Today

- Image formation
- Color
- Filtering

Image formation

- How are objects in the world captured in an image?
  - Mapping between image and world coordinates

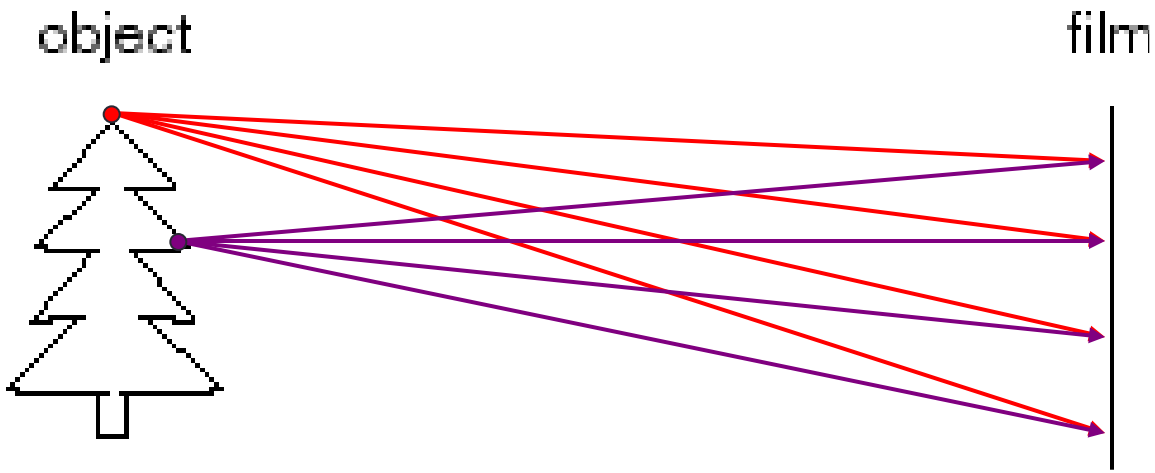
Physical parameters of image formation

- Photometric
  - Surfaces' reflectance properties
  - Direction and intensity of light reaching sensor
- Geometric
  - Type of projection
    - Pinhole camera model
    - Projective geometry
  - Camera pose
- Optical
  - Sensor's lens type
  - focal length, field of view, aperture, shutter speed

Slide credit: Kristen Grauman

What do you need to make a camera from scratch?

Image formation

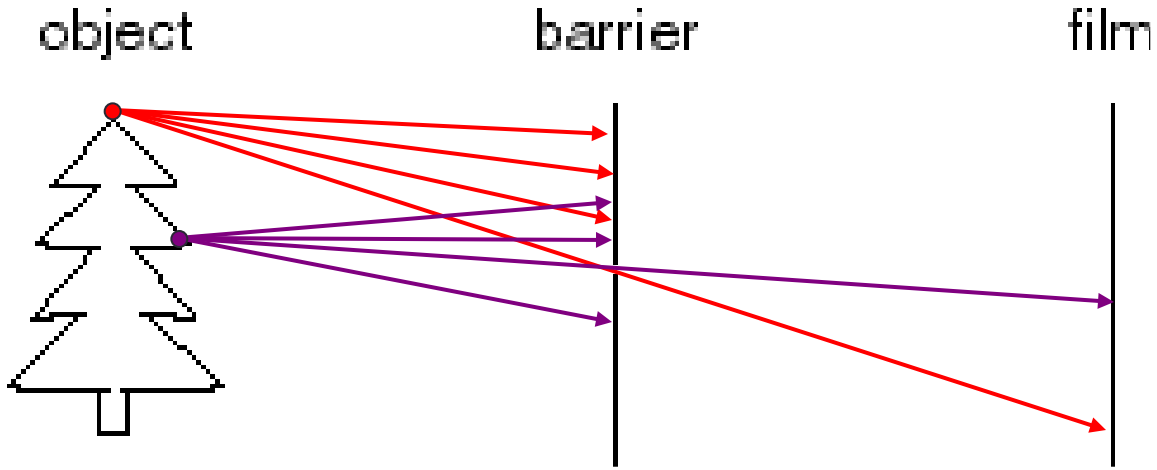


Let’s design a camera

- Idea 1: put a piece of film in front of an object
- Do we get a reasonable image?

Slide source: Seitz

Pinhole camera

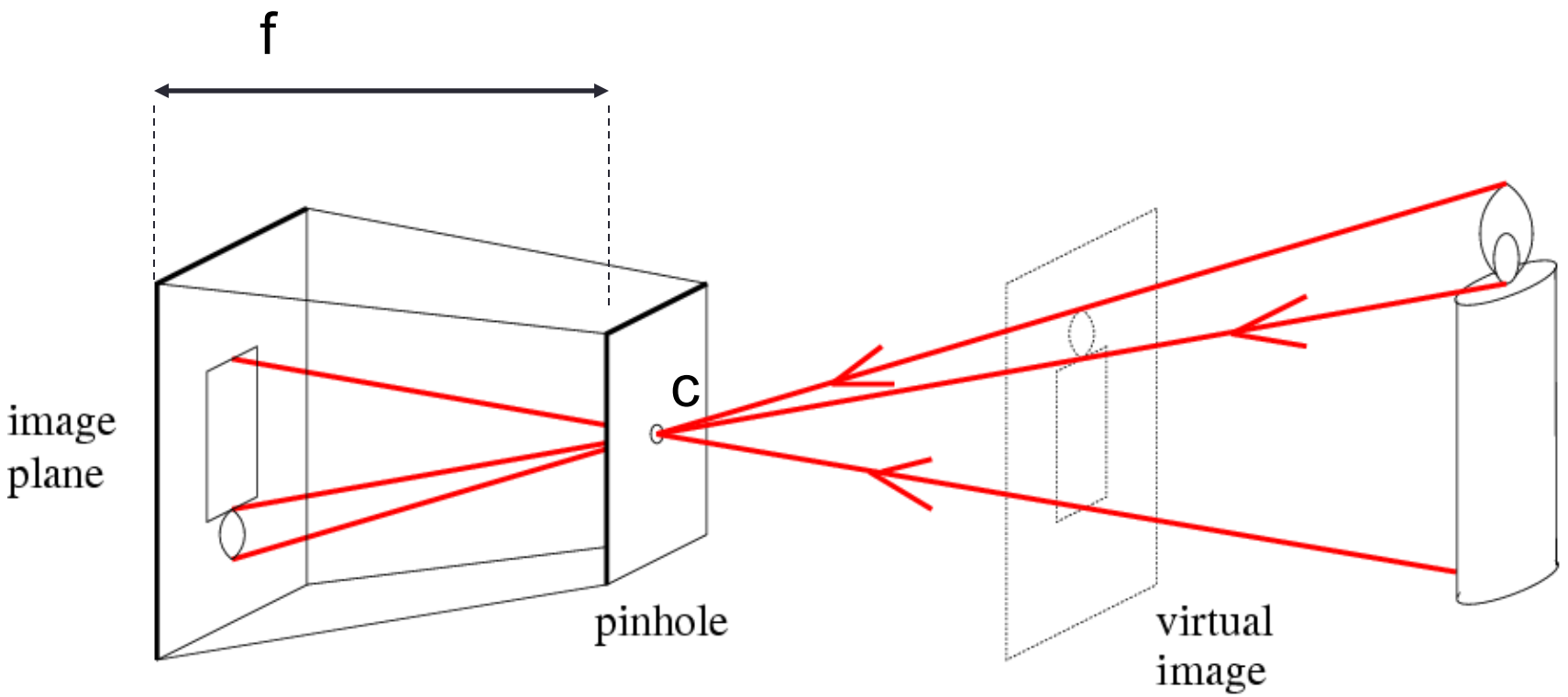


Idea 2: add a barrier to block off most of the rays

- This reduces blurring
- The opening known as the **aperture**

Slide source: Seitz

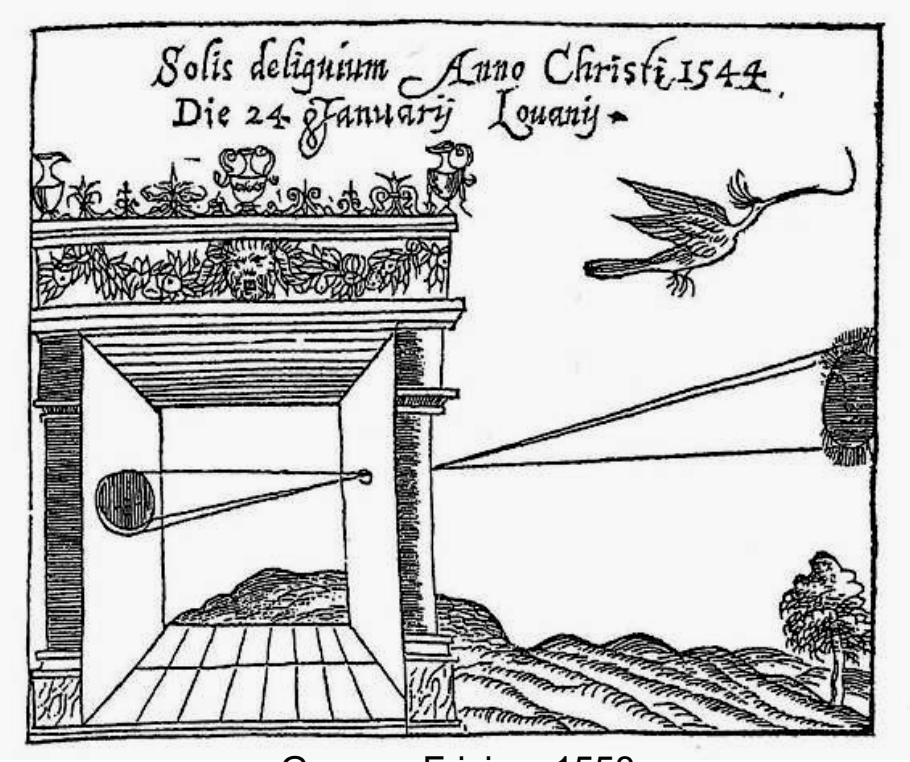
Pinhole camera



f = focal length
c = center of the camera

Figure from Forsyth

Camera Obscura

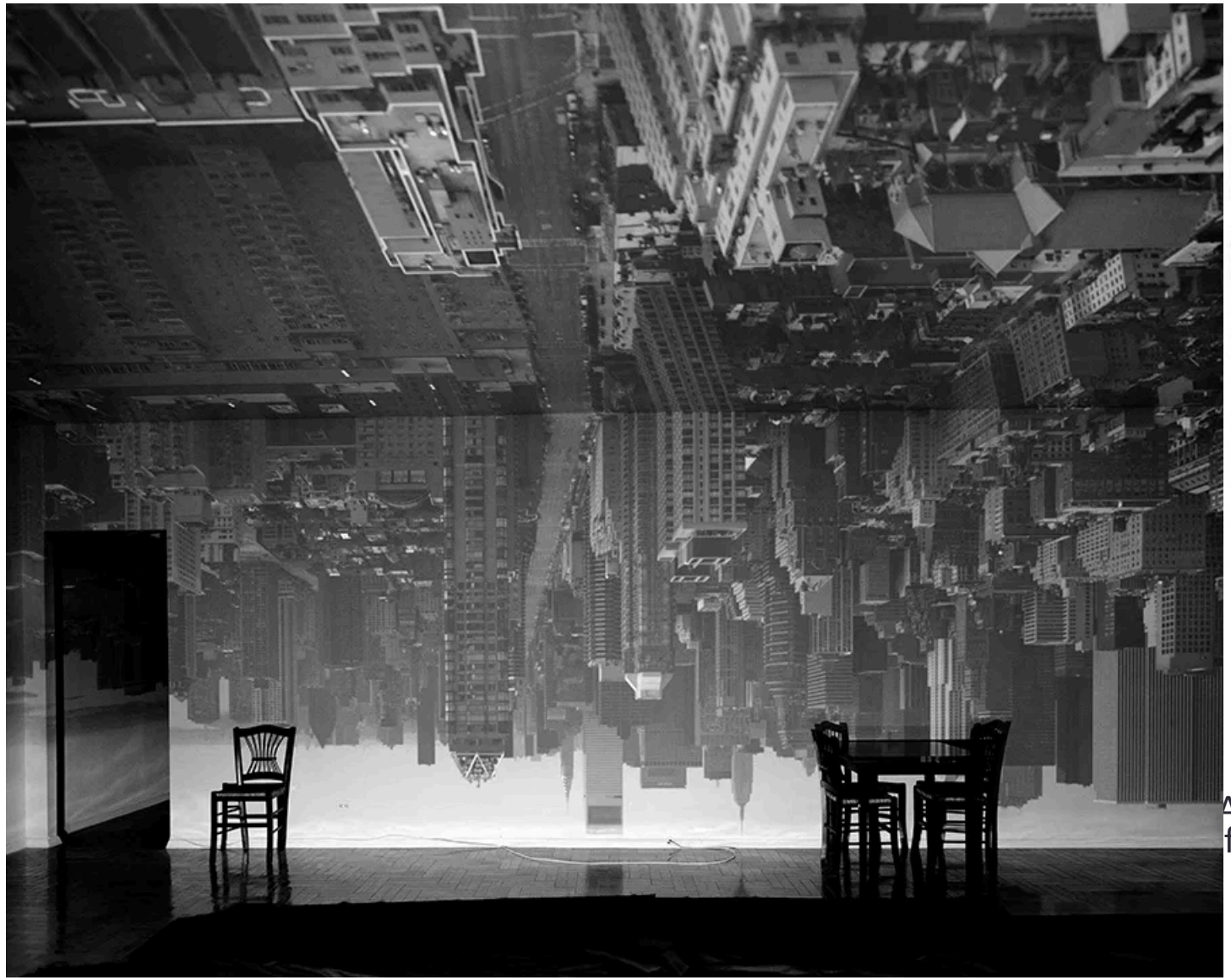


Gemma Frisius, 1558

- Basic principle known to Mozi (470-390 BCE), Aristotle (384-322 BCE)
- Drawing aid for artists: described by Leonardo da Vinci (1452-1519)

Source: A. Efros

Camera Obscura



After scouting rooms and reserving one for at least a day, Morell masks the windows except for the aperture. He controls three elements: the size of the hole, with a smaller one yielding a sharper but dimmer image; the length of the exposure, usually eight hours; and the distance from the hole to the surface on which the outside image falls and which he will photograph. He used 4 x 5 and 8 x 10 view cameras and lenses ranging from 75 to 150 mm.

After he's done inside, it gets harder. "I leave the room and I am constantly checking the weather, I'm hoping the maid reads my note not to come in, I'm worrying that the sun will hit the plastic masking and it will fall down, or that I didn't trigger the lens."

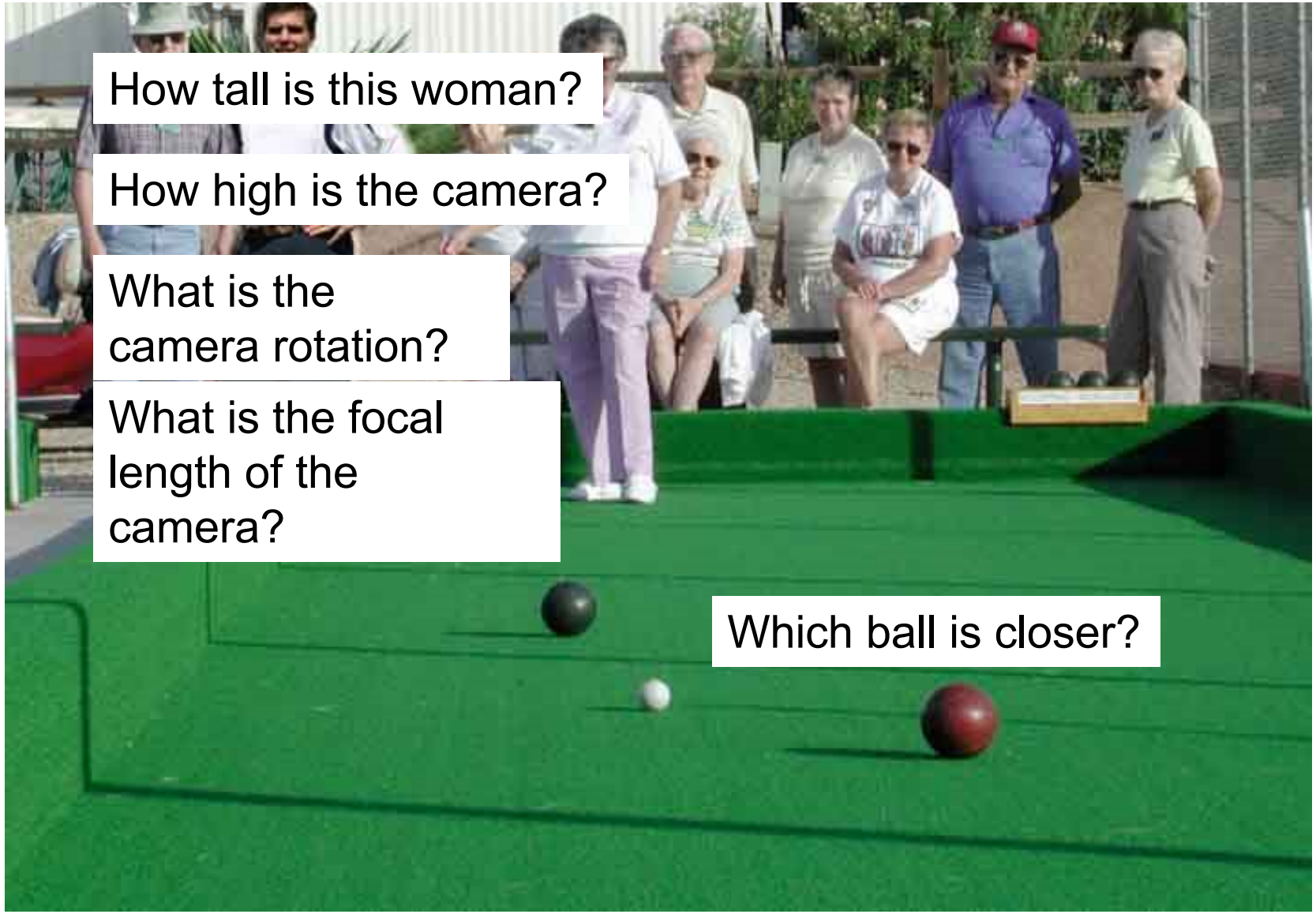
From *Grand Images Through a Tiny Opening*, **Photo District News,** February 2005

Abelardo Morell, Camera Obscura Image f Manhattan View Looking South in Large Room, 1996

http://www.abelardomorell.net/project/camera-obscura/

Abelardomorell.com

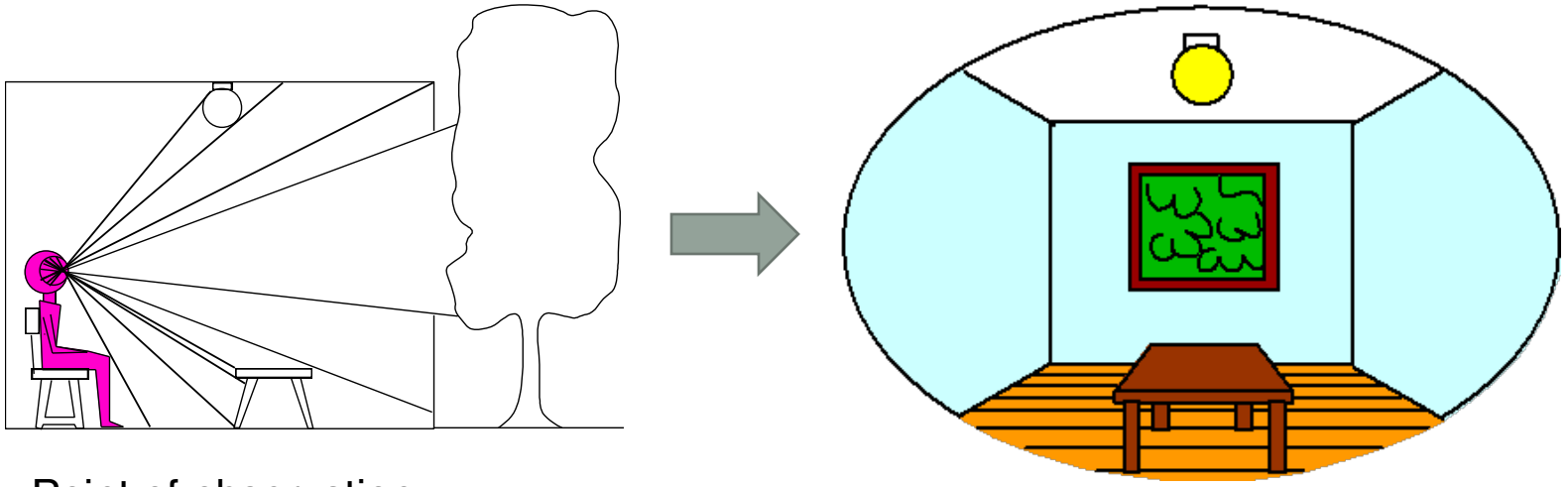
Camera and World Geometry



Dimensionality Reduction Machine (3D to 2D)

3D world

2D image



Point of observation

Figures © Stephen E. Palmer, 2002

Slide source: Seitz

Projection can be tricky…

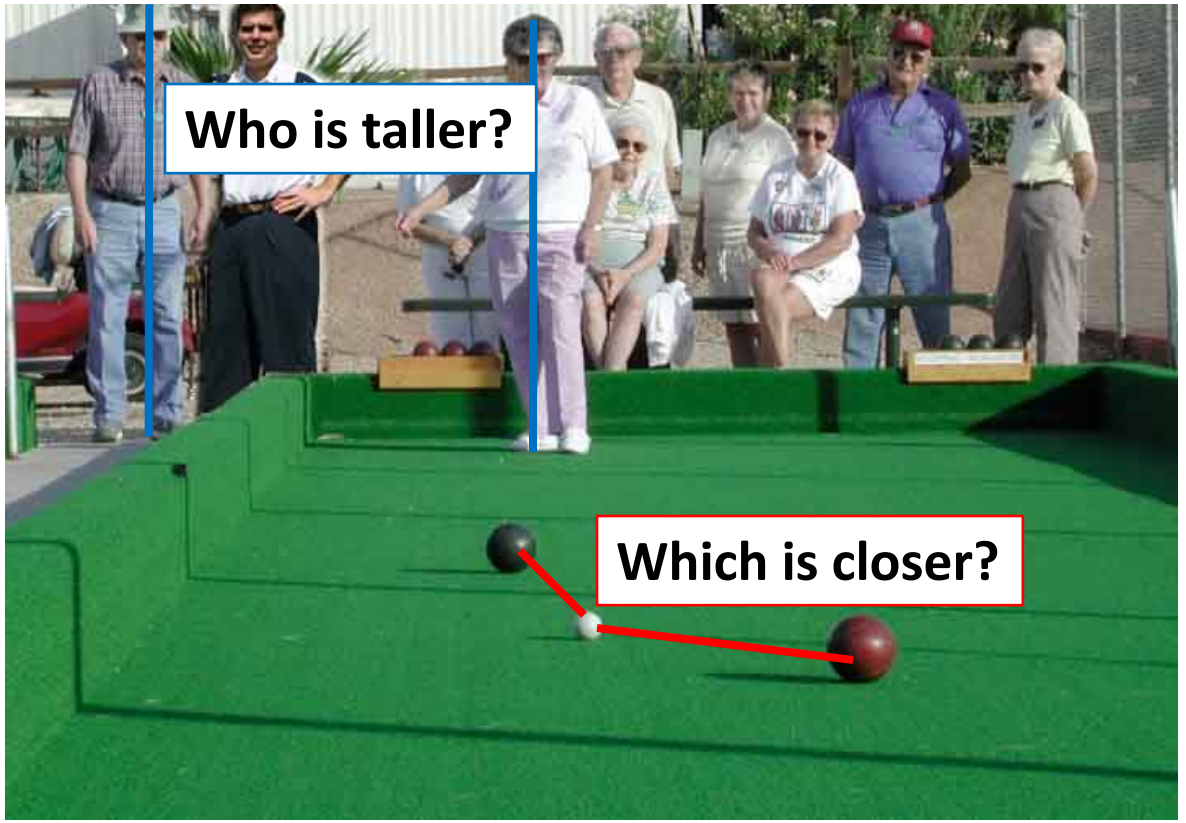
Slide source: Seitz

Projection can be tricky…

Projective Geometry

What is lost?

- Length



Length and area are not preserved

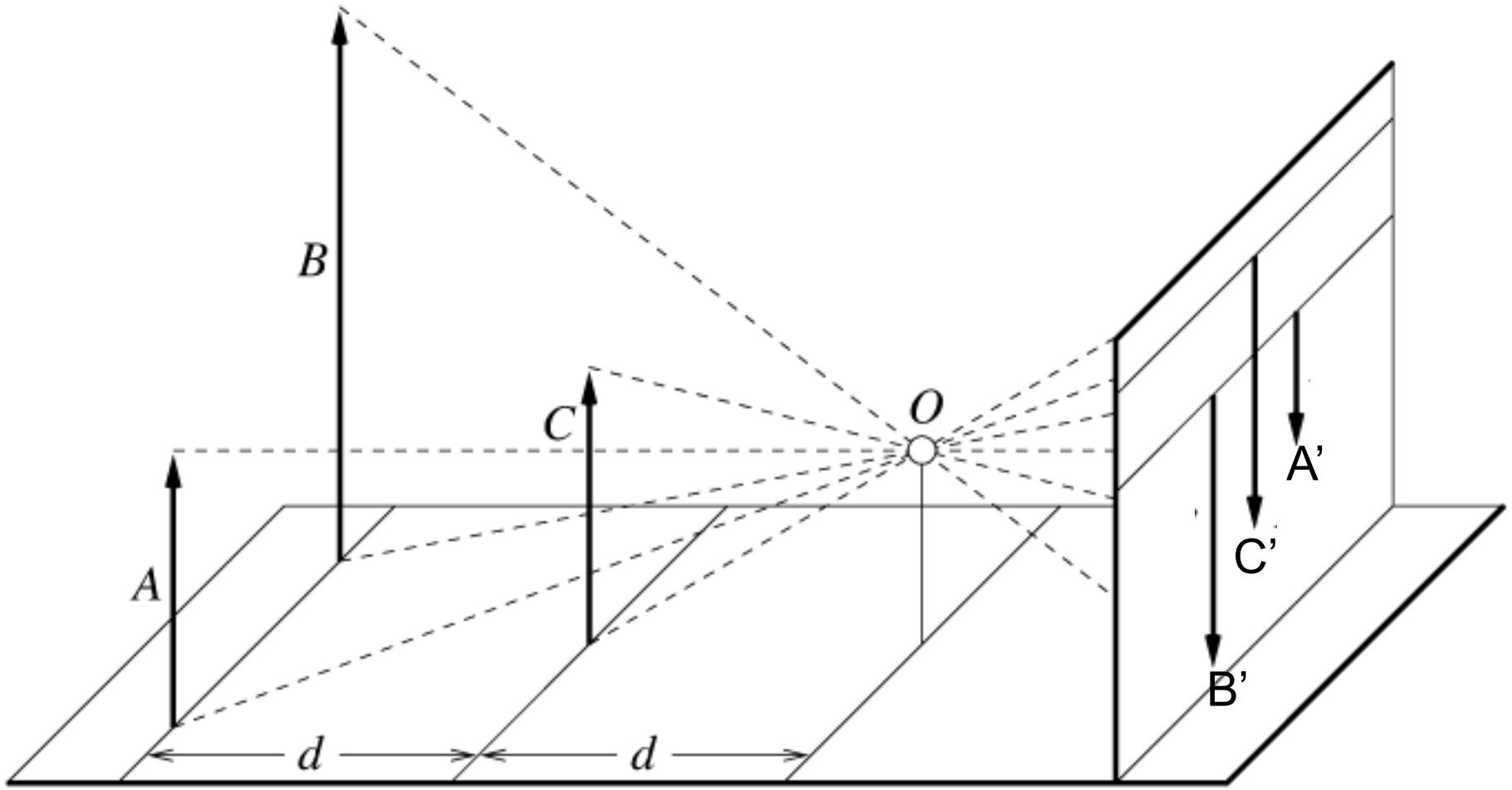
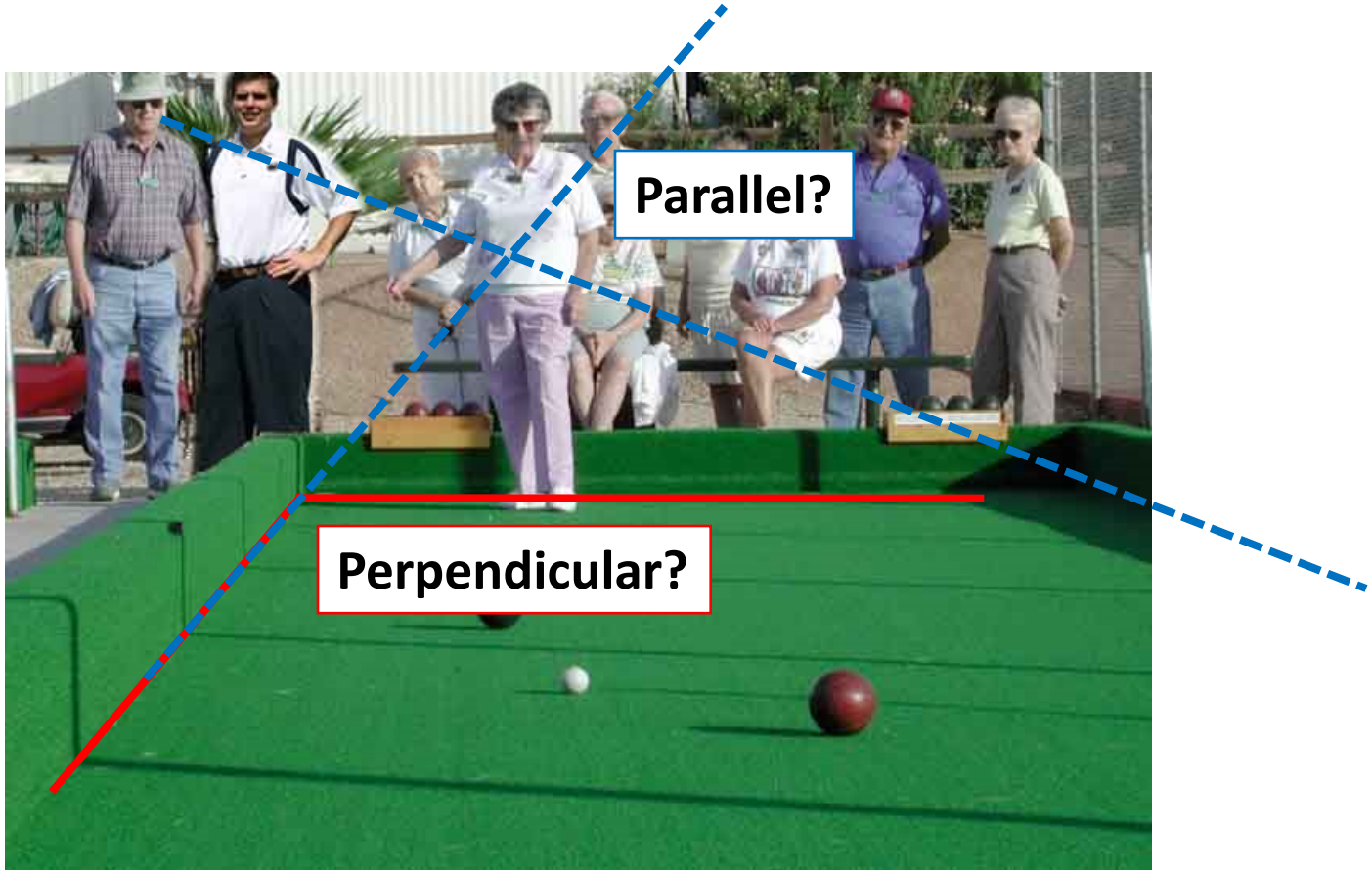


Figure by David Forsyth

Projective Geometry

What is lost?

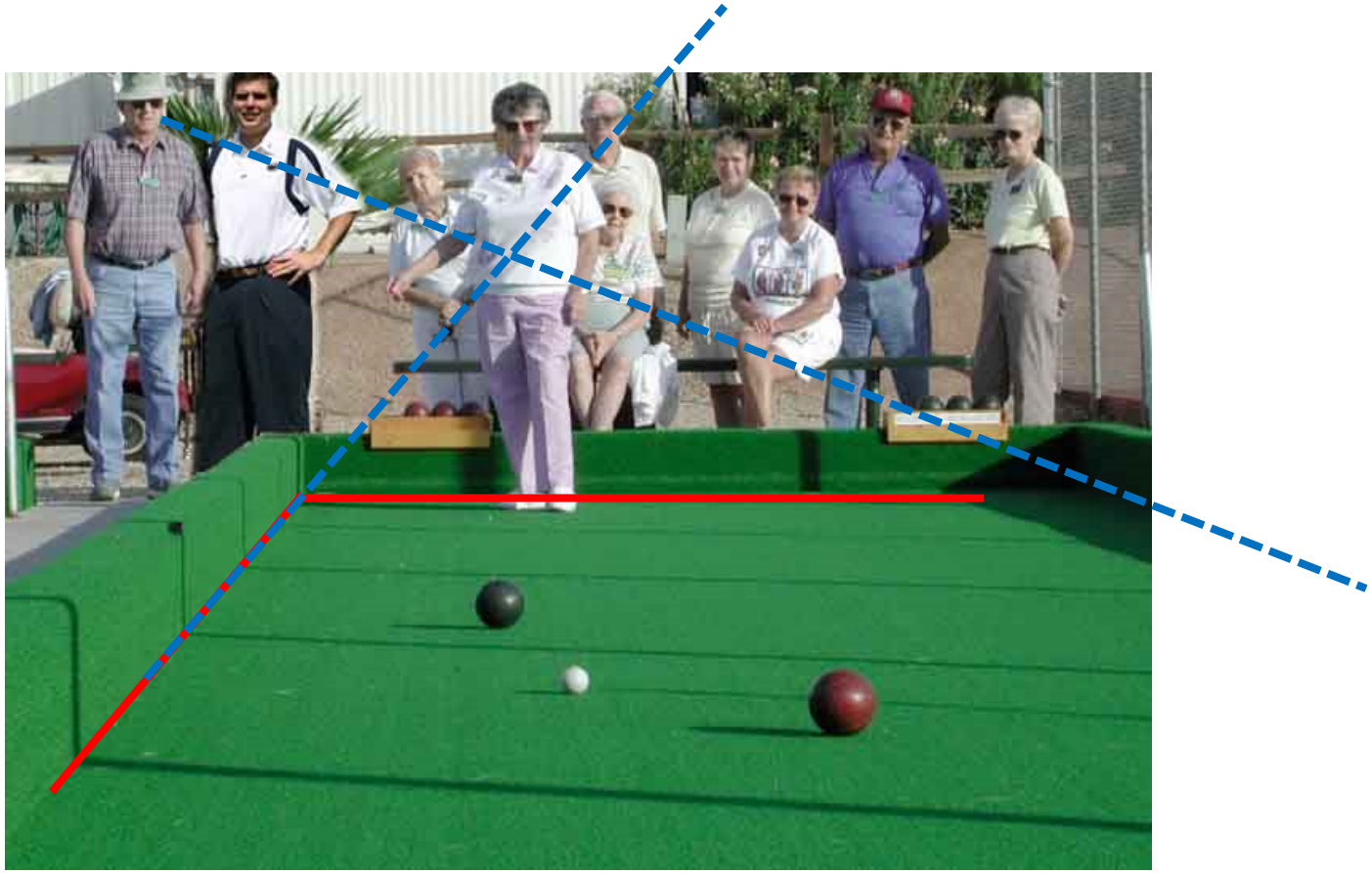
- Length
- Angles



Projective Geometry

What is preserved?

- Straight lines are still straight



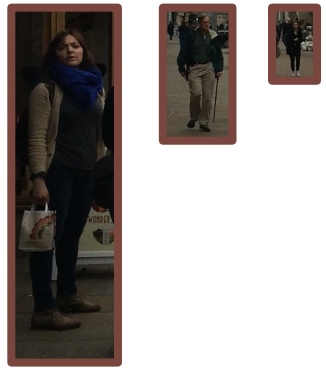
Some Facts About Projection

List of properties from M. Hebert

3D lines project to 2D lines

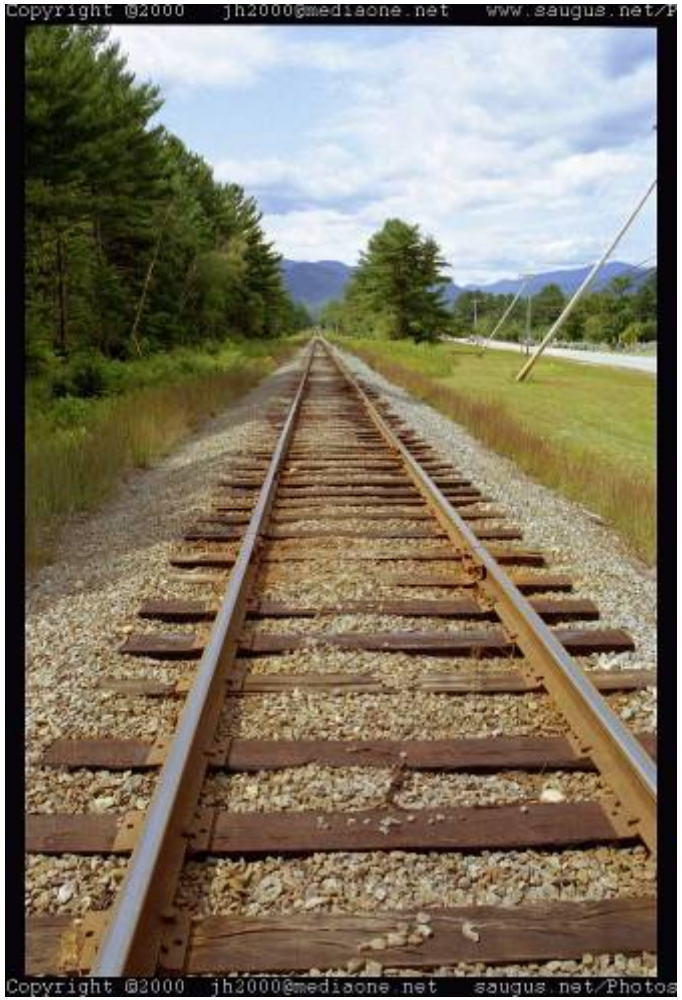
The projection of any 3D parallel lines converge at a vanishing point

Distant objects are smaller

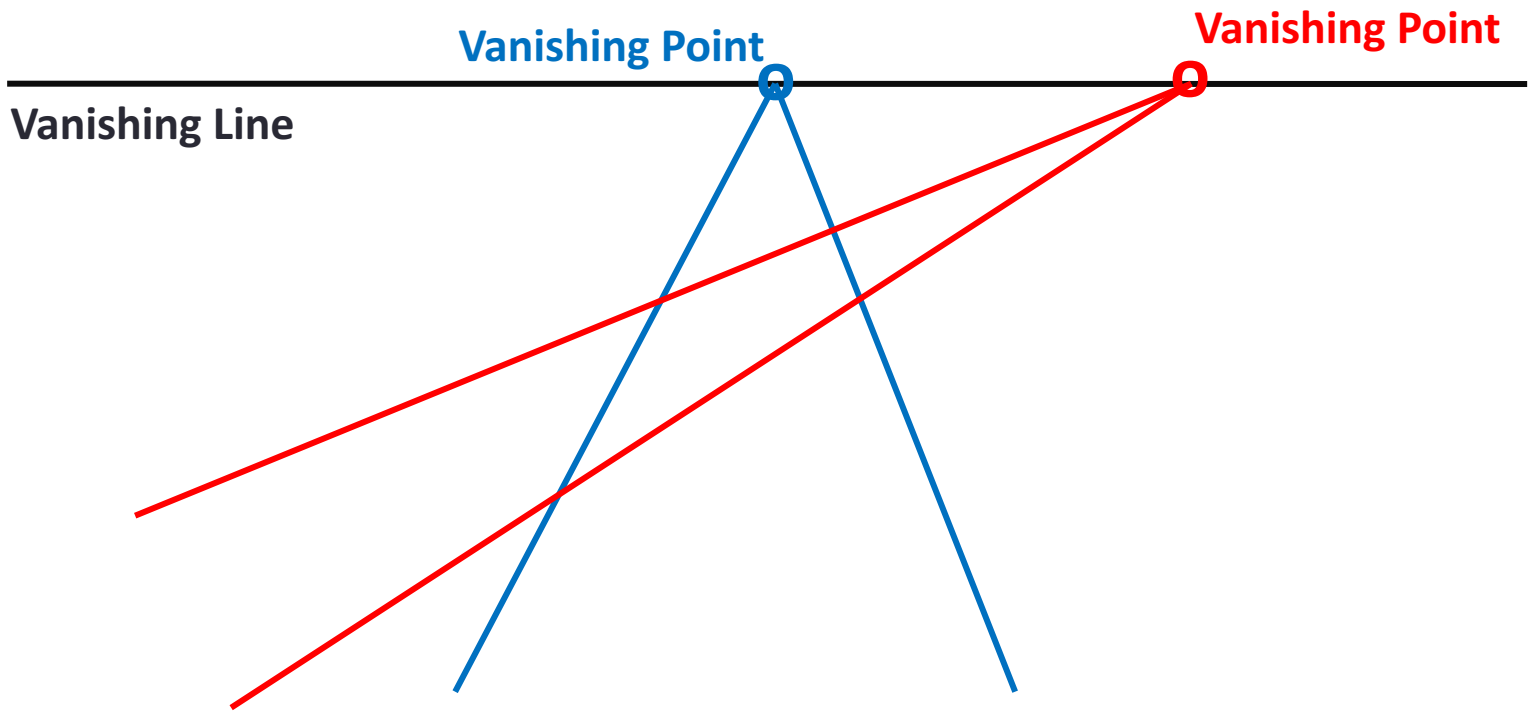


Vanishing points and lines

Parallel lines in the world intersect in the image at a “vanishing point”



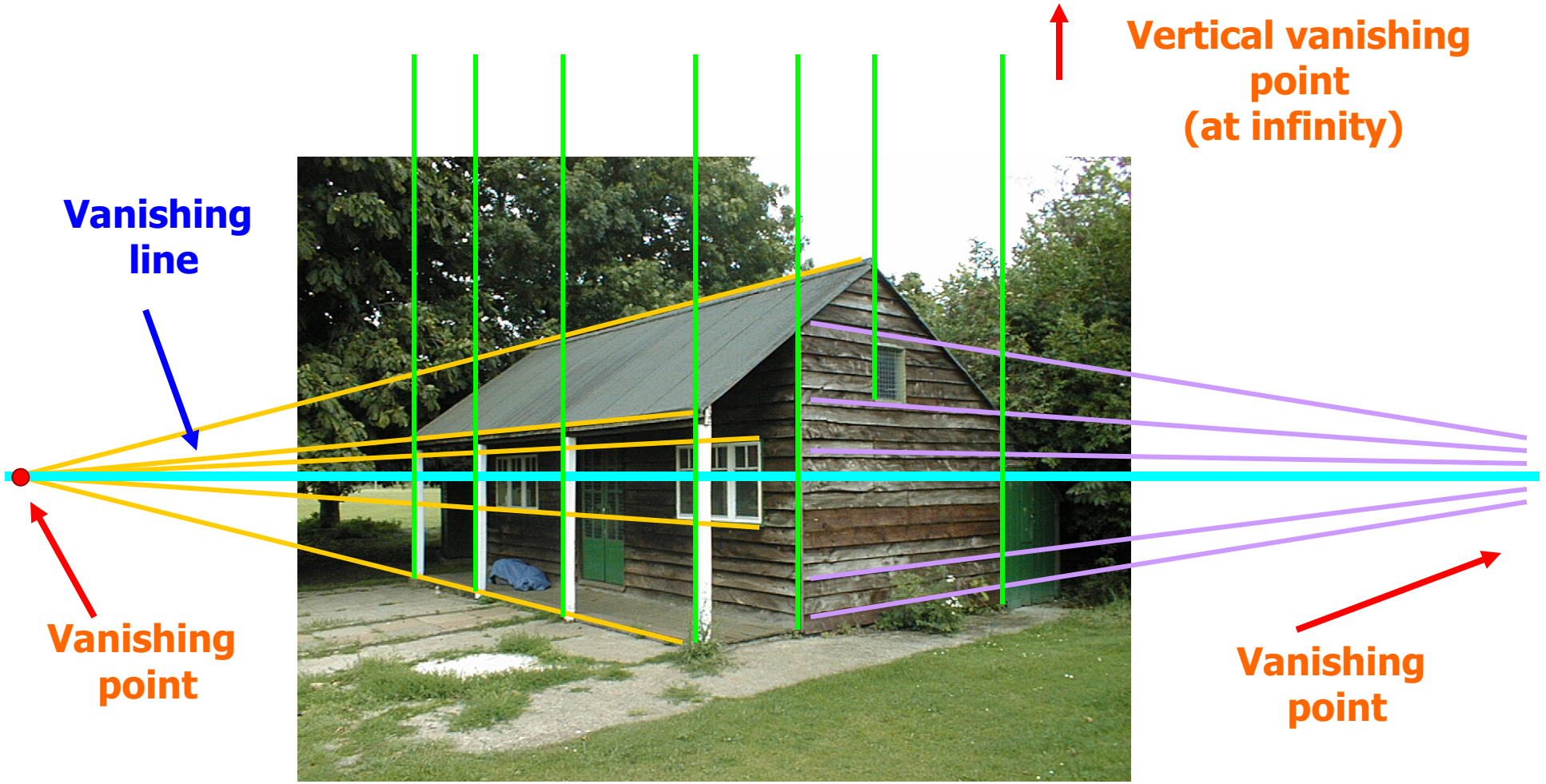
Vanishing points



- The projections of parallel 3D lines intersect at a **vanishing point**
- Vanishing point <-> 3D direction of a line
- Not all lines that intersect are parallel

Slide source: Derek Hoiem

Vanishing points

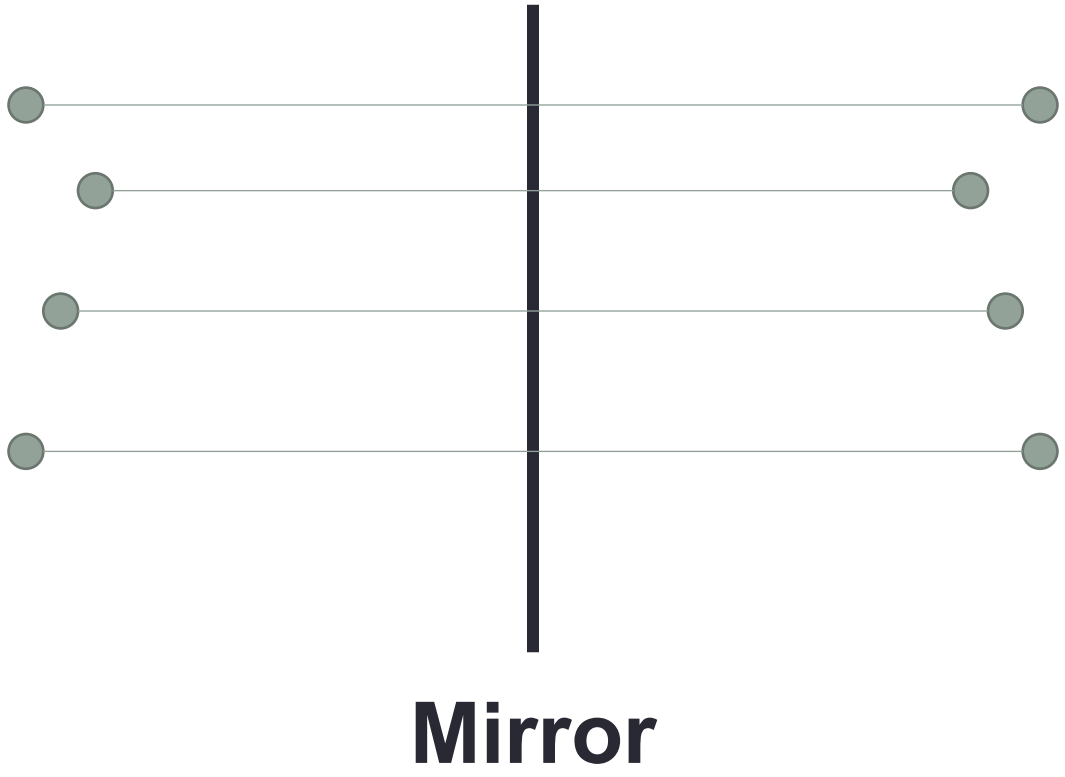


Slide from Efros, Photo from Criminisi

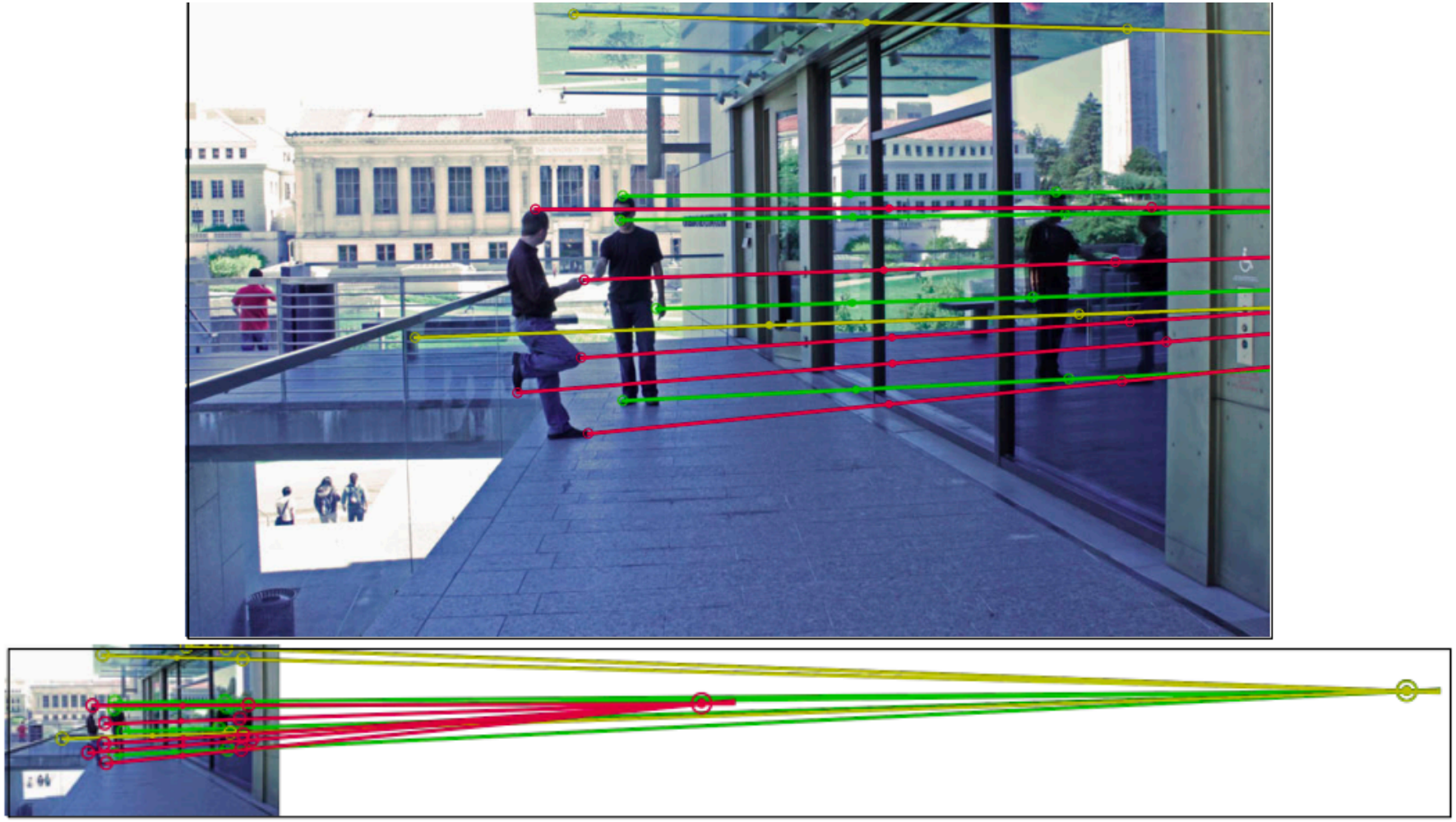
What's This Useful For?

O'Brien and Farid, SIGGRAPH 2012, Exposing Photo Manipulation with Inconsistent Reflections

What's This Useful For?

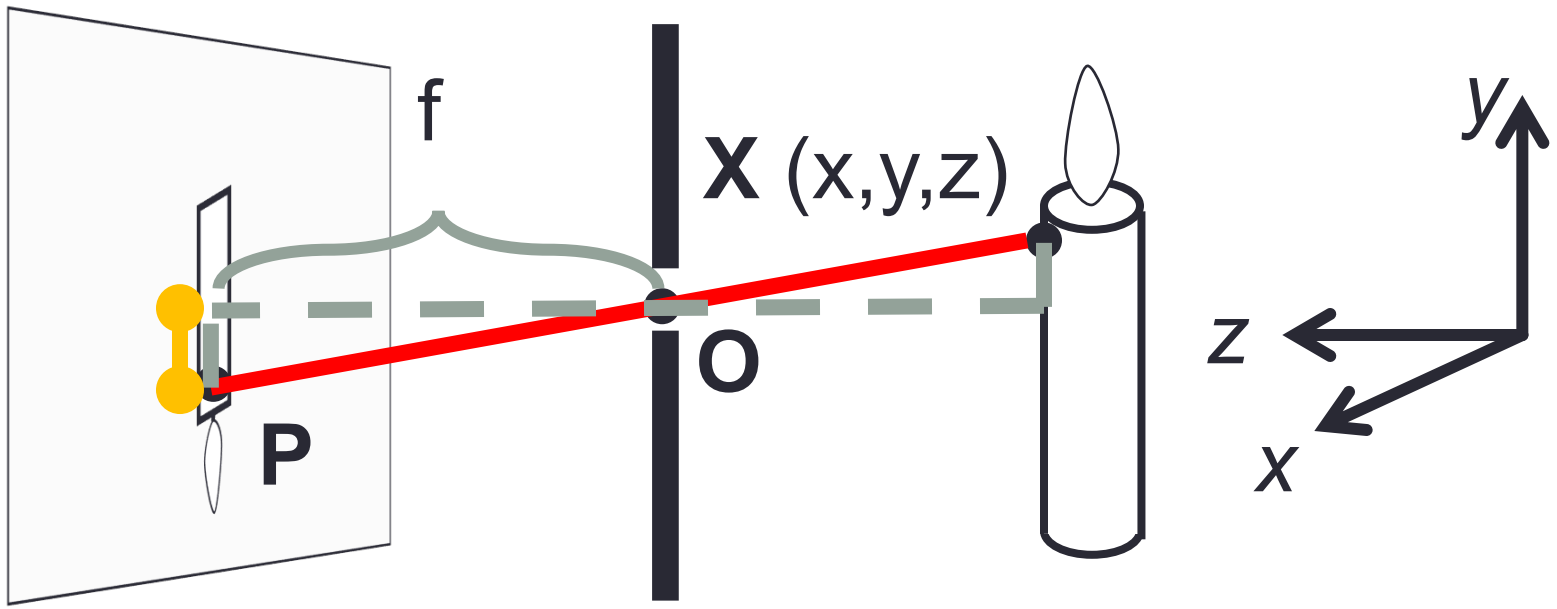


What's This Useful For?



O'Brien and Farid, SIGGRAPH 2012, Exposing Photo Manipulation with Inconsistent Reflections

Projection Equations

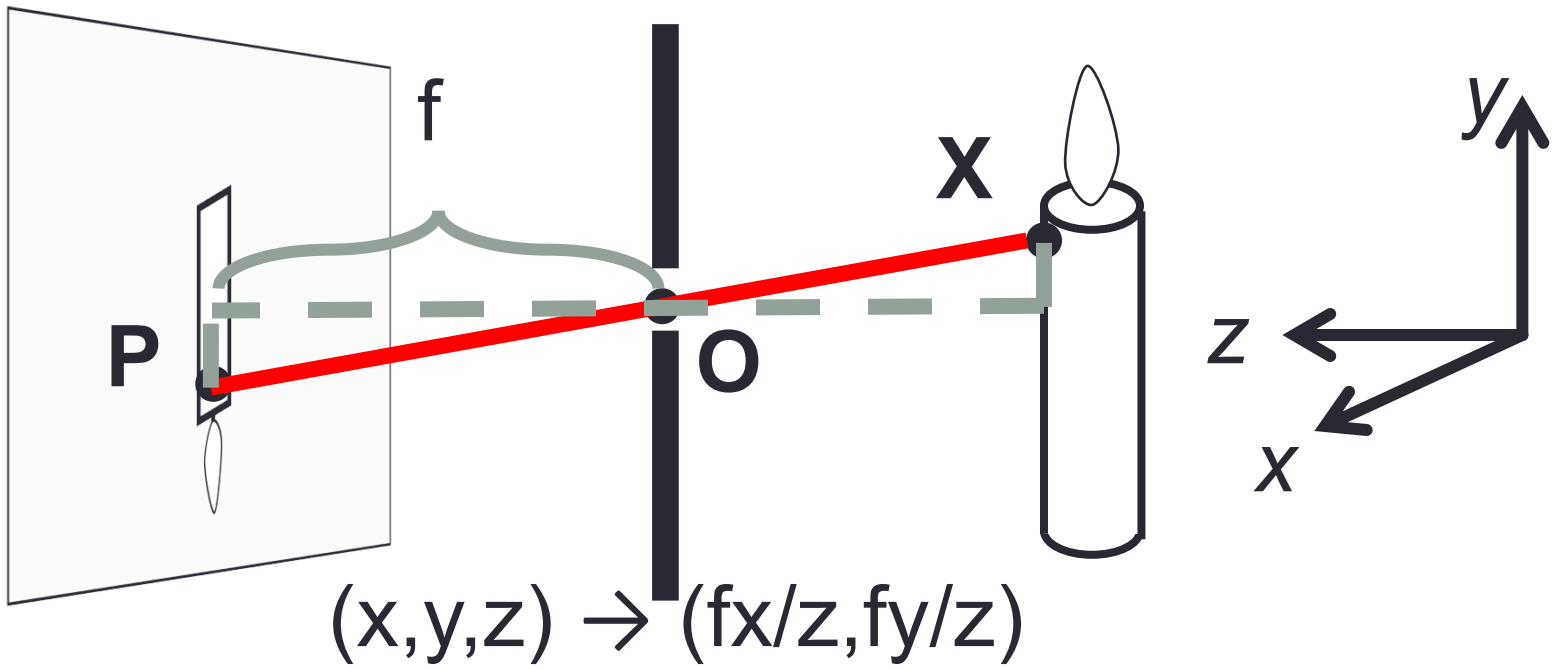


Coordinate system: **O** is origin, XY in image, Z sticks out.
XY is image plane, Z is optical axis.

(x,y,z) projects to (fx/z,fy/z) via similar triangles

Source: L Lazebnik

Projection Equation



is this linear?

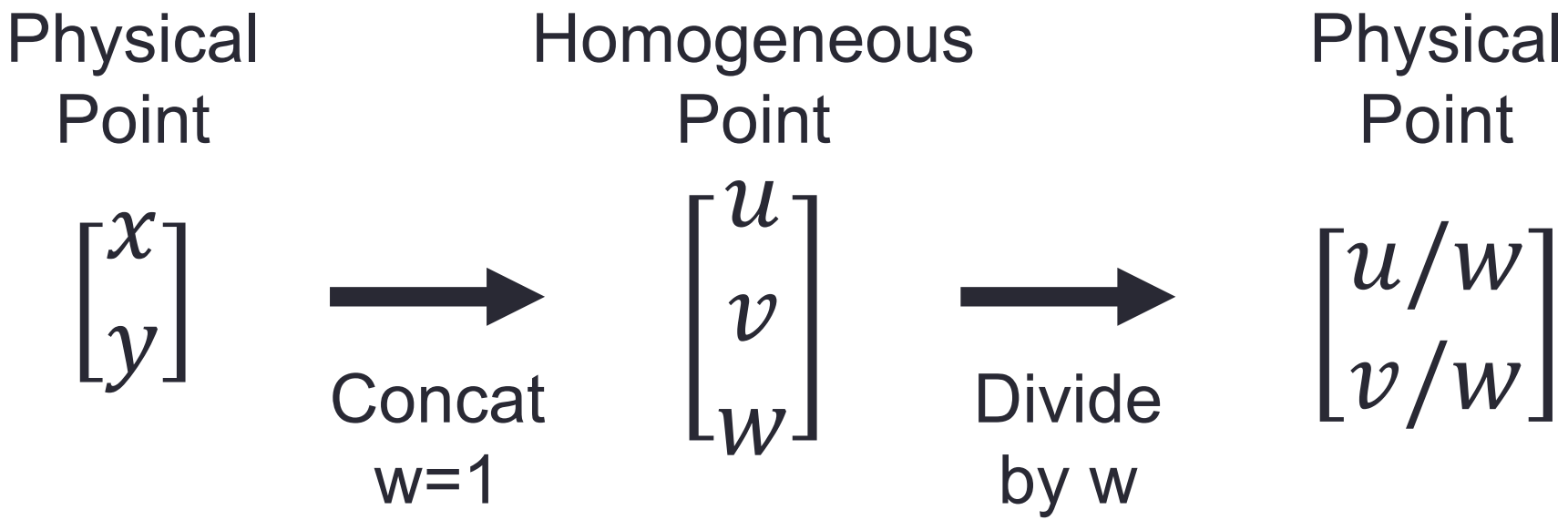
Nope: division by z is non-linear
(and risks division by 0)

Adapted from S. Seitz slide

Homogeneous Coordinates (2D)

Trick: add a dimension!

This also clears up lots of nasty special cases



What if w = 0?

Adapted from M. Hebert slide

Benefits of Homogeneous Coords

General equation of 2D line:

$$ax + by + c = 0$$

Homogeneous Coordinates

$$\boldsymbol{l}^T \boldsymbol{p} = 0, \qquad \boldsymbol{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \boldsymbol{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Slide from M. Hebert

Homogeneous coordinates

Conversion

Converting to *homogeneous* coordinates

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

homogeneous image coordinates

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

homogeneous scene coordinates

Converting *from* homogeneous coordinates

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

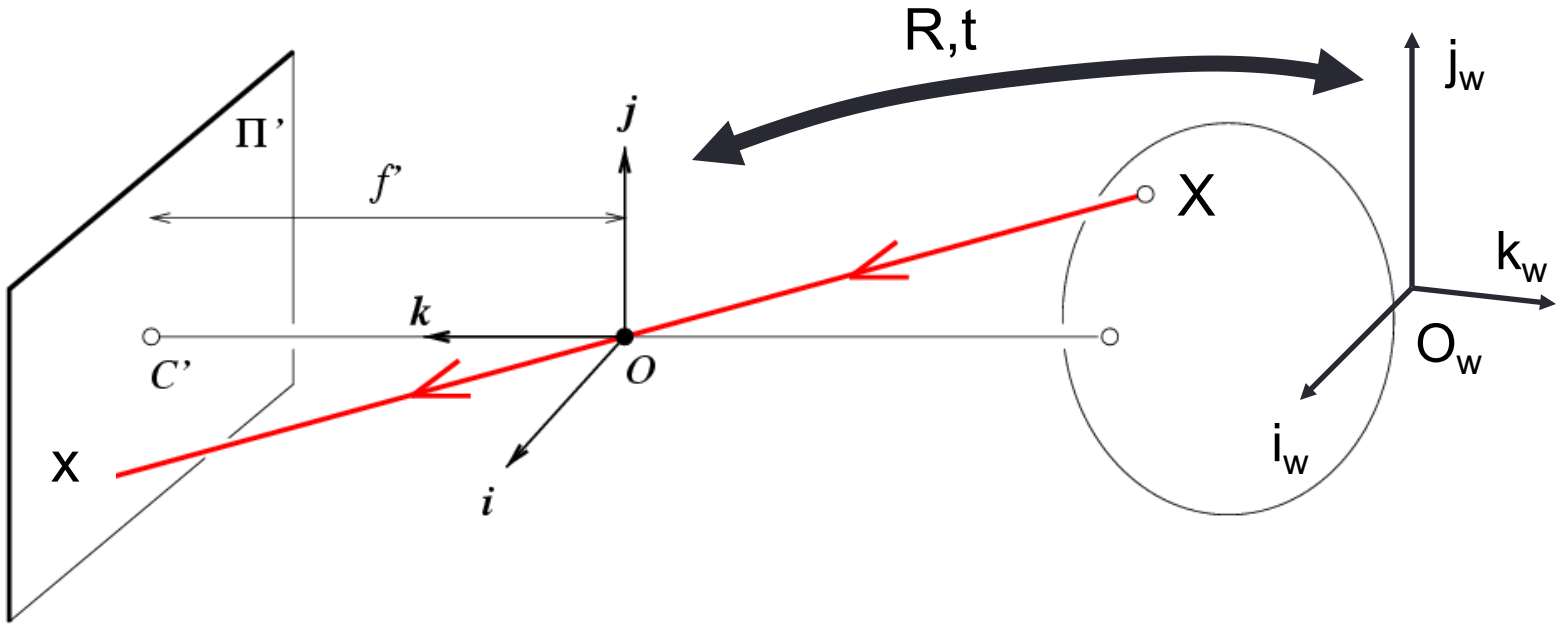
$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Benefits of Homogeneous Coords

- Lines (3D) and points (2D → 3D) are now the same dimension.
- Use the *cross* (x) and *dot product* for:
  - Intersection of lines **l** and **m**: **l** x **m**
  - Line through two points **p** and **q**: **p** x **q**
  - Point **p** on line **l**: $\mathbf{l}^T\mathbf{p}$
- Parallel lines, vertical lines become easy (compared to y=mx+b)

Slide Credit: Savarese

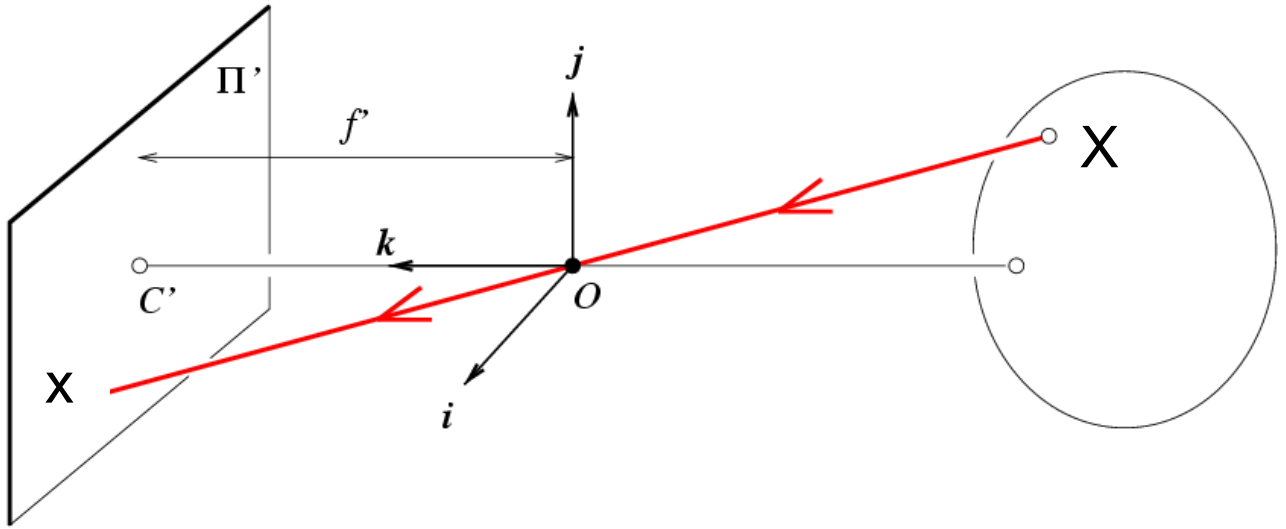
Projection matrix



$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

Projection matrix



Intrinsic Assumptions

- Unit aspect ratio
- Optical center at (0,0)
- No skew

Extrinsic Assumptions

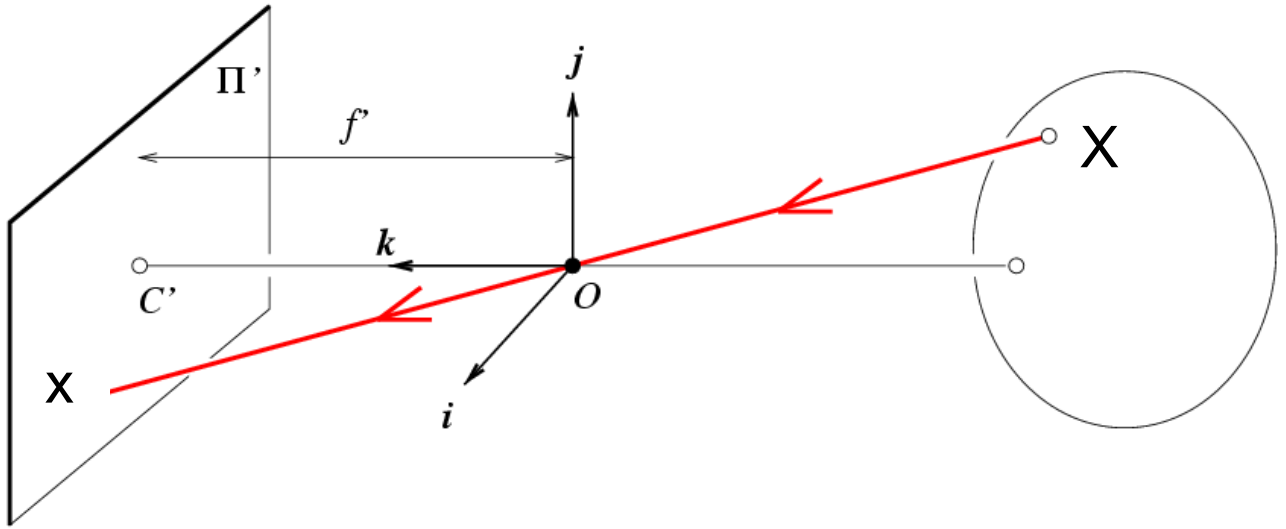
- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\ v\\ 1\end{bmatrix} = \begin{bmatrix}f & 0 & 0 & 0\\ 0 & f & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

K

Slide Credit: Savarese

Projection matrix



Intrinsic Assumptions

- Unit aspect ratio
- Optical center at (0,0)
- No skew

Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f & 0 & 0 & 0\\0 & f & 0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Slide Credit: Savarese

Remove assumption: known optical center

Intrinsic Assumptions
- Unit aspect ratio
- No skew

Extrinsic Assumptions
- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f & 0 & u_0 & 0\\0 & f & v_0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Remove assumption: square pixels

Intrinsic Assumptions
- No skew

Extrinsic Assumptions
- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\alpha & 0 & u_0 & 0\\0 & \beta & v_0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Remove assumption: non-skewed pixels

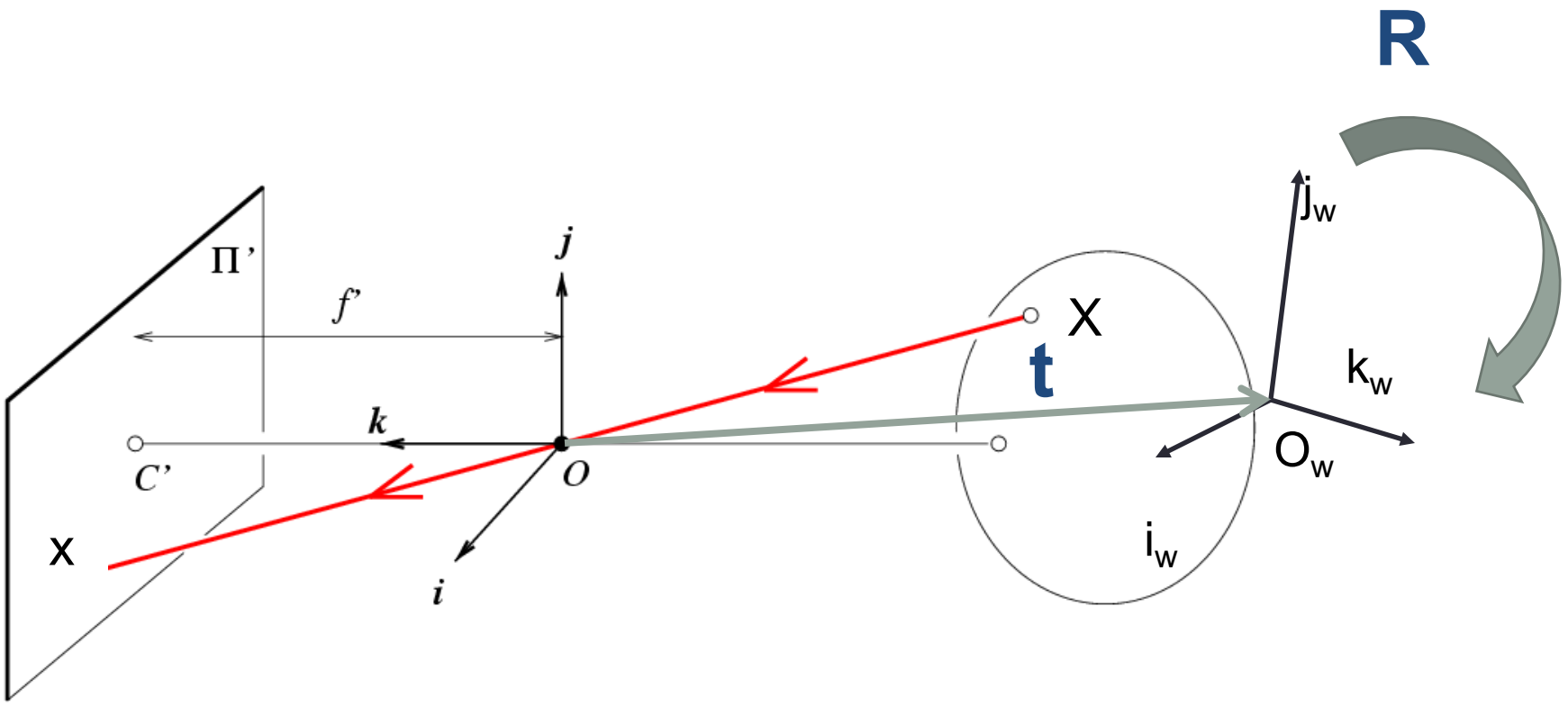
Intrinsic Assumptions

Extrinsic Assumptions
- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\alpha & s & u_0 & 0\\0 & \beta & v_0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Note: different books use different notation for parameters

Oriented and Translated Camera



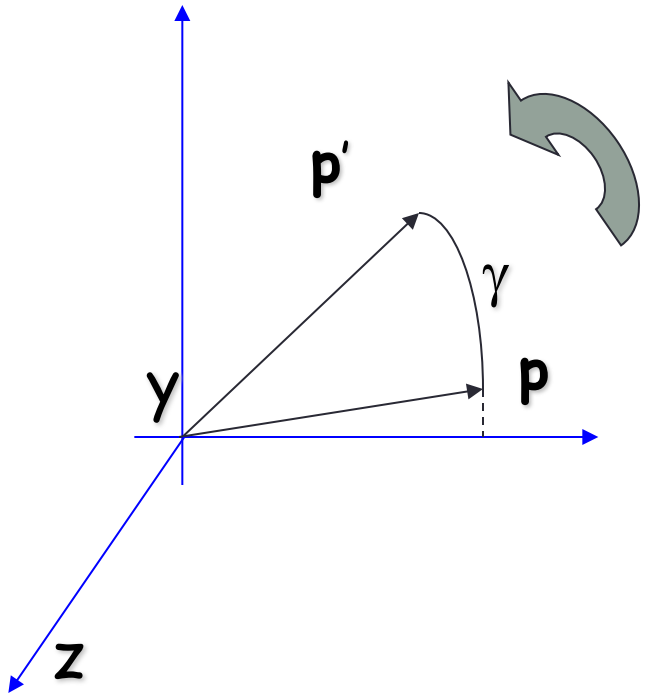
Allow camera translation

Intrinsic Assumptions Extrinsic Assumptions
- No rotation

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{t}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\ v\\ 1\end{bmatrix} = \begin{bmatrix}\alpha & 0 & u_0\\ 0 & \beta & v_0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & t_x\\ 0 & 1 & 0 & t_y\\ 0 & 0 & 1 & t_z\end{bmatrix}\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

3D Rotation of Points

Rotation around the coordinate axes, counter-clockwise:



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Allow camera rotation

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

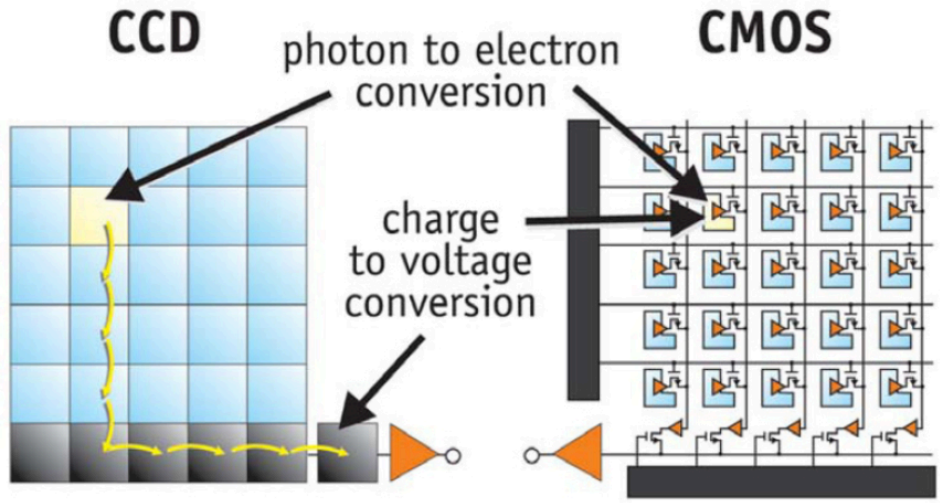
Degrees of freedom

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

5 6

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

From Photon to Photo



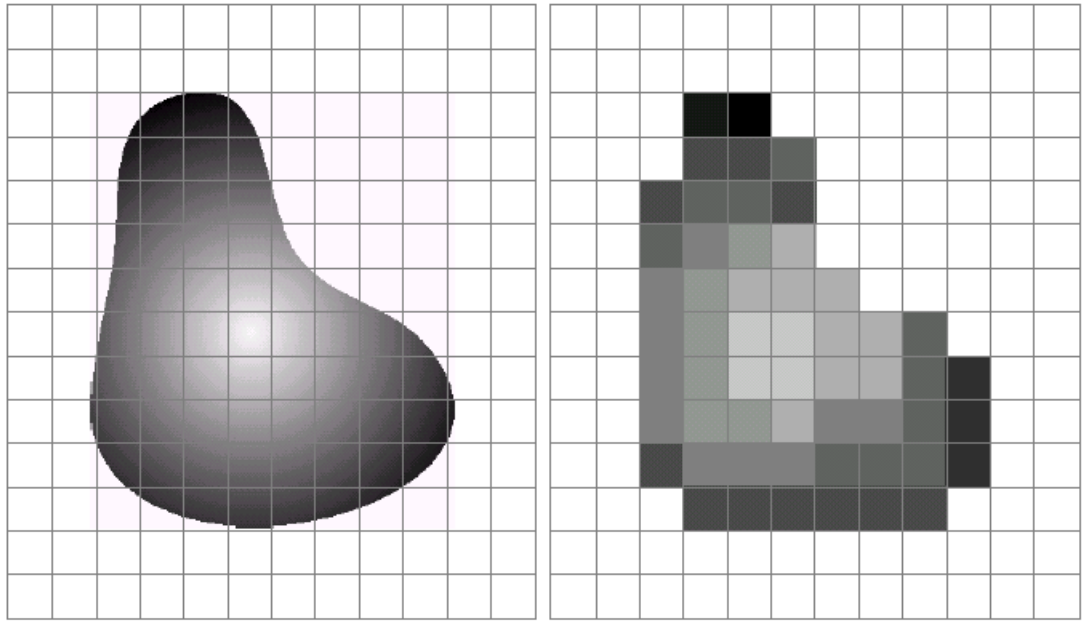
CCDs move photogenerated charge from pixel to pixel and convert it to voltage at an output node. CMOS imagers convert charge to voltage inside each pixel.

A digital camera replaces film with a sensor array

- Each cell in the array is light-sensitive diode that converts photons to electrons
- Two common types
  - **Charge Coupled Device** (CCD)
  - **Complementary metal oxide semiconductor** (CMOS)
- http://electronics.howstuffworks.com/digital-camera.htm

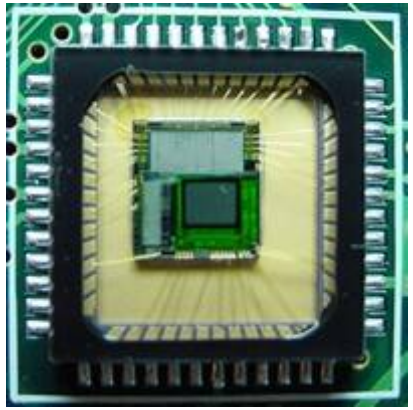
Slide by Steve Seitz

Digital Images



a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

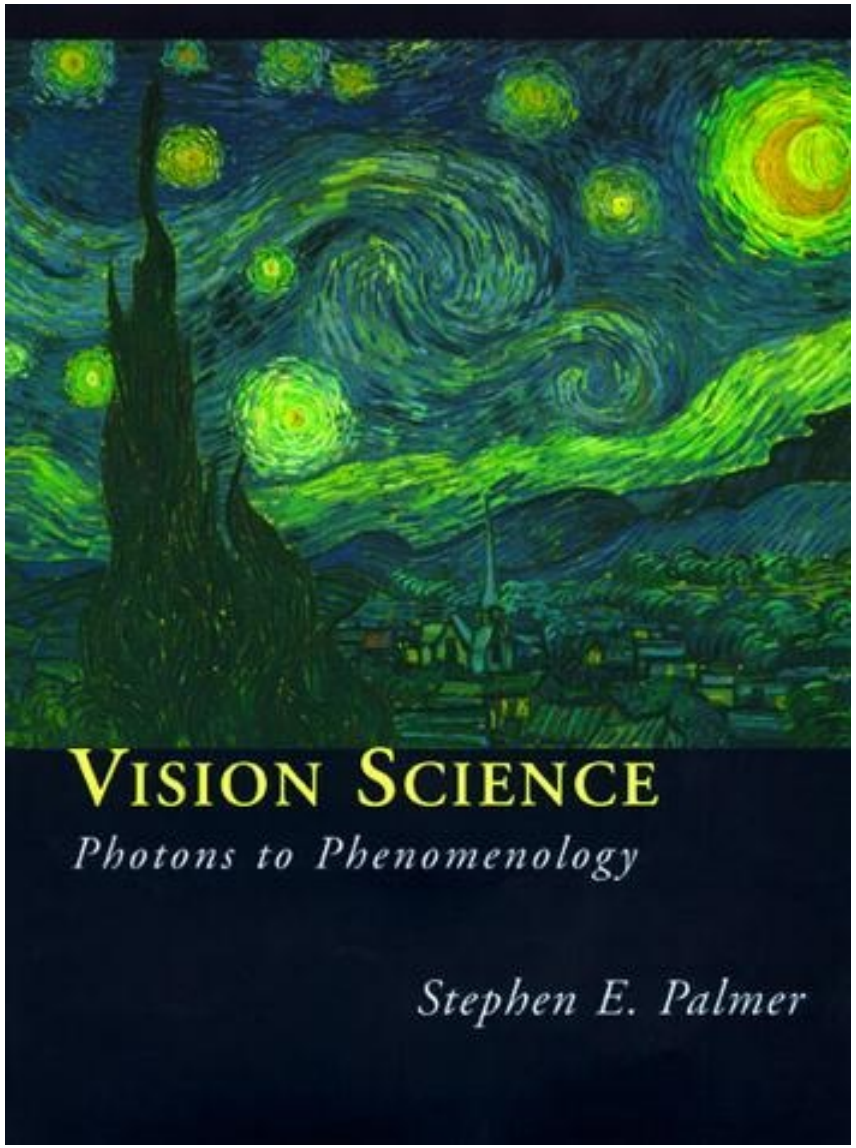


Slide credit: Derek Hoiem

What is color?

- The result of interaction between physical light in the environment and our visual system.

- A *psychological property* of our visual experiences when we look at objects and lights, *not a physical property* of those objects or lights.



Slide credit: Lana Lazebnik

Why does a visual system need color?

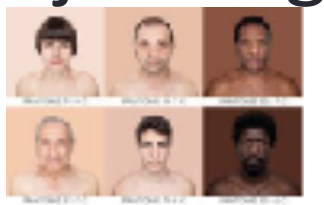
Slide credit: B. Freeman

http://www.hobbylinc.com/gr/pll/pll5019.jpg

An incomplete list

- To tell what food is edible.
- To distinguish material changes from shading changes.
- To group parts of one object together in a scene.

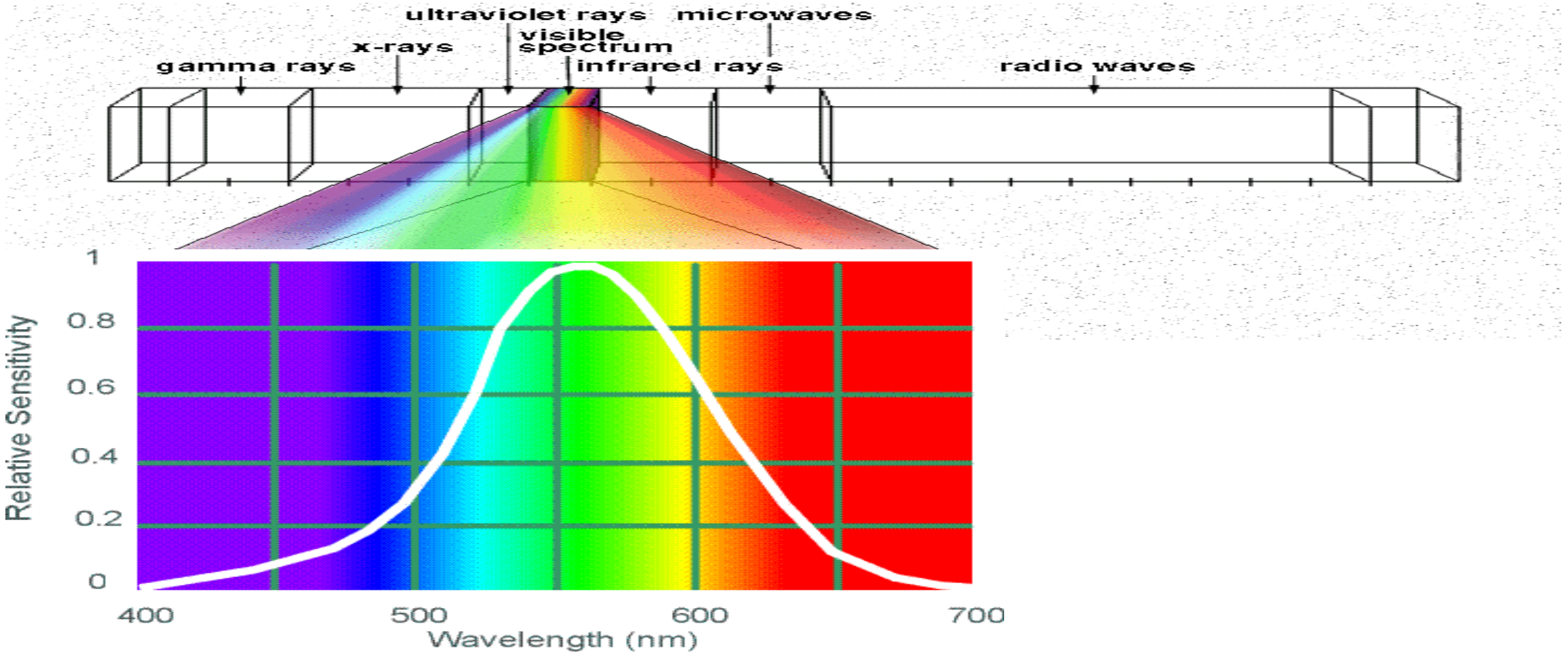
- To find people's skin.



- Check whether a person's appearance looks normal/healthy.

Slide credit: B. Freeman

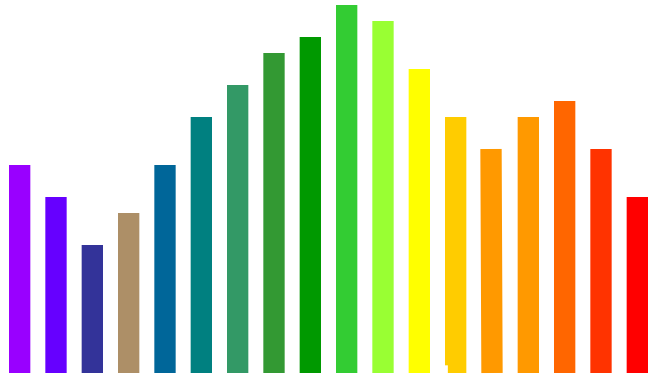
Electromagnetic Spectrum



Human Luminance Sensitivity Function

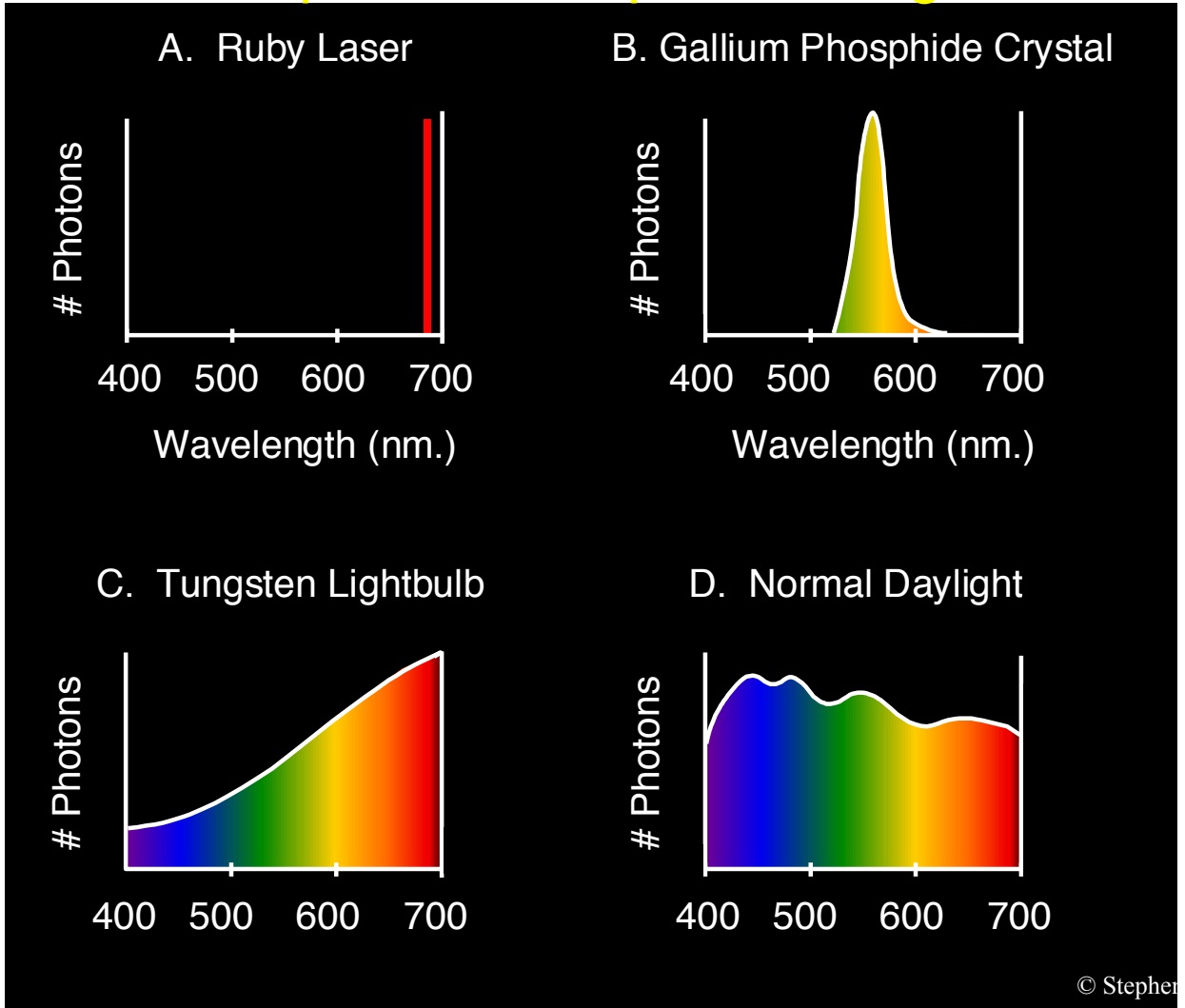
The Physics of Light

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

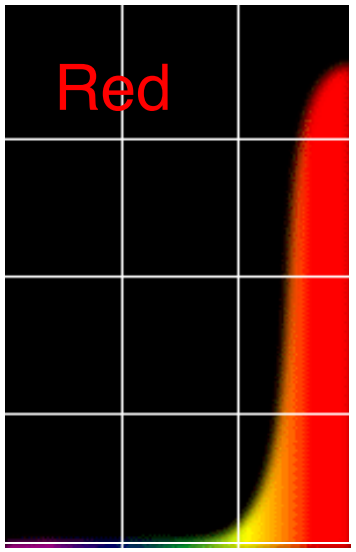


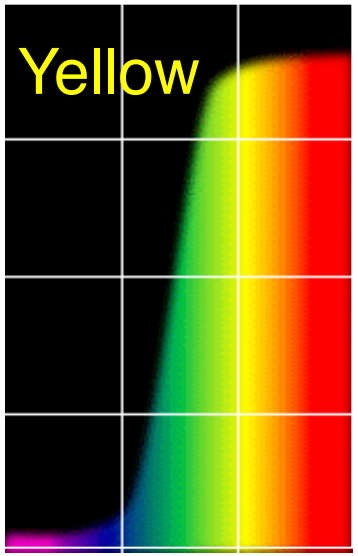
© Stephen

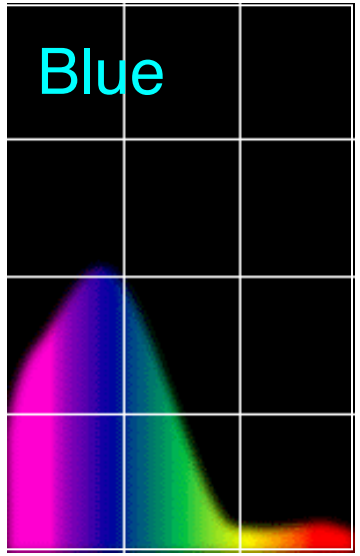
The Physics of Light

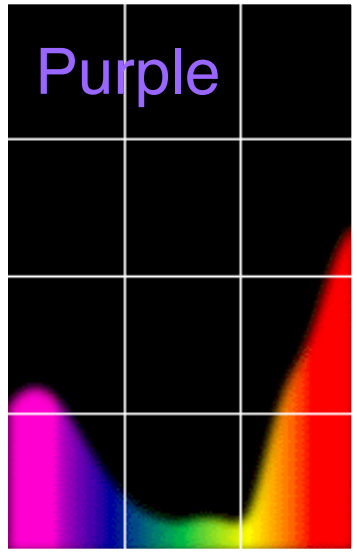
Some examples of the reflectance spectra of surfaces



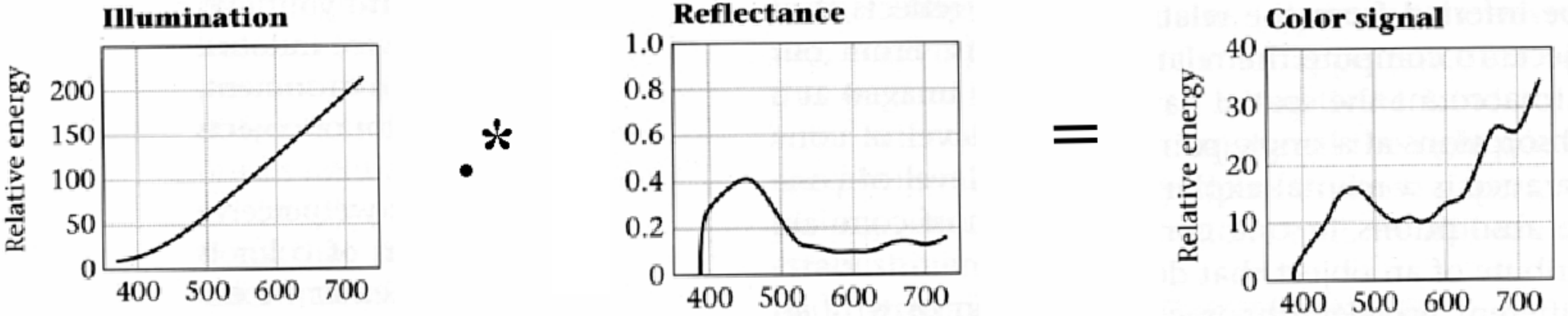




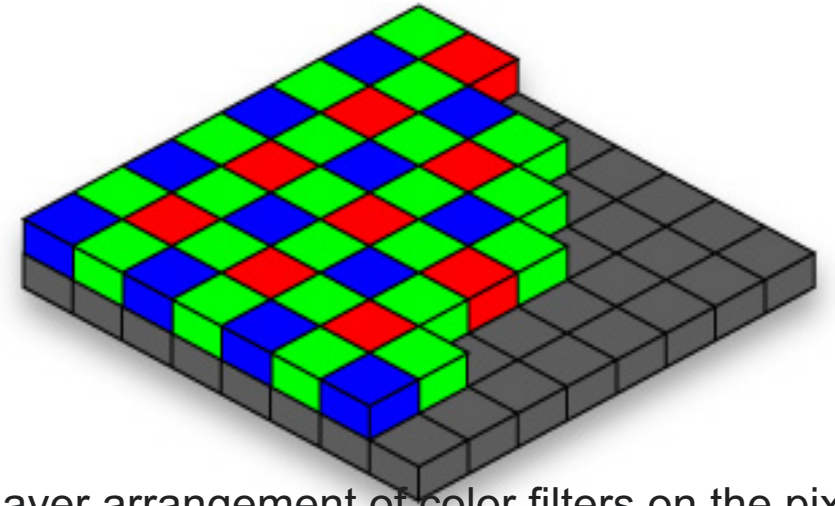


Interaction of light and surfaces

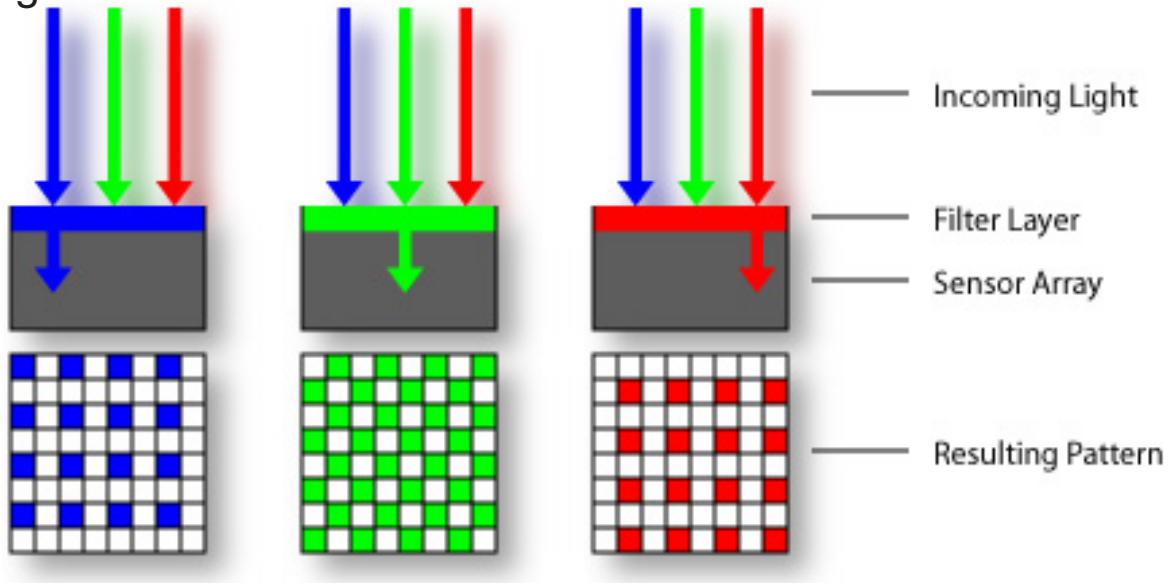
- Reflected color is the result of interaction of light source spectrum with surface reflectance
- Spectral radiometry
  - All definitions and units are now "per unit wavelength"
  - All terms are now "spectral"



Practical Color Sensing: Bayer Grid



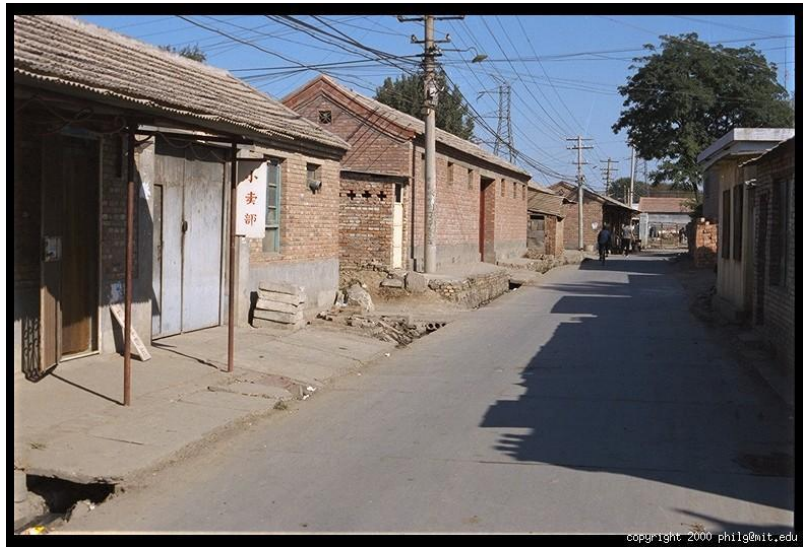
The Bayer arrangement of color filters on the pixel array of an image sensor



- Estimate RGB at ‘G’ cells from neighboring values

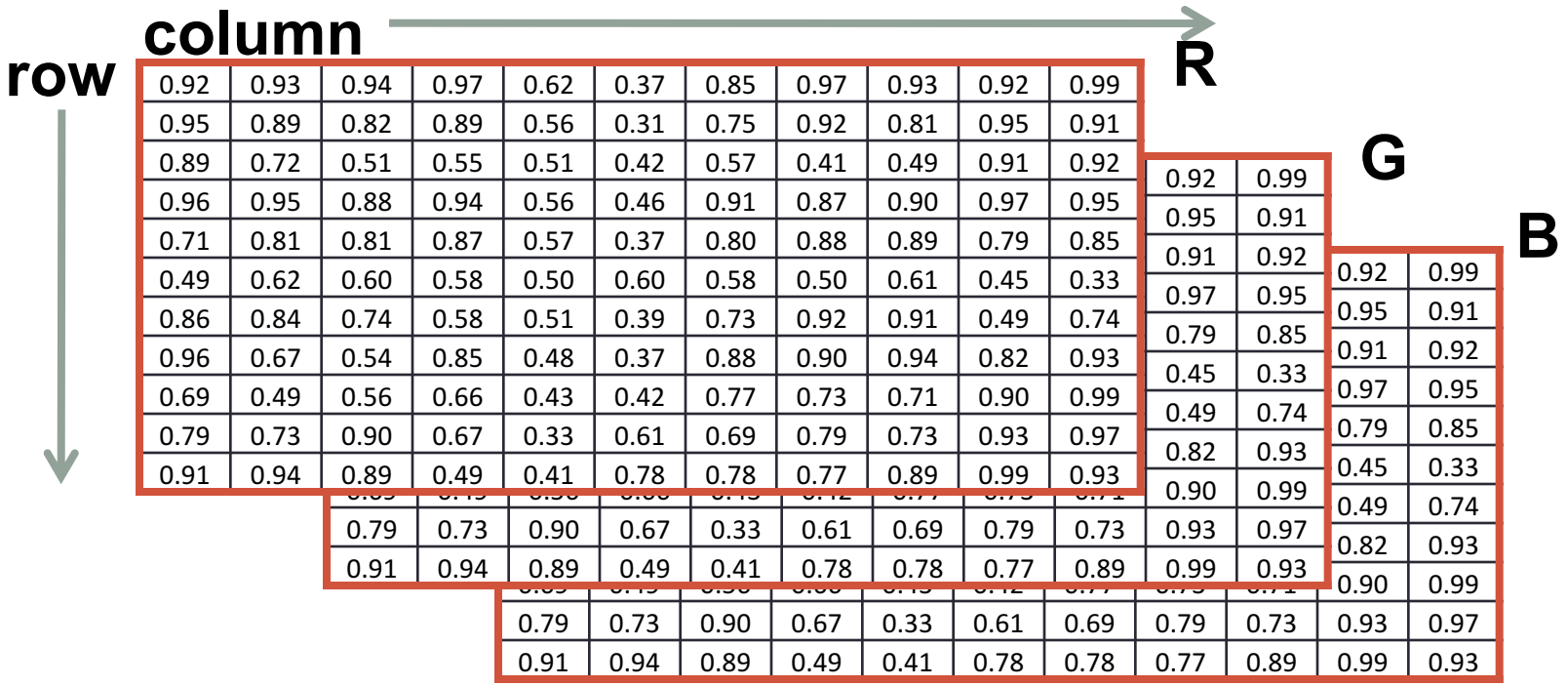
Slide by Steve Seitz

Color Image



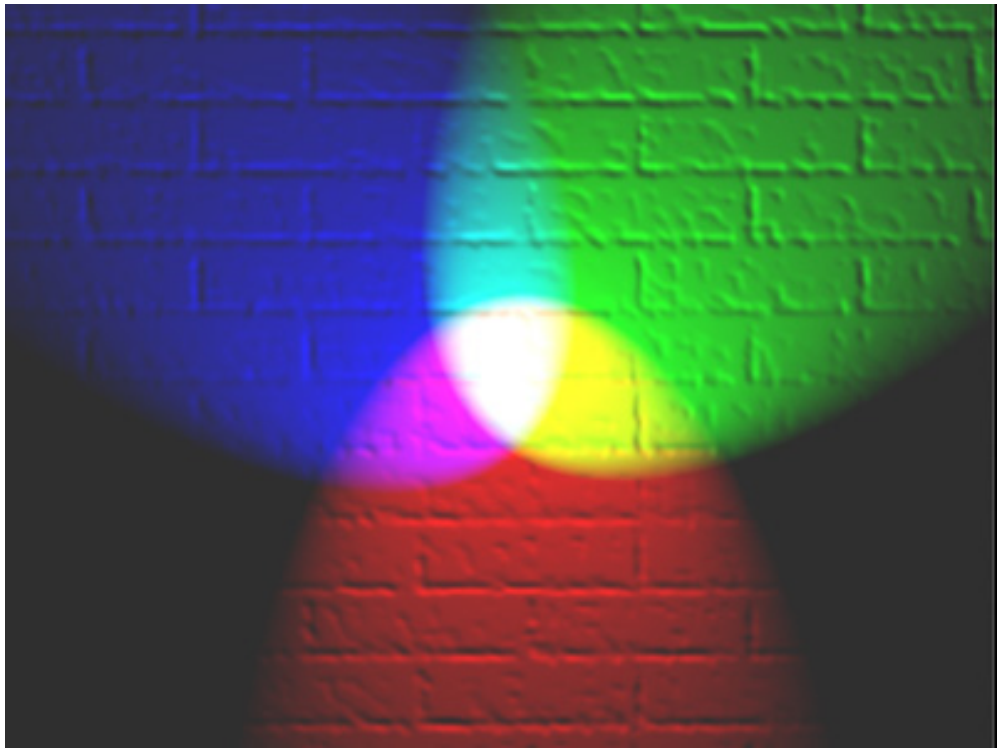
Images in Matlab

- Images represented as a matrix
- Suppose we have a NxM RGB image called “im”
  - im(1,1,1) = top-left pixel value in R-channel
  - im(y, x, b) = y pixels down, x pixels to right in the bth channel
  - im(N, M, 3) = bottom-right pixel in B-channel



Color spaces

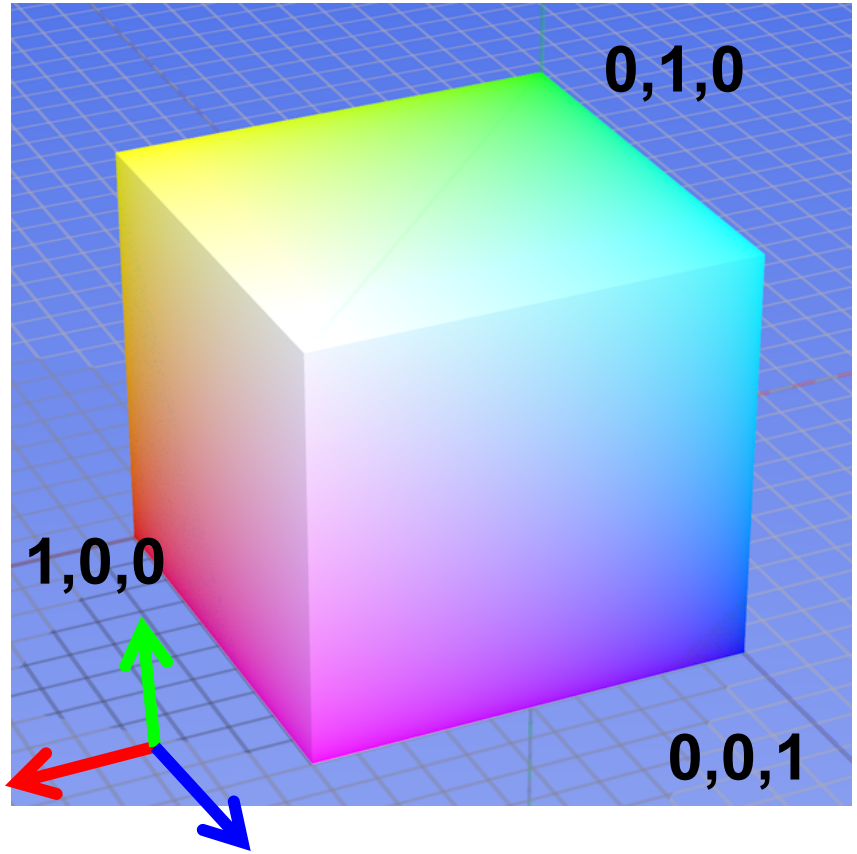
- How can we represent color?



http://en.wikipedia.org/wiki/File:RGB_illumination.jpg

Color spaces: RGB

Default color space



R
(G=0,B=0)

G
(R=0,B=0)

B
(R=0,G=0)

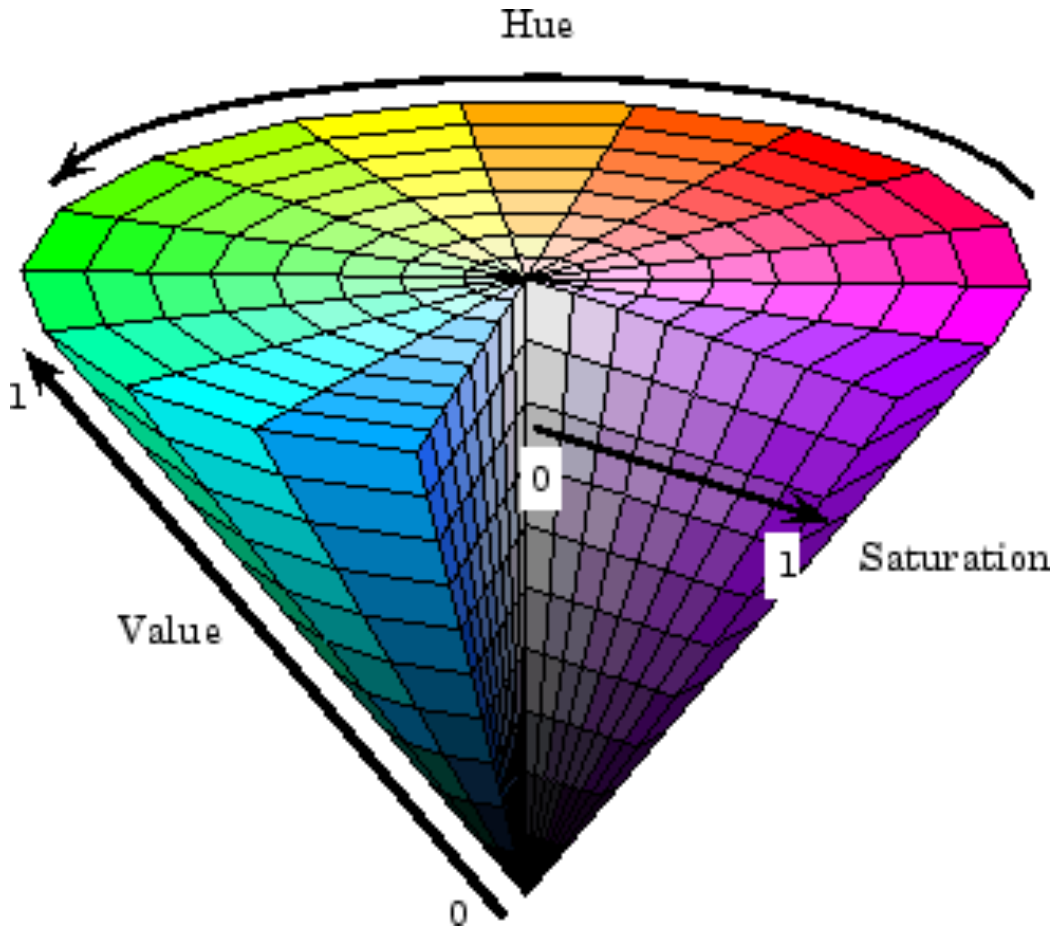
Some drawbacks

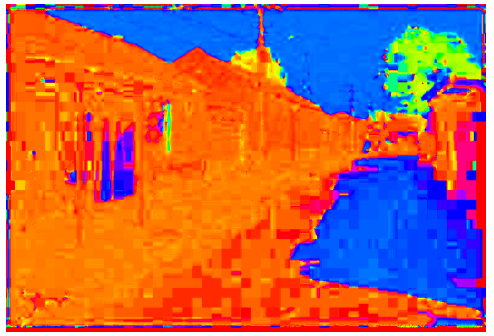
- Strongly correlated channels
- Non-perceptual

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

Color spaces: HSV

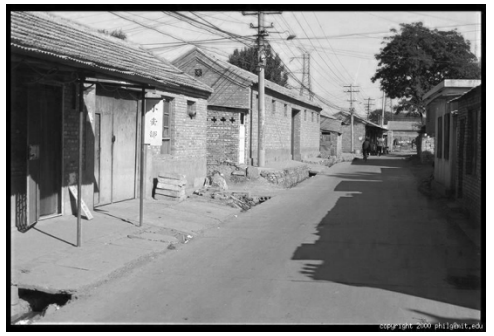
Intuitive color space





H
(S=1,V=1)

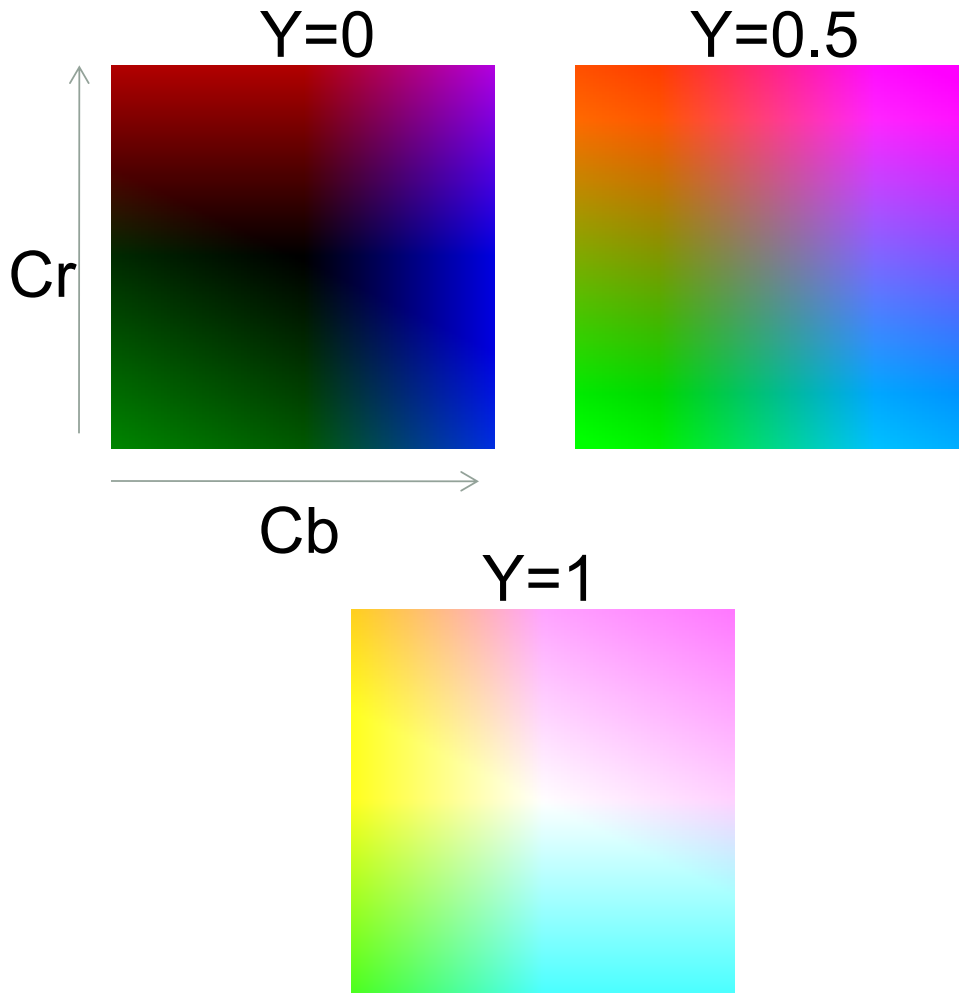
S
(H=1,V=1)



V
(H=1,S=0)

Color spaces: YCbCr

Fast to compute, good for compression, used by TV



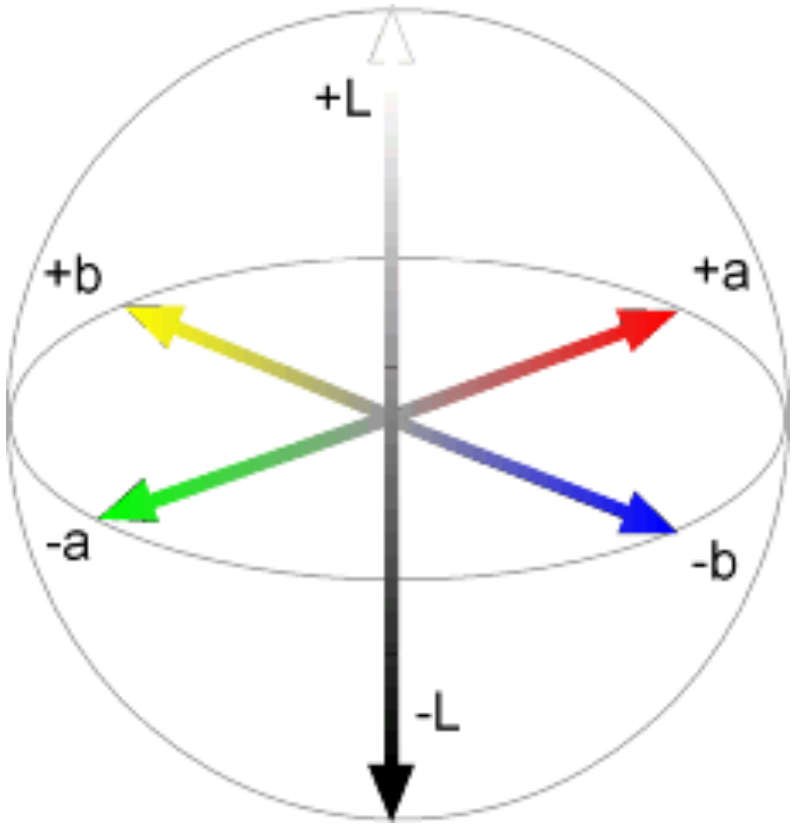
Y
(Cb=0.5,Cr=0.5)

Cb
(Y=0.5,Cr=0.5)

Cr
(Y=0.5,Cb=05)

Color spaces: L*a*b*

"Perceptually uniform"* color space



L
(a=0,b=0)

a
(L=65,b=0)

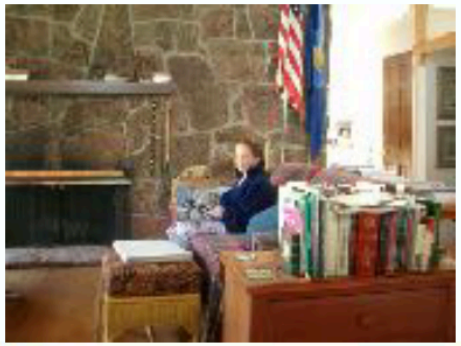
b
(L=65,a=0)

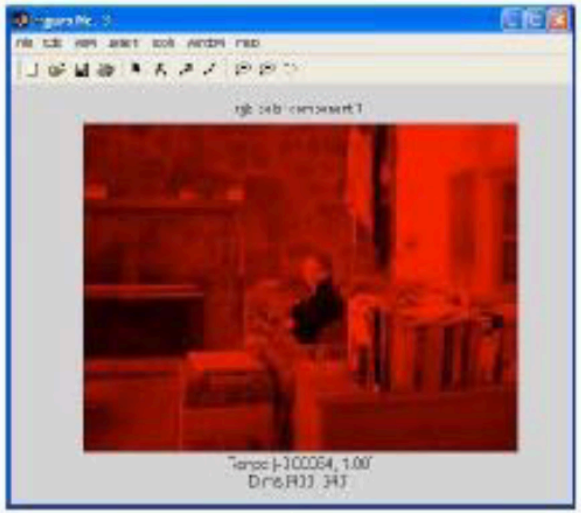
luminance, chrominance color components: Y, I, Q

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Slide credit: B. Freeman

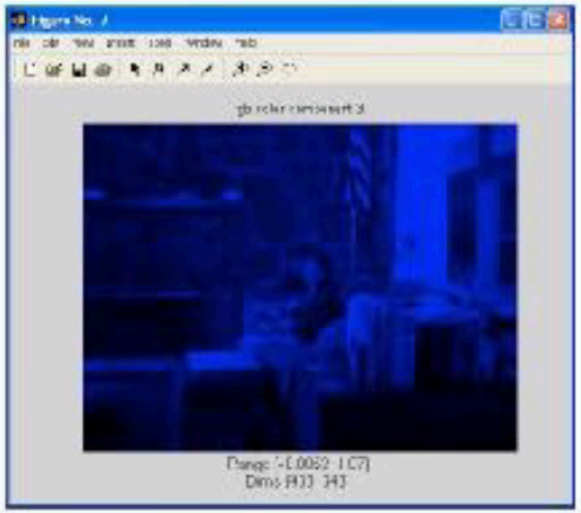
YIQ - RGB



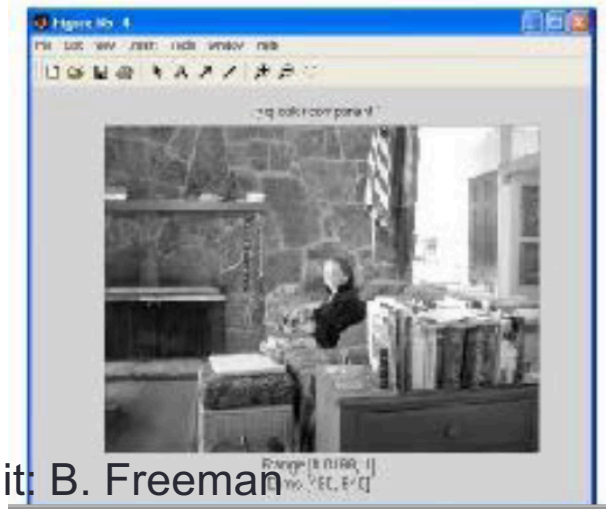


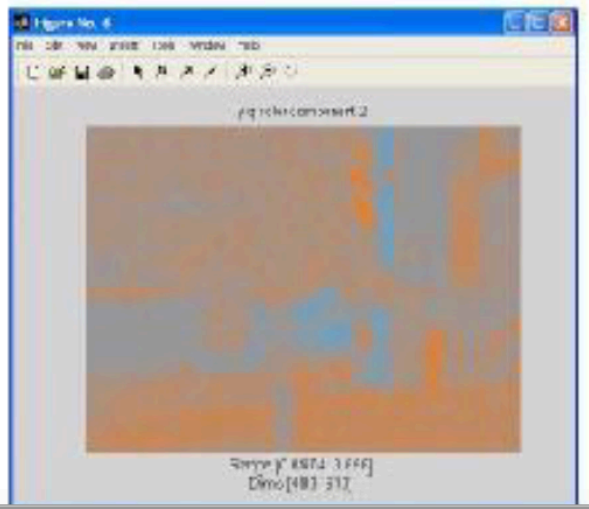
R

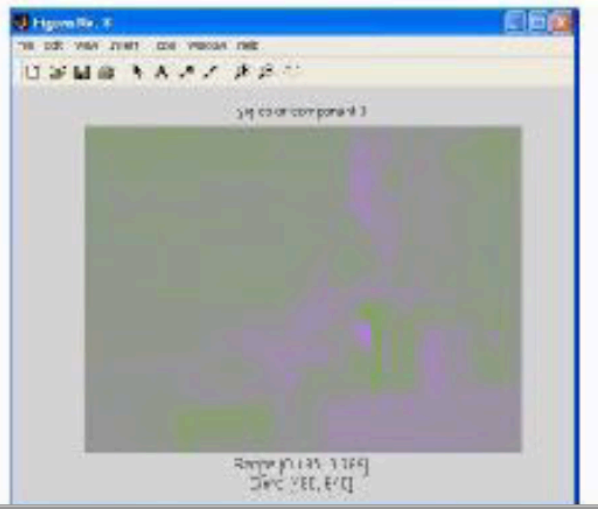
G



B







Slide credit: B. Freeman

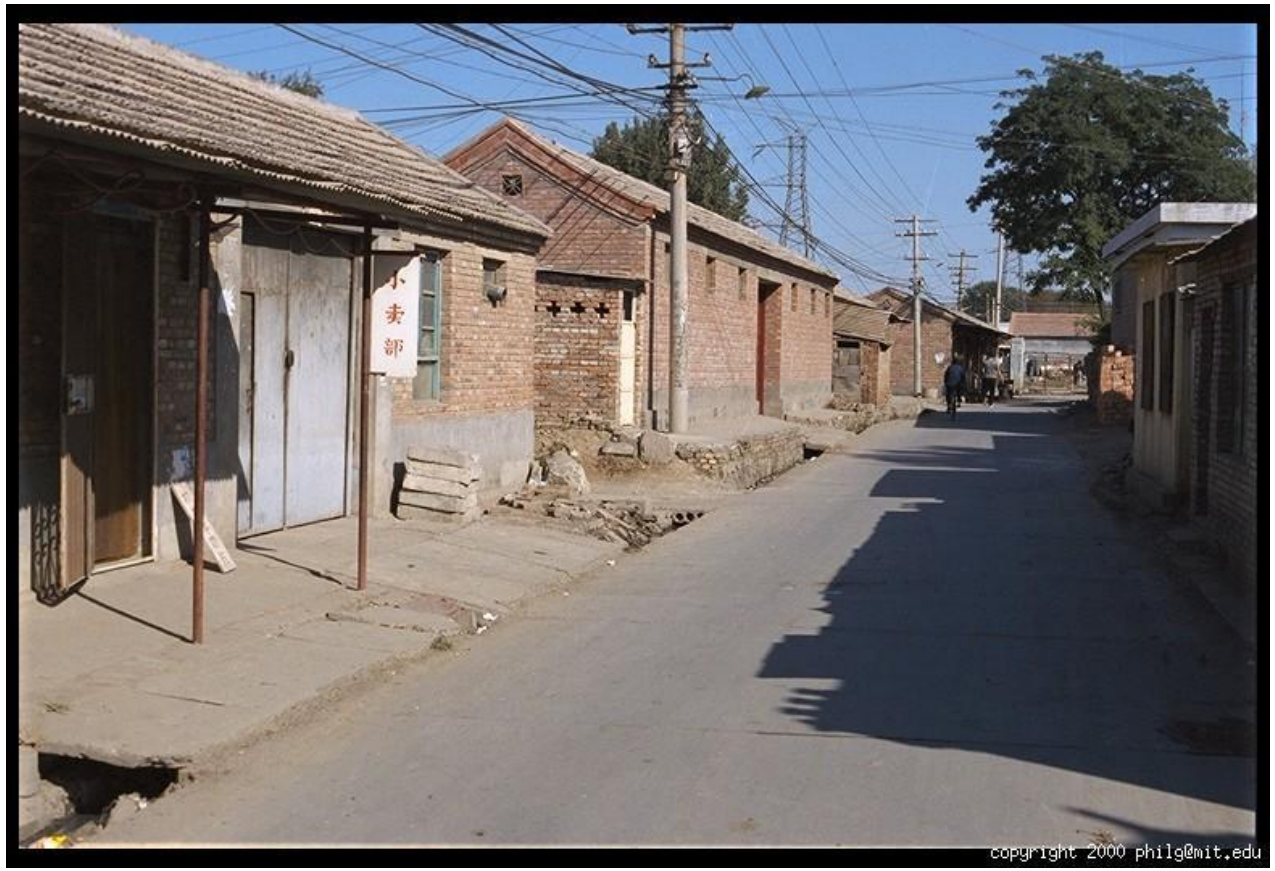
Most information in intensity

Only color shown – constant intensity

Most information in intensity

Only intensity shown – constant color

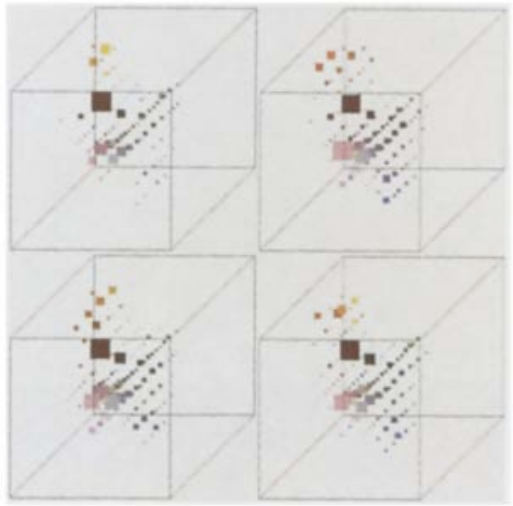
Most information in intensity

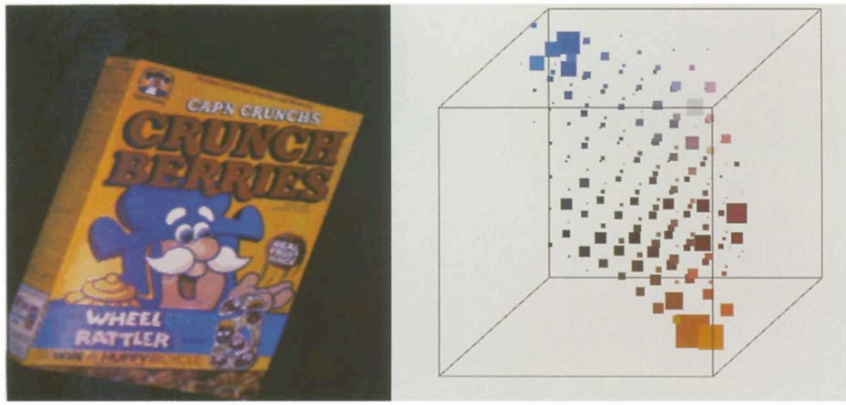


Original image

Uses of color in computer vision

Color histograms for indexing and retrieval





Swain and Ballard, Color Indexing, IJCV 1991.

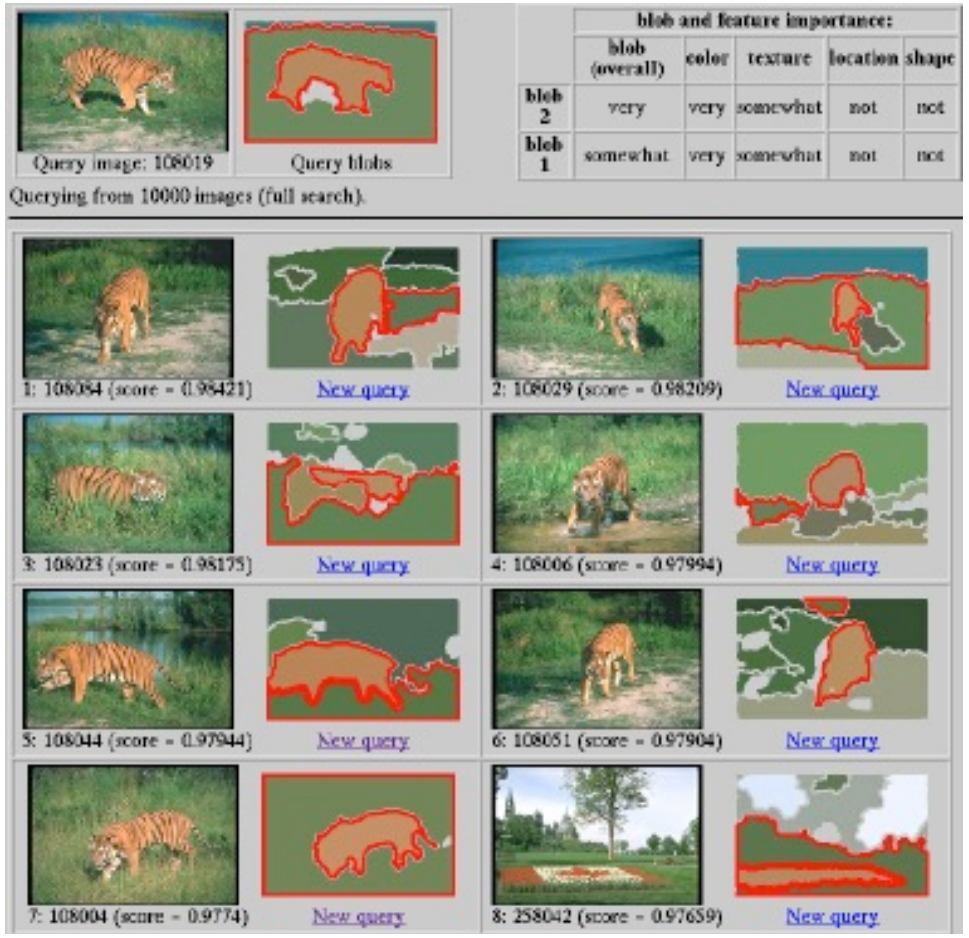
Uses of color in computer vision

Skin detection

M. Jones and J. Rehg, Statistical Color Models with Application to Skin Detection, IJCV 2002.

Uses of color in computer vision

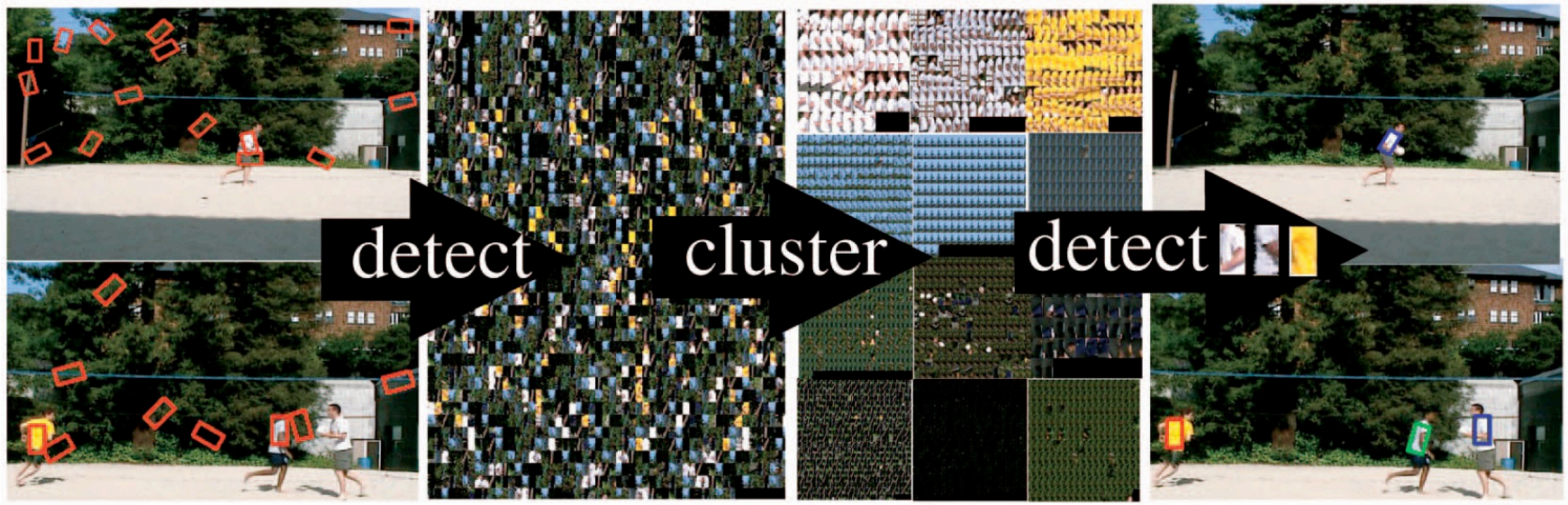
Image segmentation and retrieval

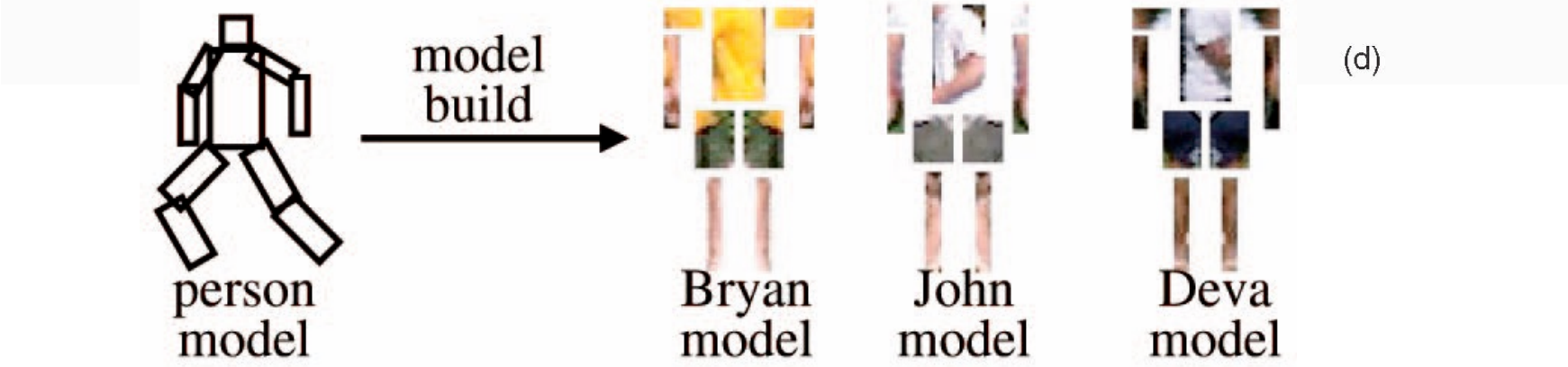


C. Carson, S. Belongie, H. Greenspan, and Ji. Malik, Blobworld: Image segmentation using Expectation-Maximization and its application to image querying, ICVIS 1999.

Uses of color in computer vision

Building appearance models for tracking



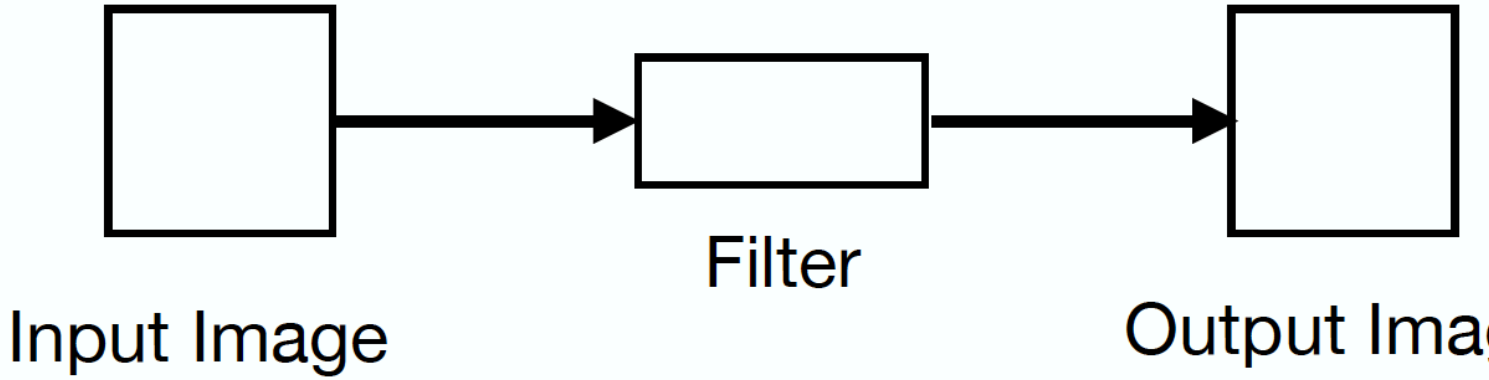


D. Ramanan, D. Forsyth, and A. Zisserman. Tracking People by Learning their Appearance. PAMI 2007.

Image Filtering

BBM416

Image Filtering



We want to remove unwanted sources of variation, and keep the information relevant for whatever task we need to solve

Source: Bill Freeman

Image filtering

- Image filtering: for each pixel, compute function of local neighborhood and output a new value
  - Function specified by a “filter” or mask saying how to combine values from neighbors.
  - Same function applied at each position
  - Output and input image are typically the same size

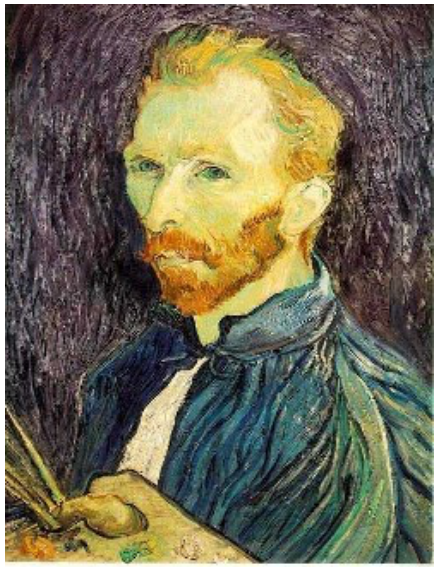
Image filtering

- Linear filtering: function is a weighted sum/difference of pixel values
- Uses of filtering:
  - Enhance images
    - Denoise, smooth, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
  - Convolutional Neural Networks

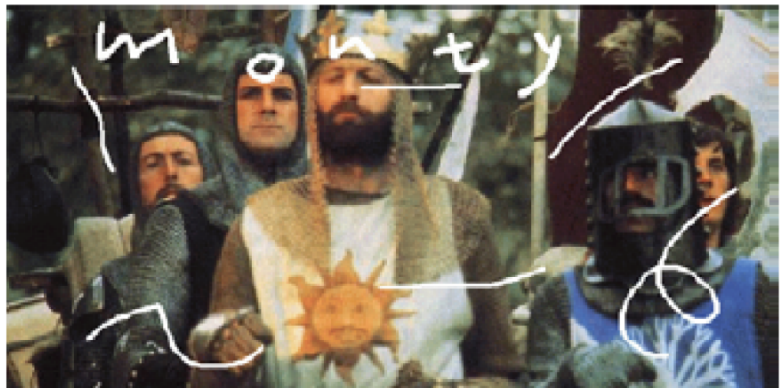
De-noising

Salt and pepper noise

Super-resolution



In-painting



Bertamio et al

Noise reduction

- Given a camera and a still scene, how can you reduce noise?

Take lots of images and average them!

What's the next best thing?

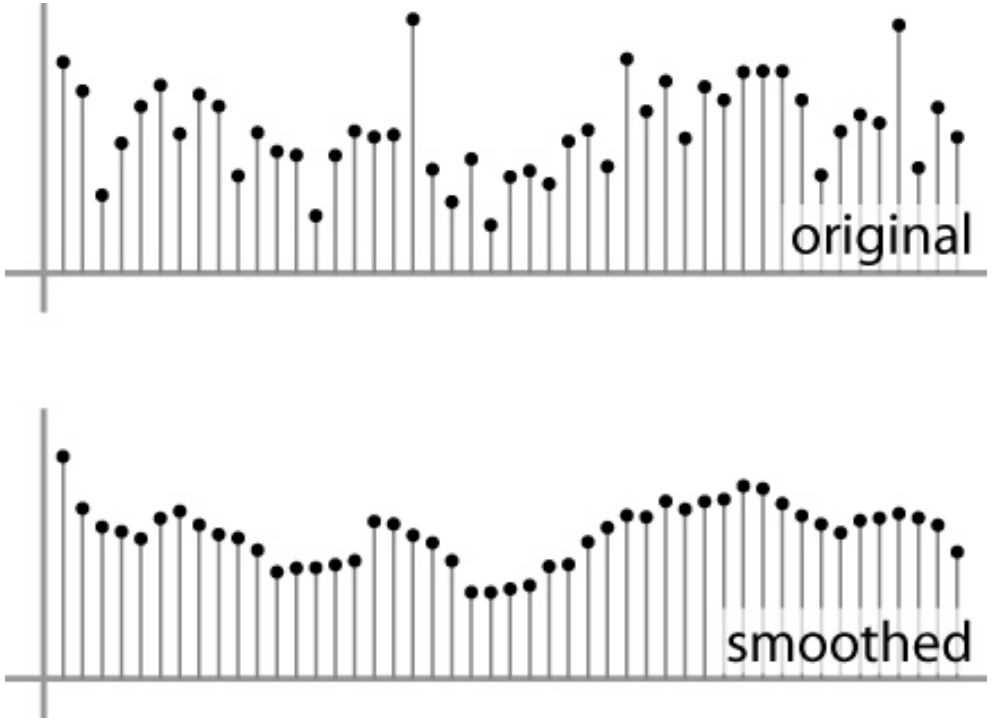
Source: S. Seitz

First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

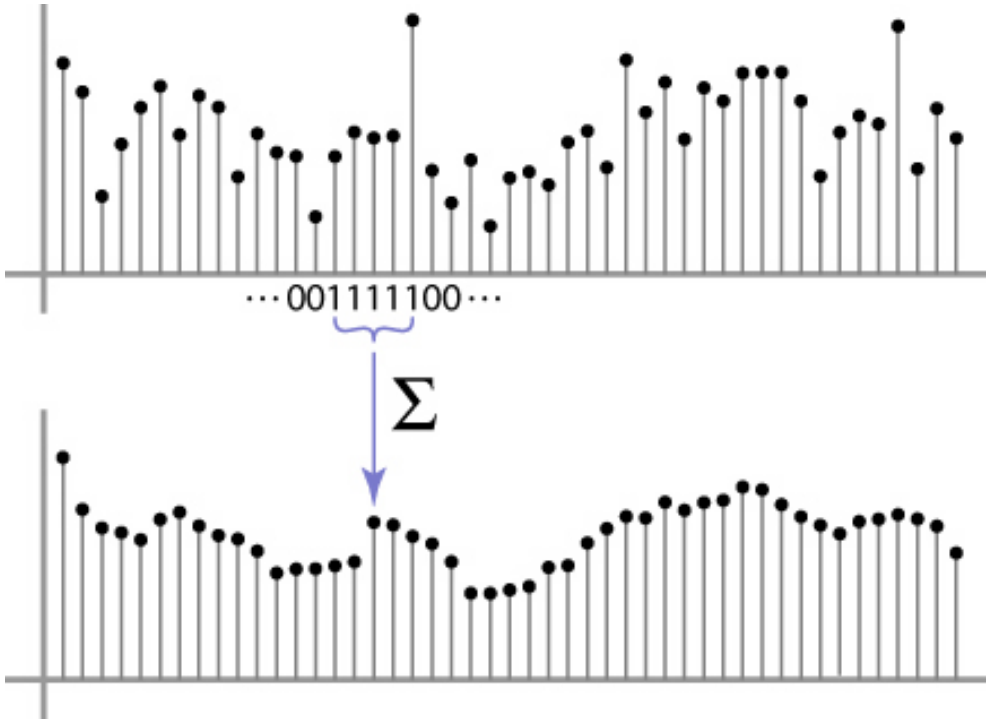
First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:



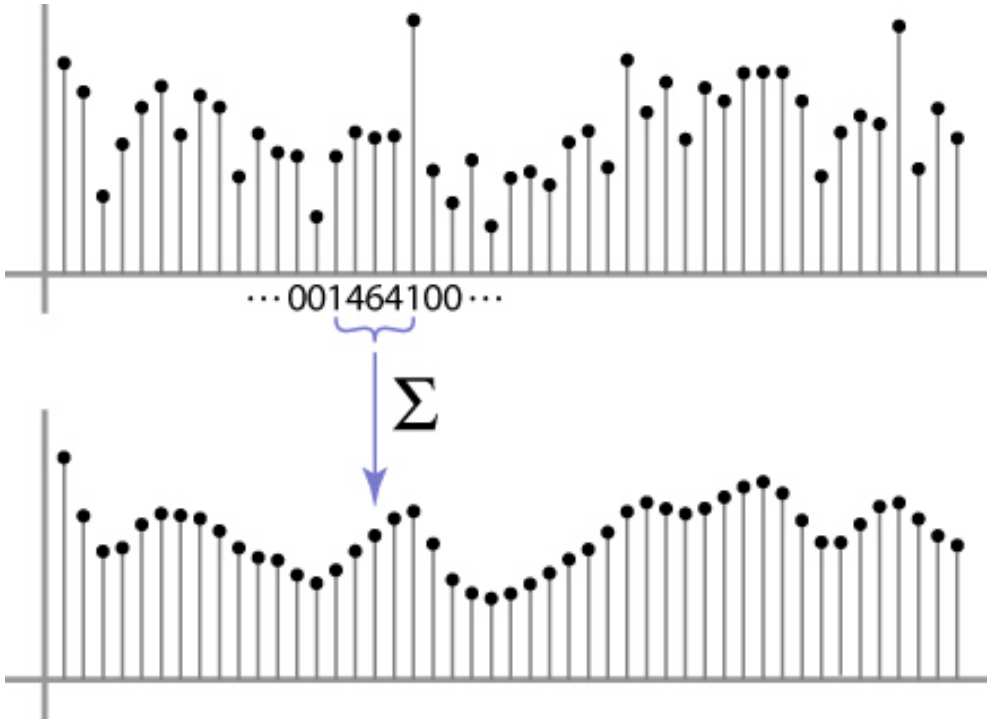
Weighted Moving Average

- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Weighted Moving Average

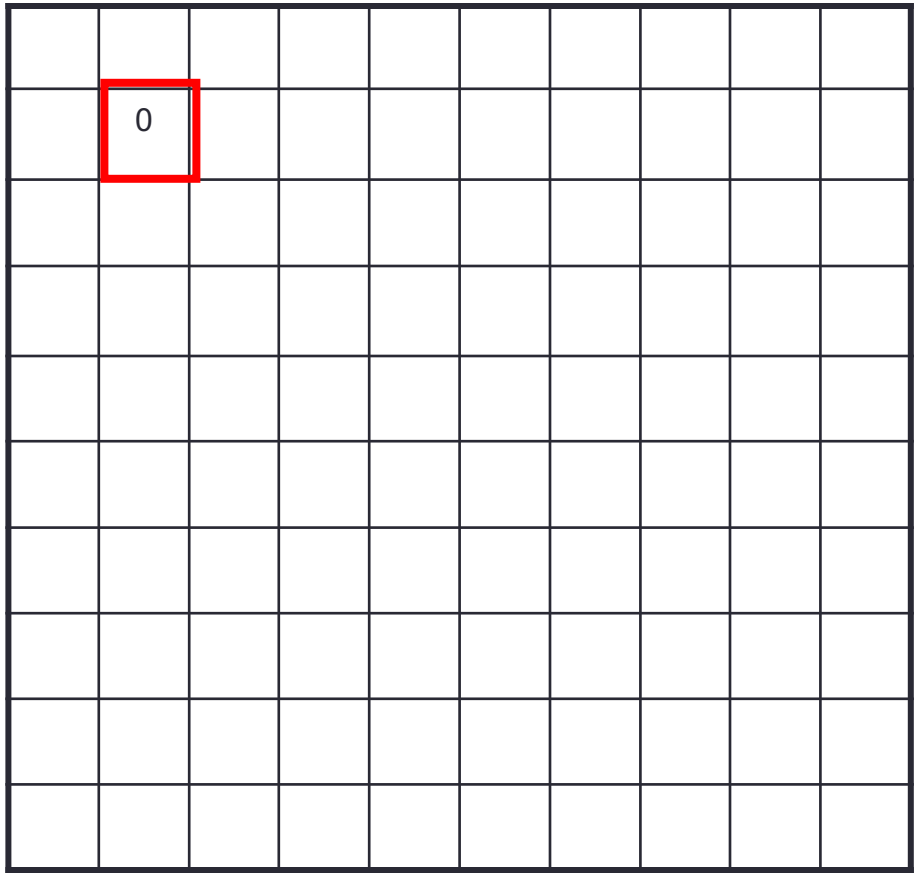
- Non-uniform weights [1, 4, 6, 4, 1] / 16



Moving Average In 2D

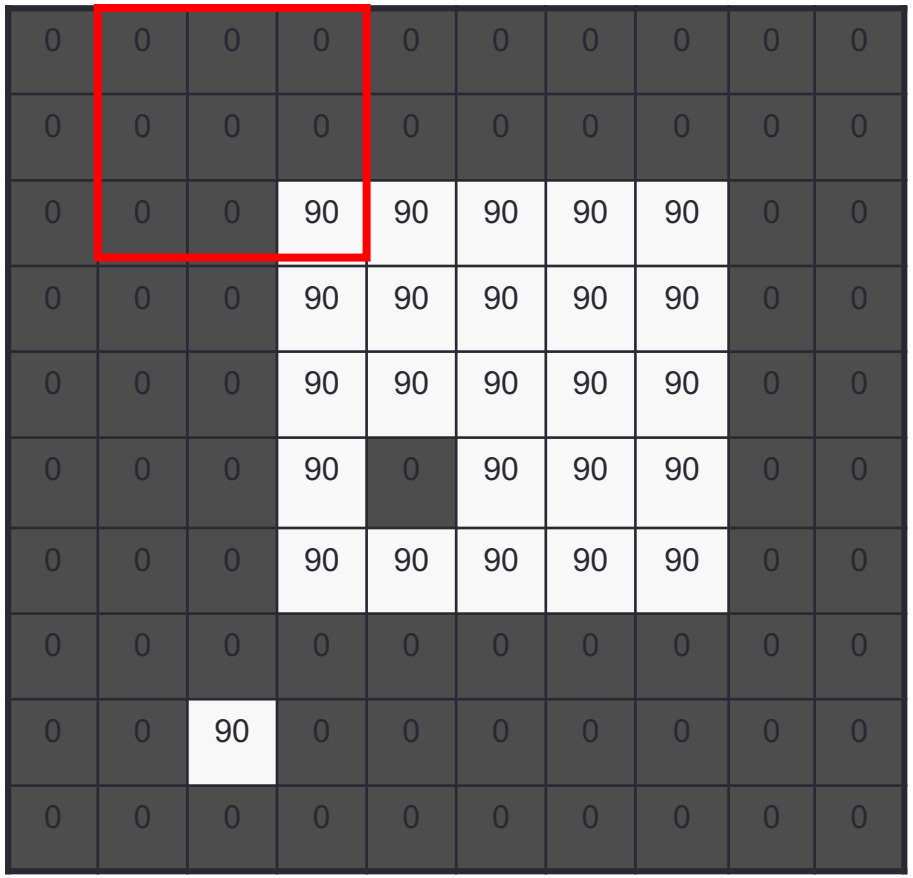
$F[x, y]$

$G[x, y]$

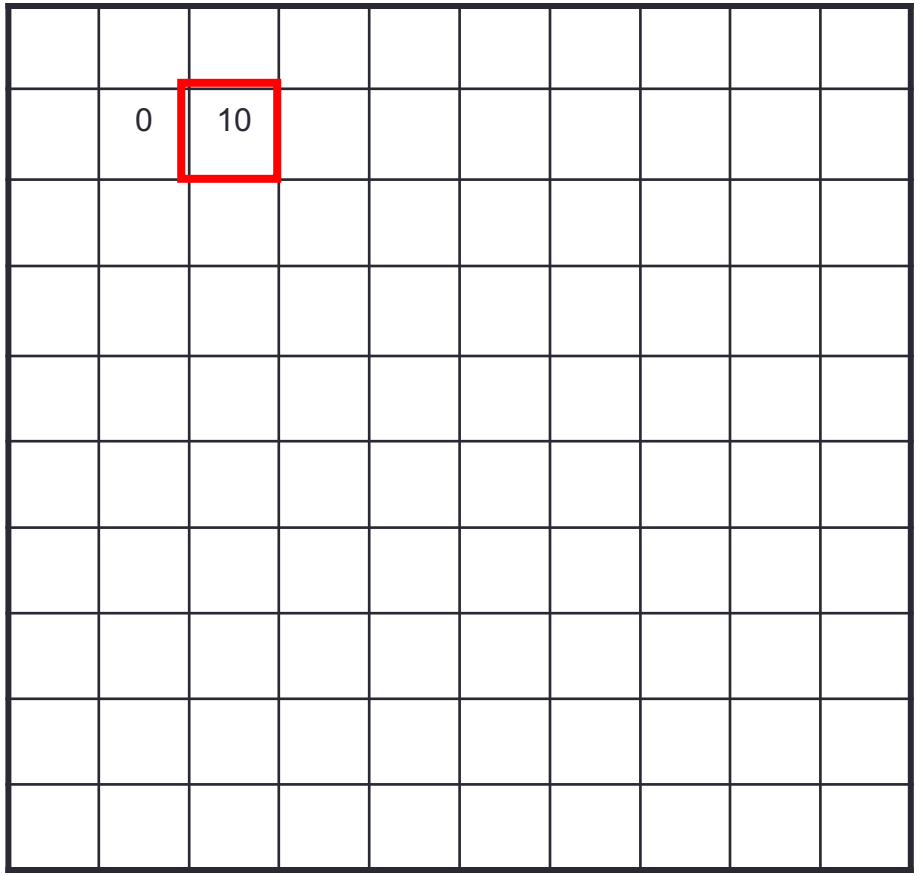


Moving Average In 2D

$F[x, y]$

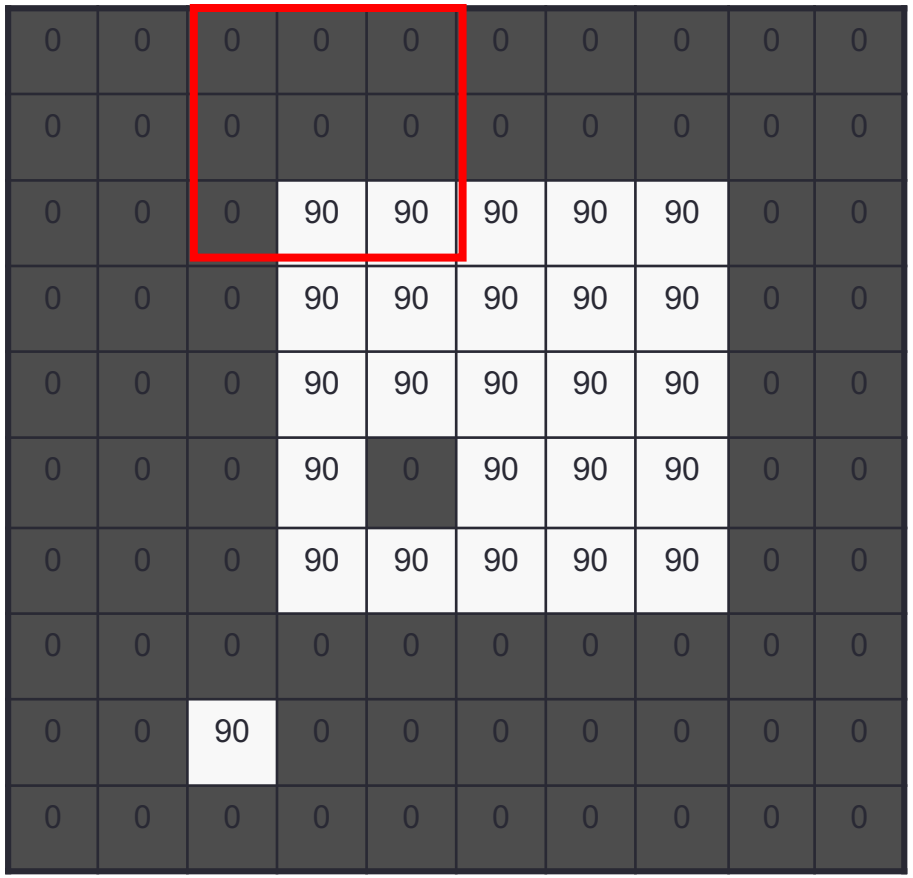


$G[x, y]$

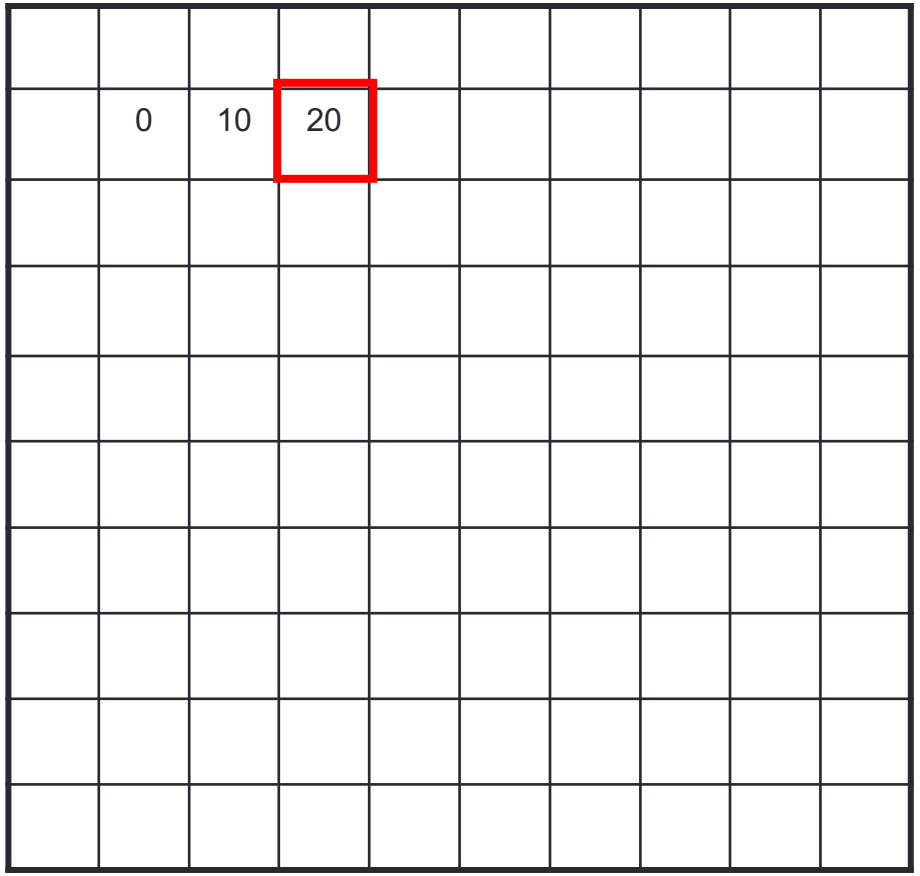


Moving Average In 2D

$F[x, y]$



$G[x, y]$



Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

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

Averaging filter

- What values belong in the kernel *H* for the moving average example?

$F[x, y]$

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

$\otimes$

$H[u, v]$

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

"box filter"

$G[x, y]$

	0	10	20	30	30				

$$G = H \otimes F$$

Linear. shift-invariant filters

- A fundamental kind of image processing operation
- Linear: preserves summation and scalar multiplication

$$f(I_1 + I_2) = f(I_1) + f(I_2)$$
$$f(aI) = af(I)$$

- Shift-invariant: commutes with shifting the image

$$f(s_{\Delta x,\Delta y}(I)) = s_{\Delta x,\Delta y}(f(I))$$

(i.e. filter does the same thing at different places in the image)

- Surprisingly, all such operations can be computed with one simple algorithm: **convolution**

Slide by S. Marschner

Convolution

- Same moving average operation, expressed mathematically:

$$c[i] = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} a[j].$$

Slide by S. Marschner

Convolution

- Simple averaging:

$$c[i] = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} a[j].$$

every sample gets the same weight

- Convolution: same idea but with *weighted* average

$$(a \star b)[i] = \sum_{j} a[j]b[i-j].$$

each sample gets its own weight (normally zero far away)

- Visually: reflect *b* and slide it over so that *b*[0] lines up with *a*[j]
- This is all convolution is: it is a **moving weighted average**

Slide by S. Marschner

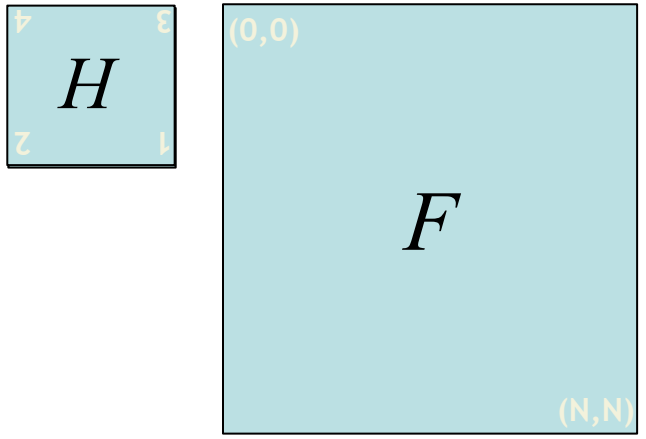
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)

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

$$G = H \star F$$

Notation for convolution operator



Slide credit: Kristen Grauman

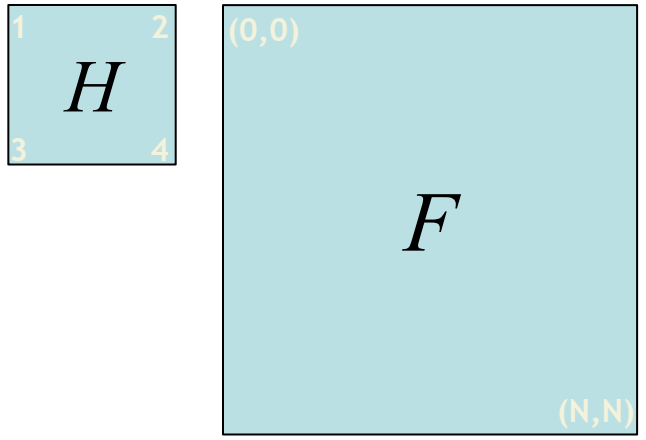
Correlation filtering

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

This is called cross-correlation, denoted $G = H \otimes F$

Filtering an image: replace each pixel with a linear combination of its neighbors.

Conceptually simpler, but not as nice as convolution



Convolution vs. correlation

Convolution

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

$$G = H \star F$$

Cross-correlation

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

$$G = H \otimes F$$

For a Gaussian or box filter, how will the outputs differ?

If the input is an impulse signal, how will the outputs differ?

Correlation vs. Convolution

- Correlation

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

Matlab:
`filter2`
`imfilter`

$$G = H \otimes F$$

- Convolution

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

Note the difference!

Matlab:
`conv2`

- Note $G = H \star F$
  - If $H[-u,-v] = H[u,v]$ (if the filter is symmetric) then correlation ≡ convolution.

Properties of Convolution

- Linear and shift-invariant
- Commutative: $a * b = b * a$

- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$

- Scalars factor out: $ka * b = a * kb = k (a * b)$

- Identity: unit impulse $e = [0, 0, 1, 0, 0]$,
  $a * e = a$

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)

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

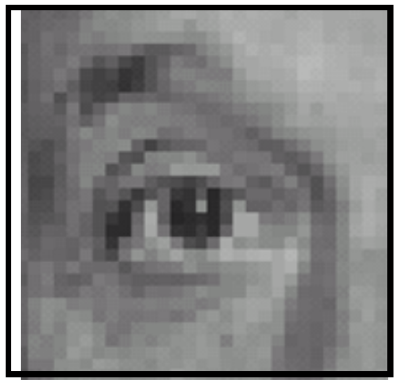
?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



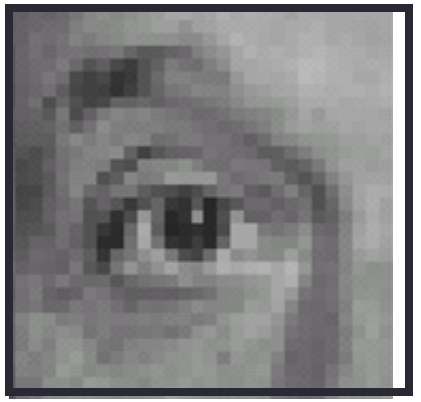
Shifted right
by 1 pixel with
convolution

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



Shifted left
by 1 pixel
with
correlation

Source: D. Lowe

Practice with linear filters

Original

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

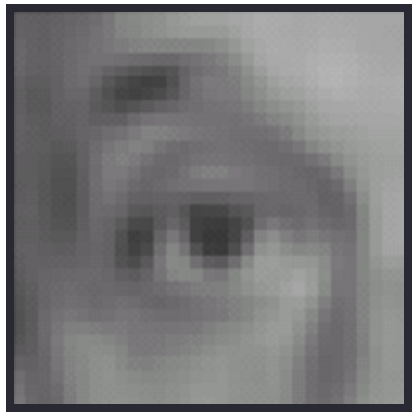
?

Source: D. Lowe

Practice with linear filters

Original

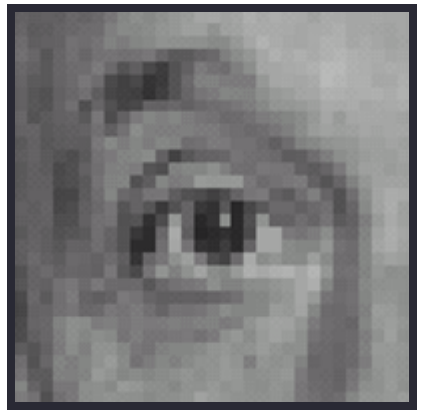
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

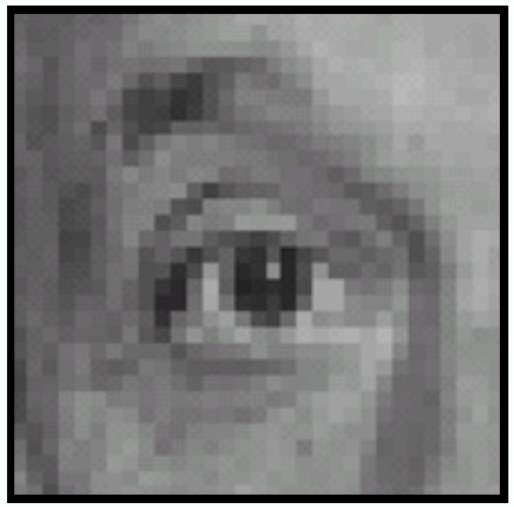


Original

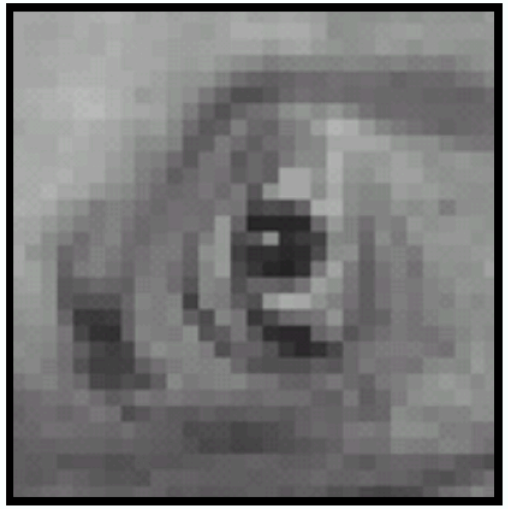
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

Source: D. Lowe

Practice with linear filters

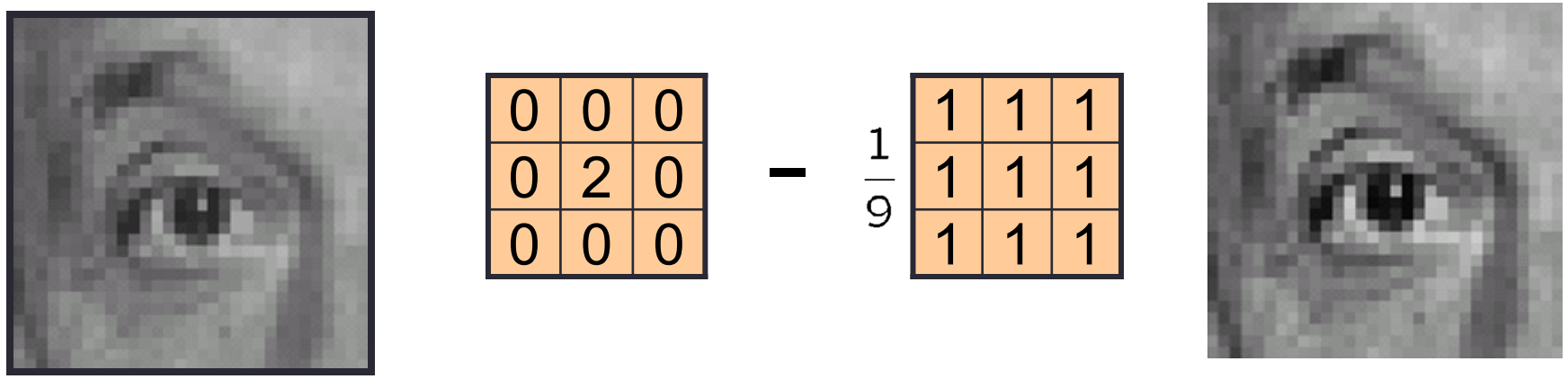


?



Interactive tool for some filters: https://setosa.io/ev/image-kernels/

Practice with linear filters



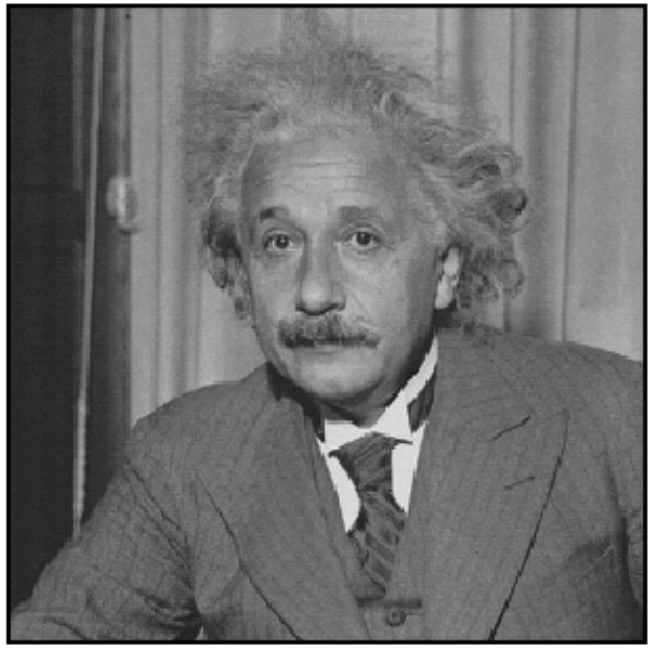
Original

Sharpening filter

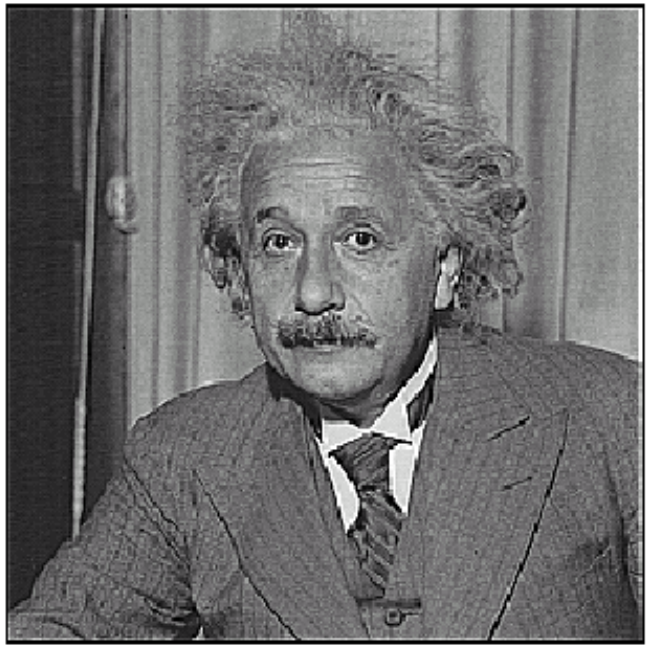
- Accentuates differences with local average

Source: D. Lowe

Sharpening



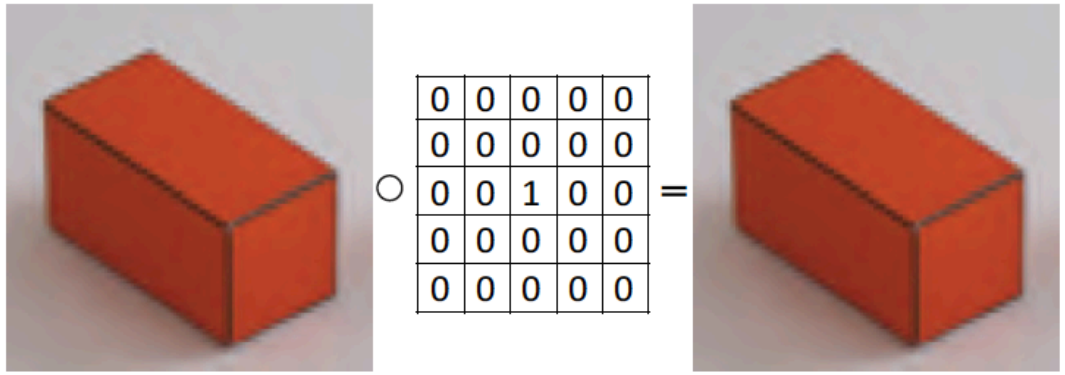
before



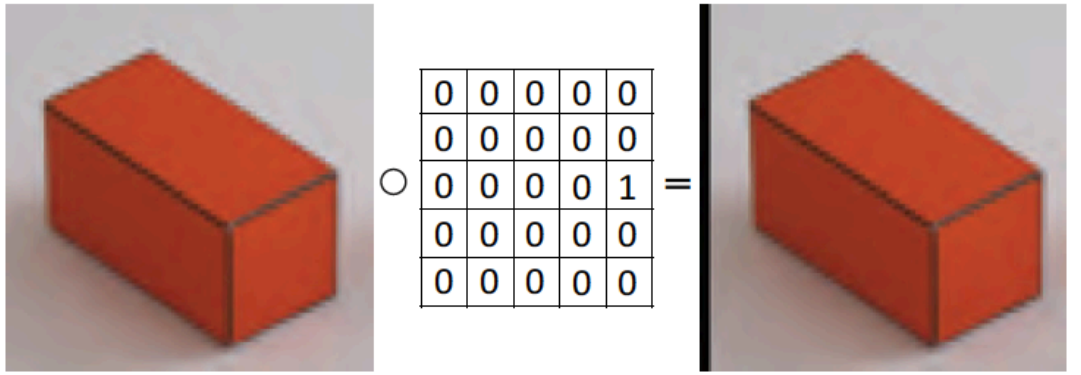
after

Source: D. Lowe

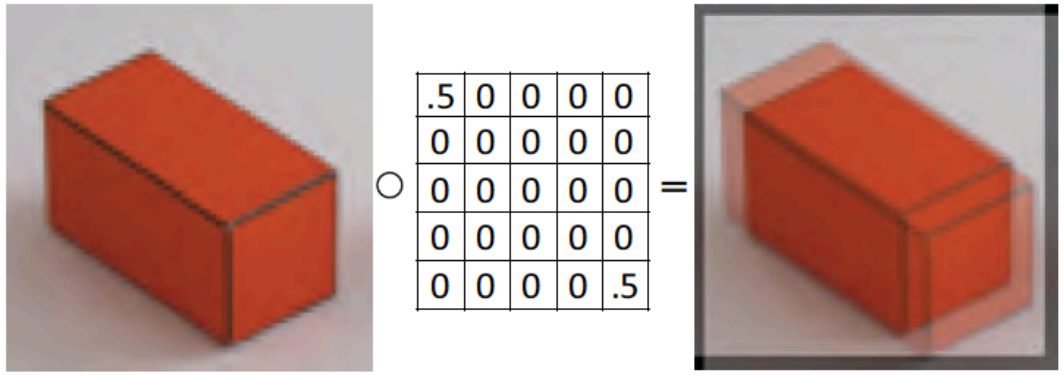
Practice with linear filters



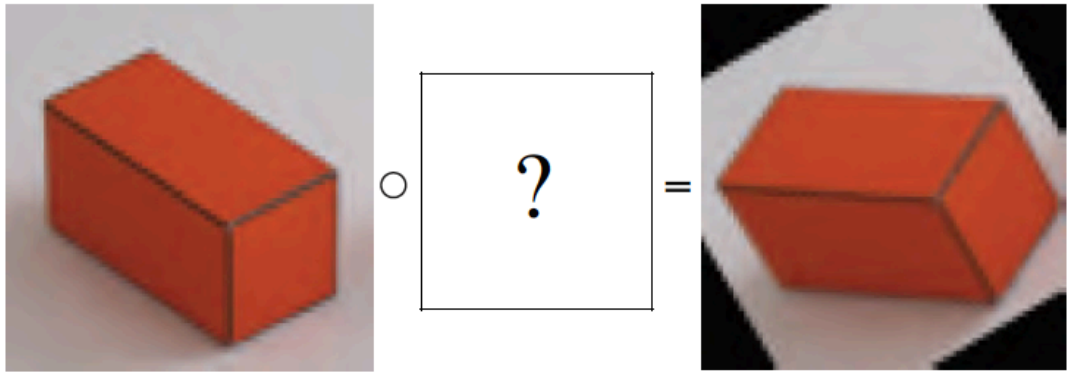
a)



b)



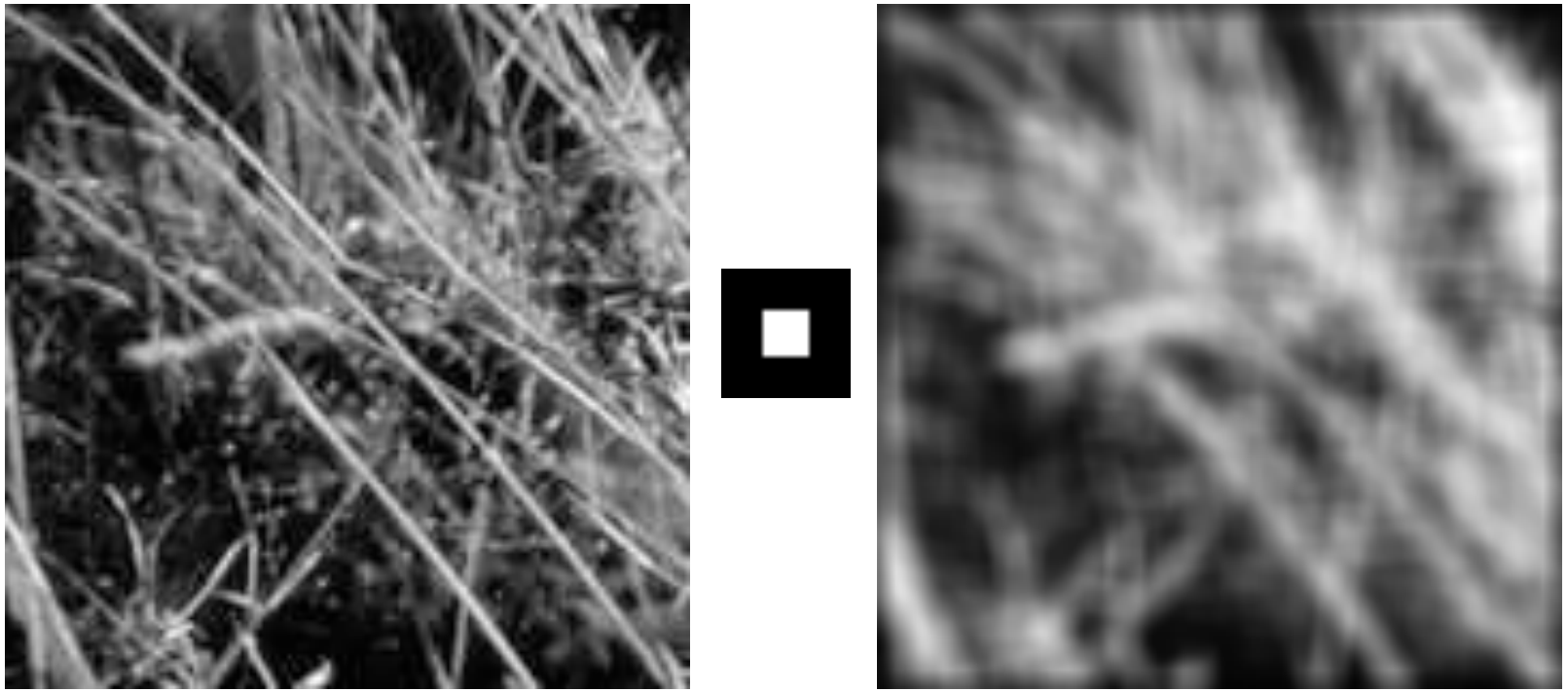
c)



d)

Smoothing with box filter revisited

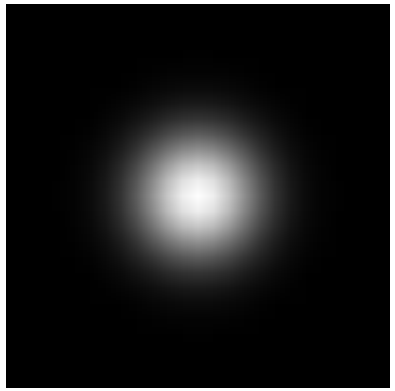
- What's wrong with this picture?
- What's the solution?



Source: D. Forsyth

Smoothing with box filter revisited

- What's wrong with this picture?
- What's the solution?
  - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center



"fuzzy blob"

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

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

$F[x, y]$

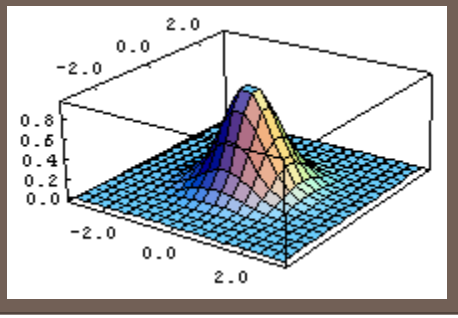
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

$H[u, v]$

This kernel is an approximation of a Gaussian function:

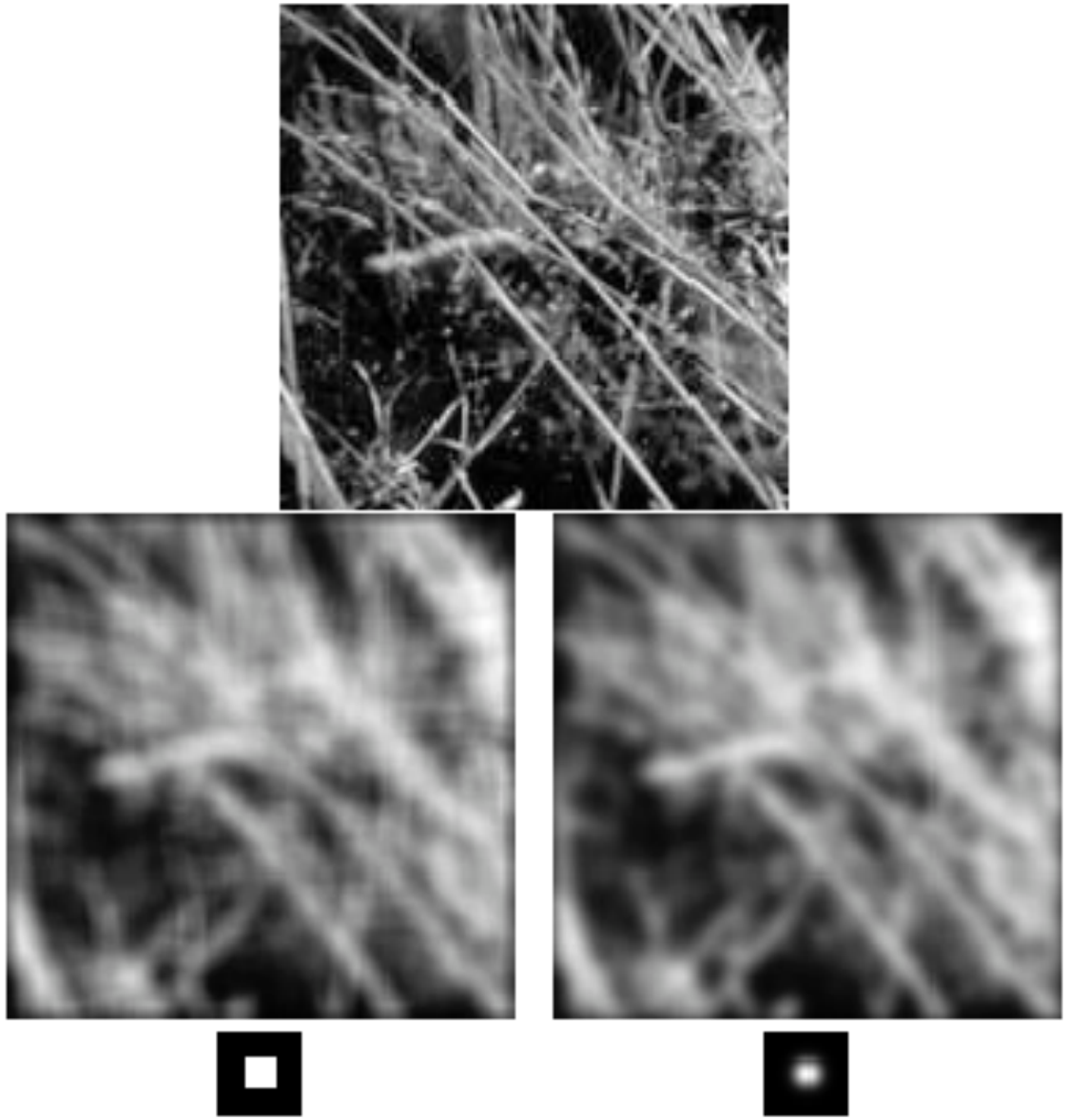
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Removes high-frequency components from the image (it is a “low-pass filter”).

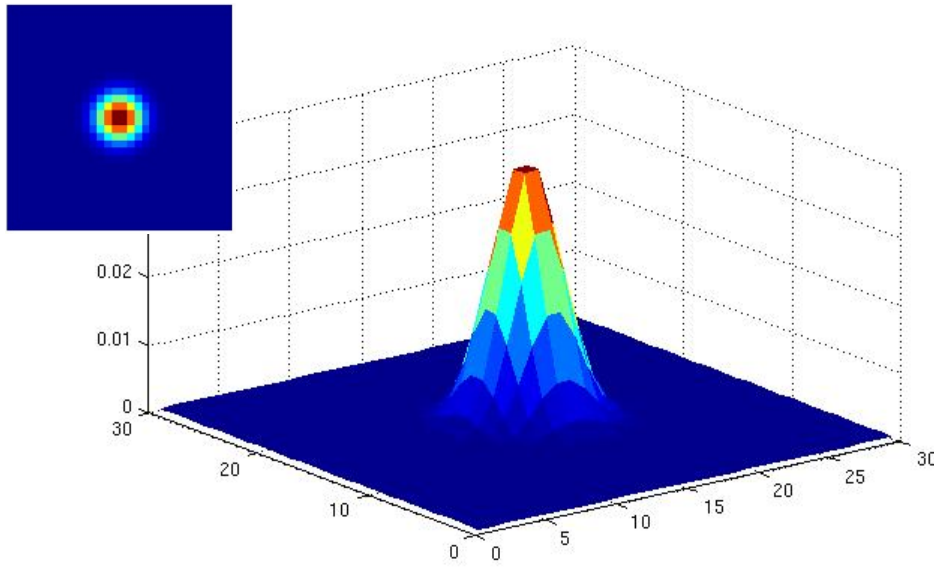
Source: S. Seitz

Gaussian vs. box filtering

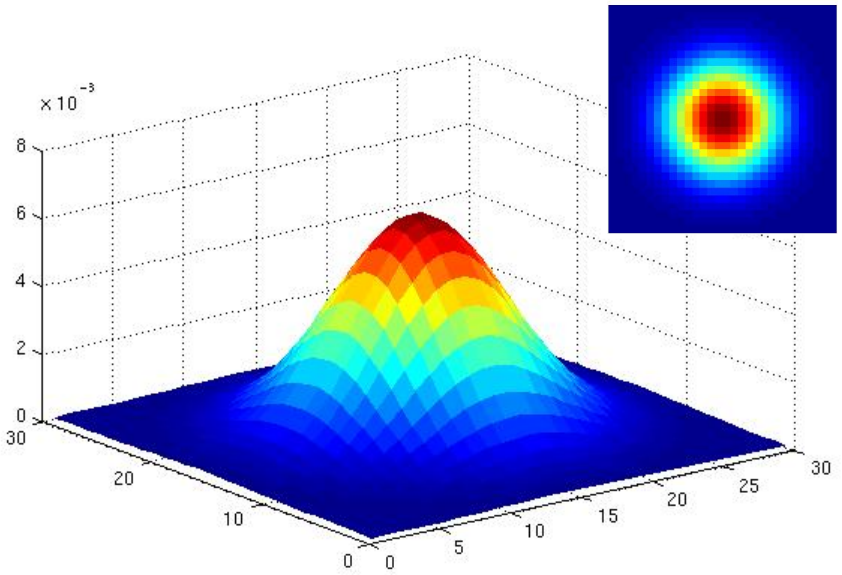


Gaussian Kernel

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



σ = 2 with 30 x 30 kernel

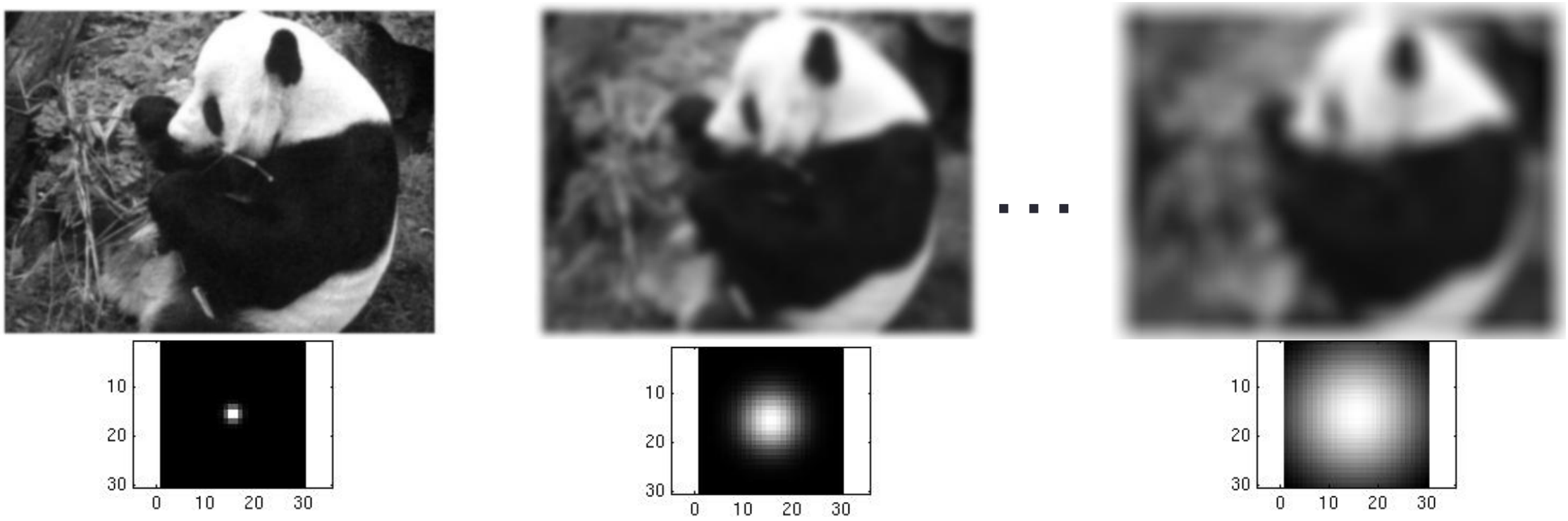


σ = 5 with 30 x 30 kernel

- Standard deviation σ: determines extent of smoothing

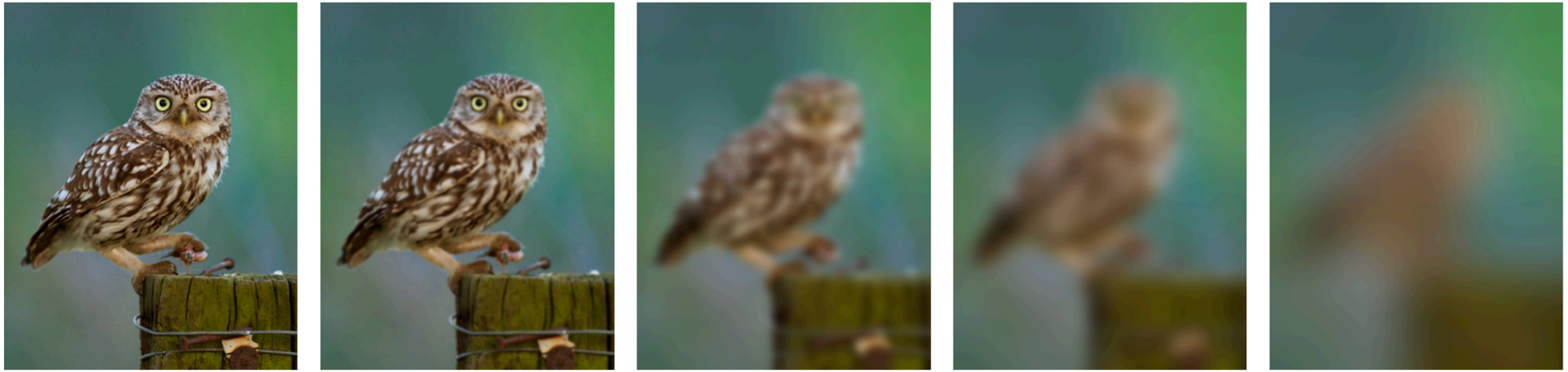
Smoothing with a Gaussian

Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



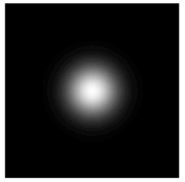

```
for sigma=1:3:10
    h = fspecial('gaussian', fsize, sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end
```

Gaussian Filters



σ = 1 pixel

σ = 5 pixels

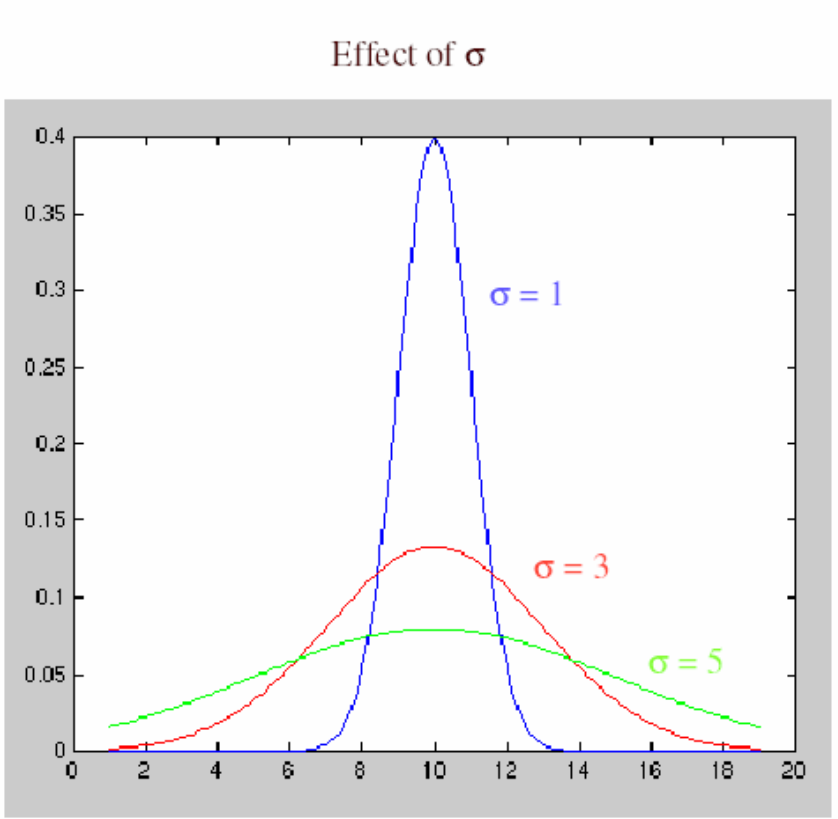


σ = 10 pixels

σ = 30 pixels

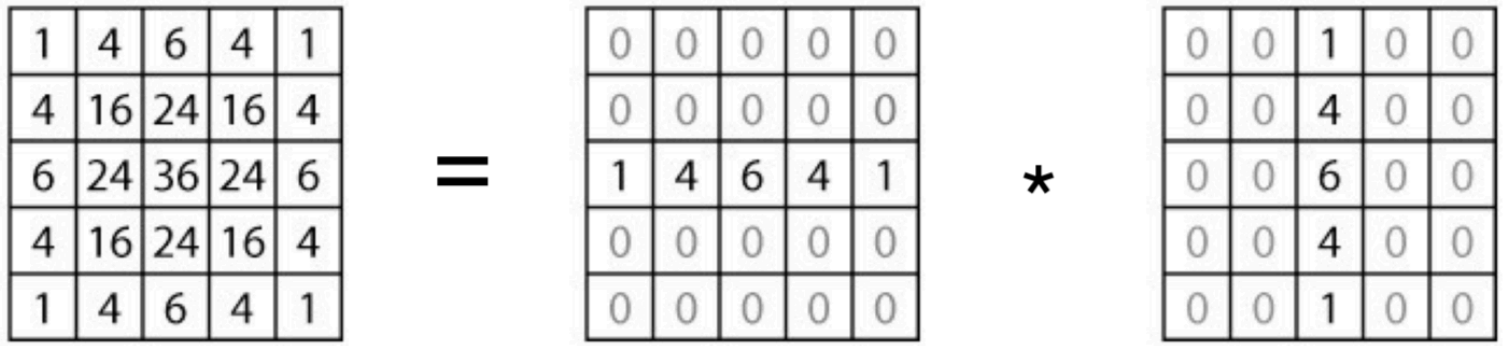
How big should the filter be?

- Values at edges should be near zero ← important!
- Rule of thumb for Gaussian: set filter half-width to about 3 σ



Separable Filtering

$$a_2[i,j] = a_1[i]a_1[j]$$



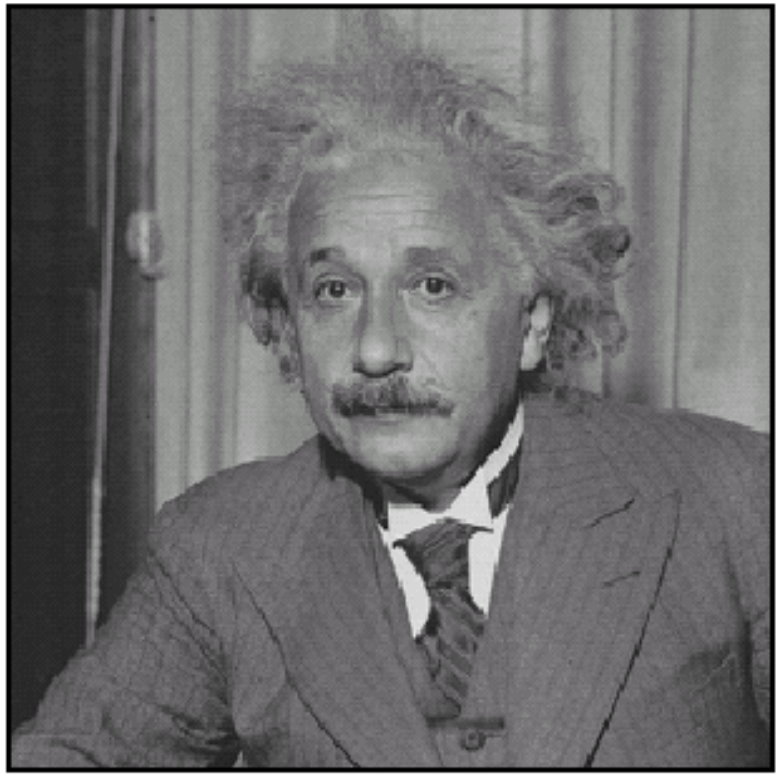
second, convolve with this

first, convolve with this

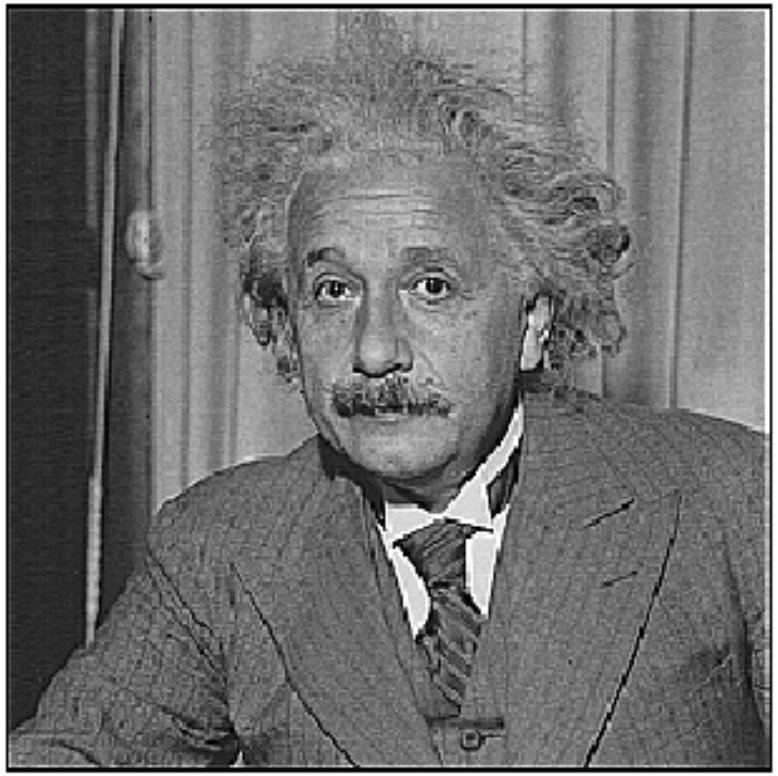
$$\sum_{i'} a_1[i'] \left(\sum_{j'} a_1[j']b[i-i', j-j'] \right)$$

Slide by S. Marschner

Sharpening revisited

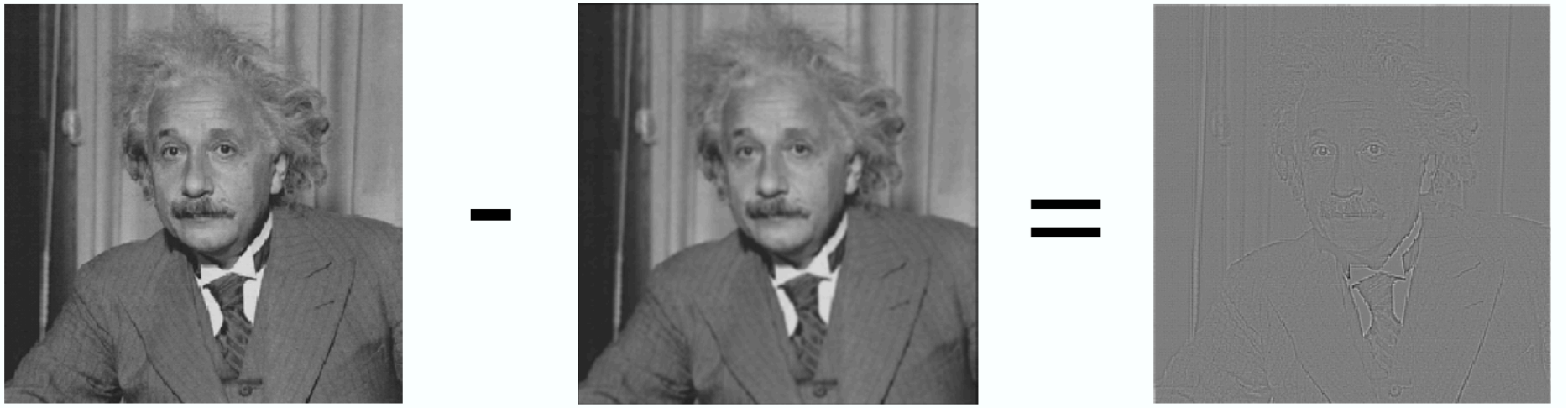


before



after

Sharpening revisited



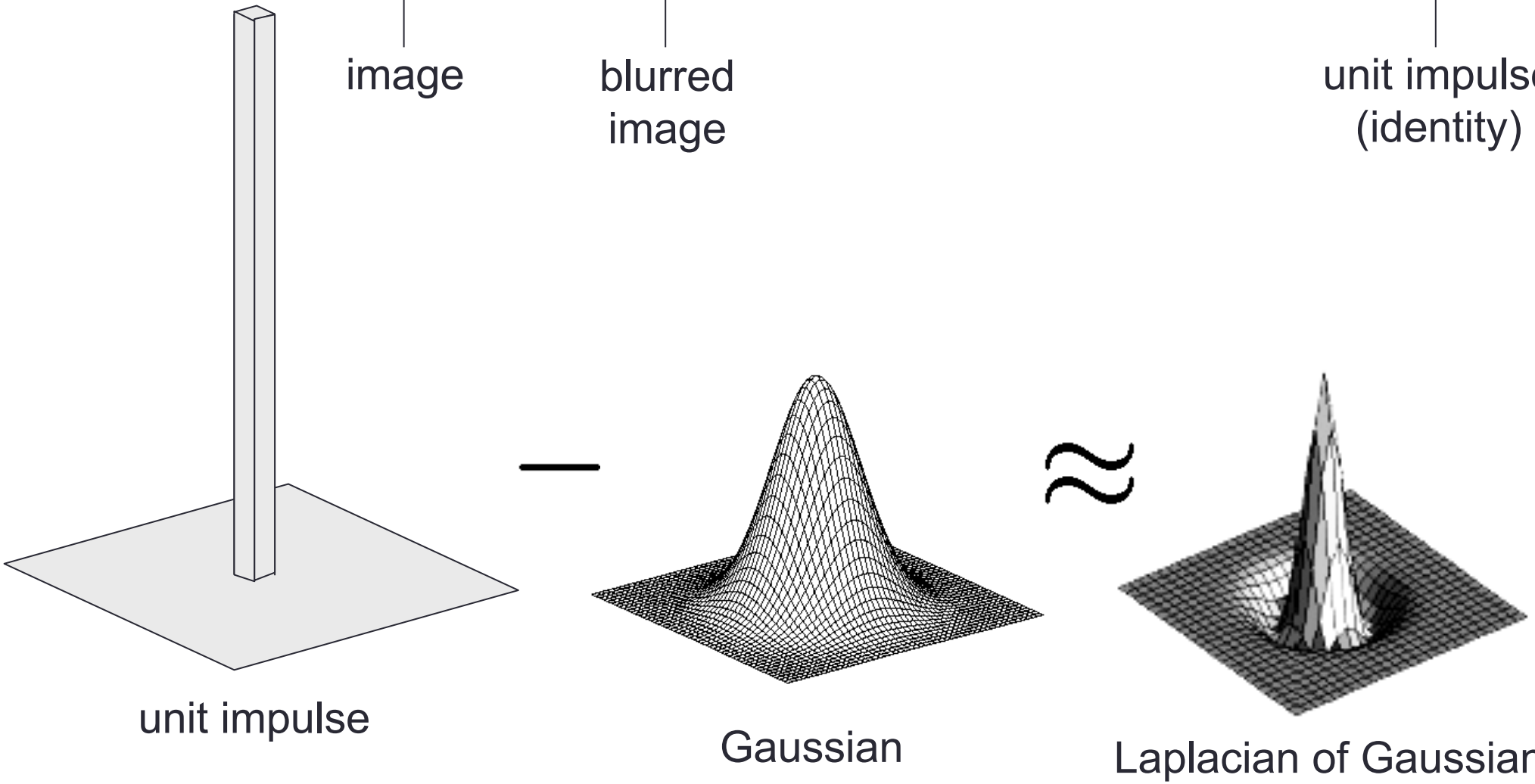
Unsharp mask filter

$$f + \alpha(f - f * g) = (1+\alpha)f - \alpha f * g = f * ((1+\alpha)e - g)$$

image

blurred image

unit impulse (identity)



unit impulse

Gaussian

Laplacian of Gaussian

Boundary issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Boundary issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods (MATLAB):
    - clip filter (black): imfilter(f, g, 0)
    - wrap around: imfilter(f, g, ‘circular’)
    - copy edge: imfilter(f, g, ‘replicate’)
    - reflect across edge: imfilter(f, g, ‘symmetric’)

Source: S. Marschner

Common Types of Noise

Original

Salt and pepper noise

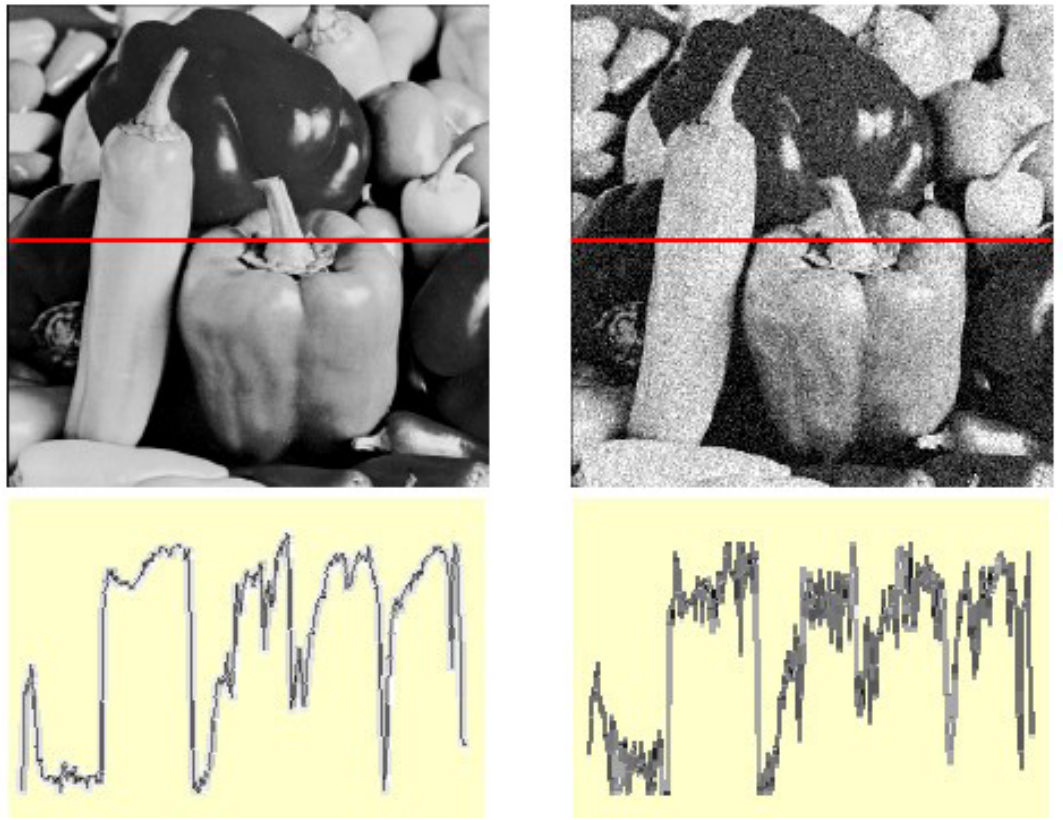
Impulse noise

Gaussian noise

- **Salt and pepper noise**: contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Source: S. Seitz

Gaussian noise



$$f(x,y) = \overbrace{\bar{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

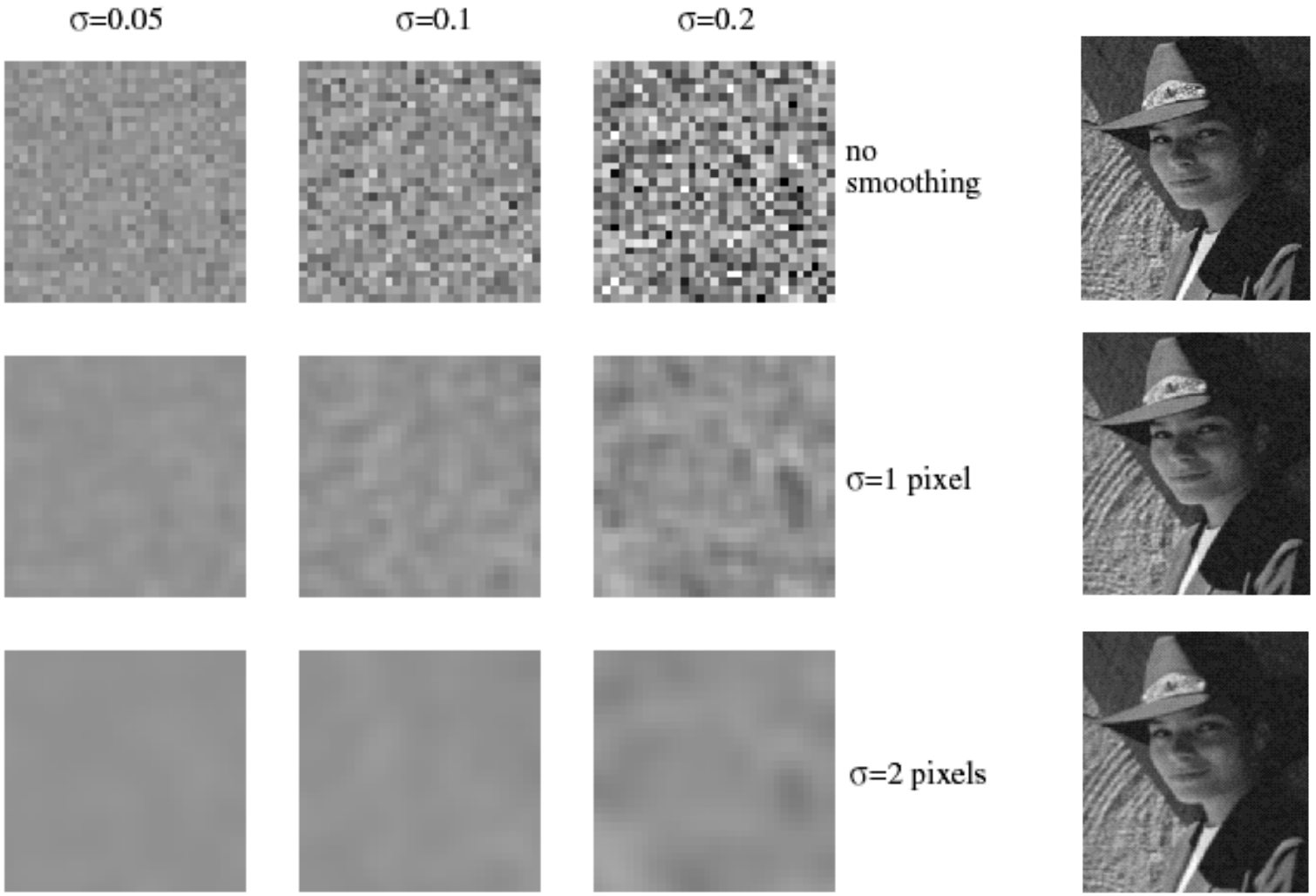
Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu, \sigma)$

```
>> noise = randn(size(im)).*sigma;

>> output = im + noise;
```

Fig: M. Hebert

Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

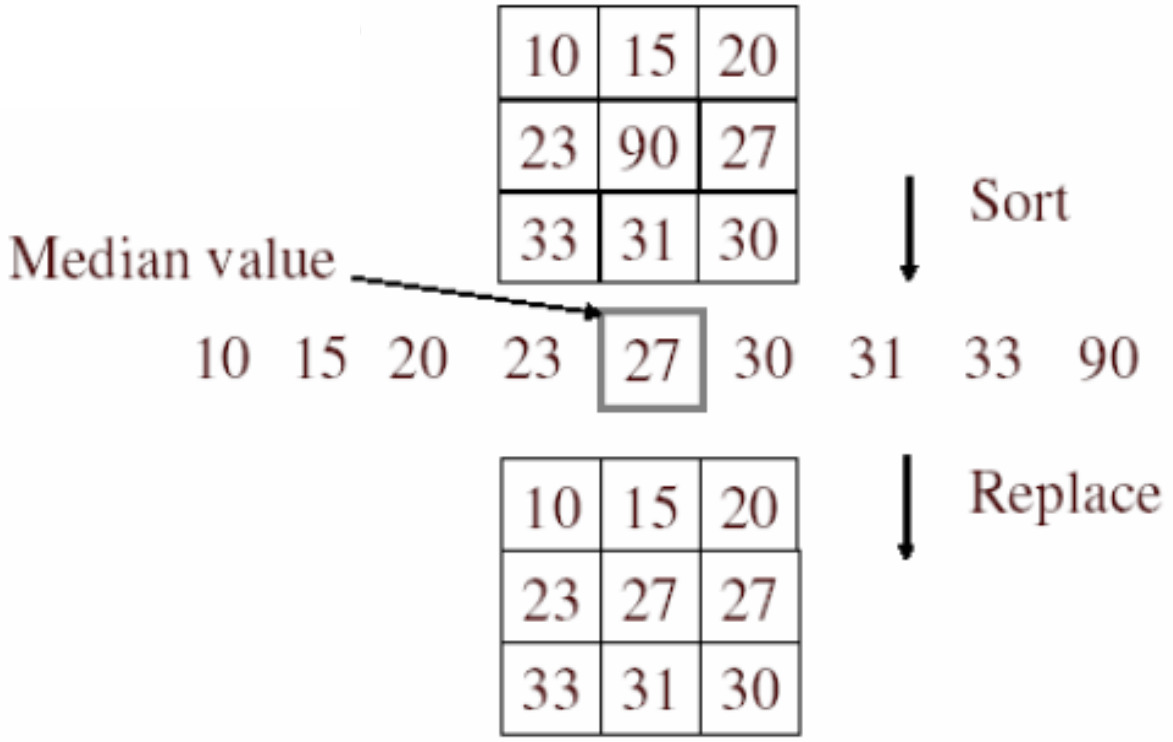
Reducing salt-and-pepper noise

3x3 5x5 7x7

- What's wrong with the results?

Alternative idea: Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window

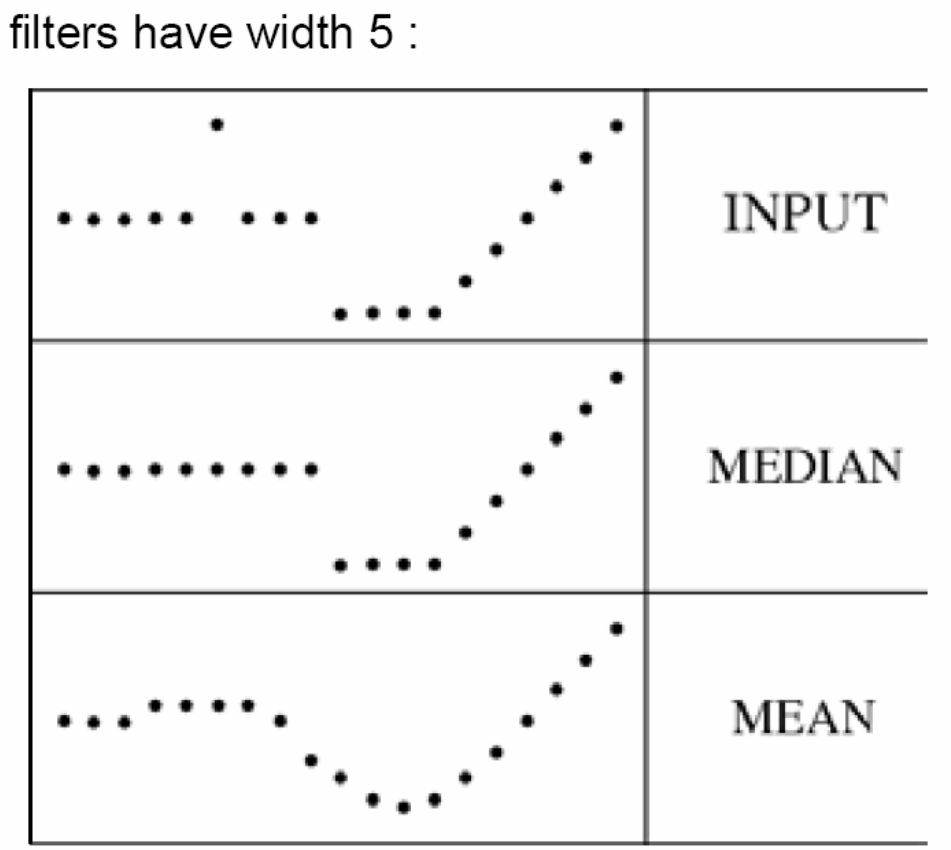


- Is median filtering linear?

Source: K. Grauman

Median filter

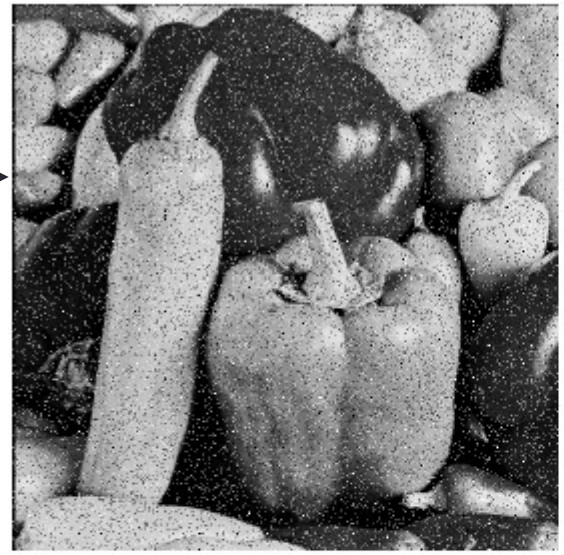
- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers
  - Edge preserving



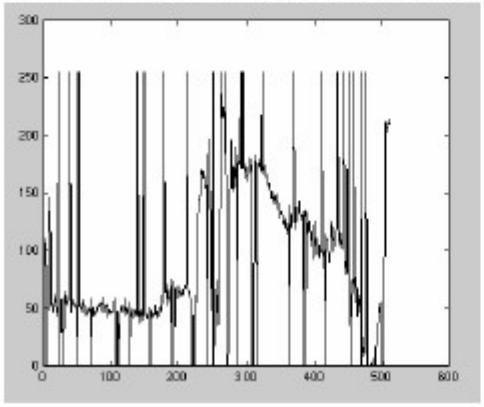
Source: K. Grauman

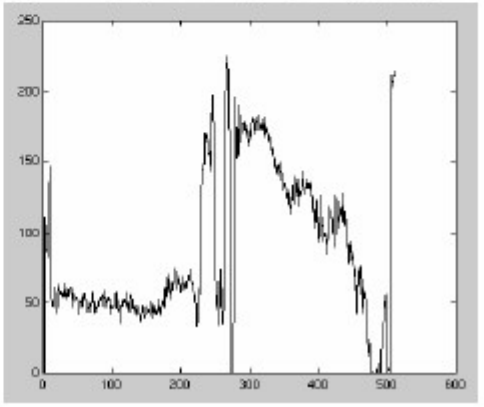
Median filter

Salt and pepper noise



Median filtered

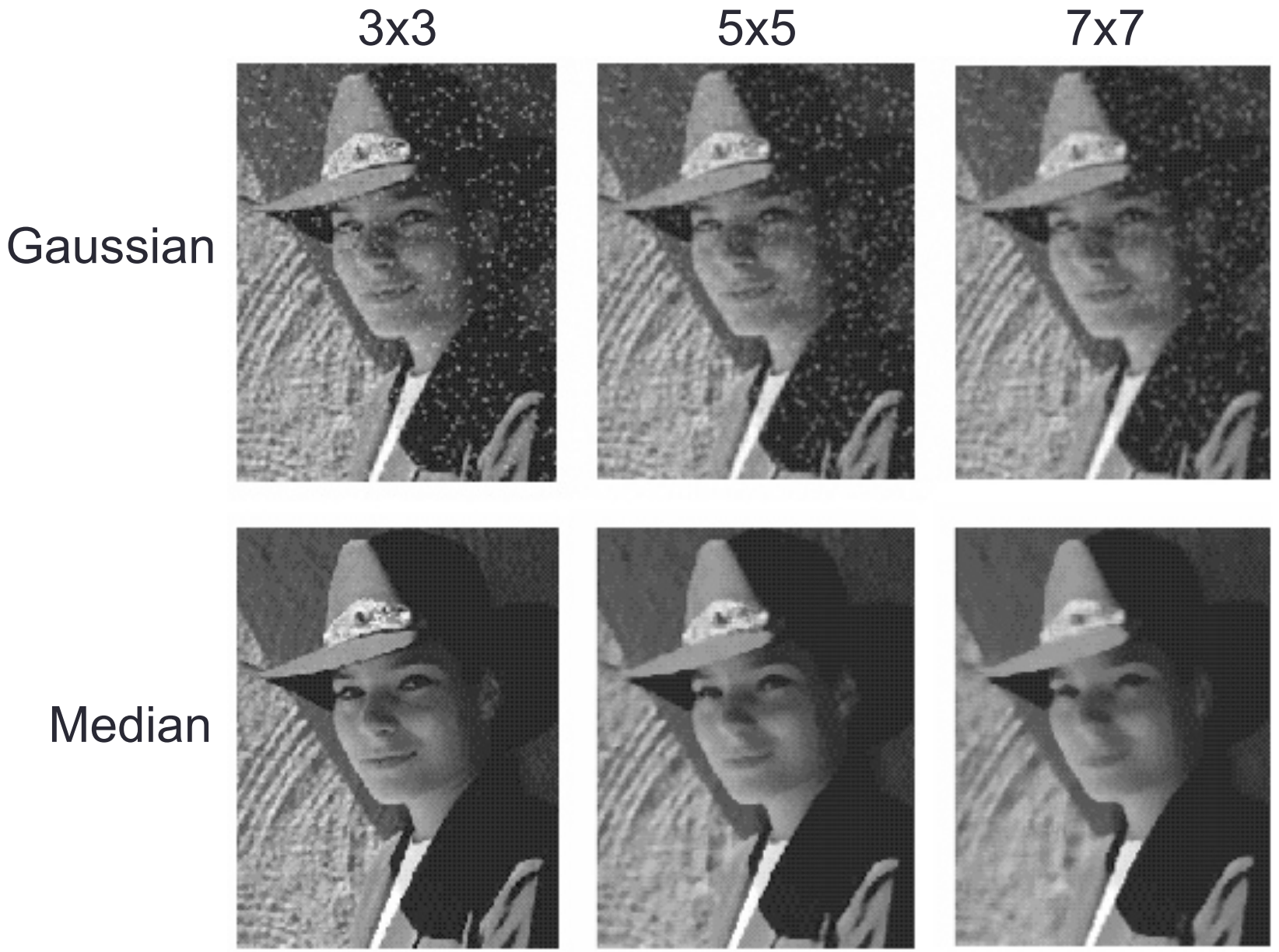




Plots of a row of the image

Source: M. Hebert

Gaussian vs. median filtering



Readings

Szeliski Book

Chapter 2 and 3