# Large-scale Data-driven Graphics and Vision: Basics of Image, Video, and Optics

Connelly Barnes

# Basics of Image, Video, Optics

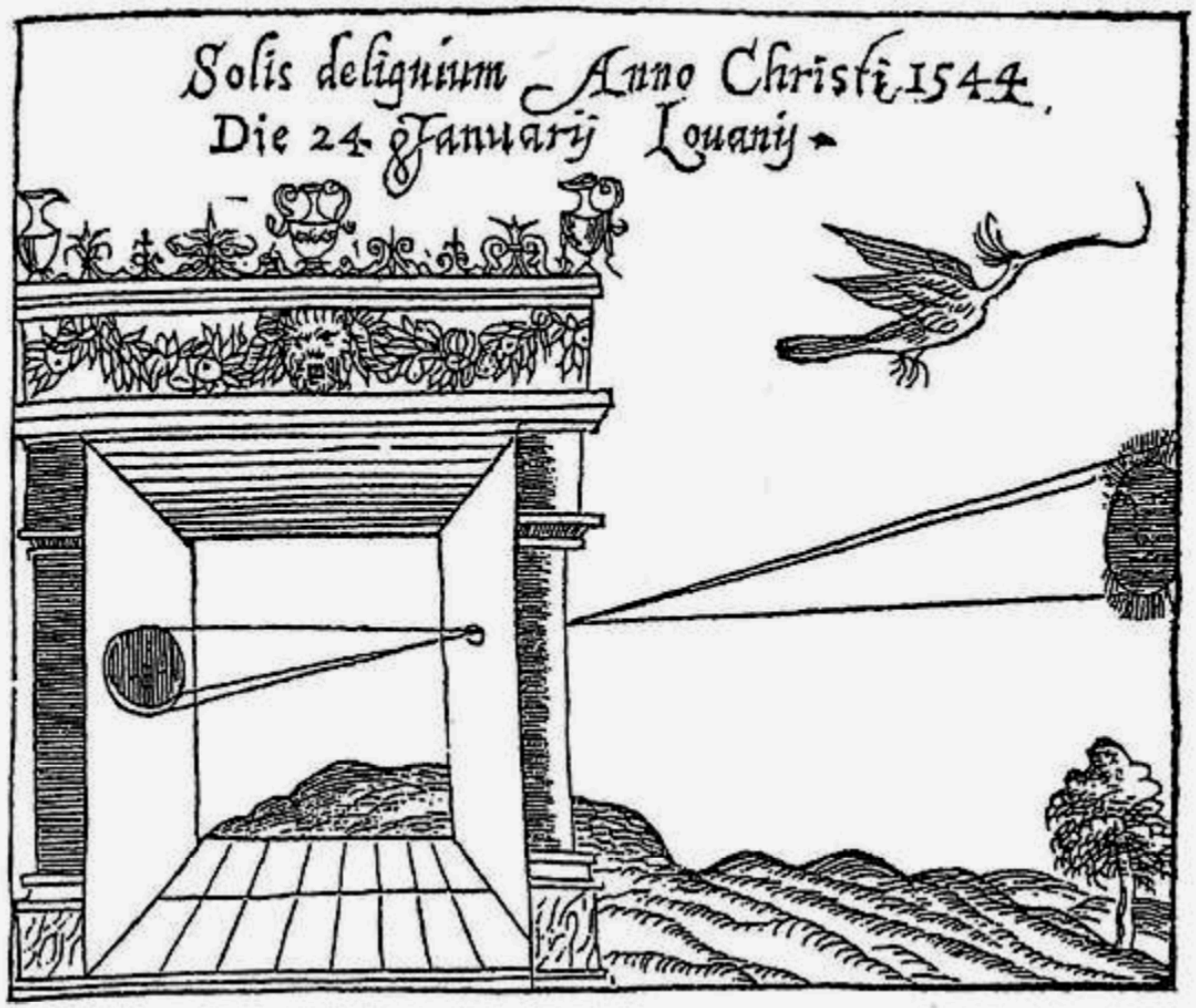
- Today:
  - Pinhole camera model
  - Modeling camera projections: homogeneous coordinates
  - Camera calibration
  - Color
  - Convolutions

# Basics of Image, Video, Optics

- Today:
  - Pinhole camera model
  - Modeling camera projections: homogeneous coordinates
  - Camera calibration
  - Color
  - Convolutions

# Camera Obscura

*Camera Obscura,* Gemma Frisius, 1558

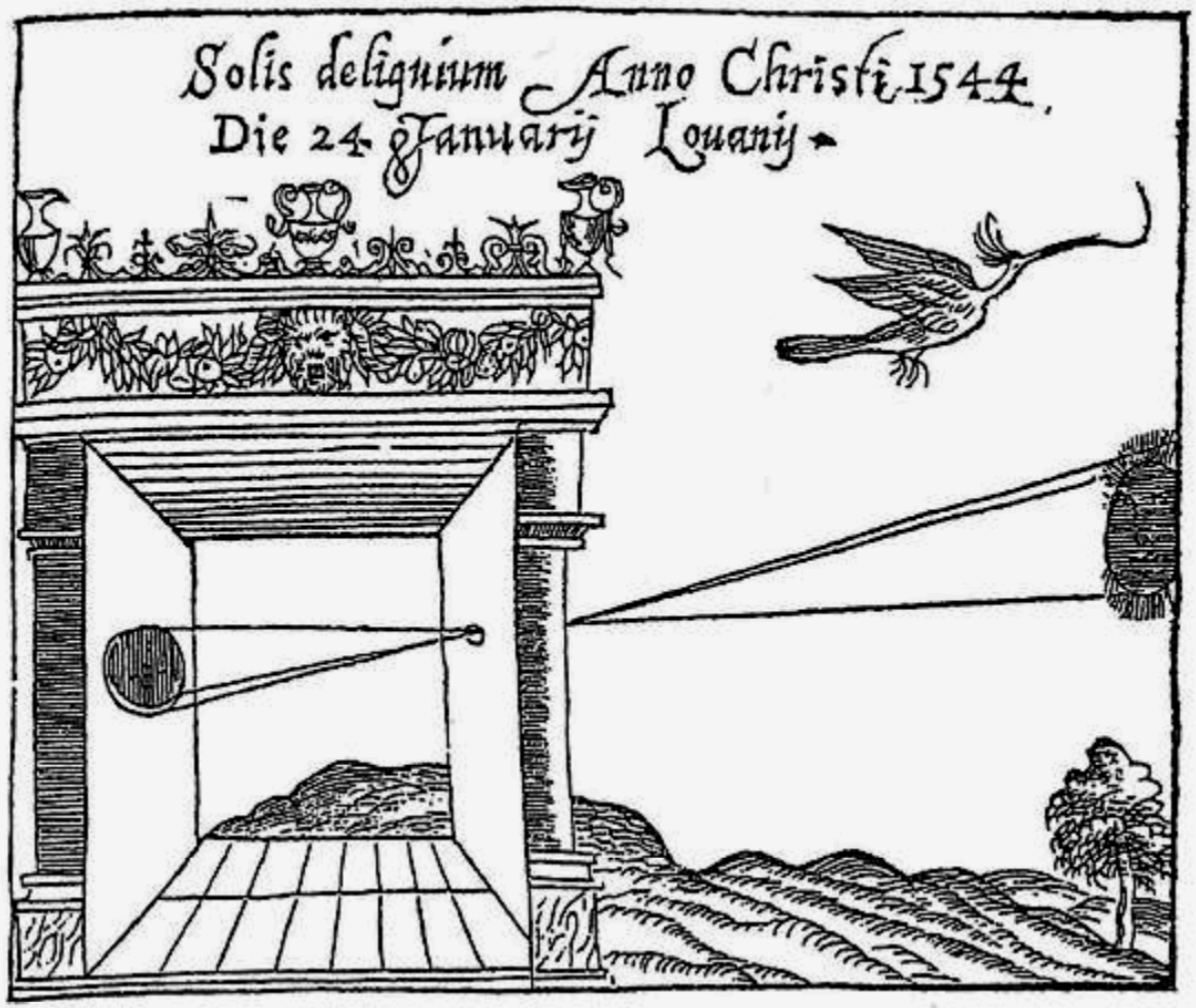



[Slide content from Efros]

Camera is Latin for chamber/room, obscura means dark

# Camera Obscura

*Camera Obscura,* Gemma Frisius, 1558




[Image from Efros]

Used by Euclid (300 B.C.) to show light travels in straight rays

# Pinhole Camera

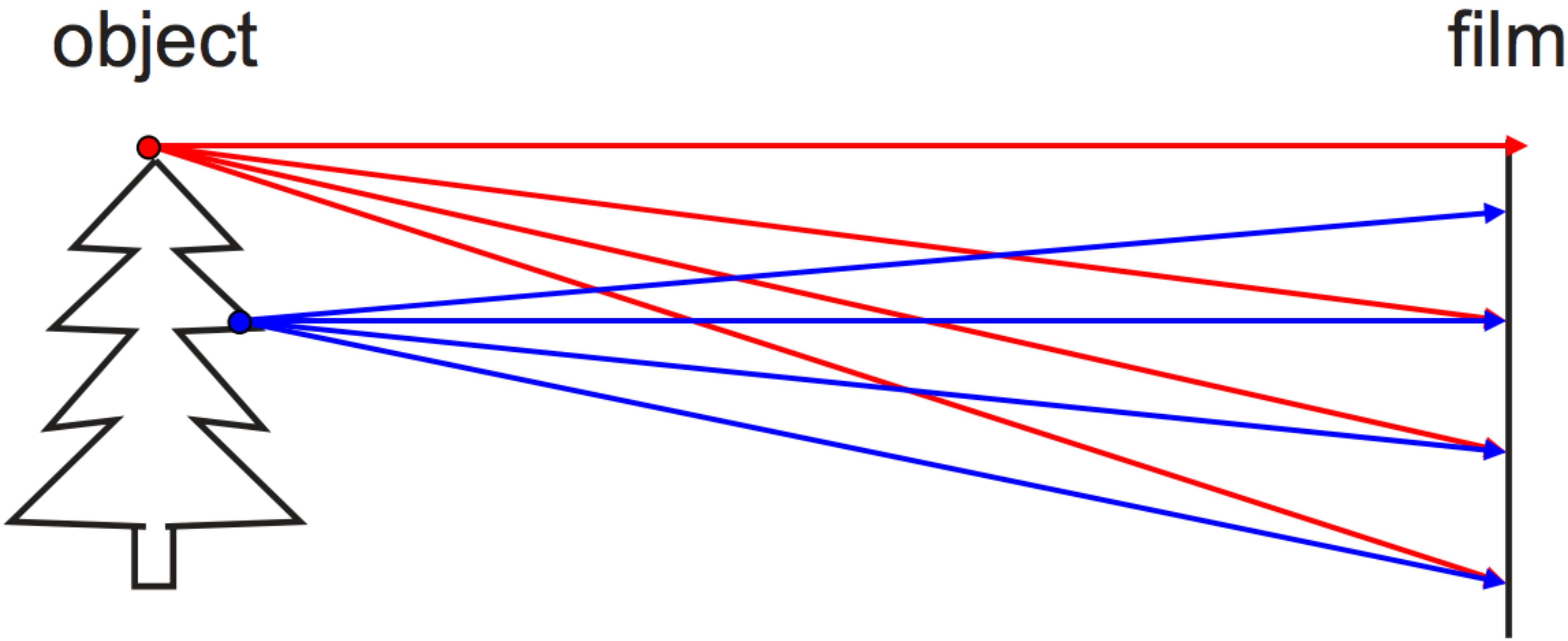



Illustration by Steve Seitz

- Blurry image results if all rays reach film

# Pinhole Camera

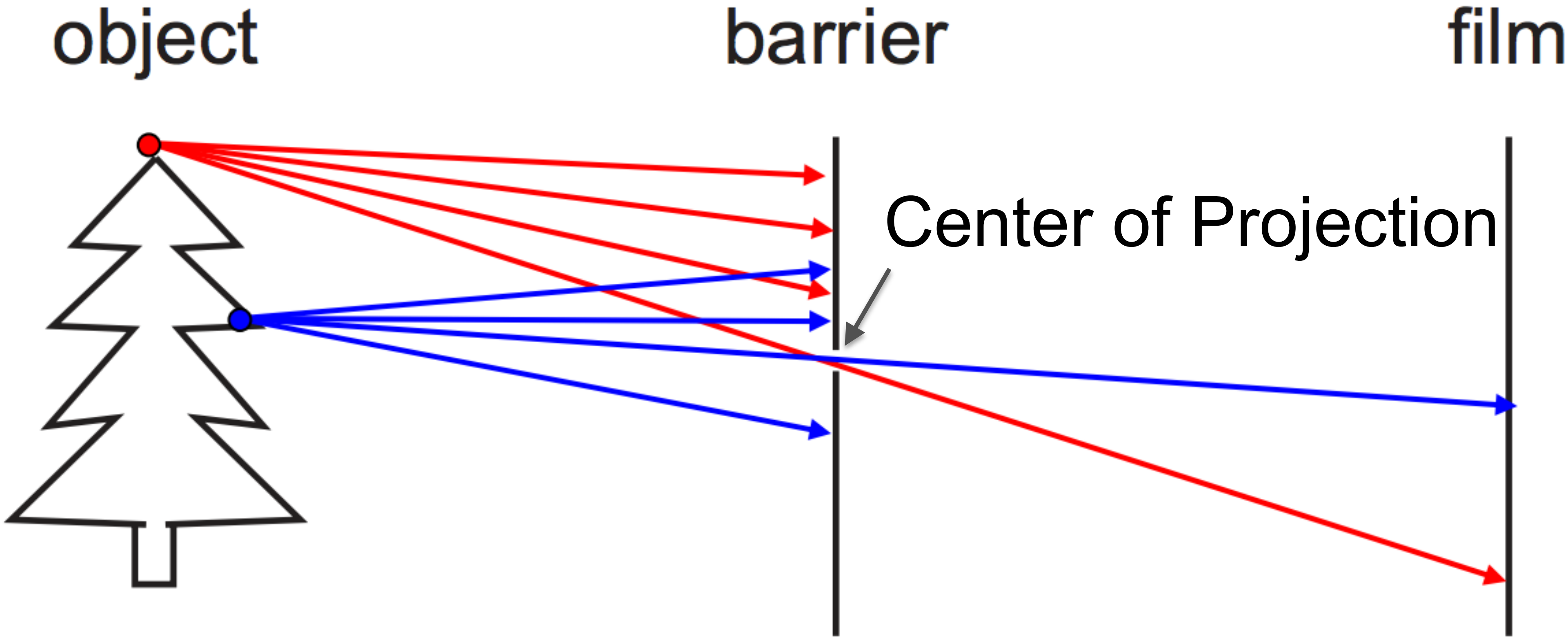
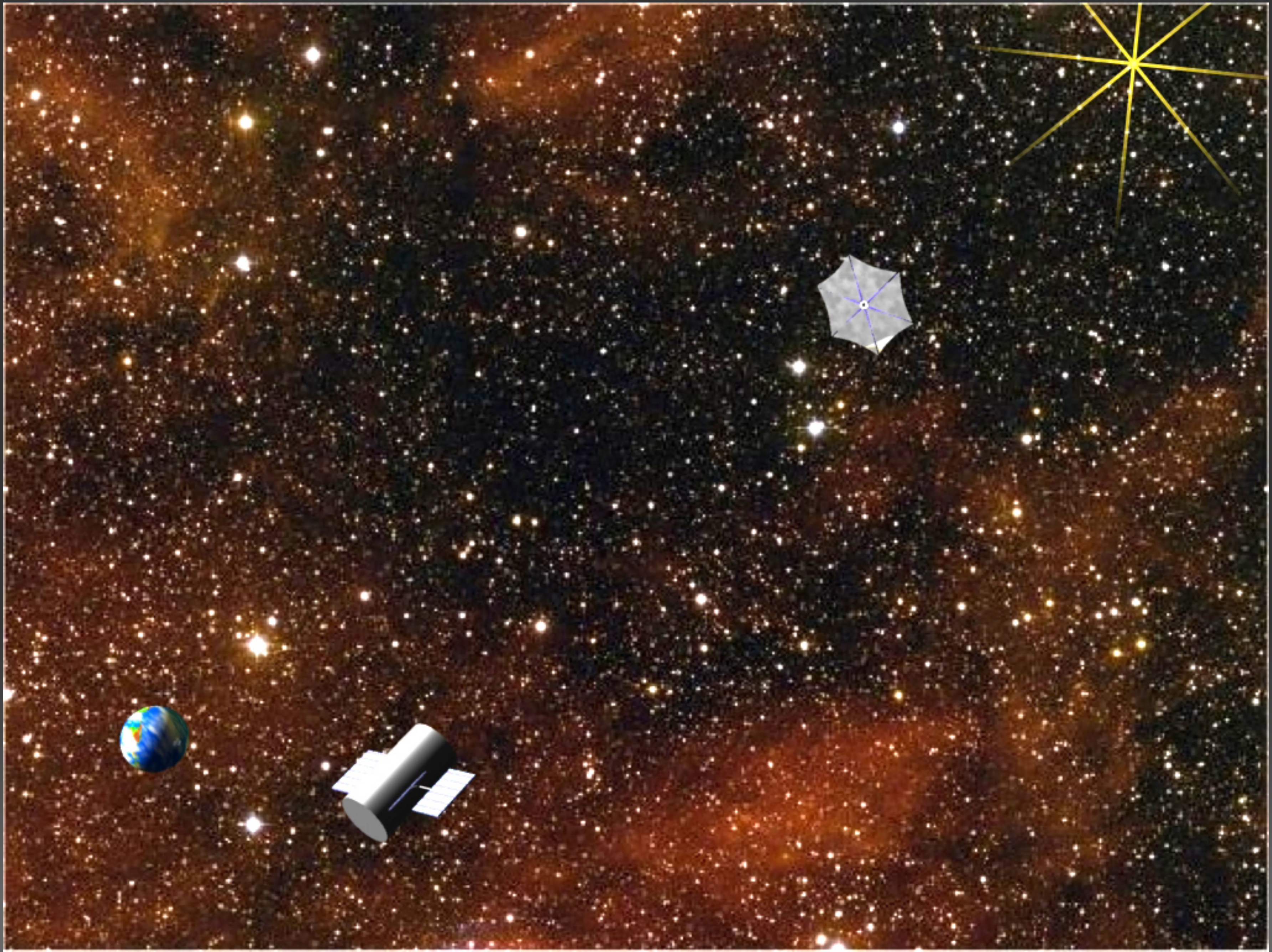


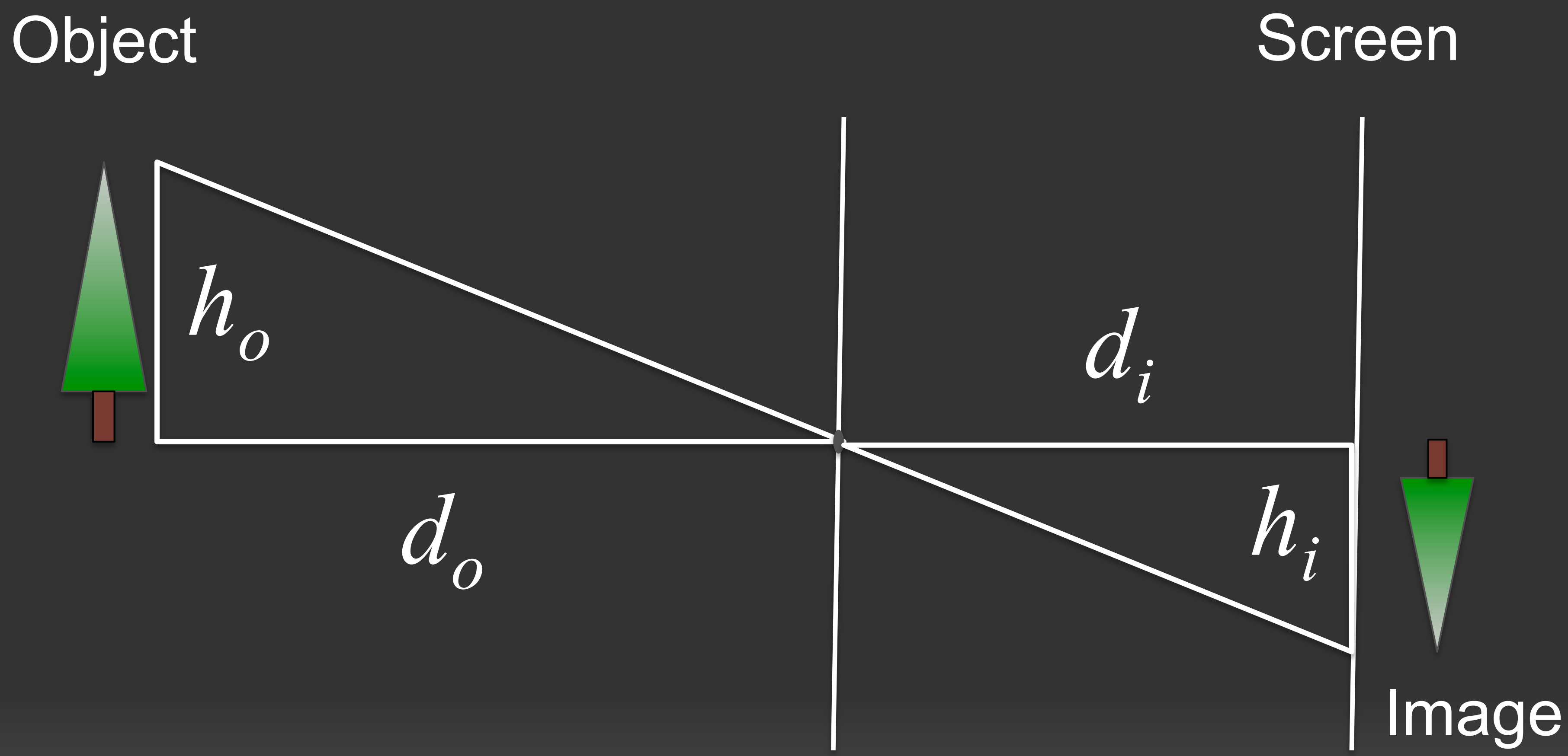

Illustration by Steve Seitz

- Pinhole camera has small aperture size.
- All objects are in focus – depth of field infinite

# New Worlds Mission (NASA)

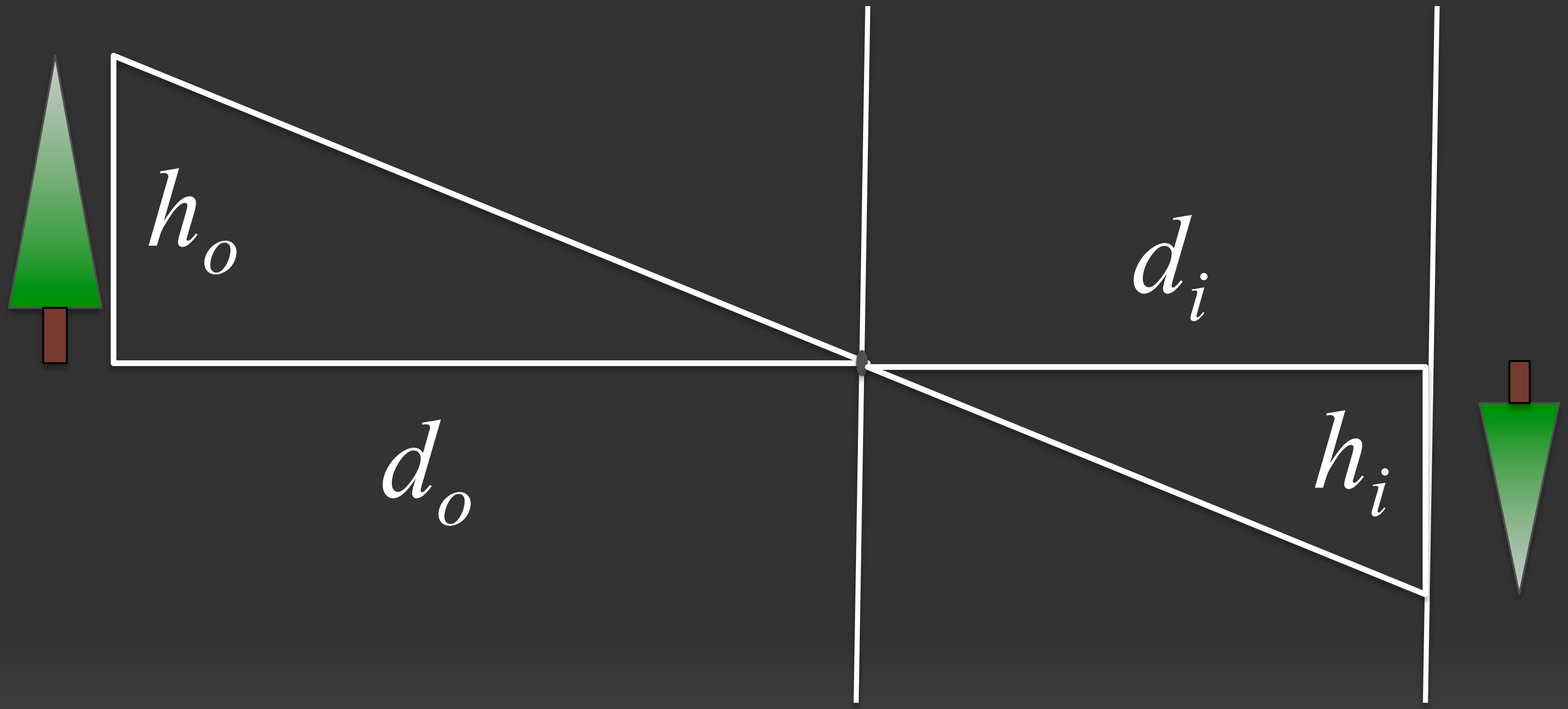


# Pinhole Camera



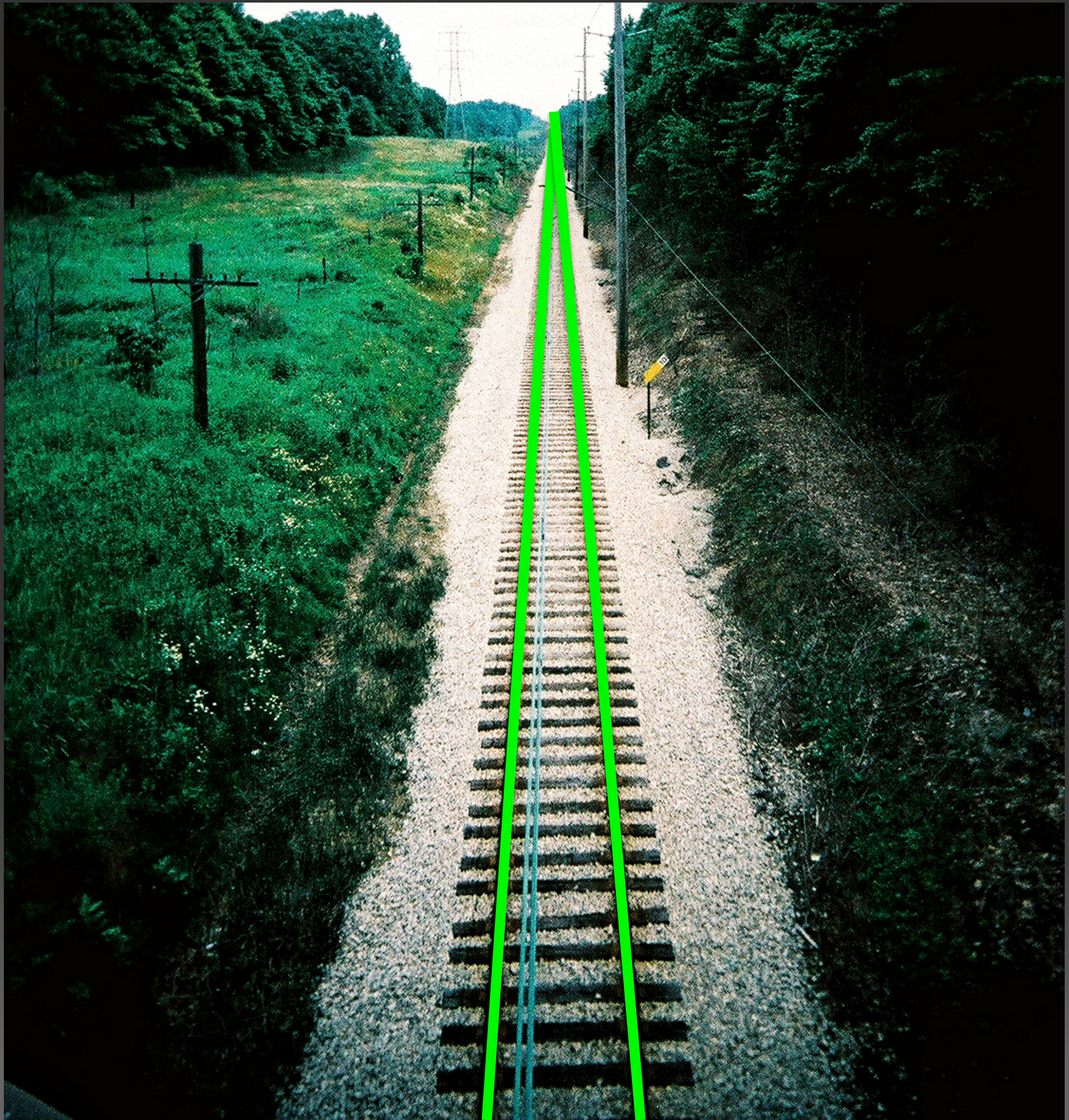

Similar triangles gives optics law: $\dfrac{h_i}{h_o} = \dfrac{d_i}{d_o}$

# Pinhole Camera




Fix object size and imaging plane distance: $h_i \propto \dfrac{1}{d_o}$

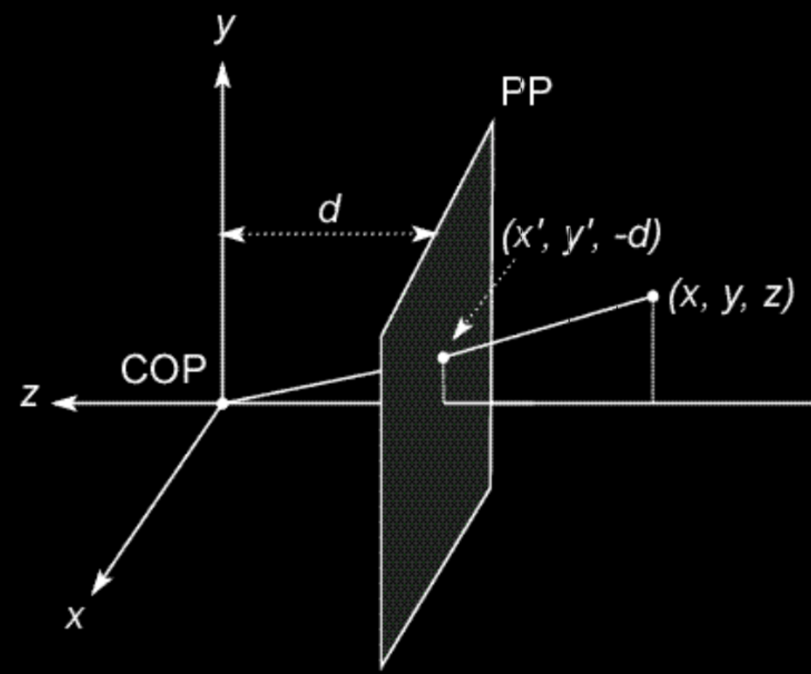
# Perspective



# Basics of Image, Video, Optics

- Today:
  - Pinhole camera model
  - Modeling camera projections: homogeneous coordinates
  - Camera calibration
  - Color
  - Convolutions

# Modeling projection

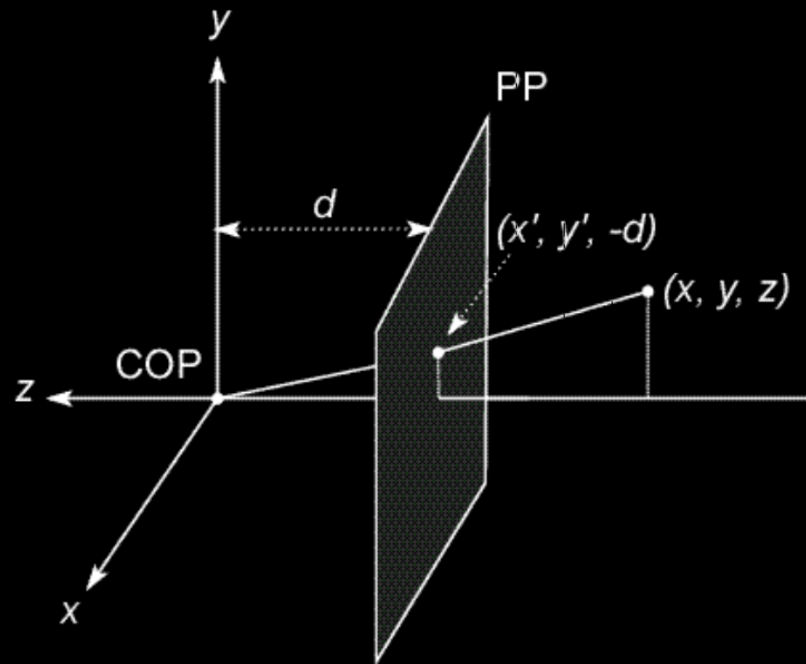



## The coordinate system

- We will use the pin-hole model as an approximation
- Put the optical center (**C**enter **O**f **P**rojection) at the origin
- Put the image plane (**P**rojection **P**lane) *in front* of the COP
  - Why?
- The camera looks down the *negative* z axis
  - we need this if we want right-handed-coordinates

Slide by Steve Seitz

# Modeling projection




## Projection equations

- Compute intersection with PP of ray from (x,y,z) to COP
- Derived using similar triangles (on board)

$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z},\ -d)$$

- We get the projection by throwing out the last coordinate:

$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z})$$

Slide by Steve Seitz

# Homogeneous coordinates

## Is this a linear transformation?

- no—division by z is nonlinear

## Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene coordinates

## Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Slide by Steve Seitz

# Perspective Projection

Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by third coordinate

This is known as **perspective projection**

- The matrix is the **projection matrix**
- Can also formulate as a 4x4

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by fourth coordinate

Slide by Steve Seitz

# Homogeneous Coordinates

- Used in graphics/vision when transforming geometry prior to projection on the screen.
  - e.g. to translate ($T$), scale ($S$), rotate ($R$), then perspective project ($P$) a point $p$:

    $\mathbf{q} = PRST\mathbf{p}$
- $PRST$ 4x4 matrices, $\mathbf{pq}$ 4x1 vectors

# Homogeneous Coordinates

- Transformation matrices found in OpenGL documentation:
  - glTranslate
  - glScale
  - glRotate
  - Derivation of Rotation Matrices in R3
- Can look at some of these on the board

# Discussion Question 1

- Suppose R is a rotation matrix by angle θ.

$$\begin{pmatrix} x^2(1-c)+c & xy(1-c)-zs & xz(1-c)+ys & 0 \\ xy(1-c)+zs & y^2(1-c)+c & yz(1-c)-xs & 0 \\ xz(1-c)-ys & yz(1-c)+xs & z^2(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where $c = \cos\theta, \quad s = \sin\theta$

- What is $R^{-1}$? Geometrically? In matrix form?

# Discussion Question?

- Will a straight line in the world become a straight line after projection on to the image plane of a camera?

- Why or why not?
- Can you prove it?

# Basics of Image, Video, Optics

- Today:
  - Pinhole camera model
  - Modeling camera projections: homogeneous coordinates
  - Camera calibration
  - Color
  - Convolutions

# Camera Calibration

- Want a 4x4 homogeneous coordinate matrix **C** that transforms world coordinates **p** (4x1) to screen coordinates **s** (4x1).
- A common convention is to drop the “z” part of the screen coordinate (depth).

# Camera Calibration (Wikipedia)

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

*R*: 3x3, *T*: 3x1, *A*: 3x3

*A*: Intrinsic parameters of camera: internal properties of the lens, sensor.

# Camera Calibration (Wikipedia)

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

*R*: 3x3, *T*: 3x1, *A*: 3x3

*R, T*: Extrinsic parameters of camera: its orientation and position (but note: *T* is not camera position).

# Intrinsic Camera Parameters (Wikipedia)

$$A = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

The parameters $\alpha_x = f \cdot m_x$ and $\alpha_y = f \cdot m_y$ represent focal length in terms of pixels, where $m_x$ and $m_y$ are the scale factors relating pixels to distance and $f$ is the focal length in terms of distance. [1] $\gamma$ represents the skew coefficient between the x and the y axis, and is often 0. $u_0$ and $v_0$ represent the principal point, which would be ideally in the centre of the image.

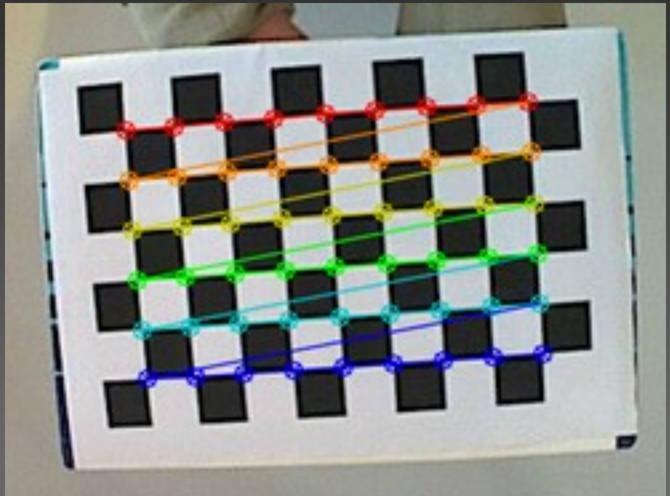
# Camera Calibration in Practice

OpenCV:

**calibrateCamera**(InputArrayOfArrays **objectPoints**,
InputArrayOfArrays **imagePoints**,
Size **imageSize**,
InputOutputArray **cameraMatrix, …)**

Estimates camera parameters given multiple views of a calibration pattern.

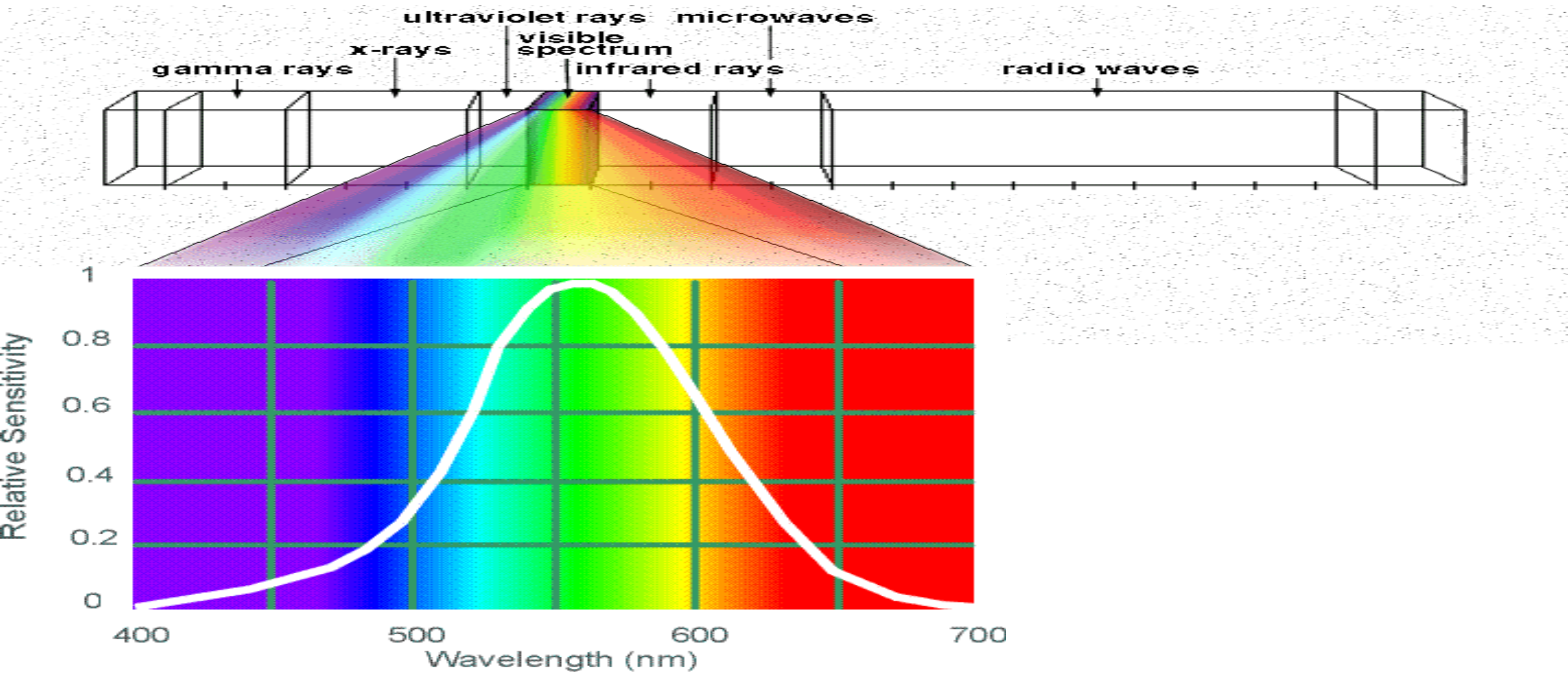
OpenCV camera calibration tutorial



# Basics of Image, Video, Optics

- Today:
  - Pinhole camera model
  - Modeling camera projections: homogeneous coordinates
  - Camera calibration
  - Color
  - Convolutions

# Electromagnetic Spectrum



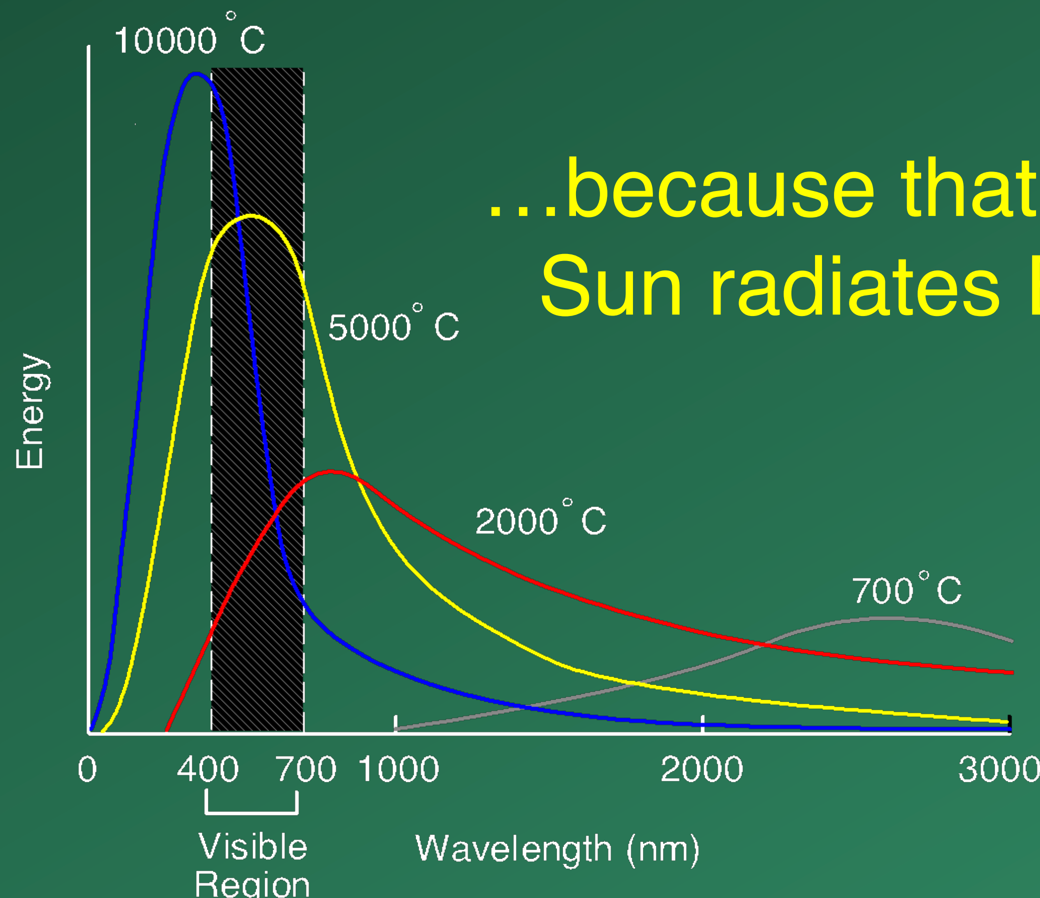

Human Luminance Sensitivity Function

http://www.yorku.ca/eye/photopik.htm

# Visible Light

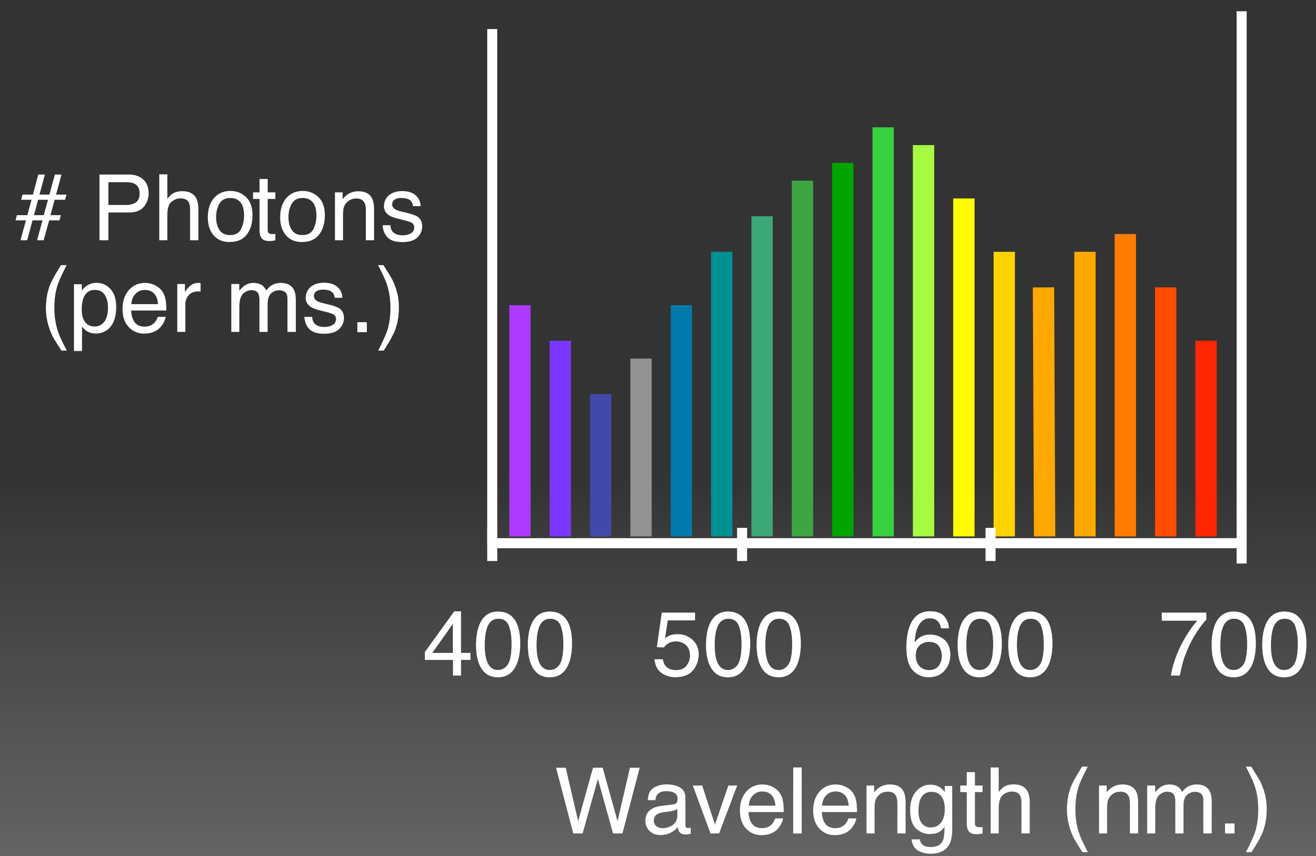
Why do we see light of these wavelengths?

…because that's where the Sun radiates EM energy



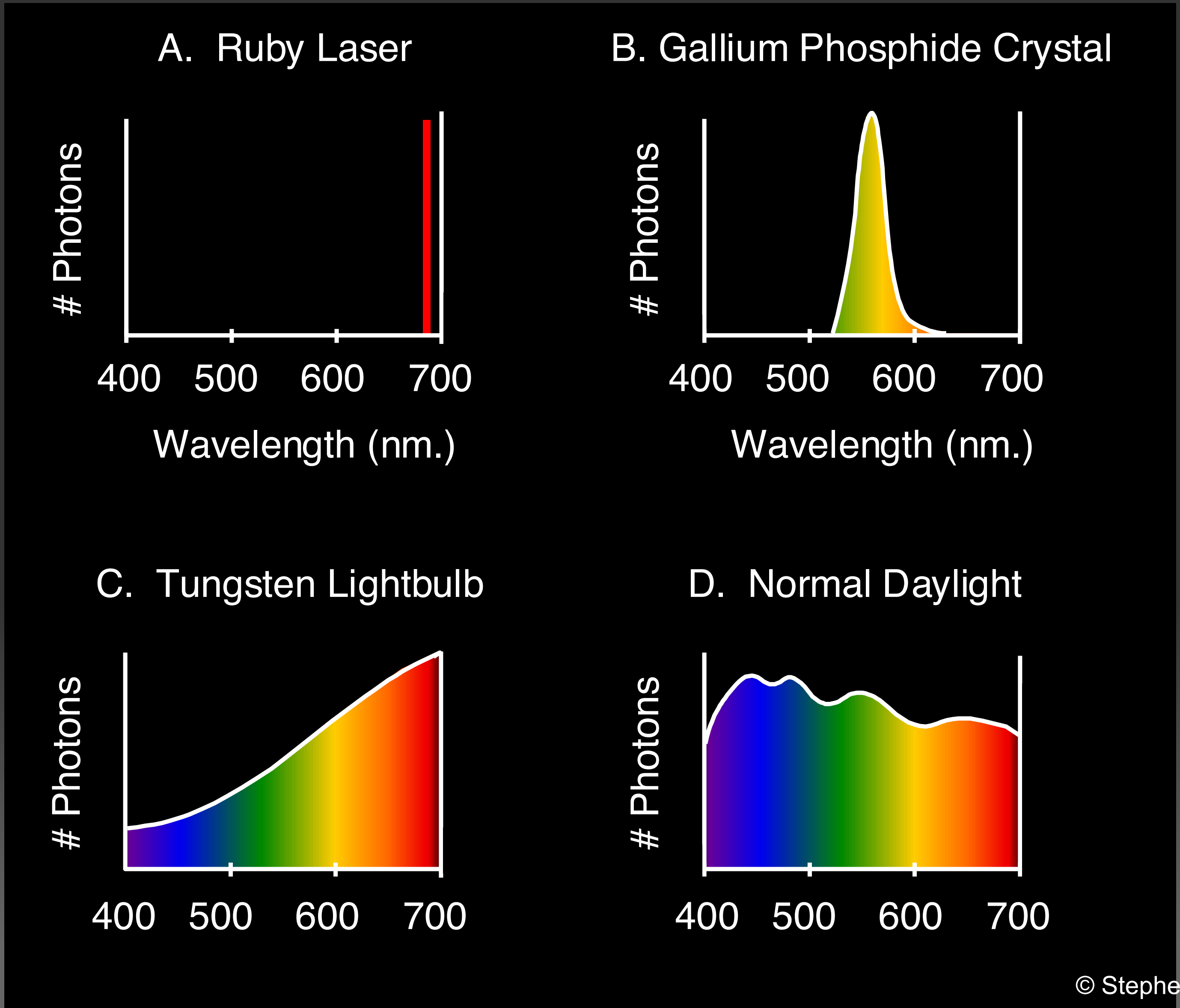
# The Physics of Light

Any patch of light can be completely described physically by its spectrum: the number of photons (per time unit) at each wavelength 400 - 700 nm.



# The Physics of Light

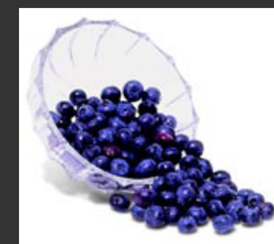
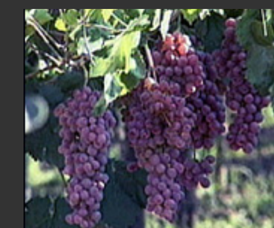
Some examples of the spectra of light sources

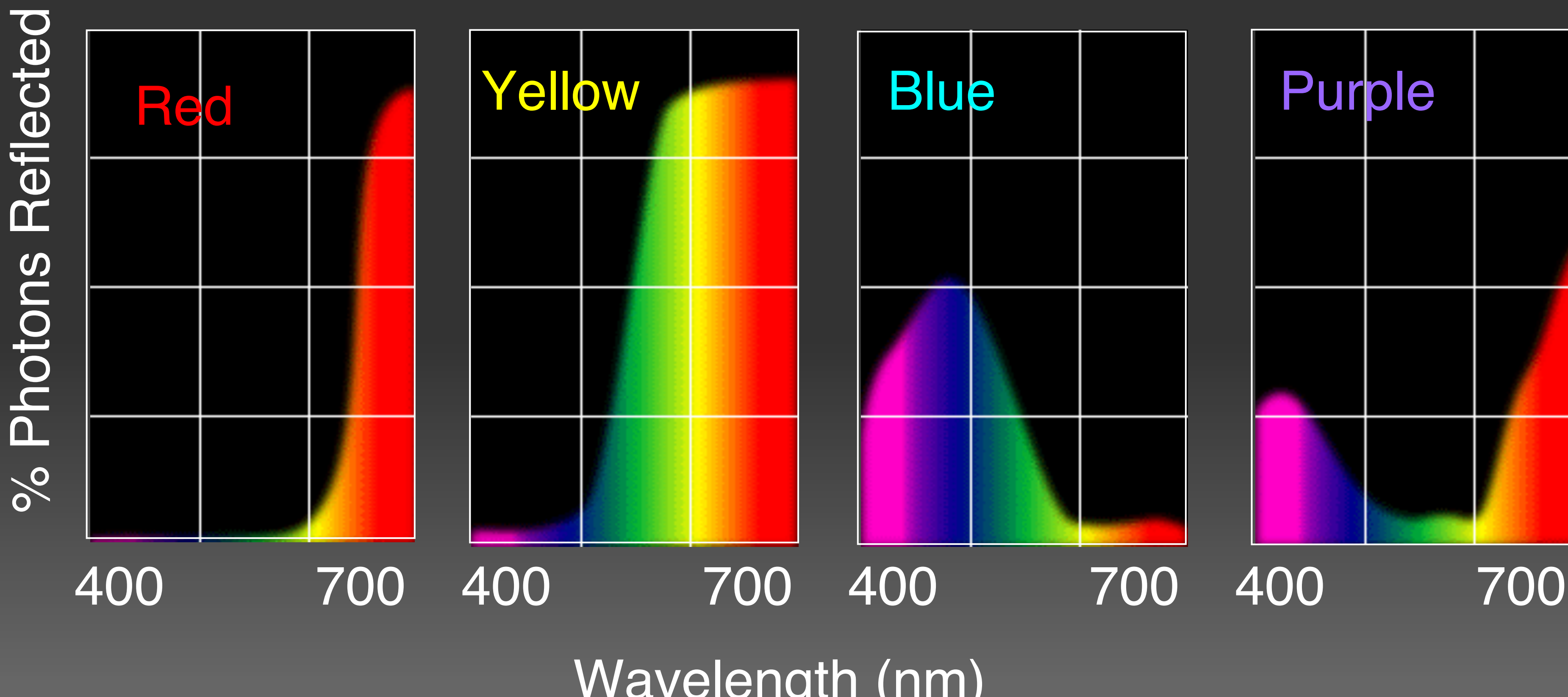


# The Physics of Light

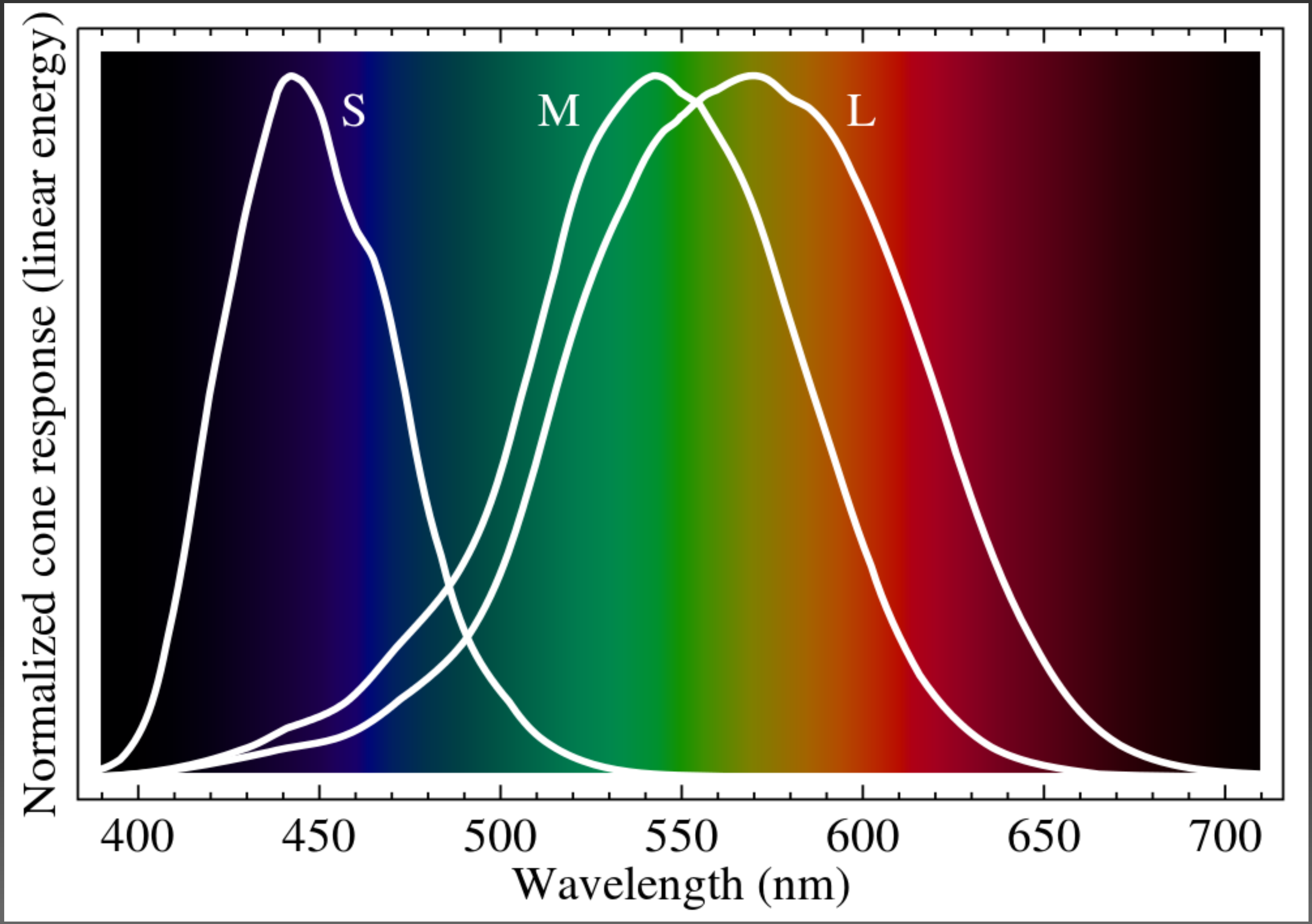
Some examples of the reflectance spectra of surfaces



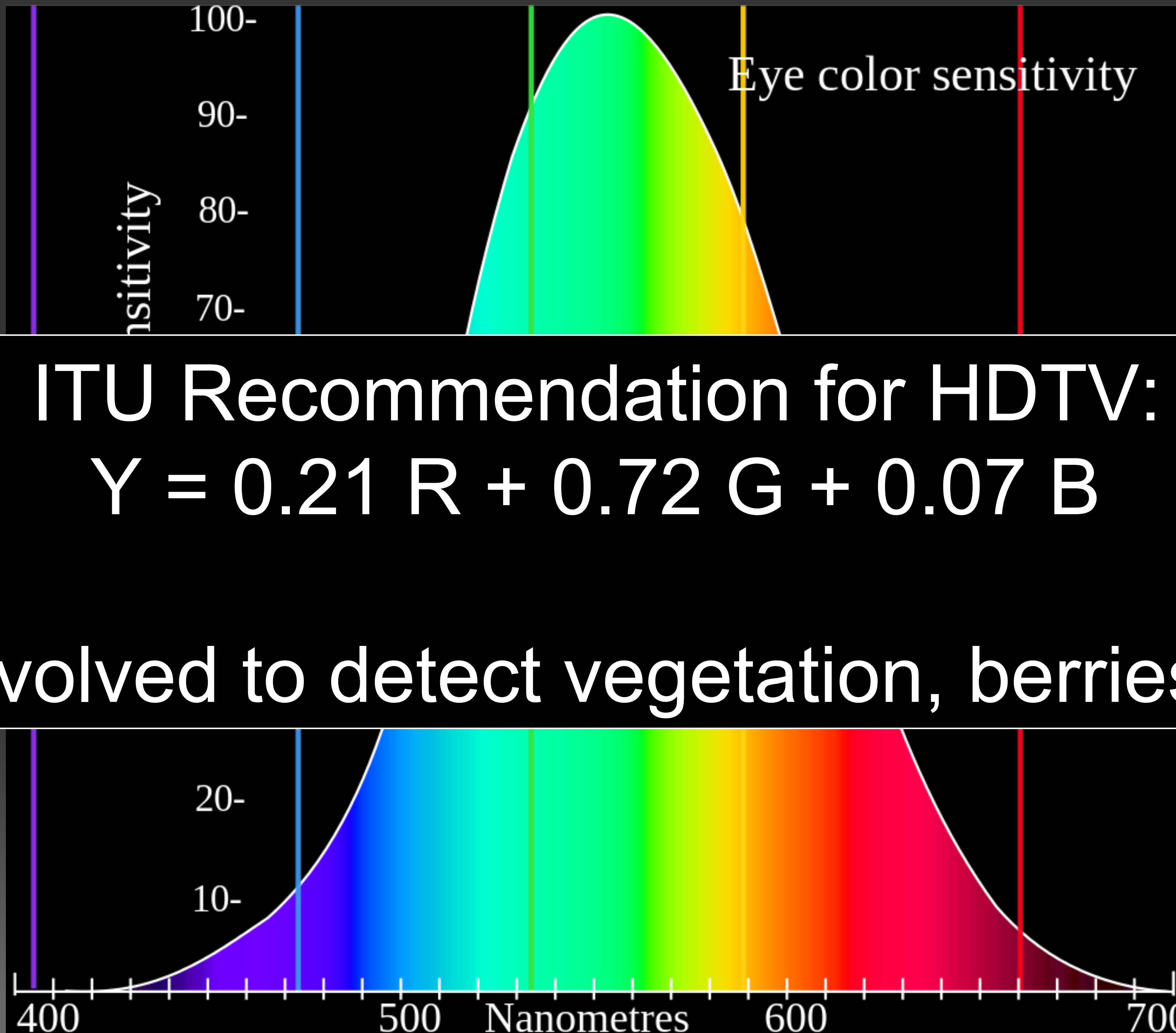





# Ordinary Human Vision (Trichromatism)



# Perceptual Sensitivity

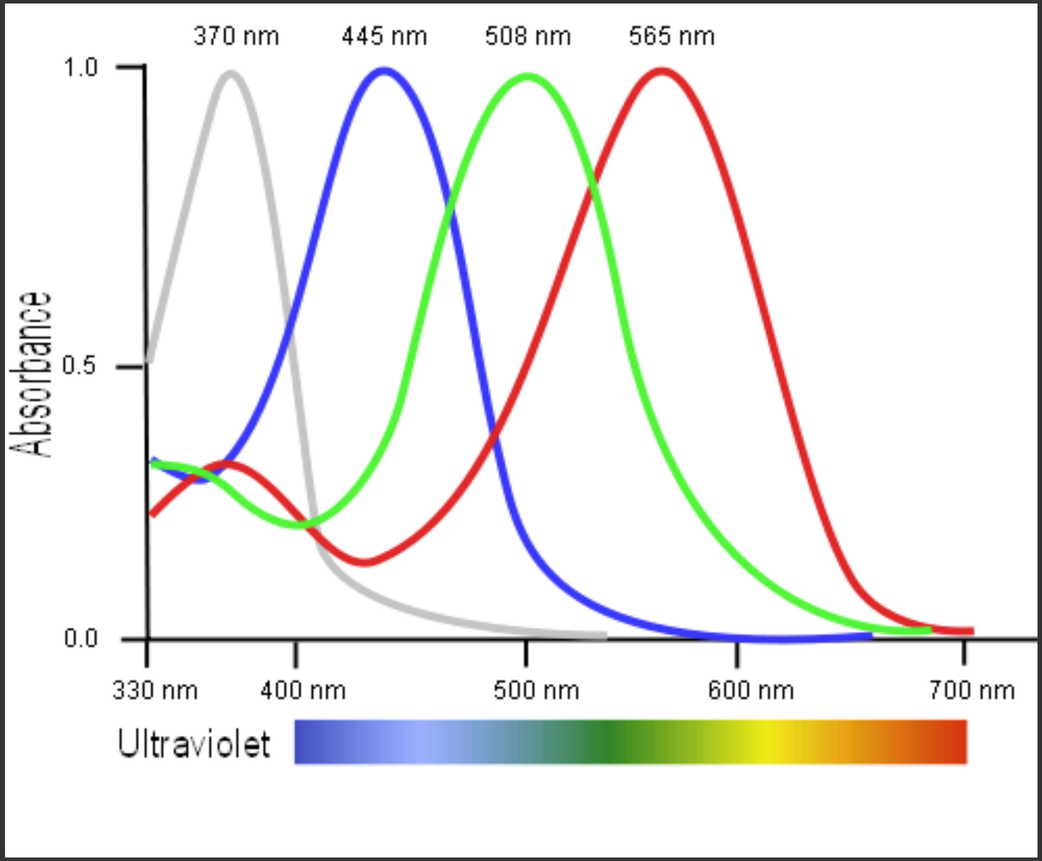



ITU Recommendation for HDTV:

$$Y = 0.21\,R + 0.72\,G + 0.07\,B$$

Evolved to detect vegetation, berries?

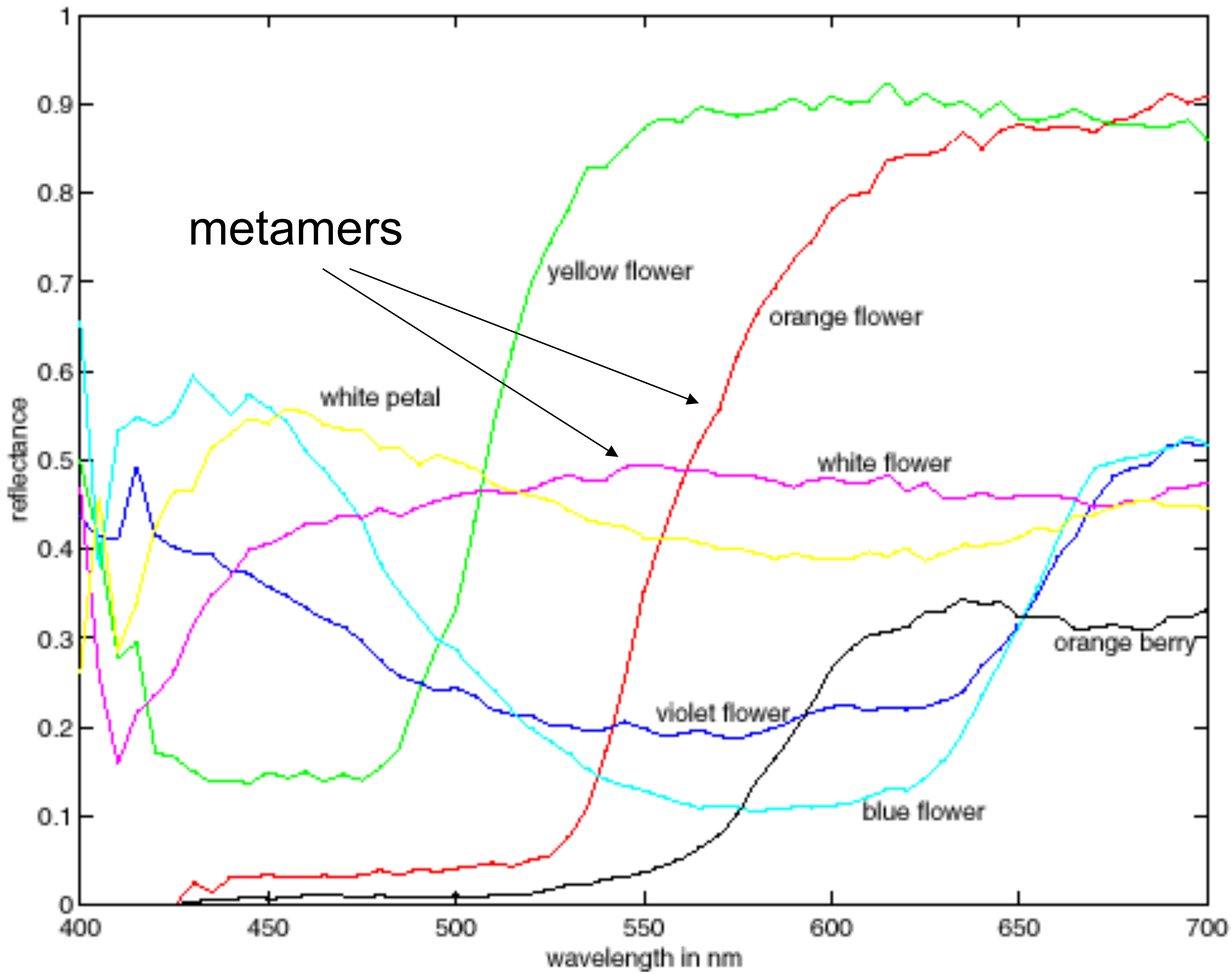
# Tetrachromatism




Bird cone responses

Most birds, and many other animals, have cones for ultraviolet light.
Some humans, mostly female, seem to have slight tetrachromatism.

# Color Spectra



# Color Image

# Images in Python/MATLAB

- Image as array: h x w x channels
  I(y,x,channel)
- Red channel, upper left corner:
  MATLAB: I(1,1,1), Python: I[0,0,0]

column → **R**

row ↓

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.92 | 0.93 | 0.94 | 0.97 | 0.62 | 0.37 | 0.85 | 0.97 | 0.93 | 0.92 | 0.99 |
| 0.95 | 0.89 | 0.82 | 0.89 | 0.56 | 0.31 | 0.75 | 0.92 | 0.81 | 0.95 | 0.91 |
| 0.89 | 0.72 | 0.51 | 0.55 | 0.51 | 0.42 | 0.57 | 0.41 | 0.49 | 0.91 | 0.92 |
| 0.96 | 0.95 | 0.88 | 0.94 | 0.56 | 0.46 | 0.91 | 0.87 | 0.90 | 0.97 | 0.95 |
| 0.71 | 0.81 | 0.81 | 0.87 | 0.57 | 0.37 | 0.80 | 0.88 | 0.89 | 0.79 | 0.85 |
| 0.49 | 0.62 | 0.60 | 0.58 | 0.50 | 0.60 | 0.58 | 0.50 | 0.61 | 0.45 | 0.33 |
| 0.86 | 0.84 | 0.74 | 0.58 | 0.51 | 0.39 | 0.73 | 0.92 | 0.91 | 0.49 | 0.74 |
| 0.96 | 0.67 | 0.54 | 0.85 | 0.48 | 0.37 | 0.88 | 0.90 | 0.94 | 0.82 | 0.93 |
| 0.69 | 0.49 | 0.56 | 0.66 | 0.43 | 0.42 | 0.77 | 0.73 | 0.71 | 0.90 | 0.99 |
| 0.79 | 0.73 | 0.90 | 0.67 | 0.33 | 0.61 | 0.69 | 0.79 | 0.73 | 0.93 | 0.97 |
| 0.91 | 0.94 | 0.89 | 0.49 | 0.41 | 0.78 | 0.78 | 0.77 | 0.89 | 0.99 | 0.93 |

**G**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | .92 | 0.99 |
| | | | | | | | | | .95 | 0.91 |
| | | | | | | | | | .91 | 0.92 |
| | | | | | | | | | .97 | 0.95 |
| | | | | | | | | | .79 | 0.85 |
| | | | | | | | | | .45 | 0.33 |
| | | | | | | | | | .49 | 0.74 |
| | | | | | | | | | .82 | 0.93 |
| | | | | | | | | | .90 | 0.99 |
| 0.79 | 0.73 | 0.90 | 0.67 | 0.33 | 0.61 | 0.69 | 0.79 | 0.73 | 0.93 | 0.97 |
| 0.91 | 0.94 | 0.89 | 0.49 | 0.41 | 0.78 | 0.78 | 0.77 | 0.89 | 0.99 | 0.93 |

**B**

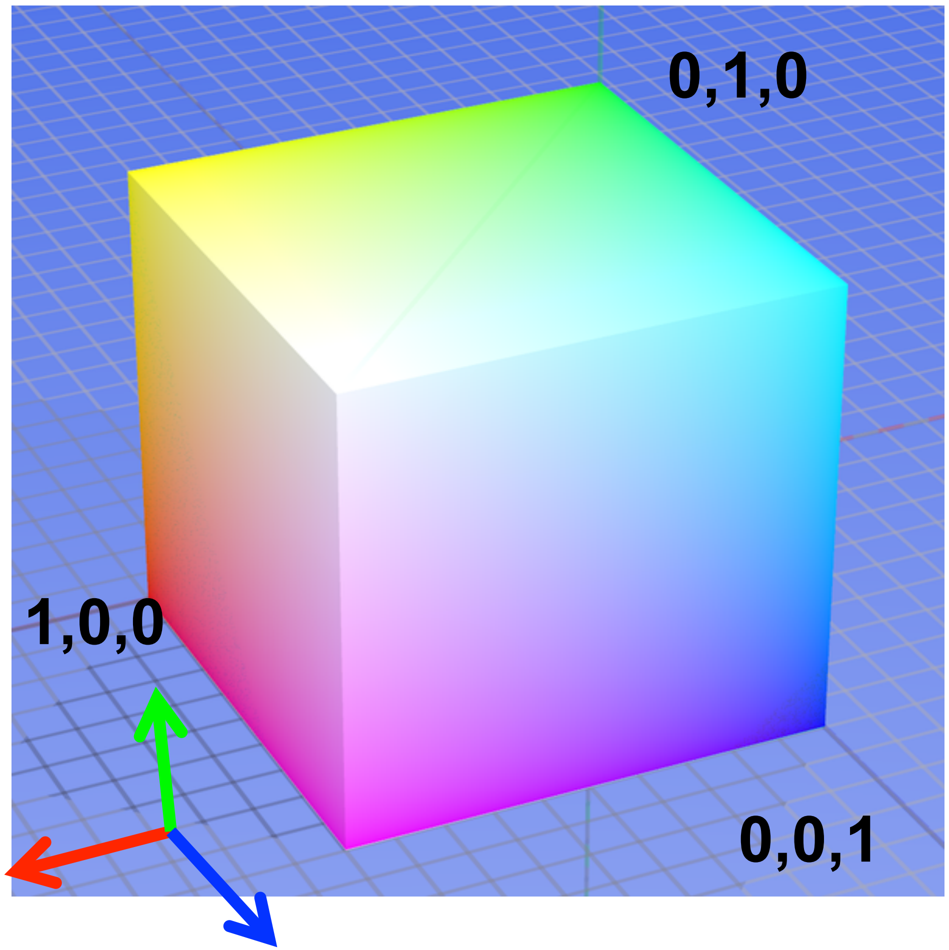
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 92 | 0.99 |
| | | | | | | | | | 95 | 0.91 |
| | | | | | | | | | 91 | 0.92 |
| | | | | | | | | | 97 | 0.95 |
| | | | | | | | | | 79 | 0.85 |
| | | | | | | | | | 45 | 0.33 |
| | | | | | | | | | 49 | 0.74 |
| | | | | | | | | | 82 | 0.93 |
| | | | | | | | | | 90 | 0.99 |
| 0.79 | 0.73 | 0.90 | 0.67 | 0.33 | 0.61 | 0.69 | 0.79 | 0.73 | 0.93 | 0.97 |
| 0.91 | 0.94 | 0.89 | 0.49 | 0.41 | 0.78 | 0.78 | 0.77 | 0.89 | 0.99 | 0.93 |

# Video Cube


$$V(t, y, x, \text{channel})$$

# Color spaces: RGB

Default color space




**R**
(G=0,B=0)

**G**
(R=0,B=0)

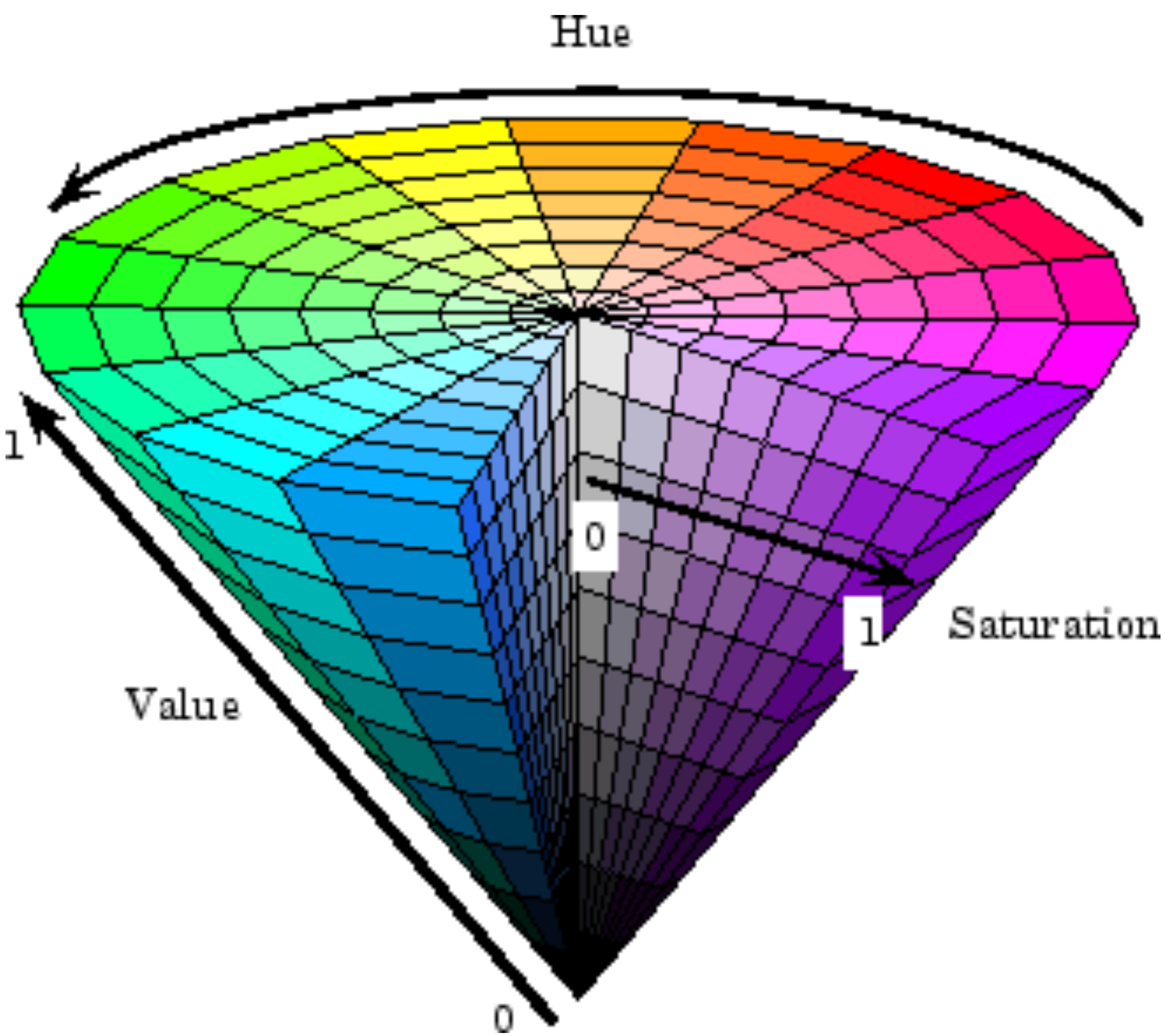
**B**
(R=0,G=0)

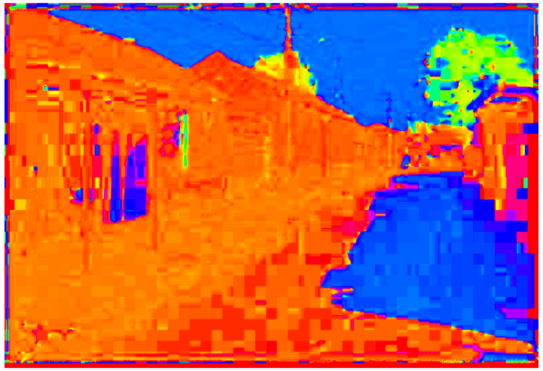
Some drawbacks
- Strongly correlated channels
- Non-perceptual

Image from: http://en.wikipedia.org/wiki/File:RGB_color_solid_cube.png

# Color spaces: HSV

Intuitive color space



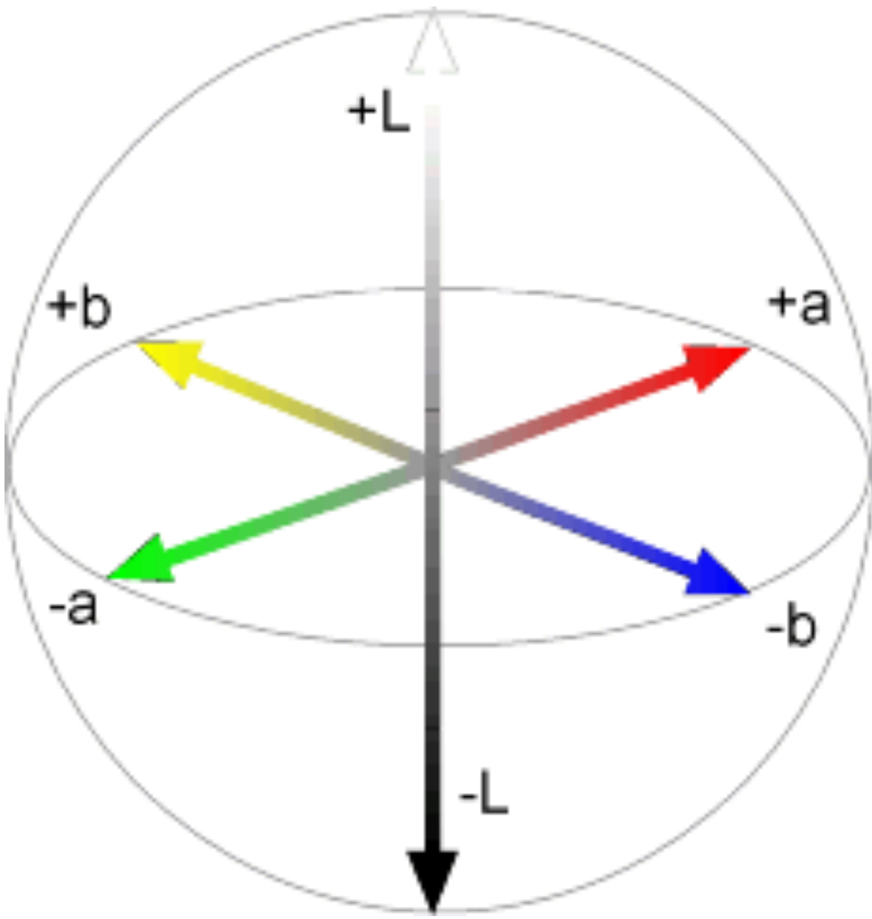
**H**
(S=1,V=1)

**S**
(H=1,V=1)

**V**
(H=1,S=0)

# Color spaces: L*a*b*

“Perceptually uniform” color space




**L**
(a=0,b=0)

**a**
(L=65,b=0)

**b**
(L=65,a=0)

# Basics of Image, Video, Optics

- Today:
  - Pinhole camera model
  - Modeling camera projections: homogeneous coordinates
  - Camera calibration
  - Color
  - Convolutions

# Convolution

**Convolution** takes a windowed average of an image *F* with a filter *H*, where the filter is flipped horizontally and vertically before being applied:

It is written:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i-u, j-v]$$

$$G = H \star F$$

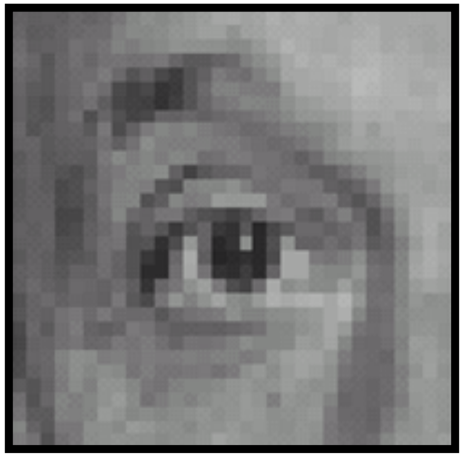
Slide by Steve Seitz

# Convolution is nice!

- Notation: $b = c \star a$
- Convolution is a multiplication-like operation
  - commutative $a \star b = b \star a$
  - associative $a \star (b \star c) = (a \star b) \star c$
  - distributes over addition $a \star (b + c) = a \star b + a \star c$
  - scalars factor out $\alpha a \star b = a \star \alpha b = \alpha(a \star b)$
  - identity: unit impulse e = [..., 0, 0, 1, 0, 0, ...]

    $a \star e = a$
- Conceptually no distinction between filter and signal
- Usefulness of associativity
  - often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - this is equivalent to applying one filter: a * $(b_1 * b_2 * b_3)$

# Practice with linear filters



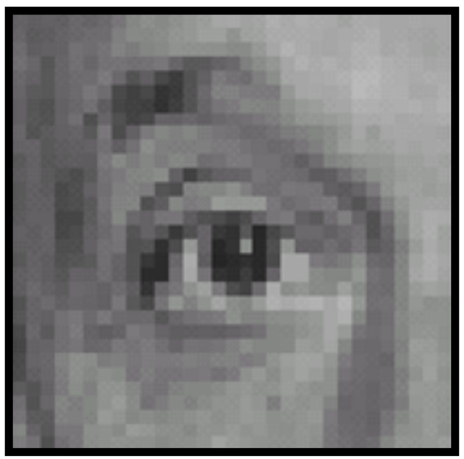
Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

?

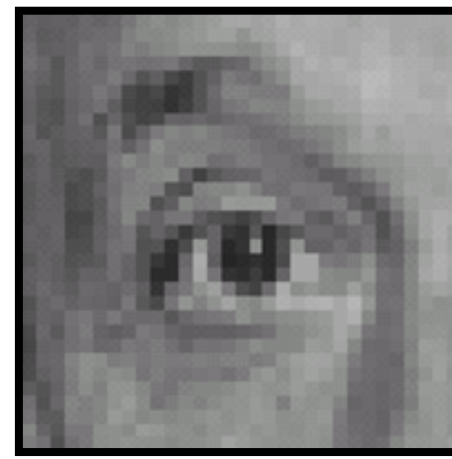
Source: D. Lowe

# Practice with linear filters



Original

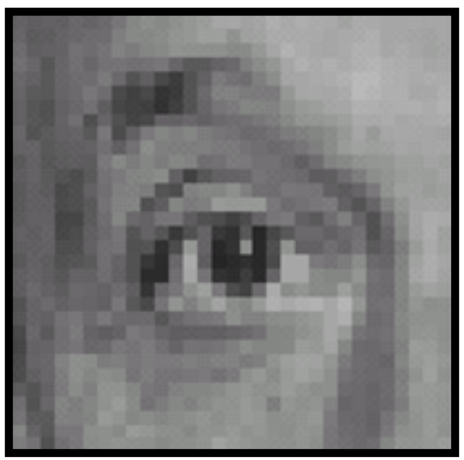
| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |



Filtered
(no change)

Source: D. Lowe

# Practice with linear filters



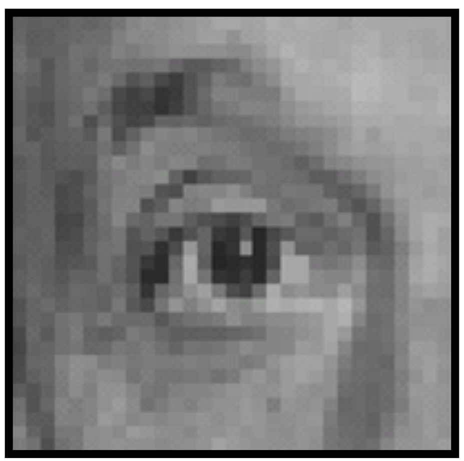
Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |

?

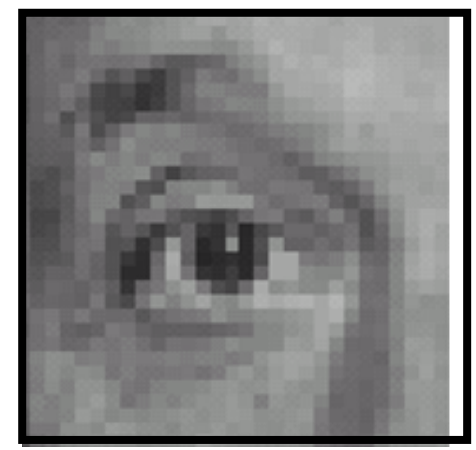
Source: D. Lowe

# Practice with linear filters



Original

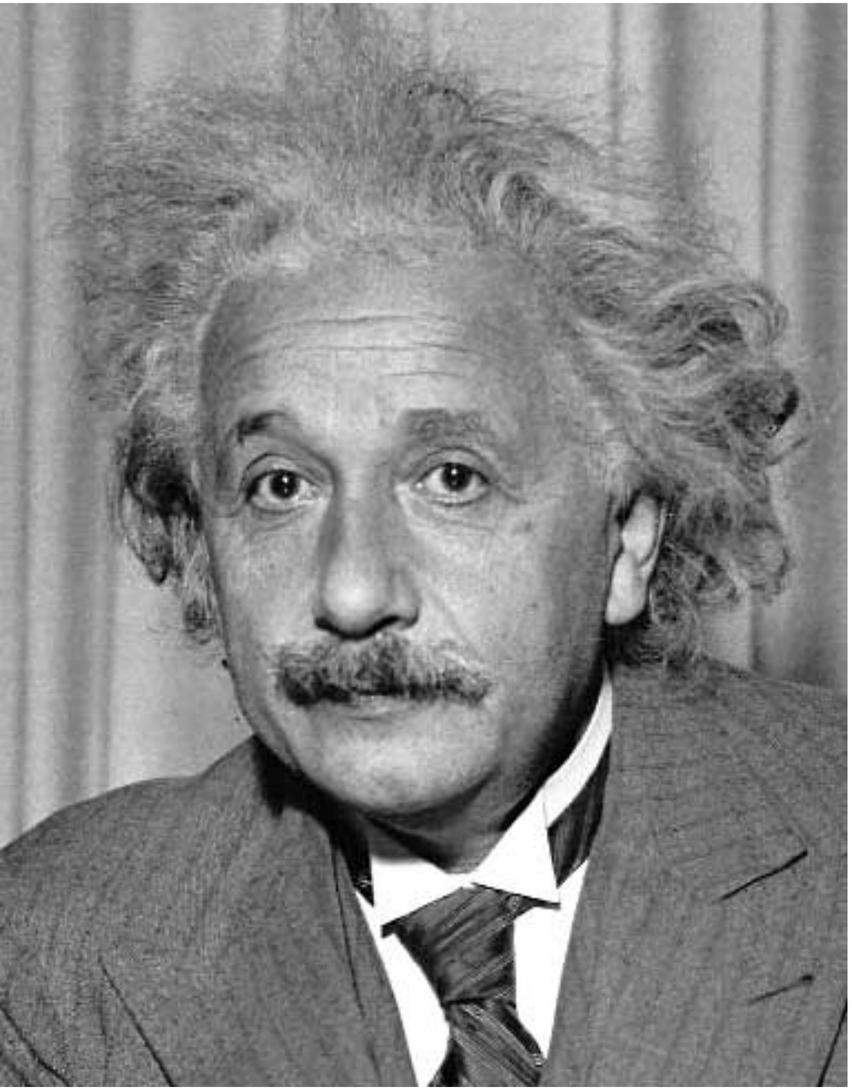
| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |



Shifted left
By 1 pixel

Source: D. Lowe

# Other filters



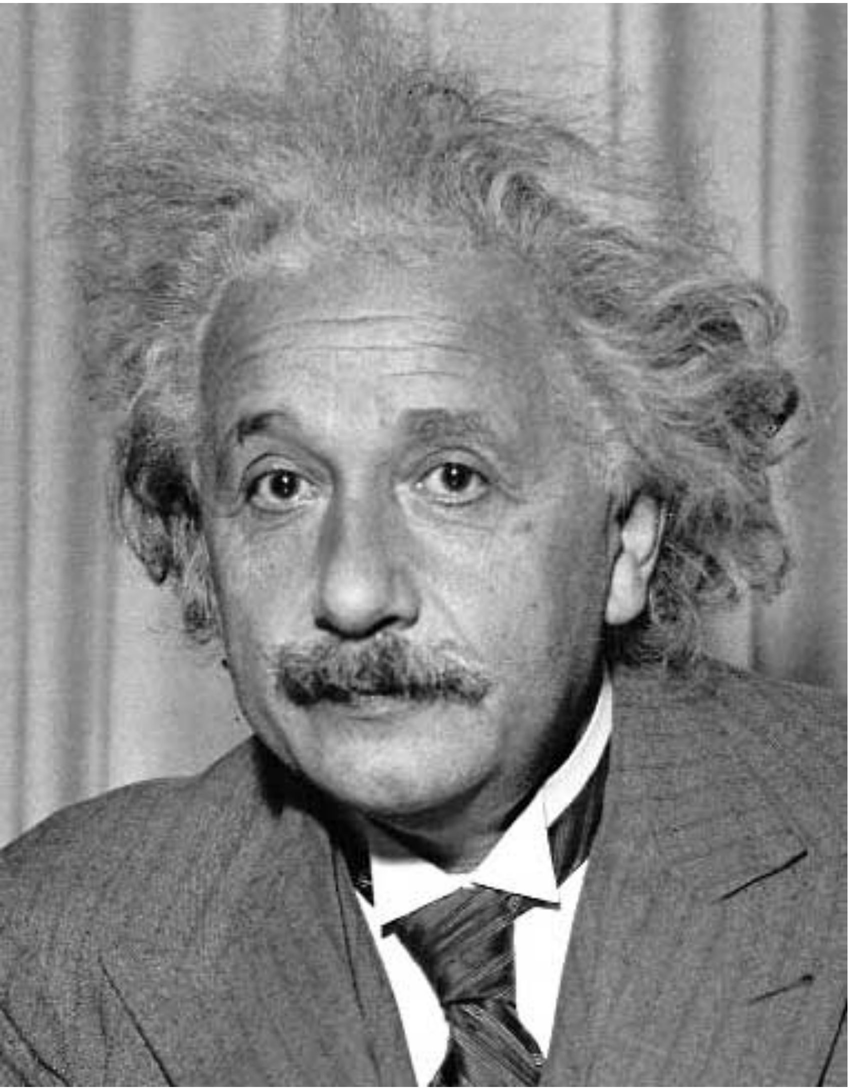
| | | |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

Sobel

?

Separable (show on board)

# Other filters

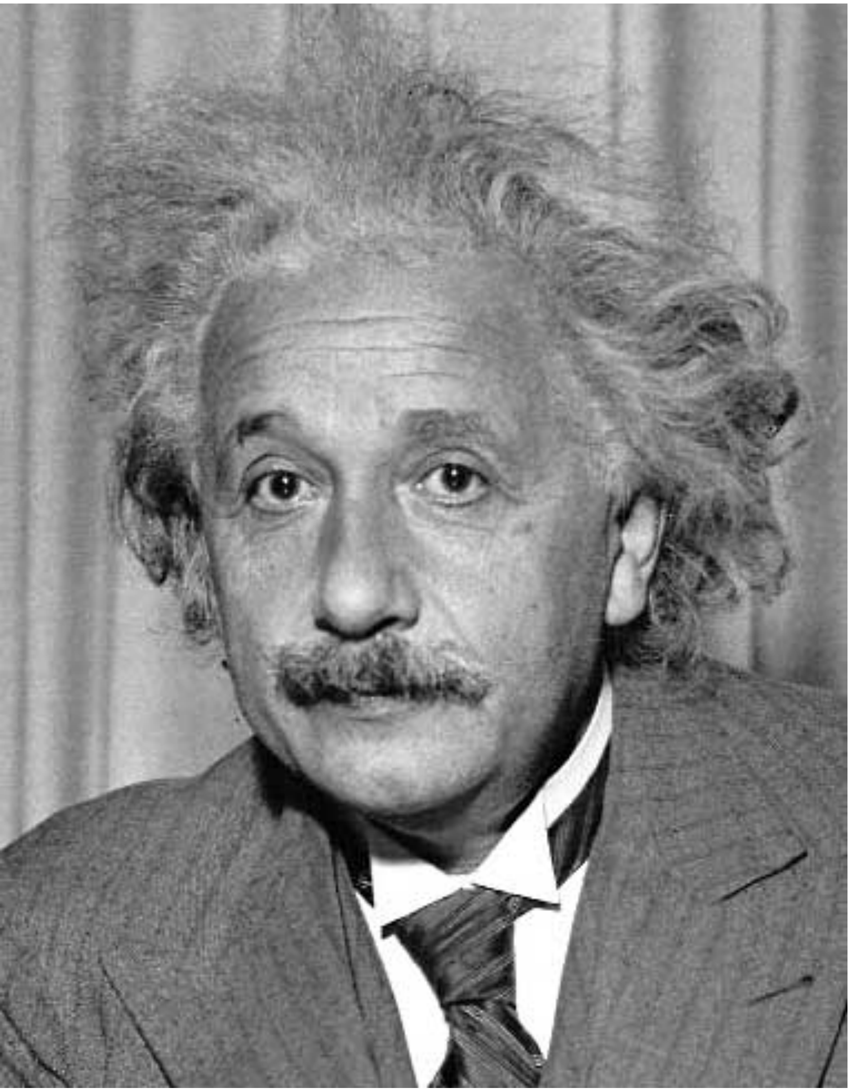


| | | |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

Sobel

Vertical Edge
(absolute value)

# Other filters



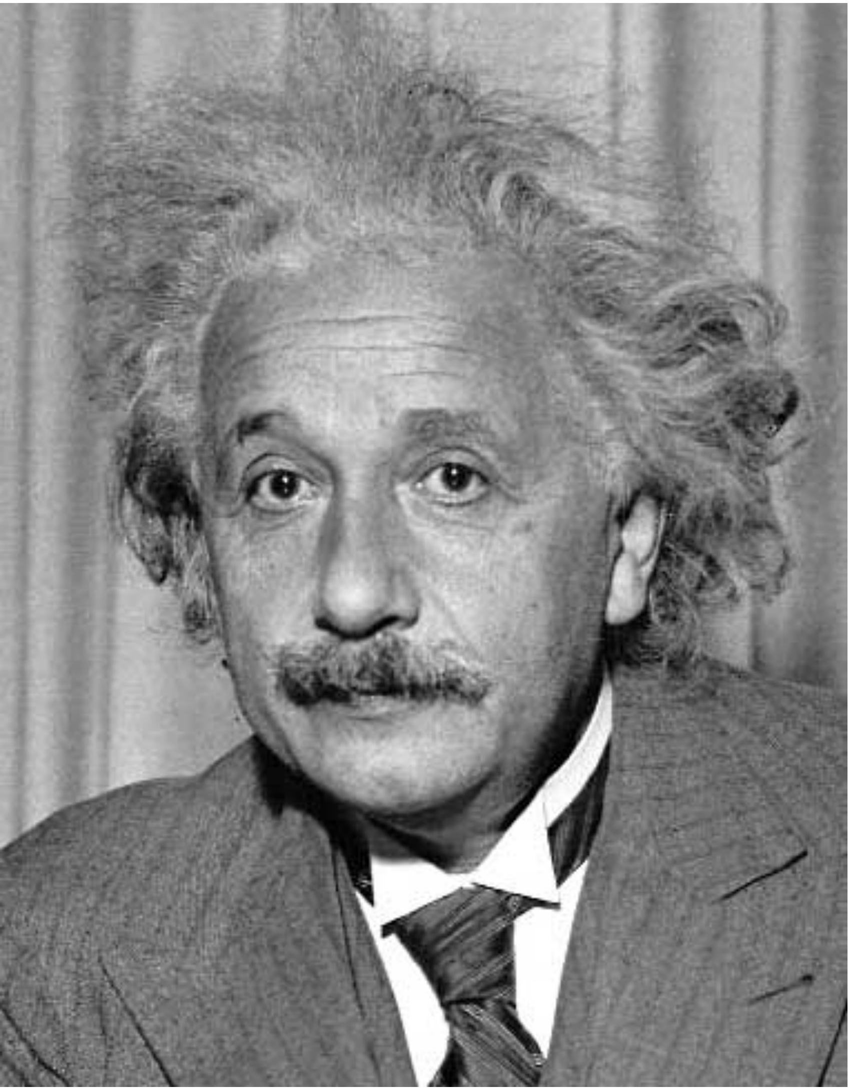
| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

Sobel

?

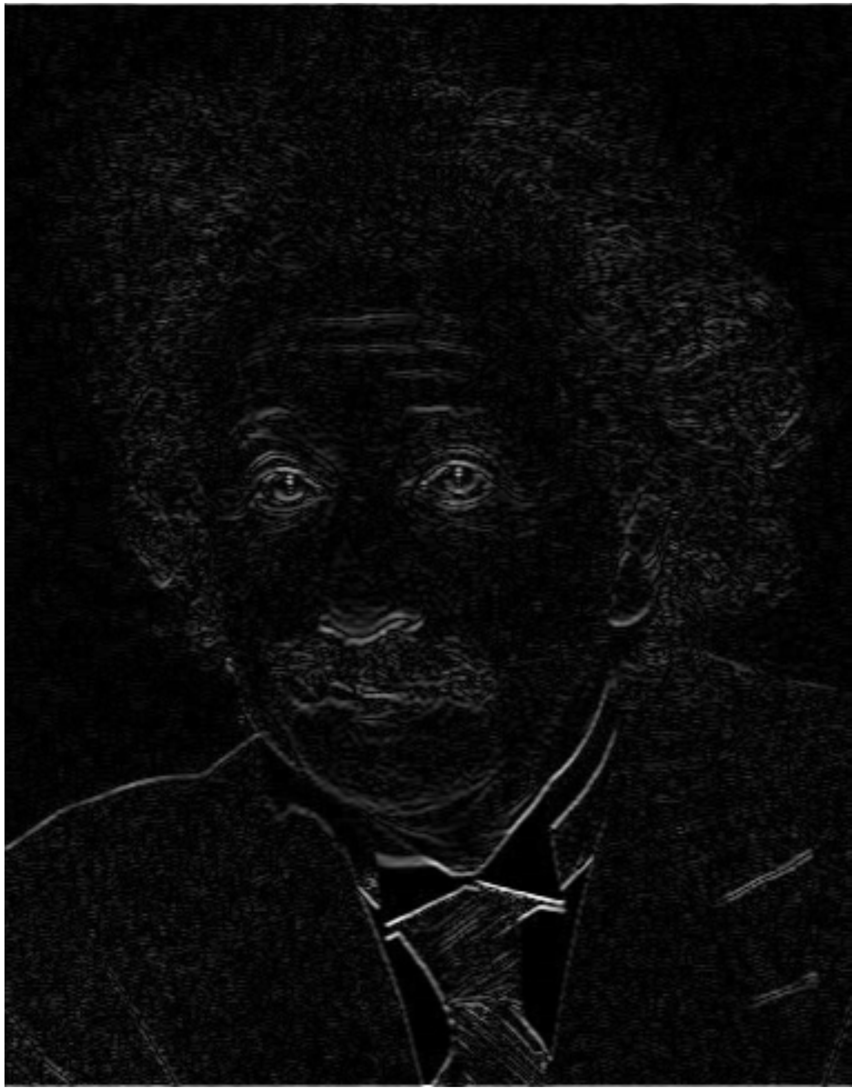
Separable (show on board)

# Other filters



| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

Sobel

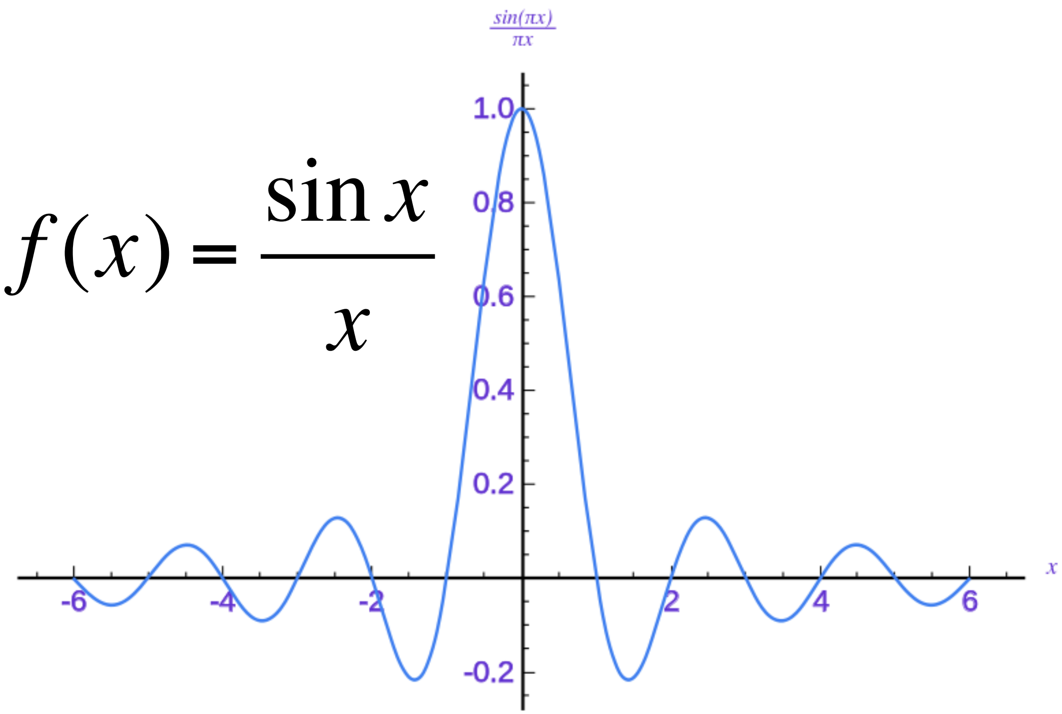


Horizontal Edge
(absolute value)

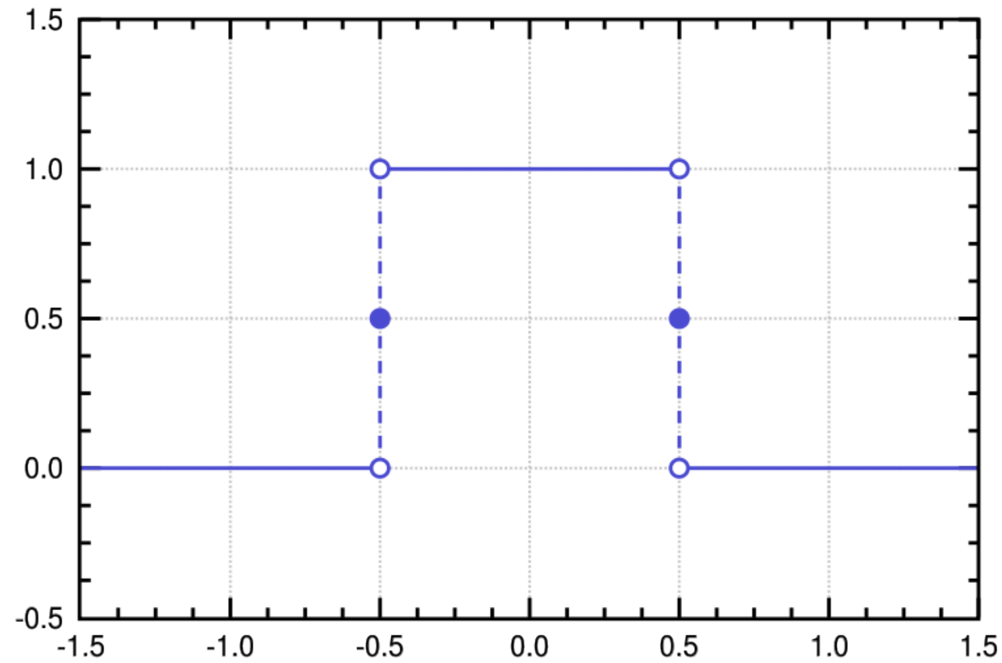
Separable (show on board)

# Sinc filter

$$f(x) = \frac{\sin x}{x}$$




Spatial Kernel

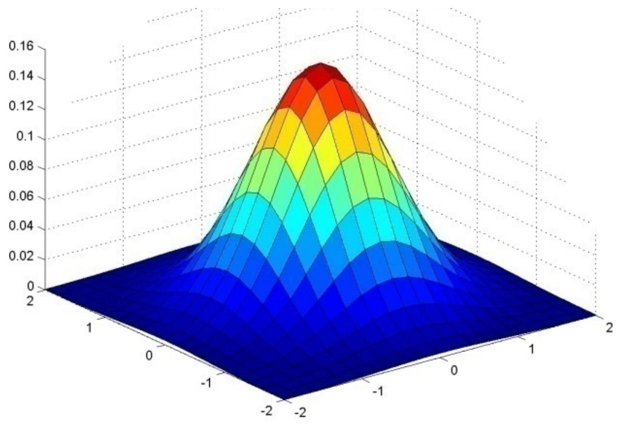


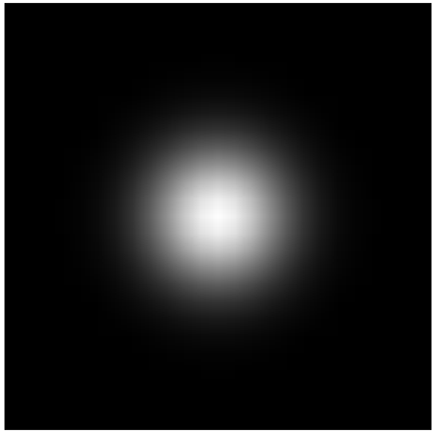

Frequency Response

- Ideal for Nyquist-Shannon: removes high frequencies
- Often a bit higher quality than Gaussian
- But can introduce ringing (oscillations) due to sine

# Important filter: Gaussian

Weight contributions of neighboring pixels by nearness



| | | | | |
|---|---|---|---|---|
| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.022 | 0.097 | 0.159 | 0.097 | 0.022 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |

5 x 5, $\sigma = 1$

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

Same shape in spatial and frequency domain
(Fourier transform of Gaussian is Gaussian)

Slide credit: Christopher Rasmussen

# Gaussian filters

## Remove "high-frequency" components from the image (low-pass filter)

- Images become more smooth

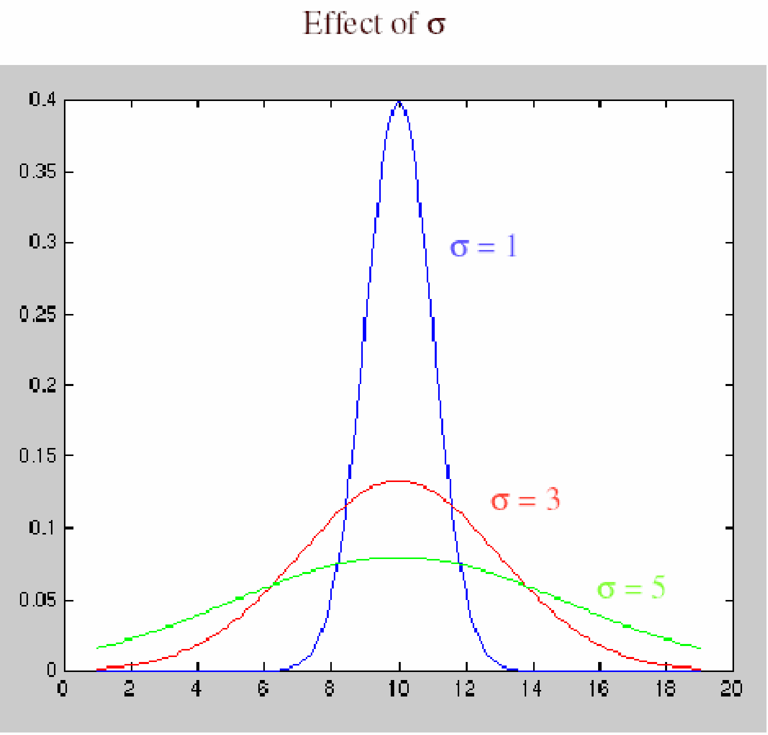
## Convolution with self is another Gaussian

- So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
- Convolving two times with Gaussian kernel of width $\sigma$ is same as convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

# Practical matters

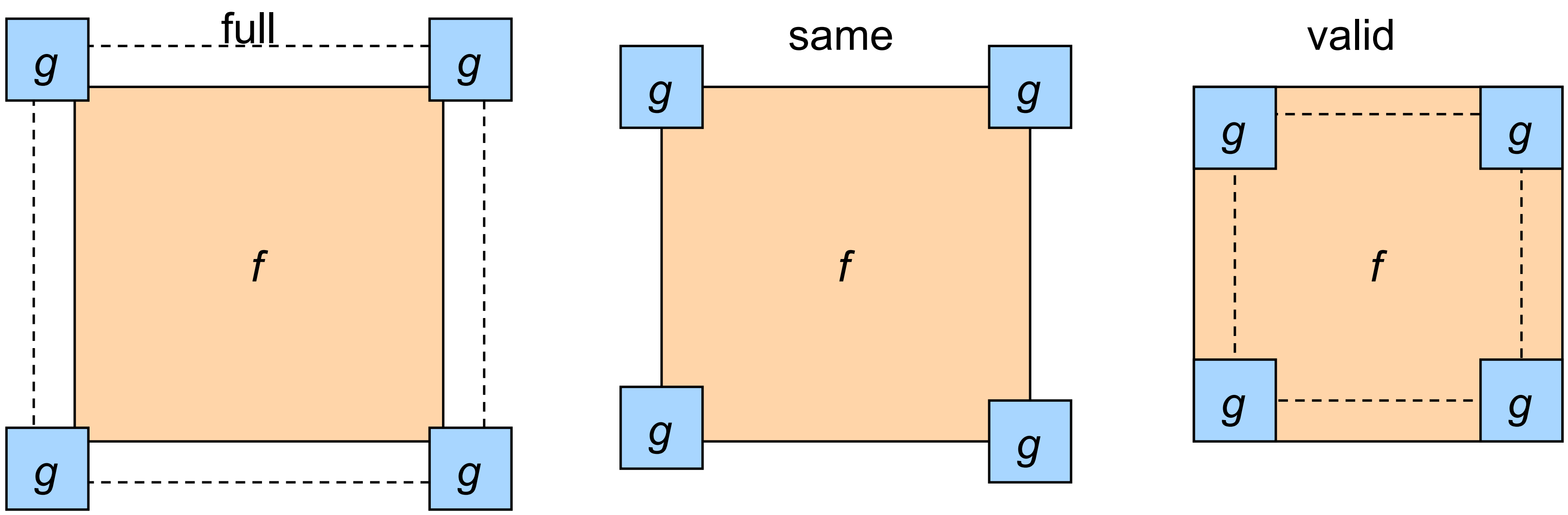
- How big should the filter be?
- Values at edges should be near zero
- Rule of thumb for Gaussian: set filter half-width to about 3 $\sigma$
- Normalize truncated kernel. Why?




Side by Derek Hoiem

# Size of Output?

- MATLAB: conv2(g,f,shape)
- Python: scipy.signal.convolve2d(g,f,shape)
  - *shape* = 'full': output size is sum of sizes of f and g
  - *shape* = 'same': output size is same as f
  - *shape* = 'valid': output size is difference of sizes of f, g

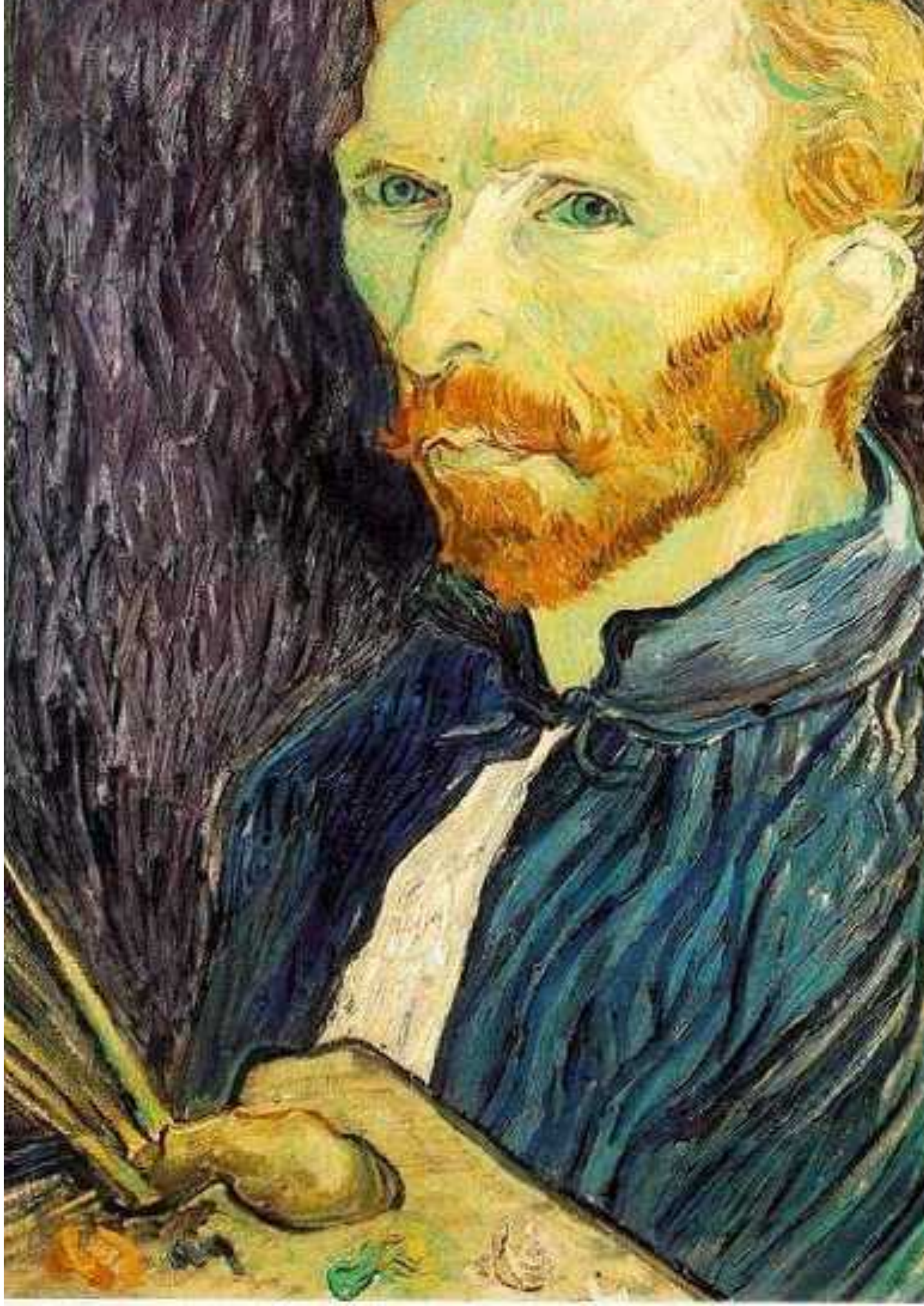



Source: S. Lazebnik

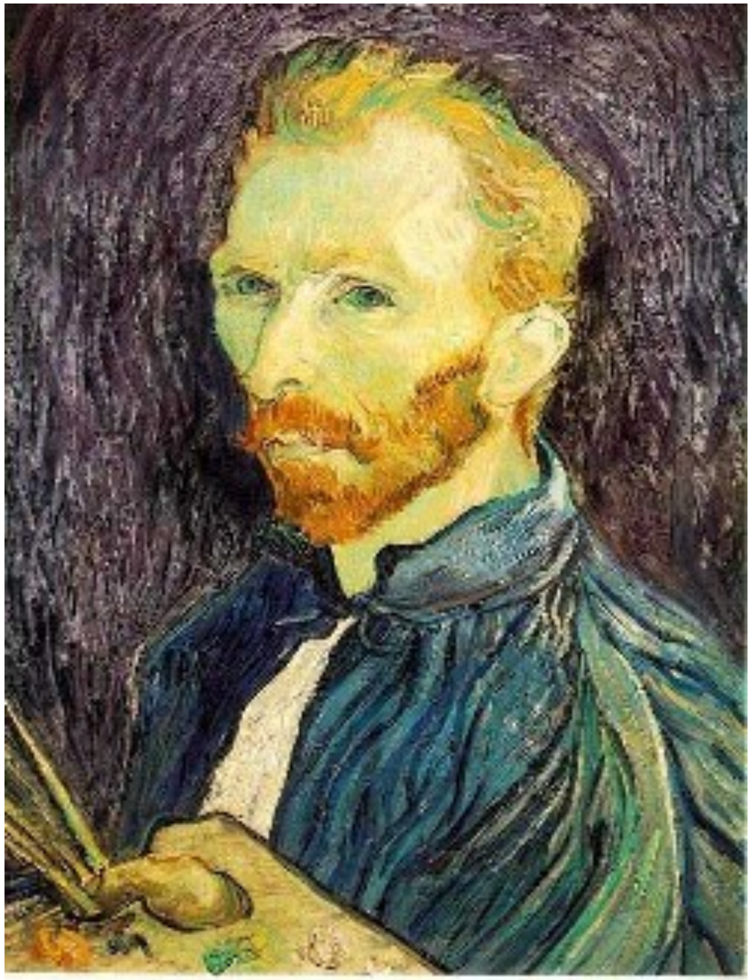
# Image half-sizing

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?



# Image sub-sampling



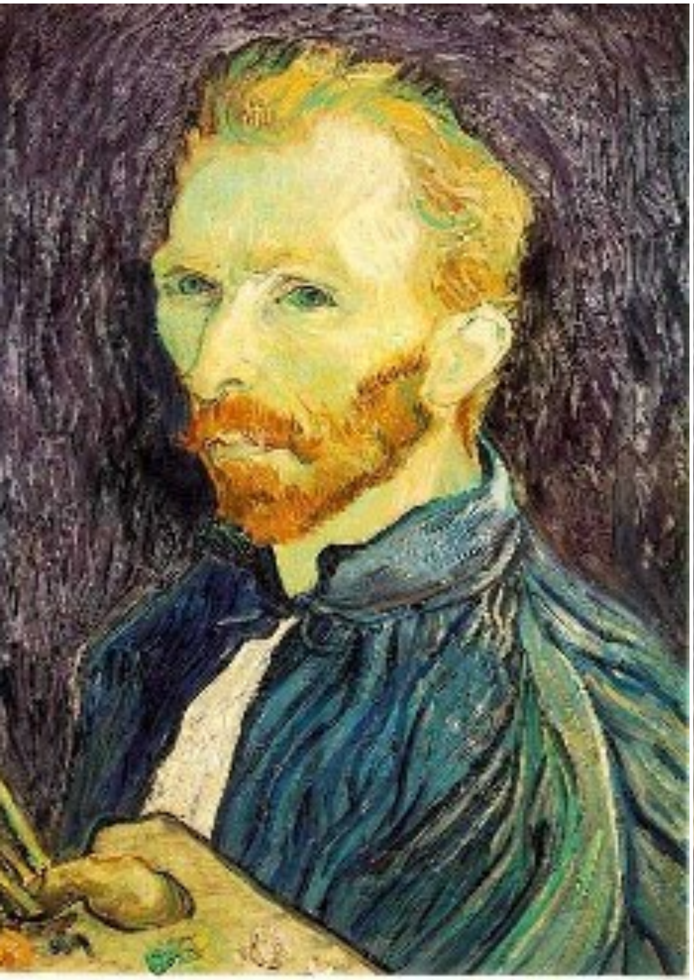
1/4

1/8

Throw away every other row and column to create a 1/2 size image - called *image sub-sampling*

Slide by Steve Seitz

# Image sub-sampling



1/2

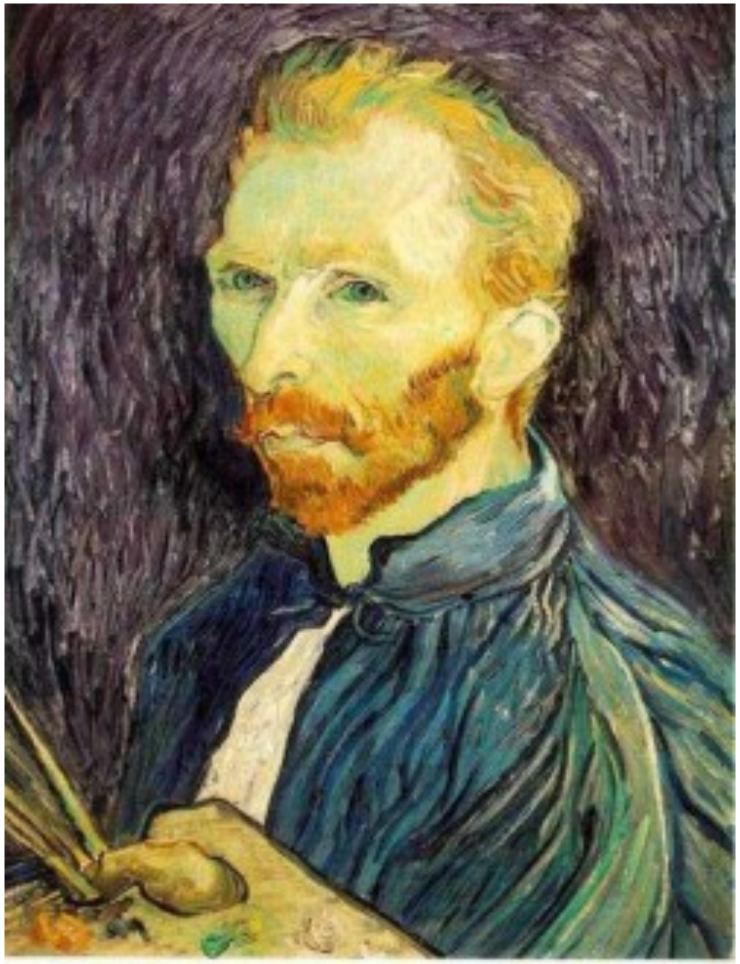
1/4 (2x zoom)

1/8 (4x zoom)

Aliasing! What do we do?

Slide by Steve Seitz

# Gaussian (lowpass) pre-filtering



Gaussian 1/2

G 1/4

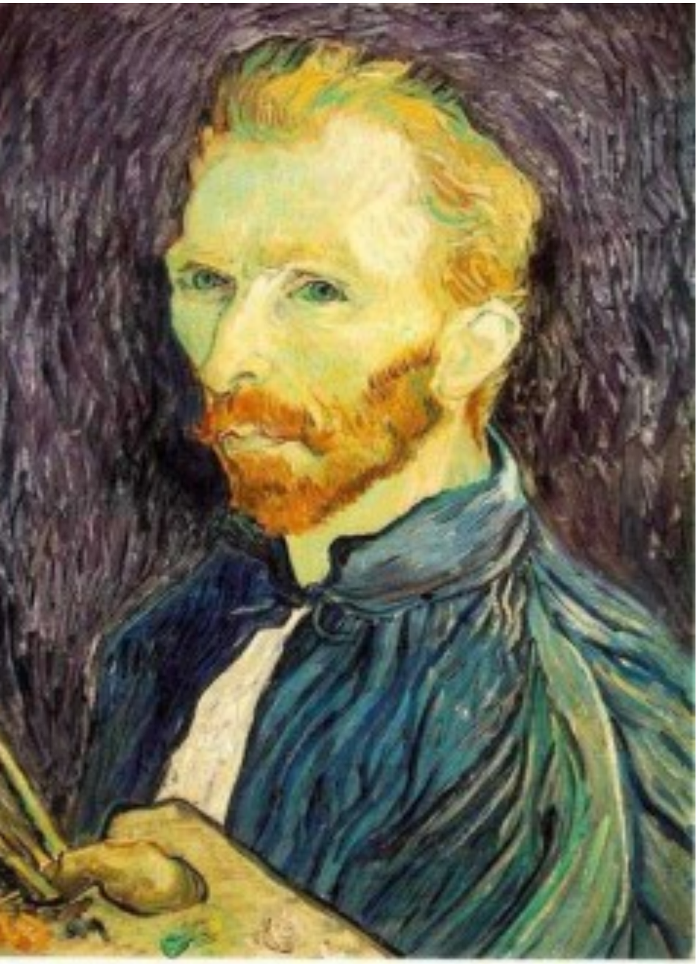
G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?

Slide by Steve Seitz

# Subsampling with Gaussian pre-filtering



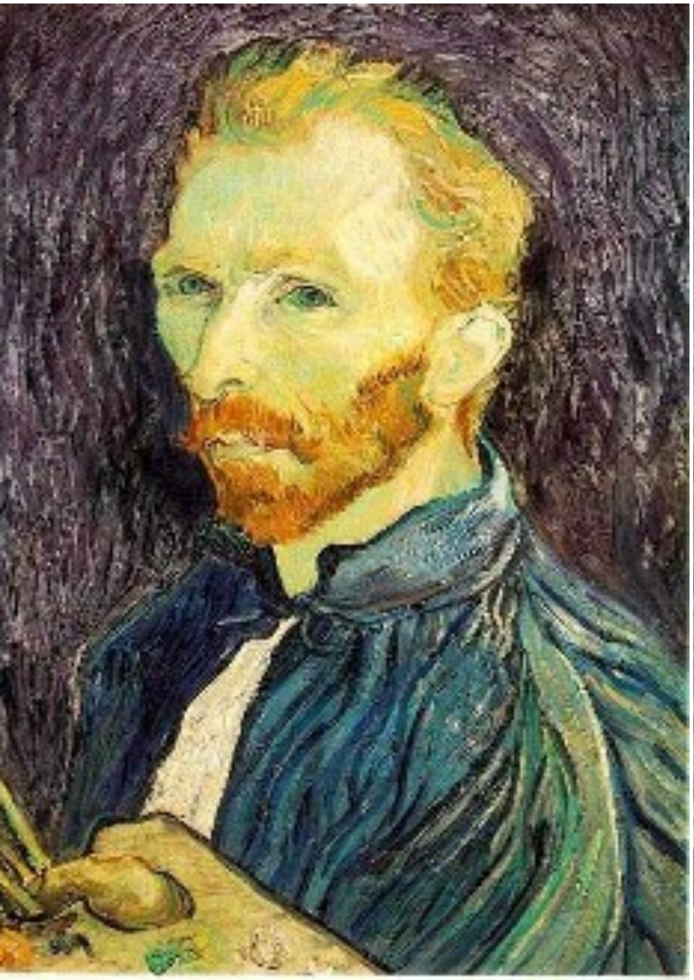
Gaussian 1/2

G 1/4

G 1/8

Slide by Steve Seitz

# Compare with...



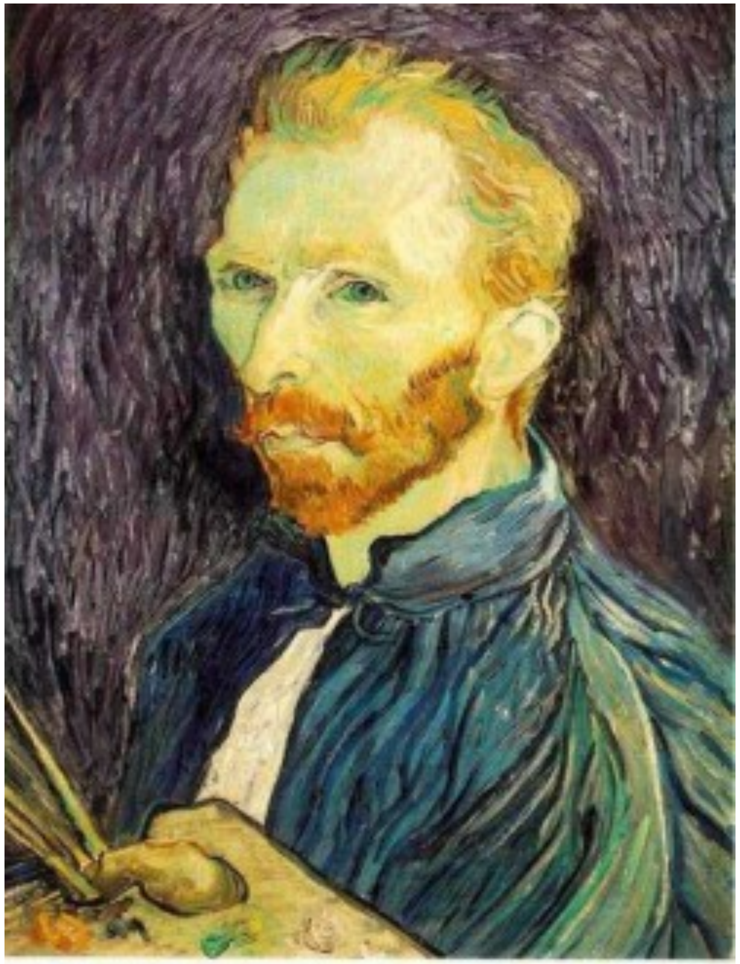
1/2

1/4 (2x zoom)

1/8 (4x zoom)

Slide by Steve Seitz

# Gaussian (lowpass) pre-filtering



Gaussian 1/2

G 1/4

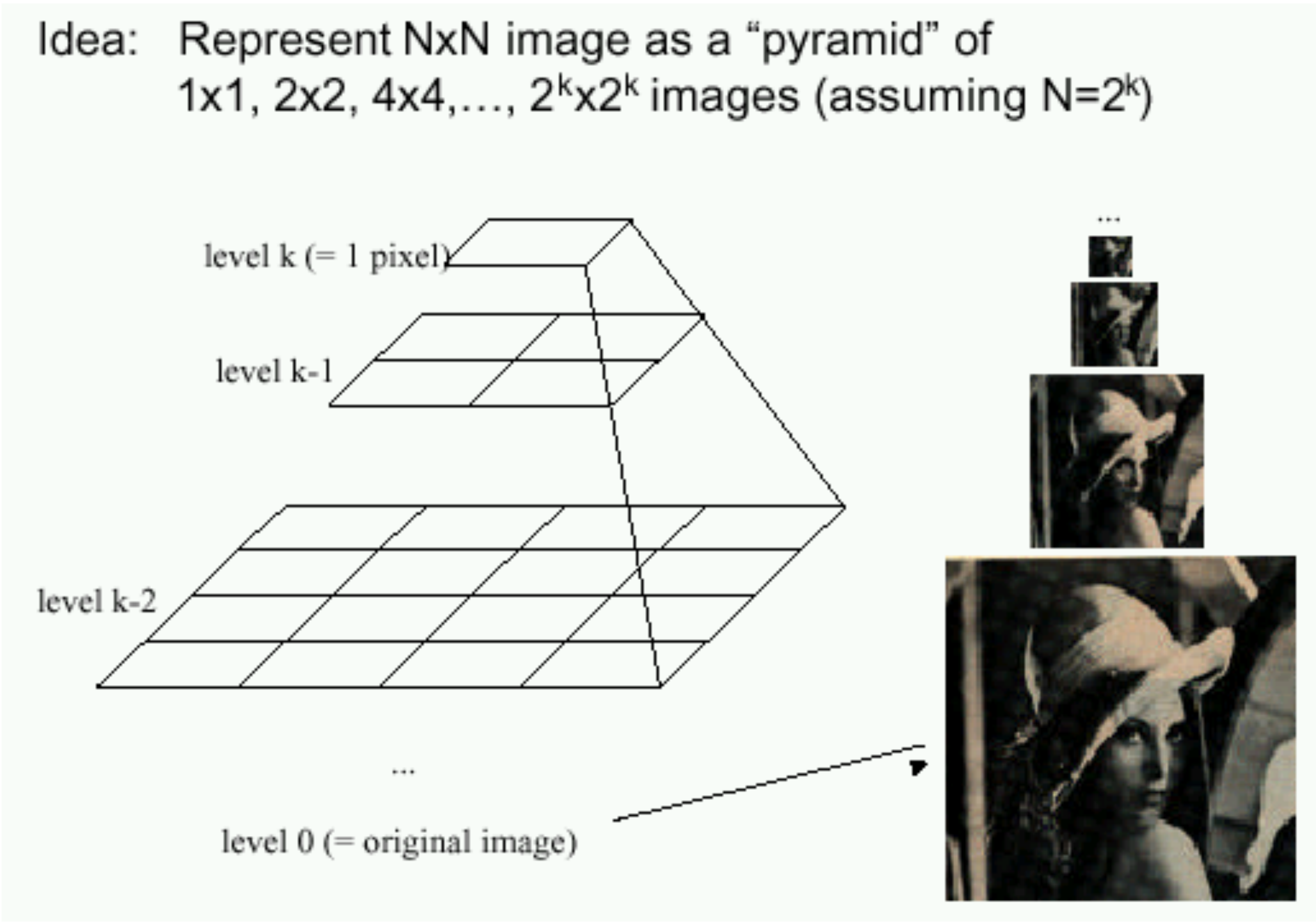
G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?
- How can we speed this up?

Slide by Steve Seitz

# Image Pyramids




Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

Slide by Steve Seitz

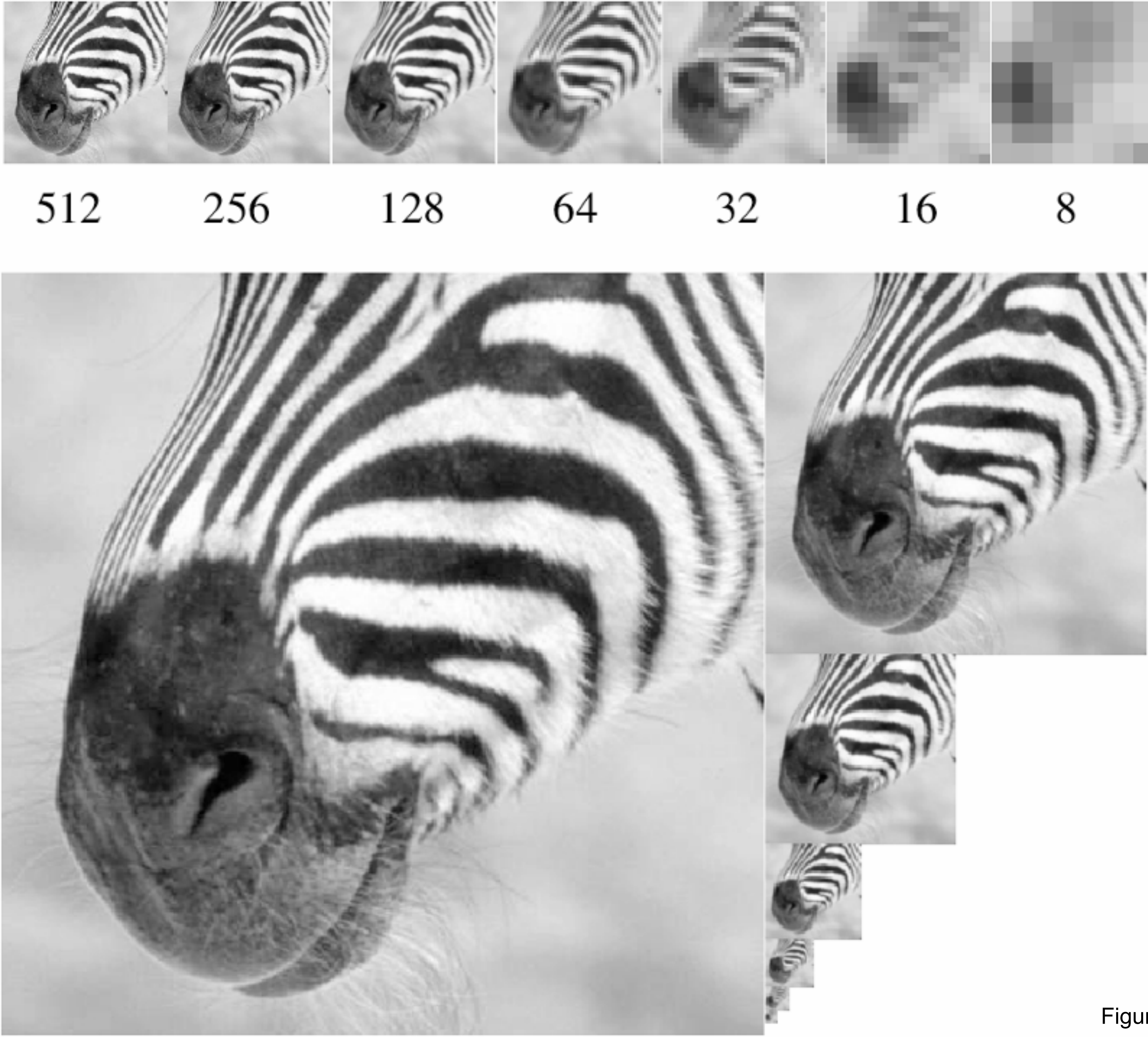



Figure from David Forsyth