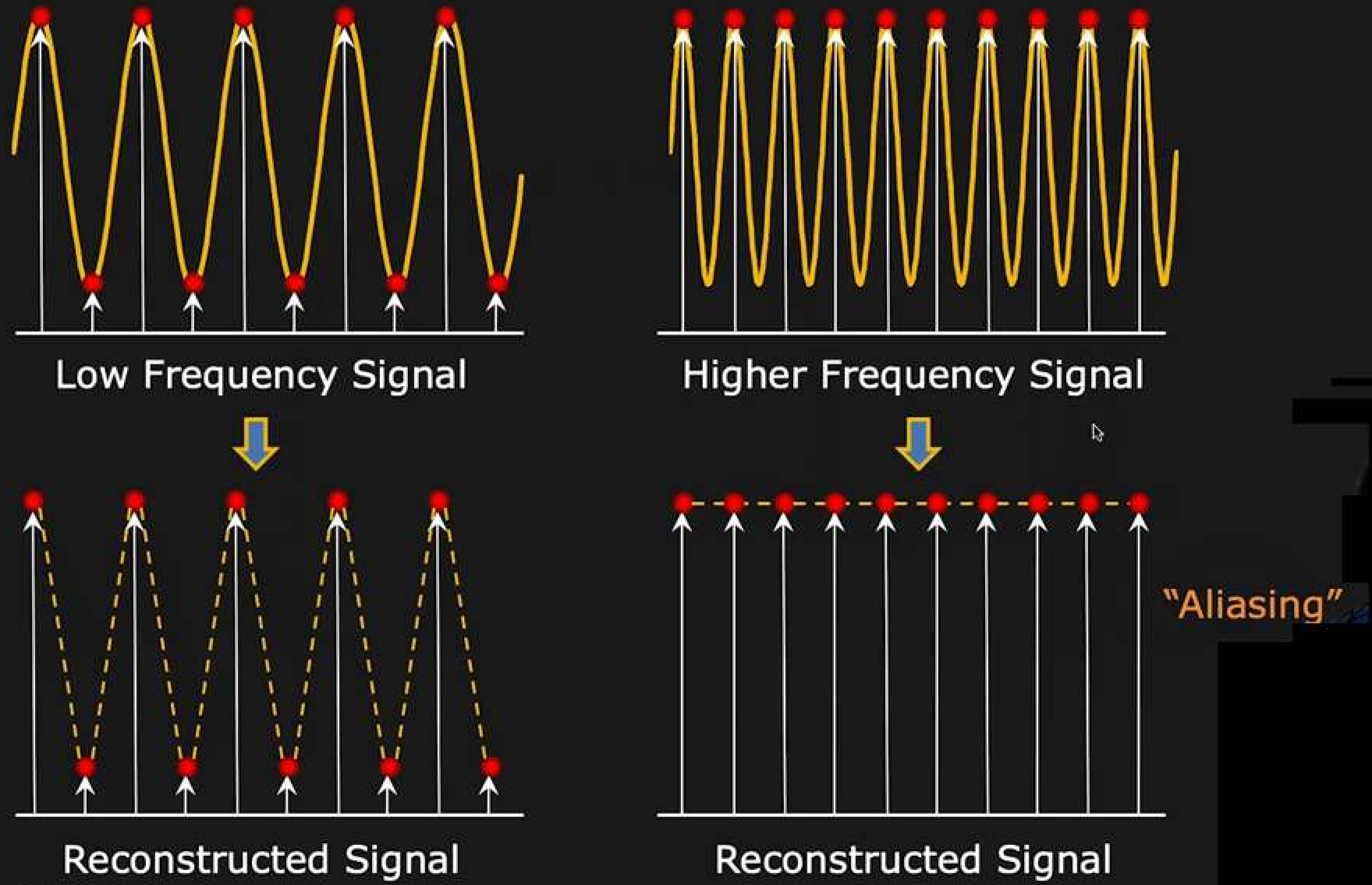
Low Frequency Signal



Higher Frequency Signal

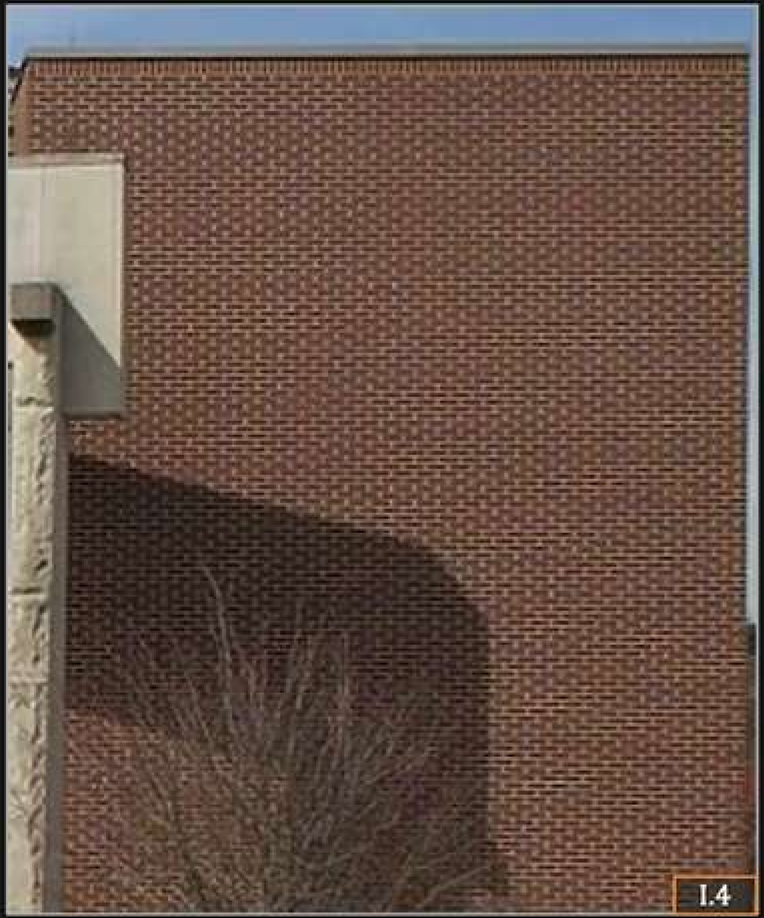
© 2020 Shree K. Nayar

Sampling Problem



© 2020 Shree K. Nayar

Sampling Problem



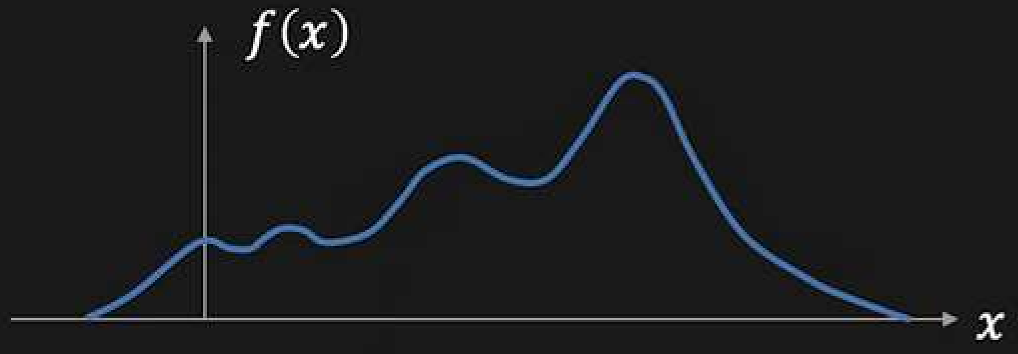
"Well sampled" image

"Under sampled" image
(visible aliasing artifacts)

© 2020 Shree K. Nayar

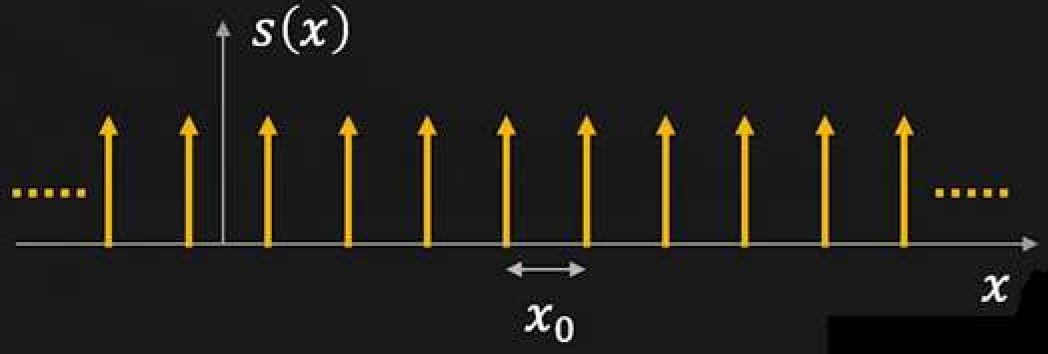
Sampling Theory

Continuous Signal:



Shah Function (Impulse Train):

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nx_0)$$



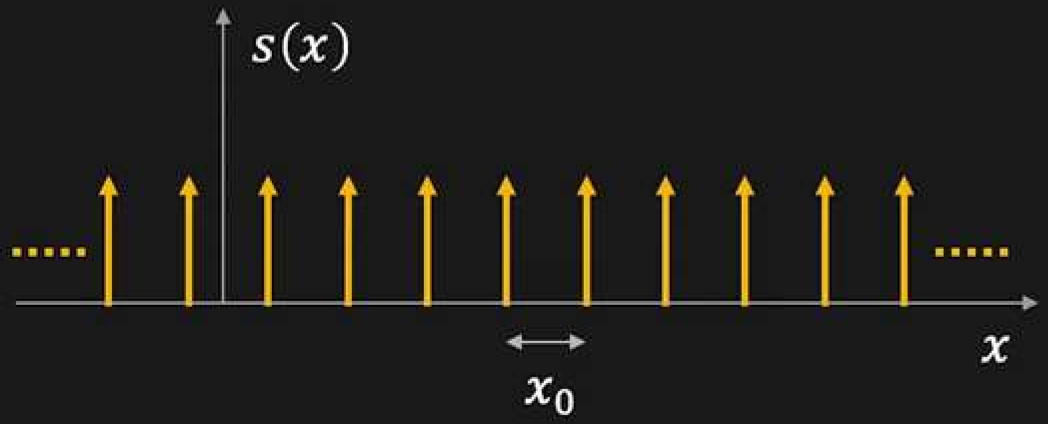
Sampled Function: $\boxed{f_s(x) = f(x)s(x)}$

© 2020 Shree K. Nayar

Shah Function (Impulse Train)

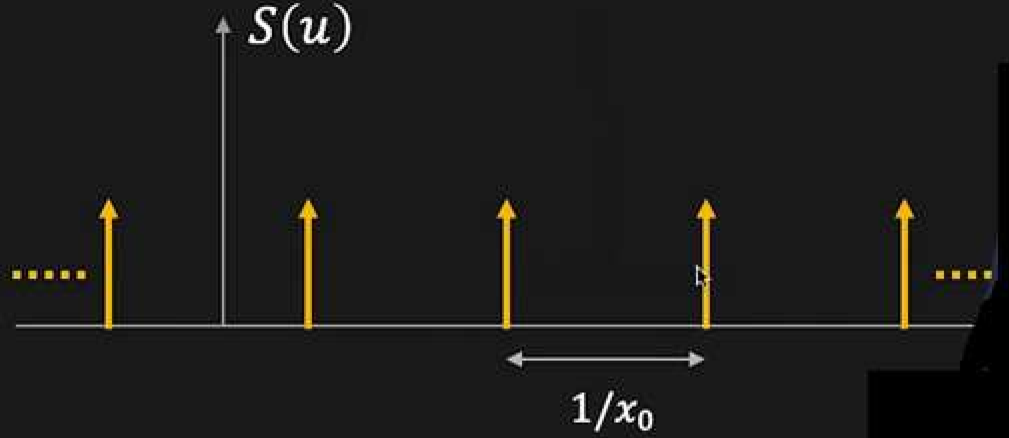
Shah Function (Spatial Domain):

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nx_0)$$



Shah Function (Fourier Domain):

$$S(u) = \frac{1}{x_0} \sum_{n=-\infty}^{\infty} \delta\left(x - \frac{n}{x_0}\right)$$



© 2020 Shree K. Nayar

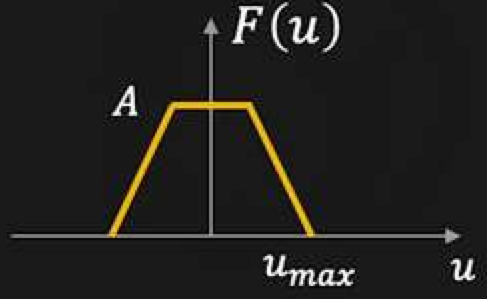
Fourier Analysis of Sampled Signal

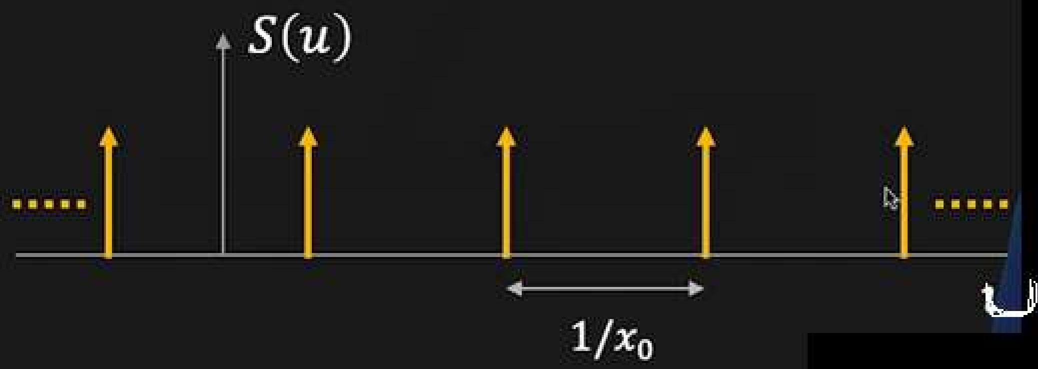
Sampled Signal:

$$f_s(x) = f(x)s(x) = f(x)\sum \delta(x - nx_0)$$

$$F_s(u) = F(u) * S(u) = F(u) * \frac{1}{x_0}\sum \delta(u - n/x_0)$$

For example:





© 2020 Shree K. Nayar

Fourier Analysis of Sampled Signal

Sampled Signal:

$$f_s(x) = f(x)s(x) = f(x)\sum \delta(x - nx_0)$$

$$F_s(u) = F(u) * S(u) = F(u) * \frac{1}{x_0}\sum \delta(u - n/x_0)$$

If $u_{max} \leq \frac{1}{2x_0}$

© 2020 Shree K. Nayar

Aliasing

Sampled Signal:

$$f_s(x) = f(x)s(x) = f(x)\sum \delta(x - nx_0)$$

$$F_s(u) = F(u) * S(u) = F(u) * \frac{1}{x_0}\sum \delta(u - n/x_0)$$

If $u_{max} > \frac{1}{2x_0}$ Aliasing

© 2020 Shree K. Nayar

Nyquist Theorem

Can we recover $f(x)$ from $f_s(x)$? In other words, can we recover $F(u)$ from $F_s(u)$?

Only if $u_{max} \leq \frac{1}{2x_0}$ (Nyquist Frequency)

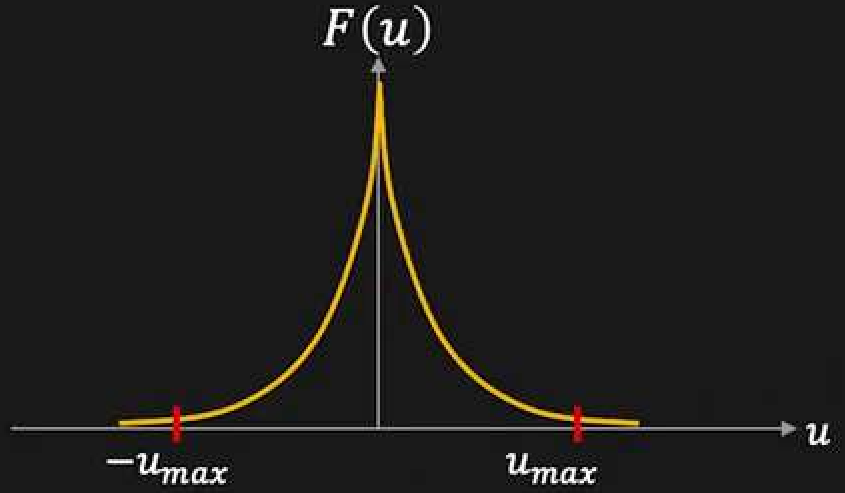
$$F(u) = F_s(u)C(u)$$

$$C(u) = \begin{cases} x_0, & |u| < 1/2x_0 \\ 0, & Otherwise \end{cases}$$

© 2020 Shree K. Nayar

Aliasing in Digital Imaging

Aliasing occurs when imaging a scene (signal) that has frequencies above the image sensor's Nyquist Frequency



Typical Power Spectrum of Natural Scenes

Aliasing artifacts usually occur in the form of Moiré patterns

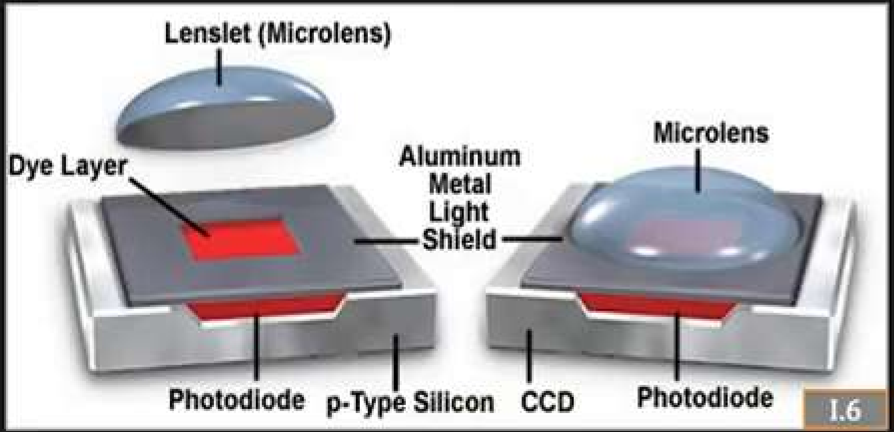
© 2020 Shree K. Nayar

Minimizing the Effects of Aliasing

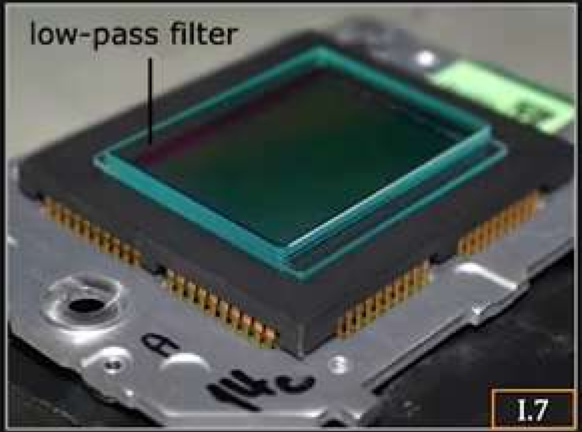
Band Limit: Clip the signal above the Nyquist frequency.

Effectively, "blur" the scene before sampling.

Sensors use two strategies.



Pixels are area-samplers
(box-averaging filter)



Use optical low-pass filte[redacted]
(anti-aliasing filter)

© 2020 Shree K. Nayar

Thank You

Kumar Bipin | linux.kbp@gmail.com