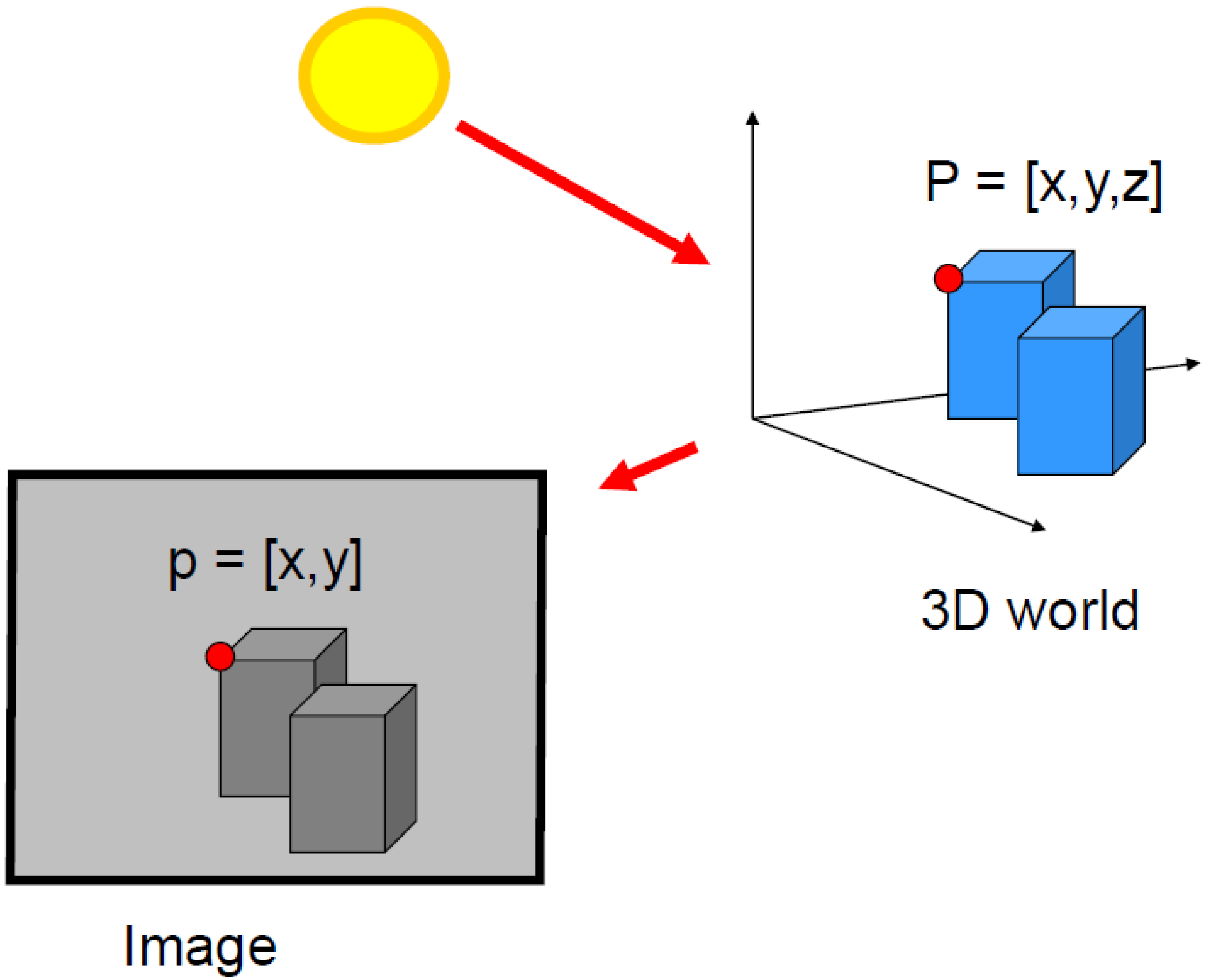
https://www.reddit.com/r/Utah/comments/177ymi6/the_partial_eclipse_shadow_through_my_trees_st/

Welcome back!

- Optional project 0 is out.
- Project 1 will be out soon.
- Read Szeliski 2.1, especially 2.1.4
- Today
  - Image projection
  - Filtering

From the 3D to 2D



Slide credit Fei Fei Li

Chapter 2

Image formation

Camera Obscura used for Tracing

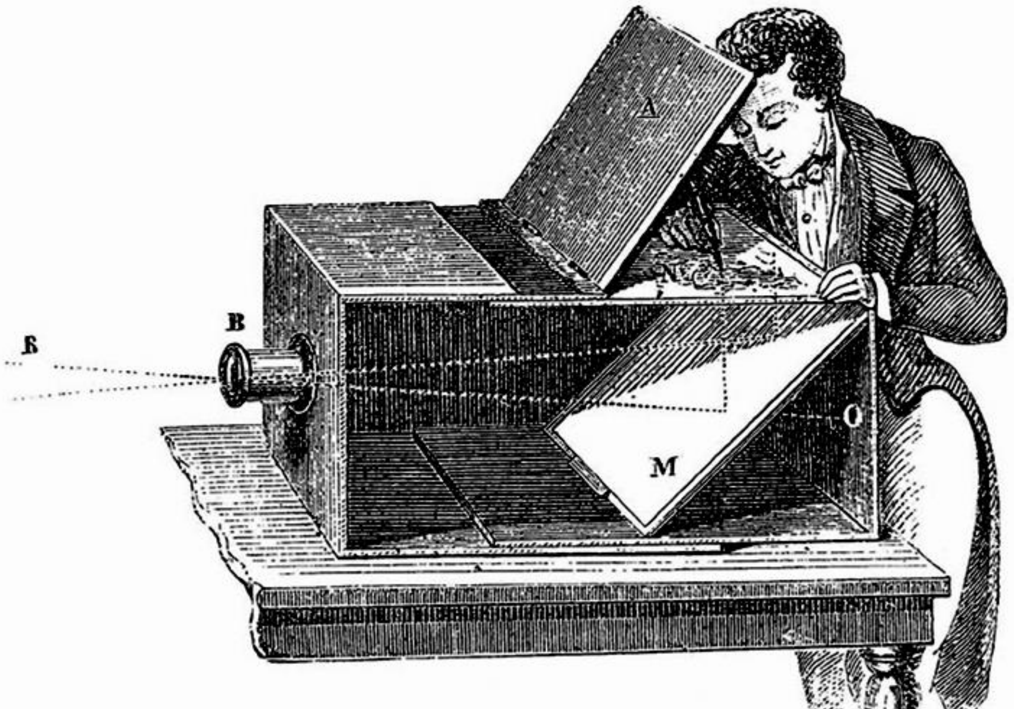


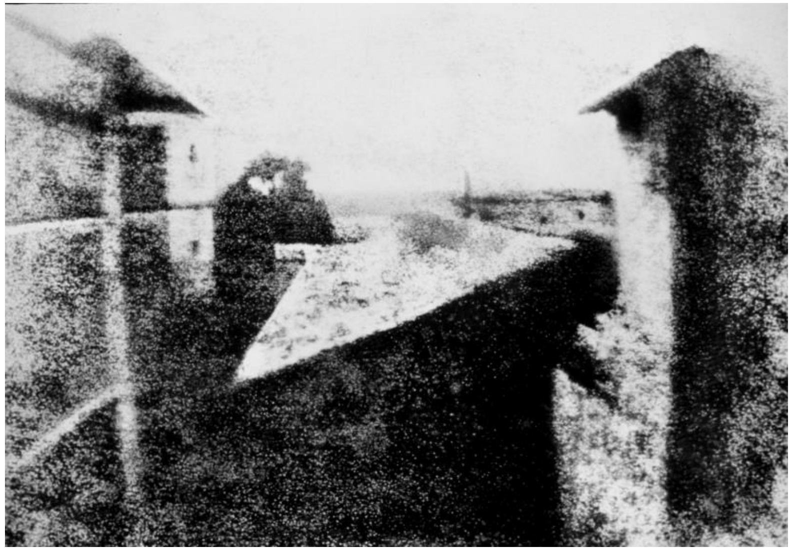
Fig. 434.

Lens Based Camera Obscura, 1568

First Photograph

Oldest surviving photograph

– Took 8 hours on pewter plate

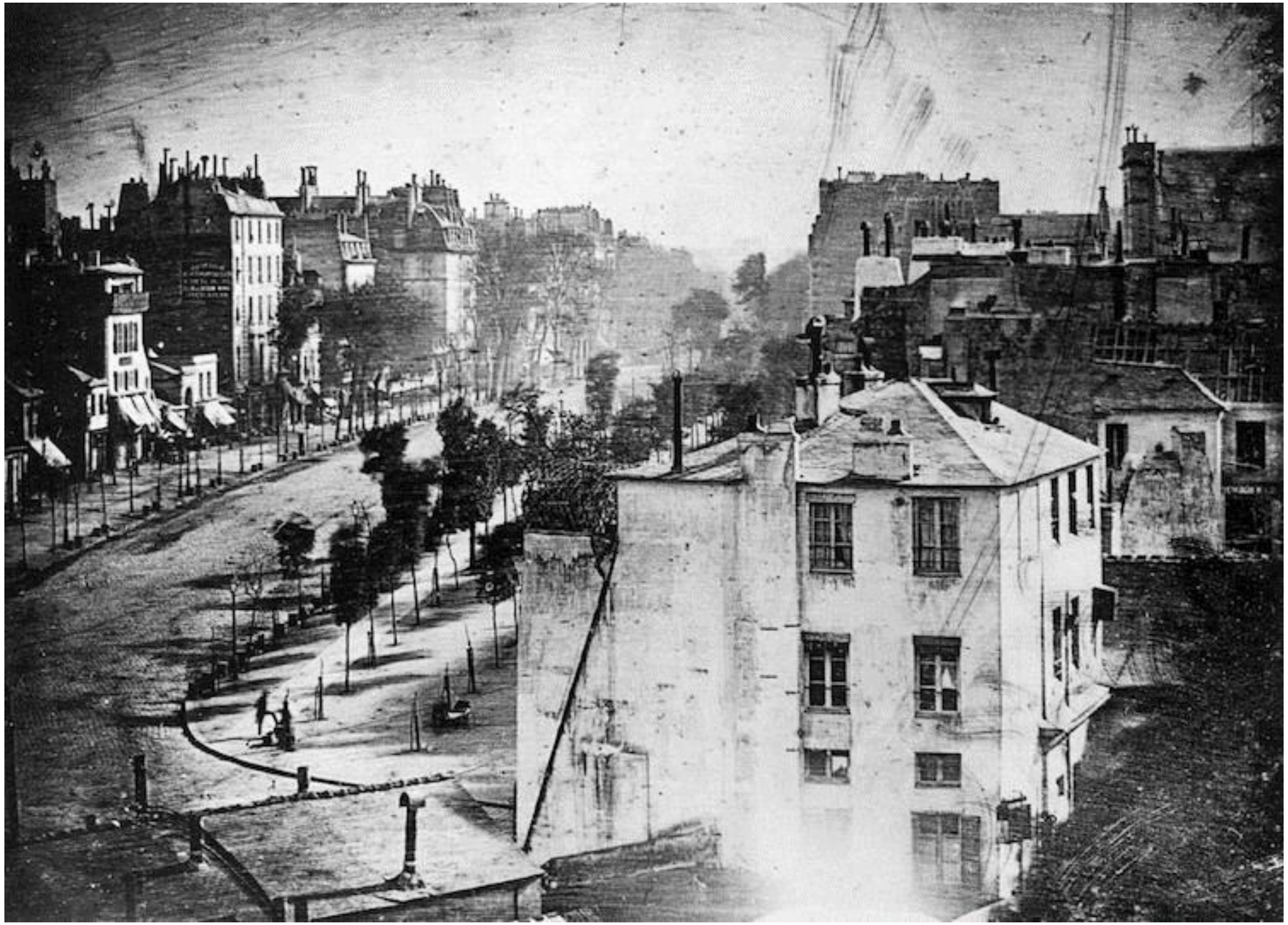


Joseph Niepce, 1826

Photograph of the first photograph

On display at UT Austin

Niepce later teamed up with Daguerre, who eventually created Daguerrotypes



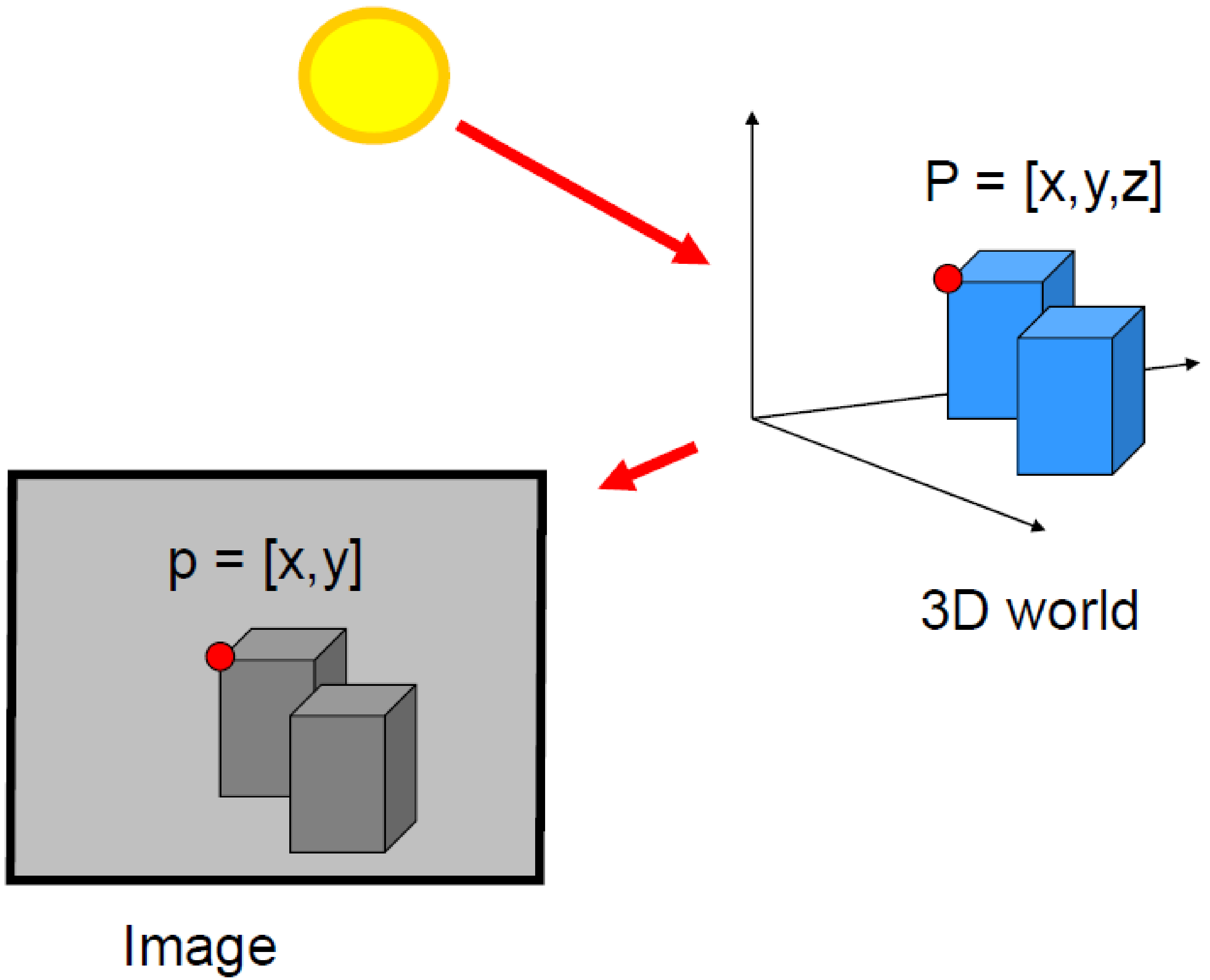
“Louis Daguerre—the inventor of daguerreotype—shot what is not only the world's oldest photograph of Paris, but also the first photo with humans. The 10-minute long exposure was taken in 1839 in Place de la République and it's just possible to make out two blurry figures in the left-hand corner.”

Source

Great history lesson on the chemistry and engineering challenges of early photography from the "Technology Connections" YouTube channel.

https://www.youtube.com/watch?v=wbbH77rYaa8&list=PLv0jwu7G_DFV6yW240e6CbiwCLaZ0Z6PV

From the 3D to 2D

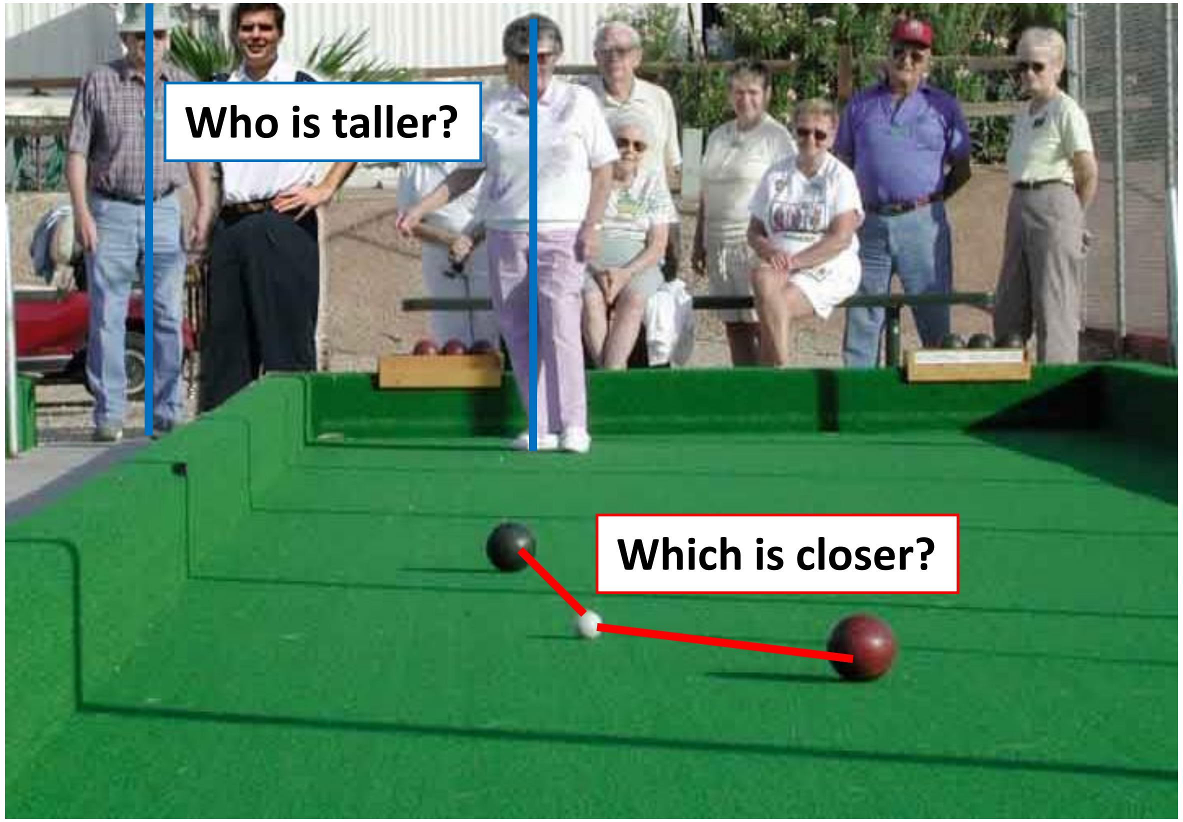


Slide credit Fei Fei Li

Projective Geometry

What is lost?

- Length



Length and area are not preserved

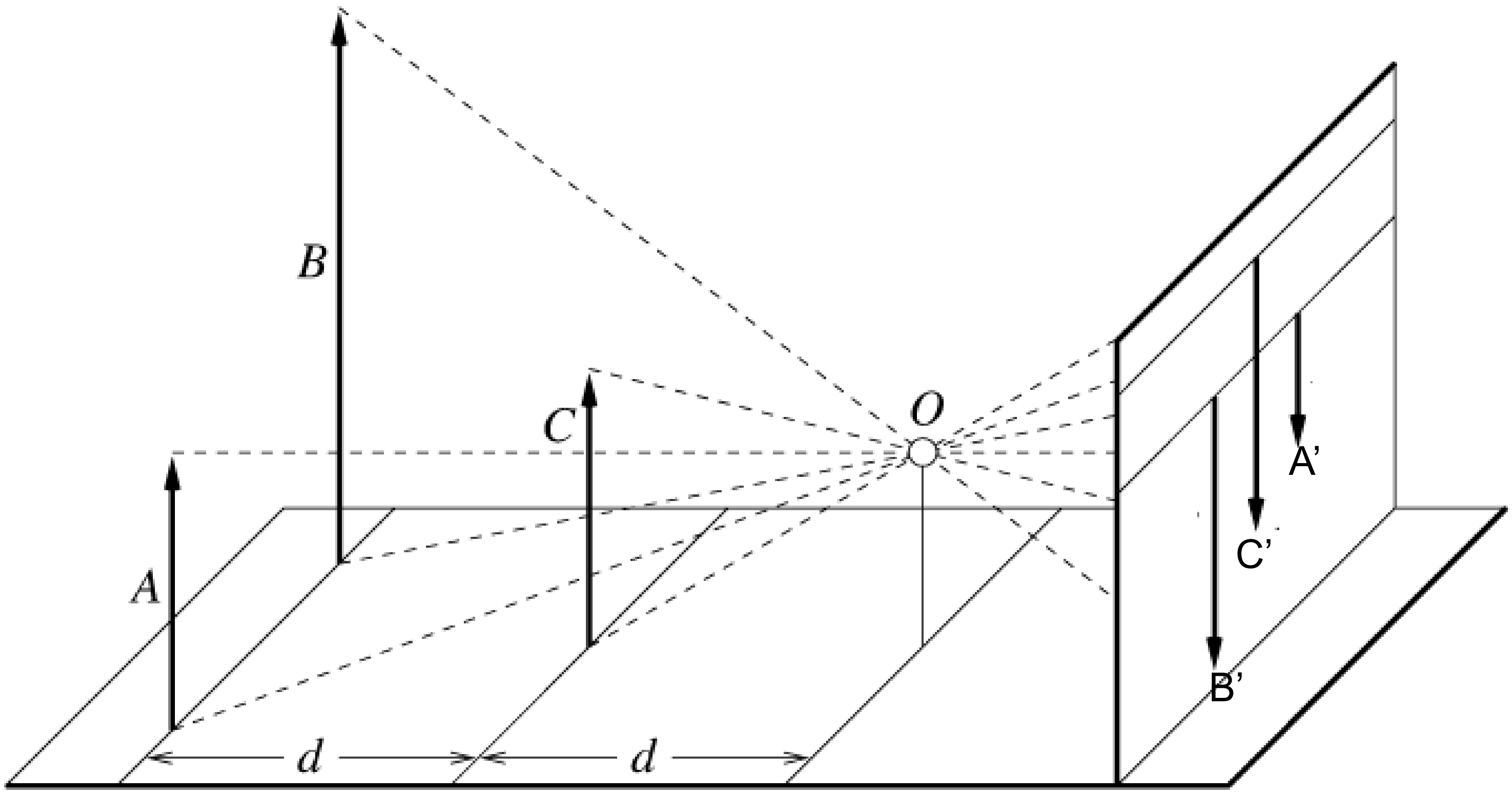
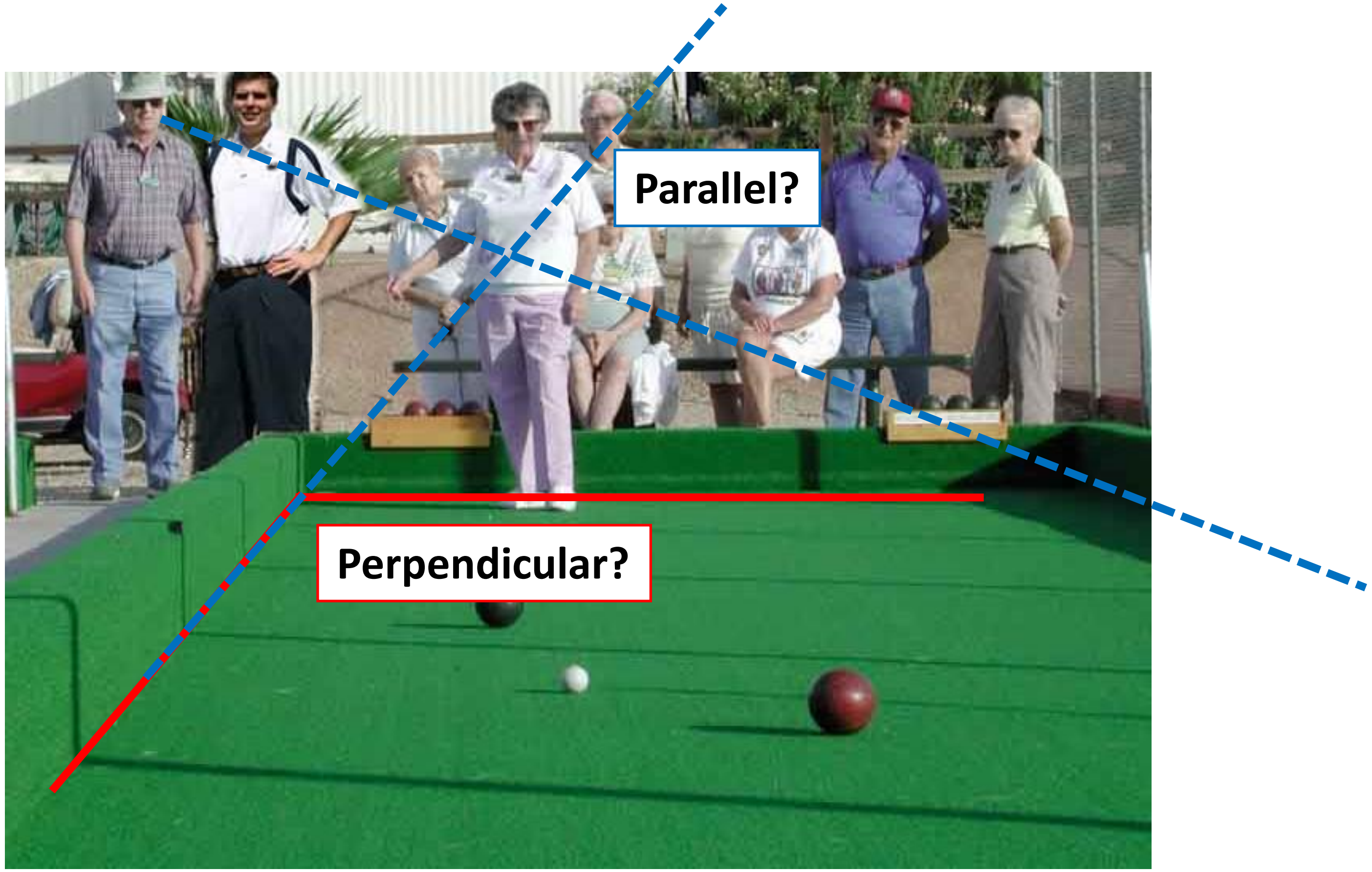


Figure by David Forsyth

Projective Geometry

What is lost?

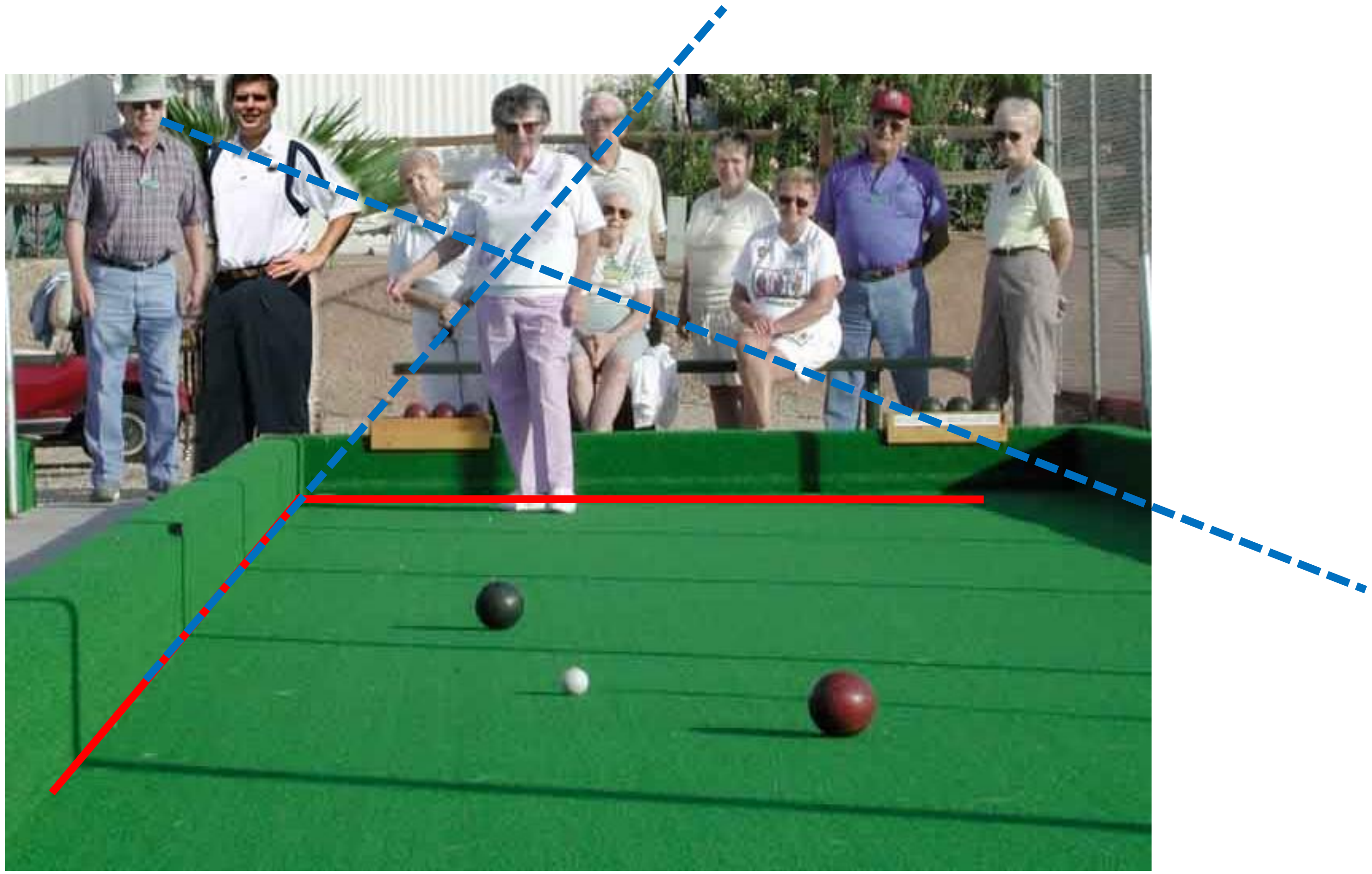
- Length
- Angles



Projective Geometry

What is preserved?

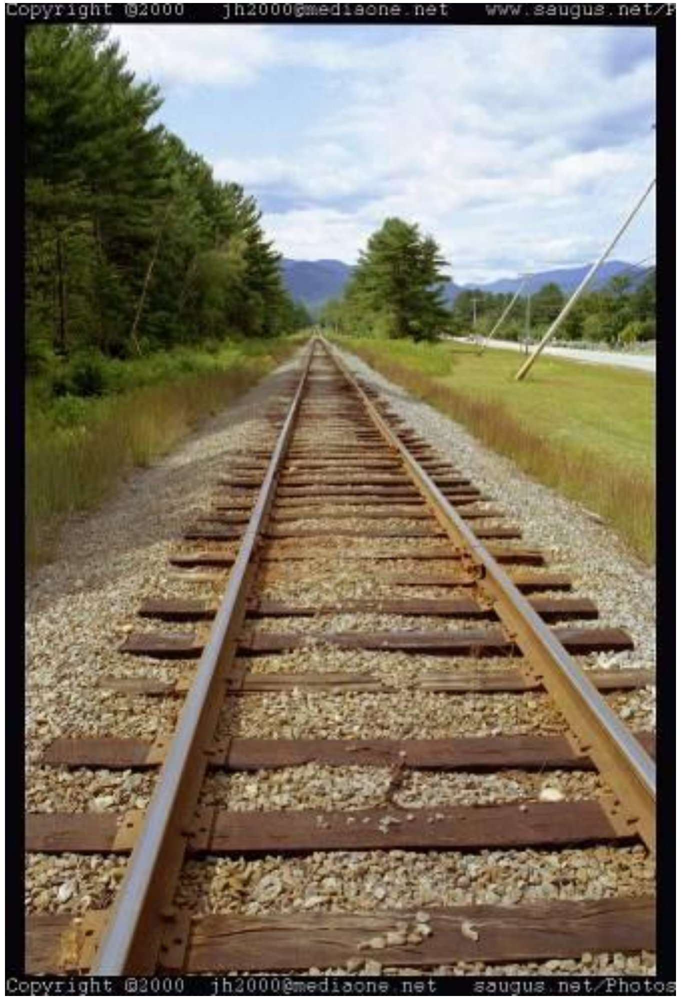
- Straight lines are still straight



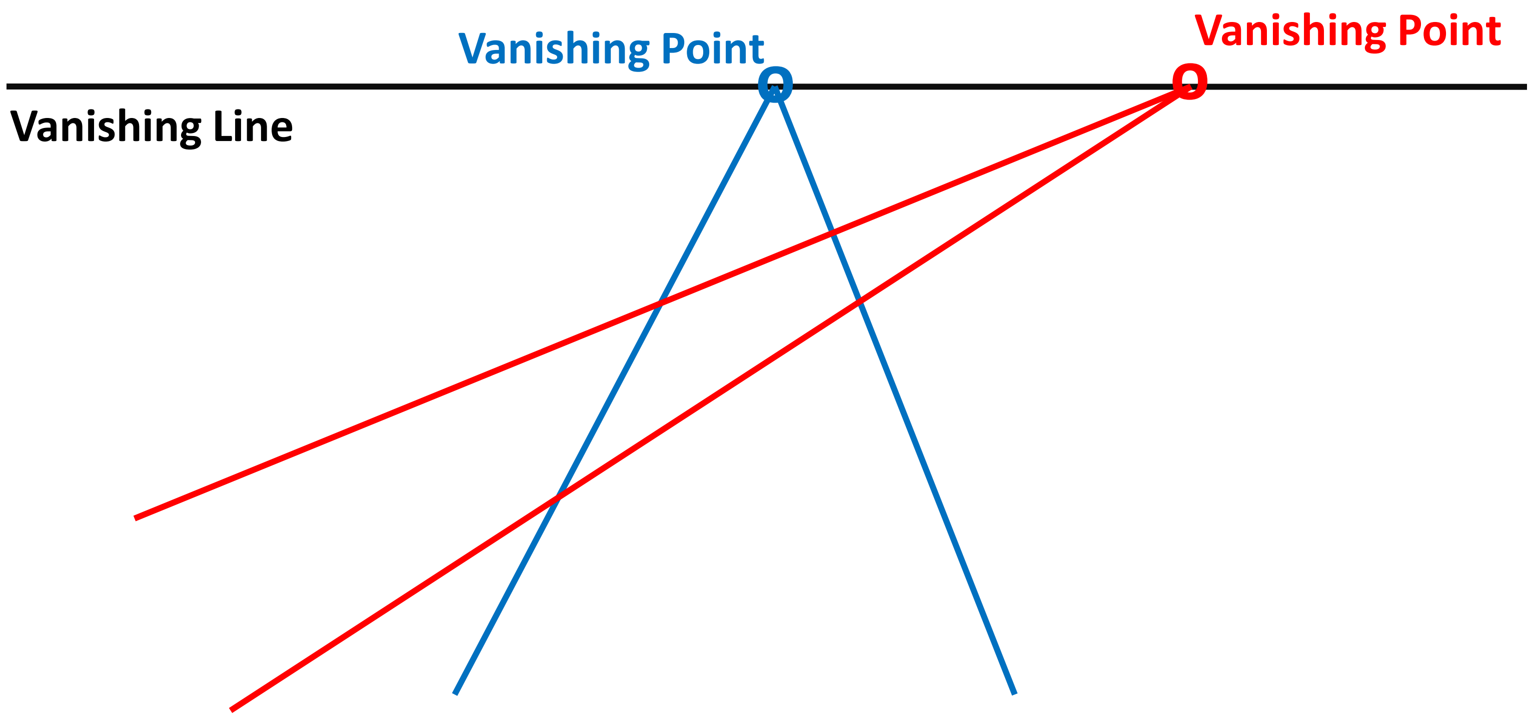
The pinhole camera model preserves straight lines, but real cameras might not

Vanishing points and lines

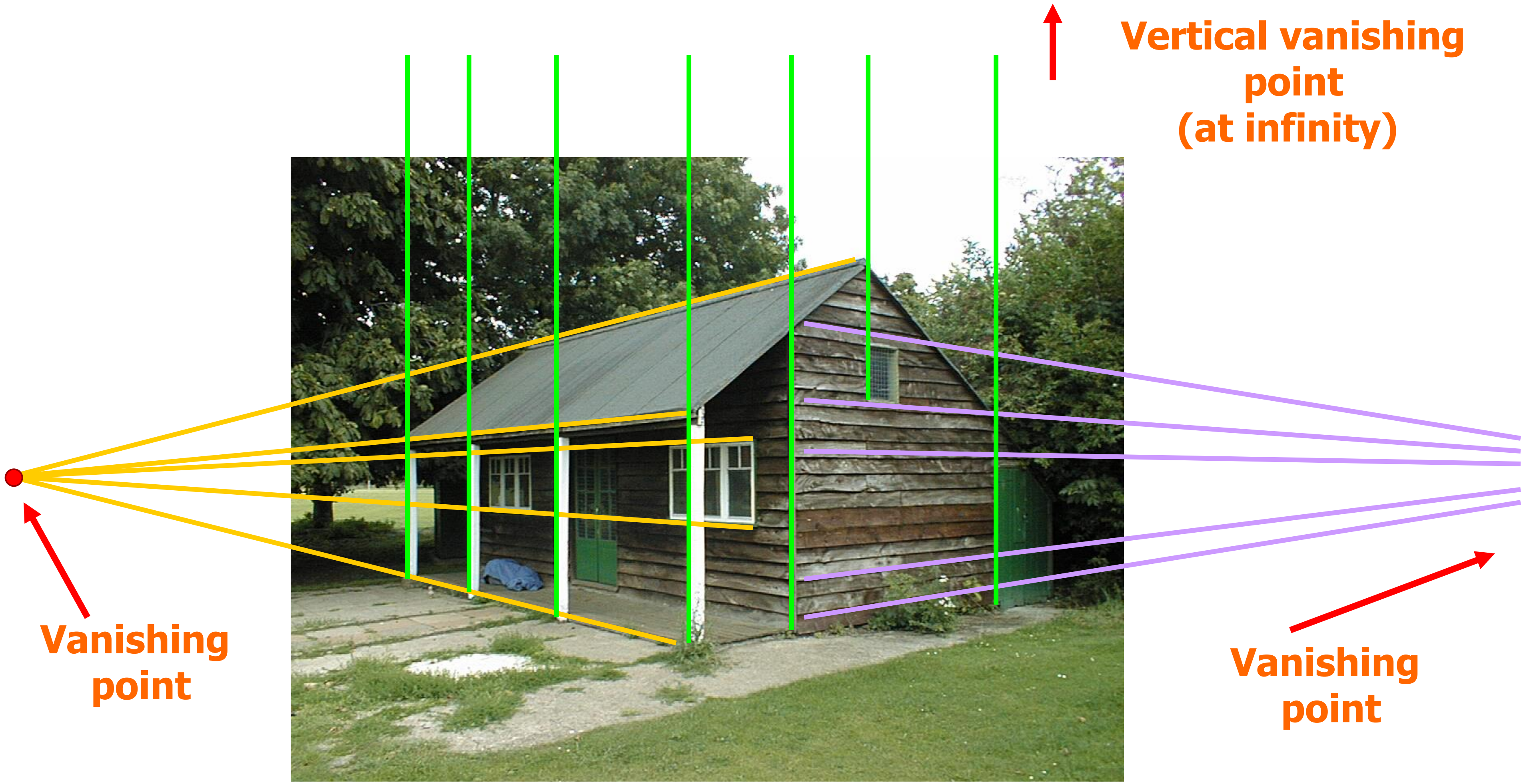
Parallel lines in the world intersect in the image at a “vanishing point”



Vanishing points and lines

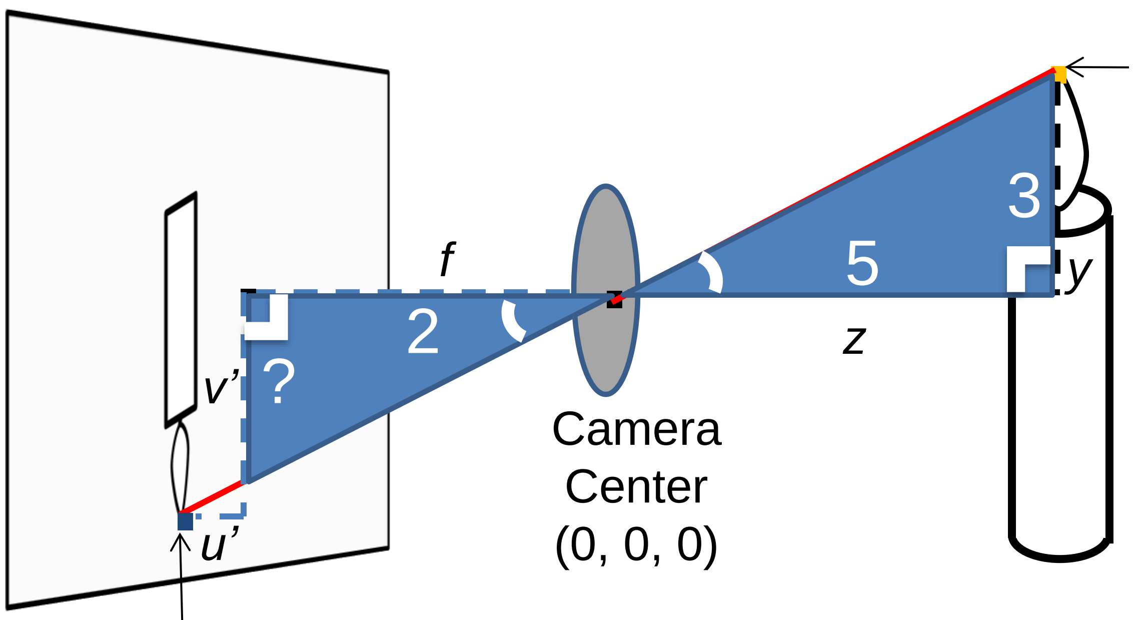


Vanishing points and lines



Slide from Efros, Photo from Criminisi

Projection: world coordinates→image coordinates



$$X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$x = \begin{bmatrix} u' \\ v' \end{bmatrix}$$

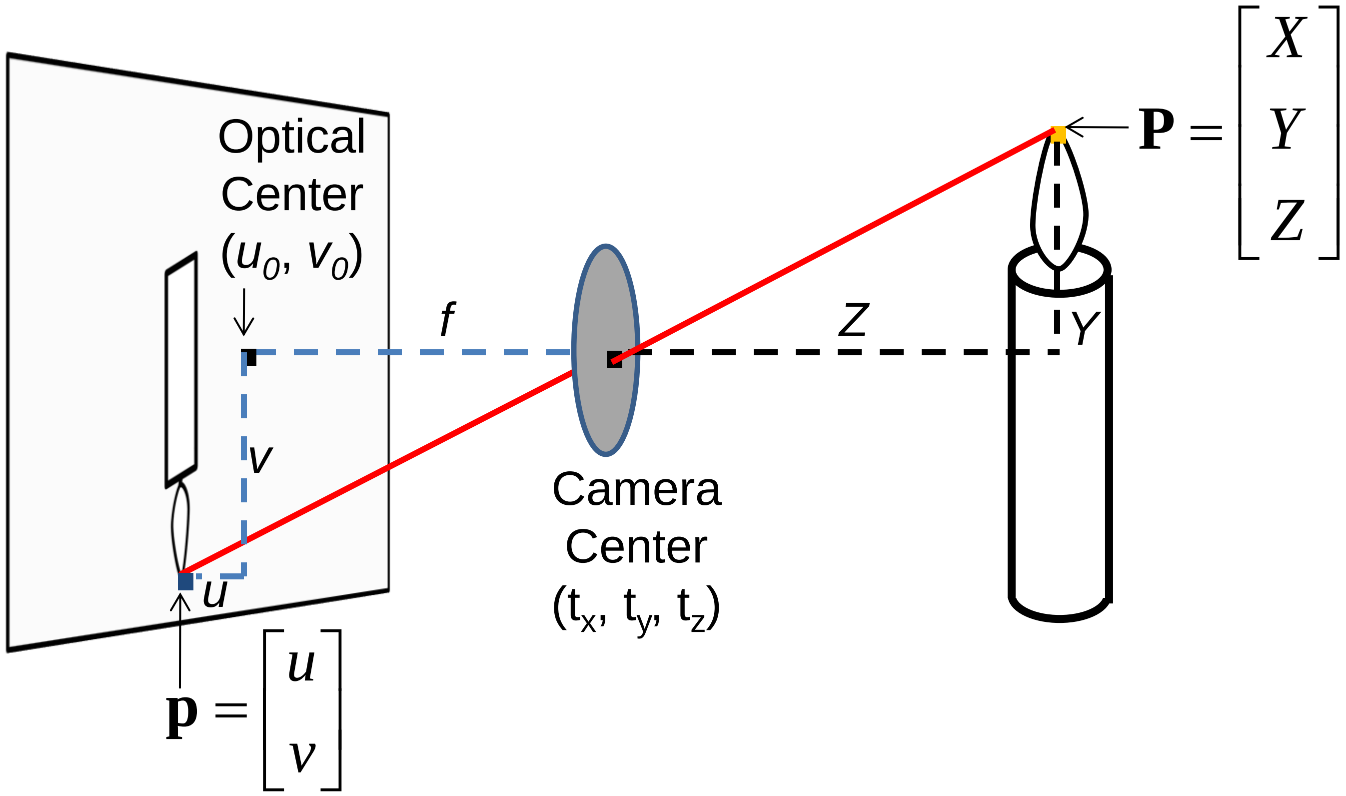
If x = 2, y = 3, z = 5, and
f = 2
What are u' and v'?

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

$$u' = -x * \frac{f}{z} \qquad u' = -2 * \frac{2}{5}$$

$$v' = -y * \frac{f}{z} \qquad v' = -3 * \frac{2}{5}$$

Projection: world coordinates→image coordinates



How do we handle the general case?

Interlude: why does this matter?

Relating multiple views

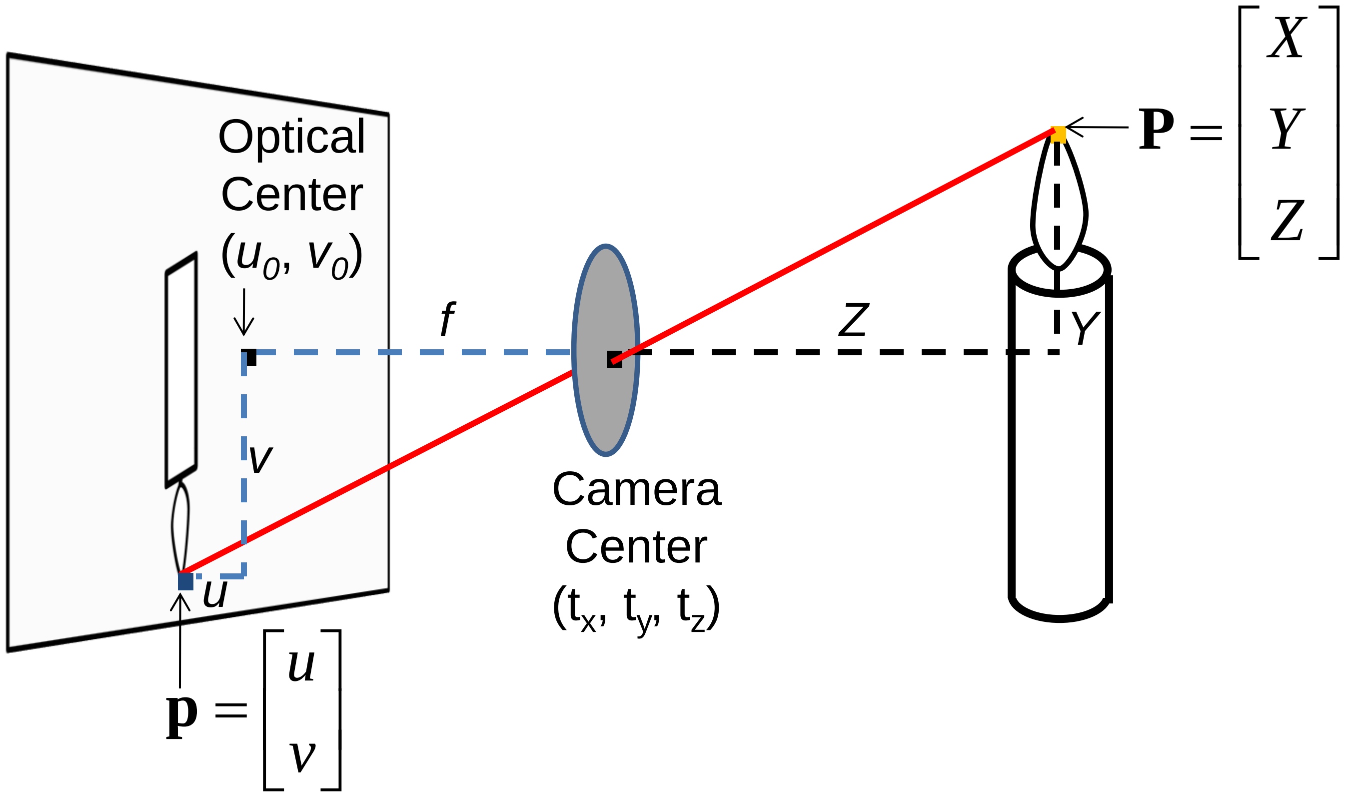
Photo Tourism

Exploring photo collections in 3D

Noah Snavely   Steven M. Seitz   Richard Szeliski
University of Washington   *Microsoft Research*

SIGGRAPH 2006

Projection: world coordinates→image coordinates



How do we handle the general case?

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

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

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

Homogeneous coordinates

Invariant to scaling

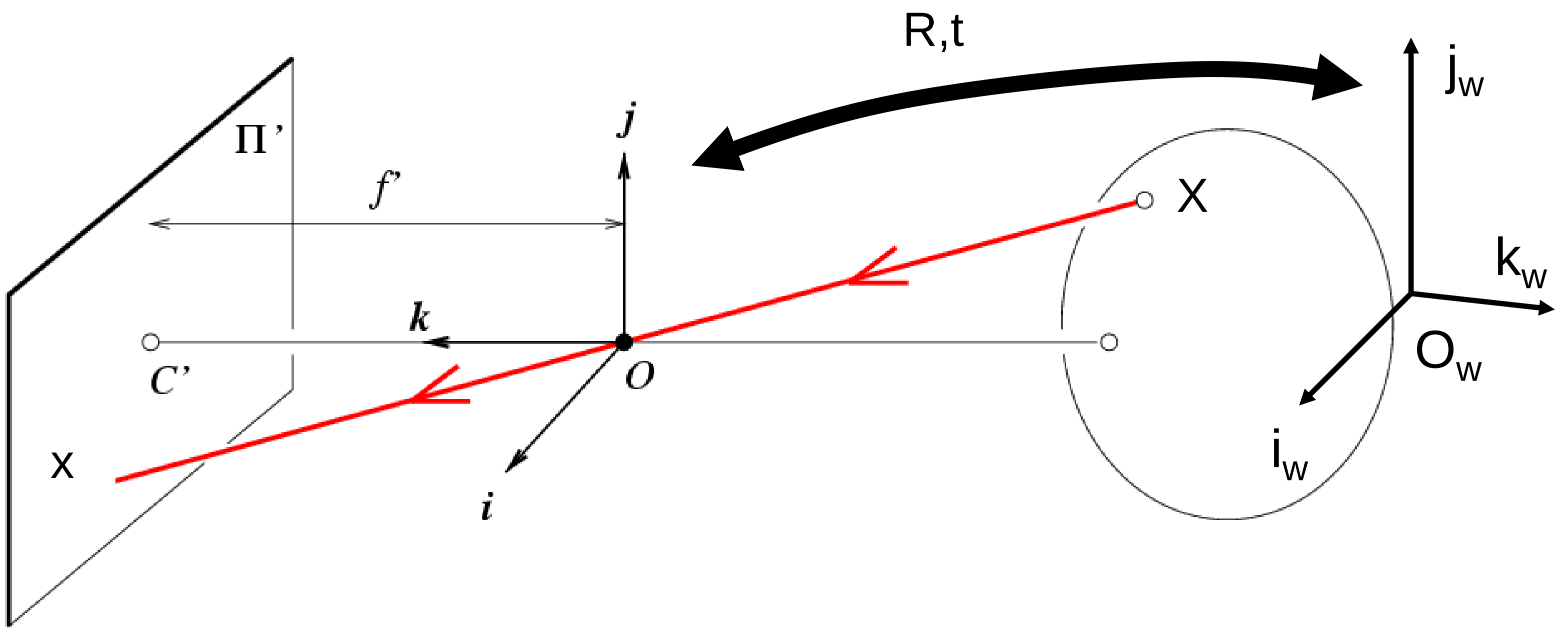
$$k\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{kx}{kw} \\ \frac{ky}{kw} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}$$

Homogeneous Coordinates — Cartesian Coordinates

Point in Cartesian is ray in Homogeneous

Slide Credit: Savarese

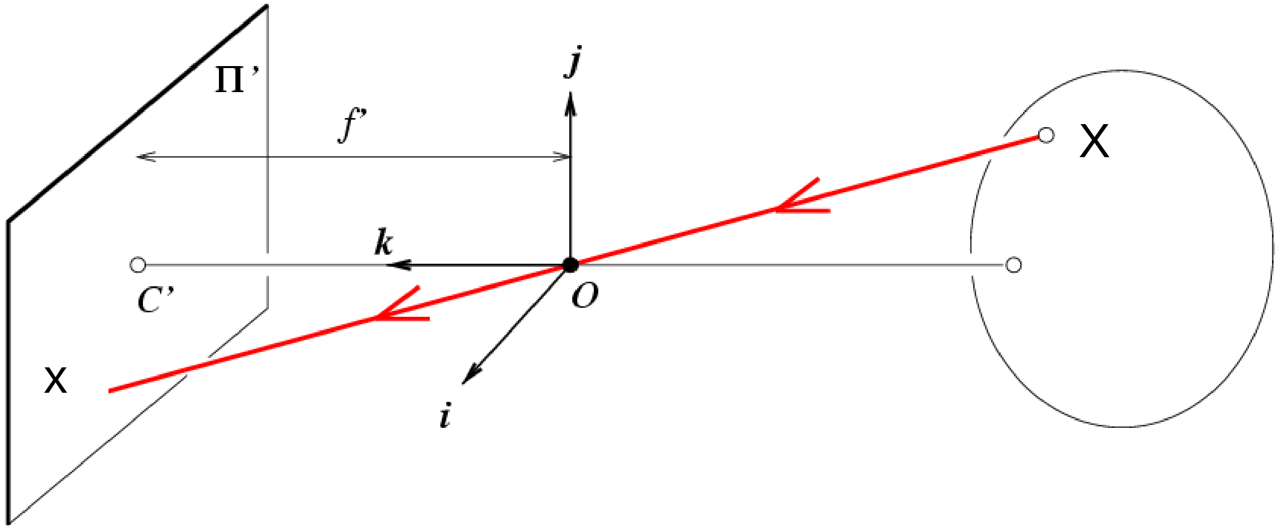
Projection matrix



$$\mathbf{x} = \mathbf{K}\left[\mathbf{R} \quad \mathbf{t}\right]\mathbf{X}$$

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

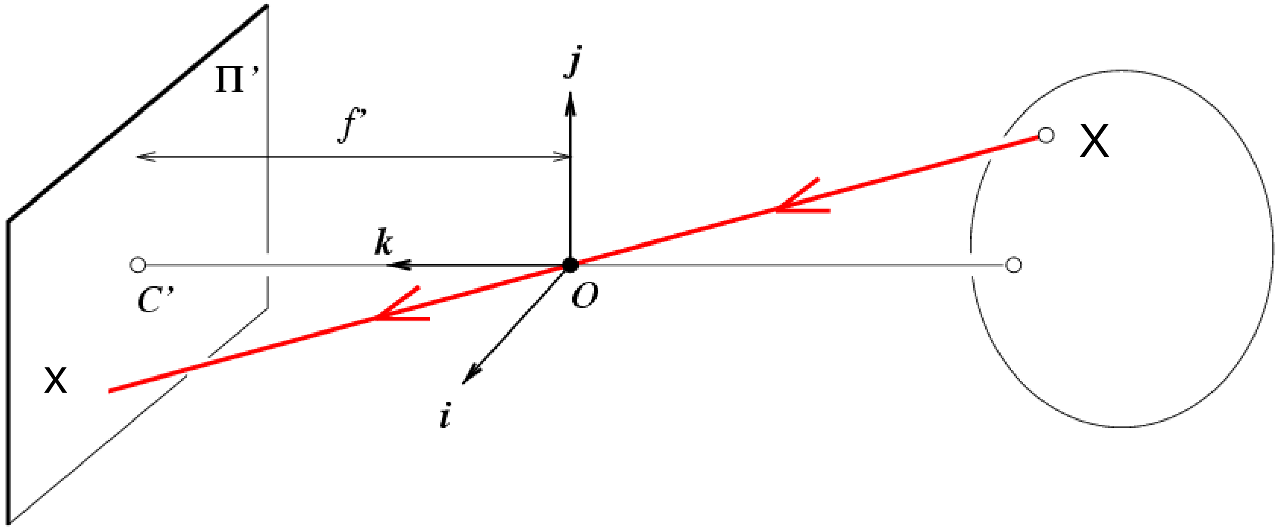
- No rotation
- Camera at (0,0,0)

K

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

Slide Credit: Savarese

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: center pixel is (0,0)

Intrinsic Assumptions
- Unit aspect ratio
- No skew

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

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

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

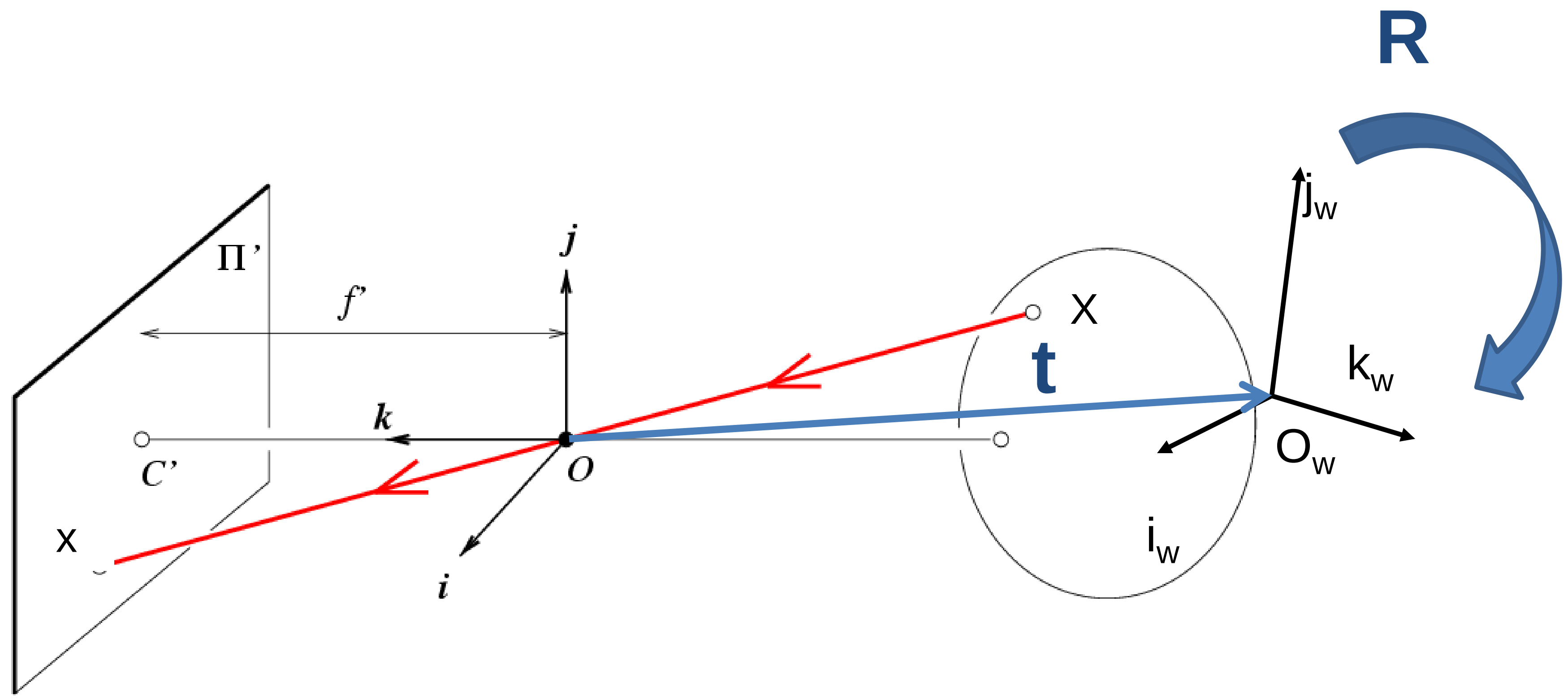
Intrinsic Assumptions

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

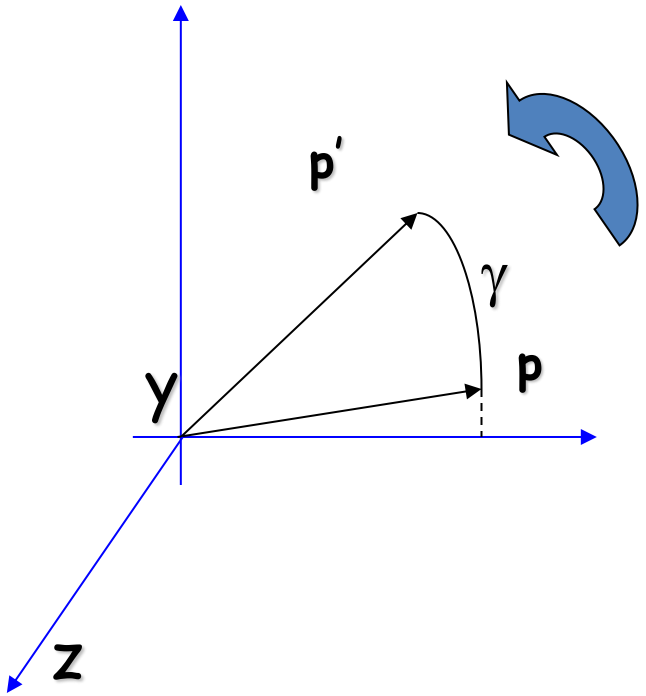
Extrinsic Assumptions
- No rotation

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

Slide Credit: Saverese

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

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

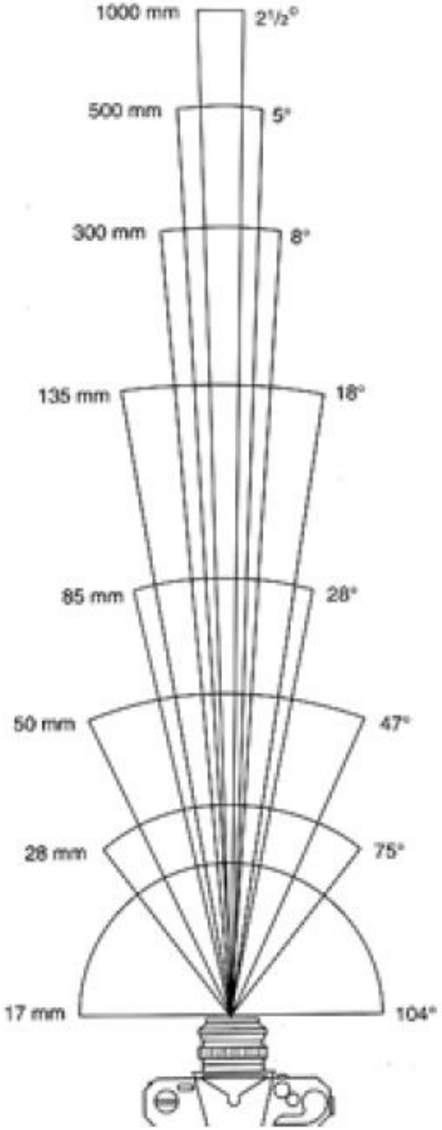
$$w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\alpha & s & u_0\\0 & \beta & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\r_{21} & r_{22} & r_{23} & t_y\\r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Degrees of freedom

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

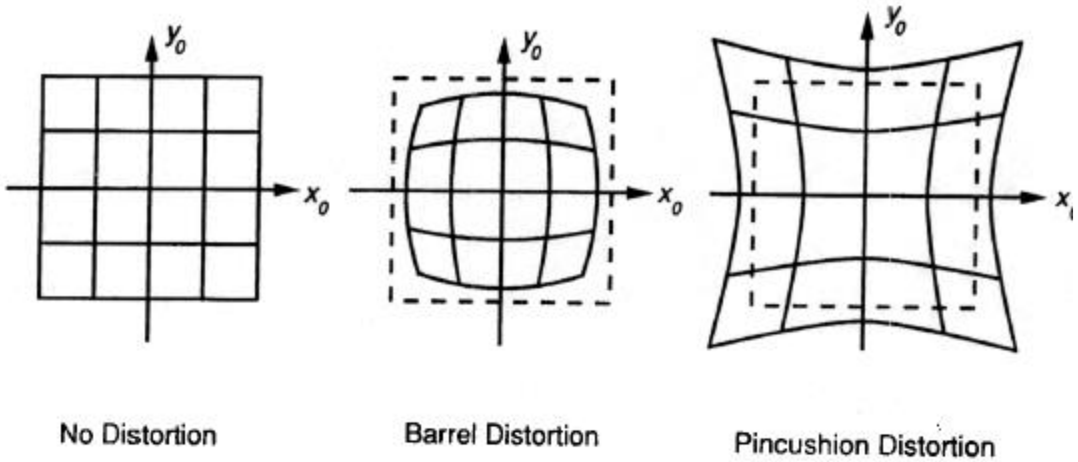
$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \overset{5}{\begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{6}{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Field of View (Zoom, focal length)



From London and Upton

Beyond Pinholes: Radial Distortion

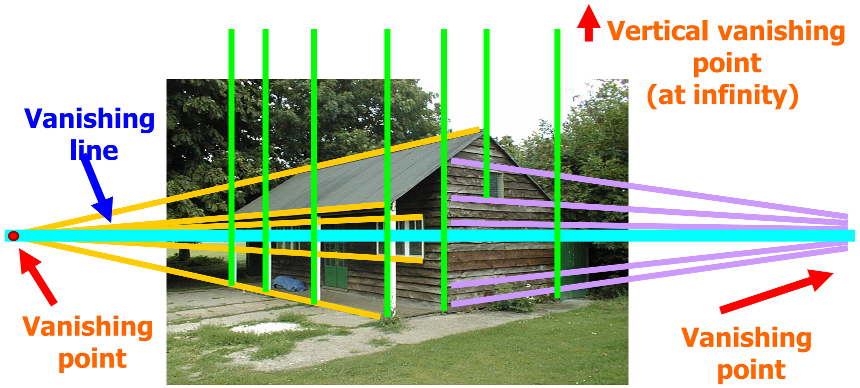


Corrected Barrel Distortion

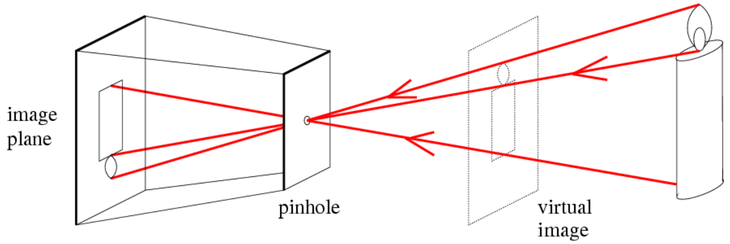
Image from Martin Habbecke

Things to remember

- Vanishing points and vanishing lines



- Pinhole camera model and camera projection matrix



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

- Homogeneous coordinates

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

Reminder: read your book

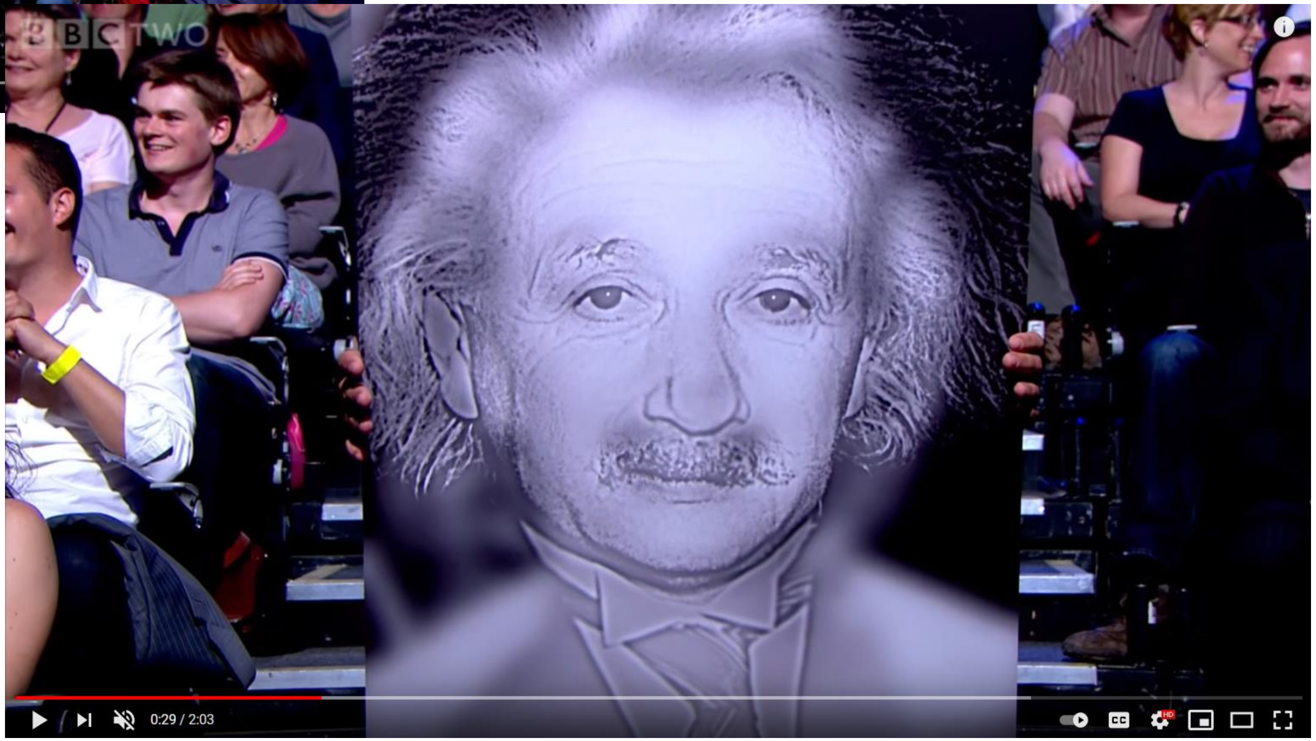
- Lectures have assigned readings
- Szeliski 2.1 and especially 2.1.4 cover the geometry of image formation

Image Filtering

Computer Vision

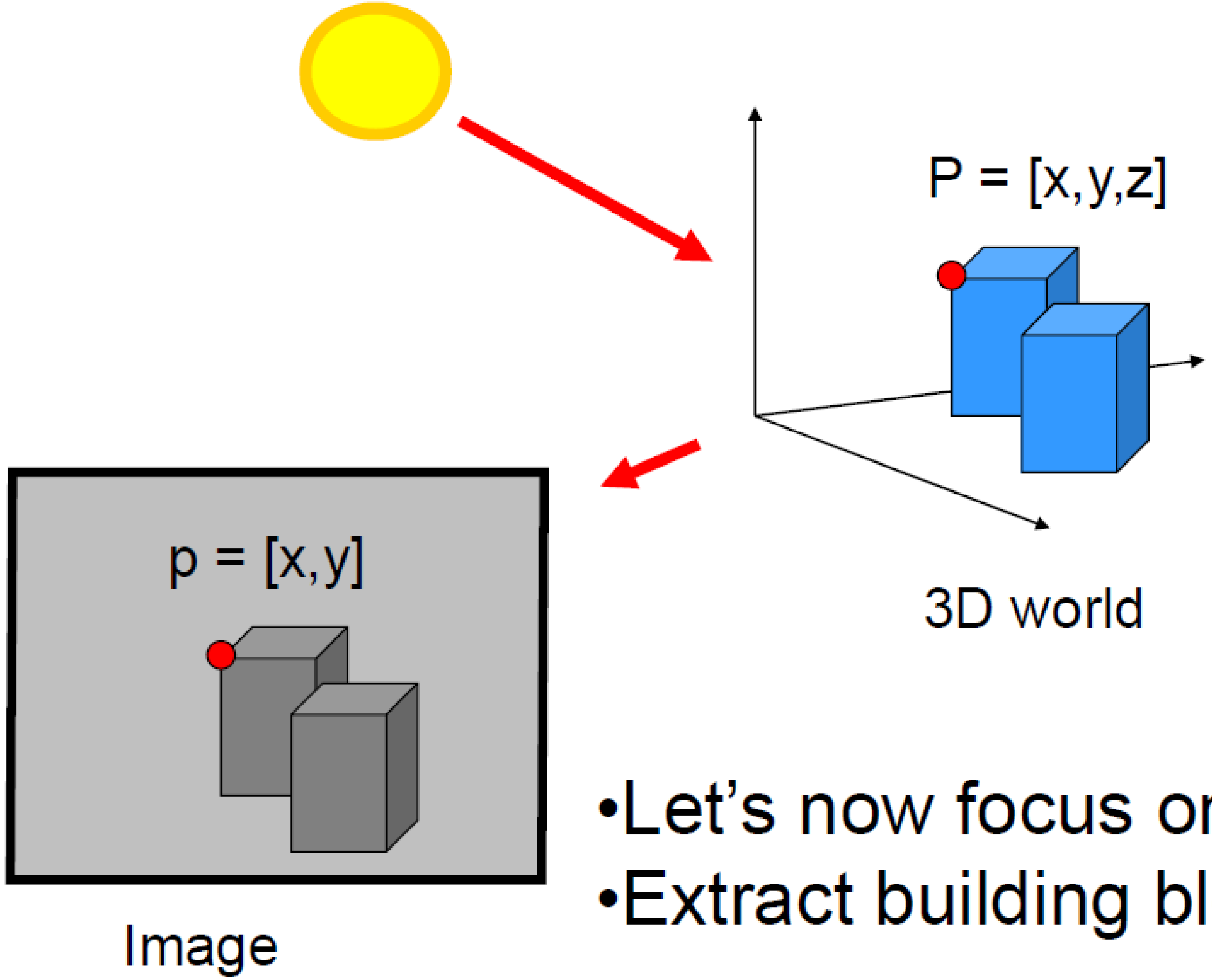
James Hays

Many slides by Derek Hoiem



BBC Clip: https://www.youtube.com/watch/OlumoQ05gS8

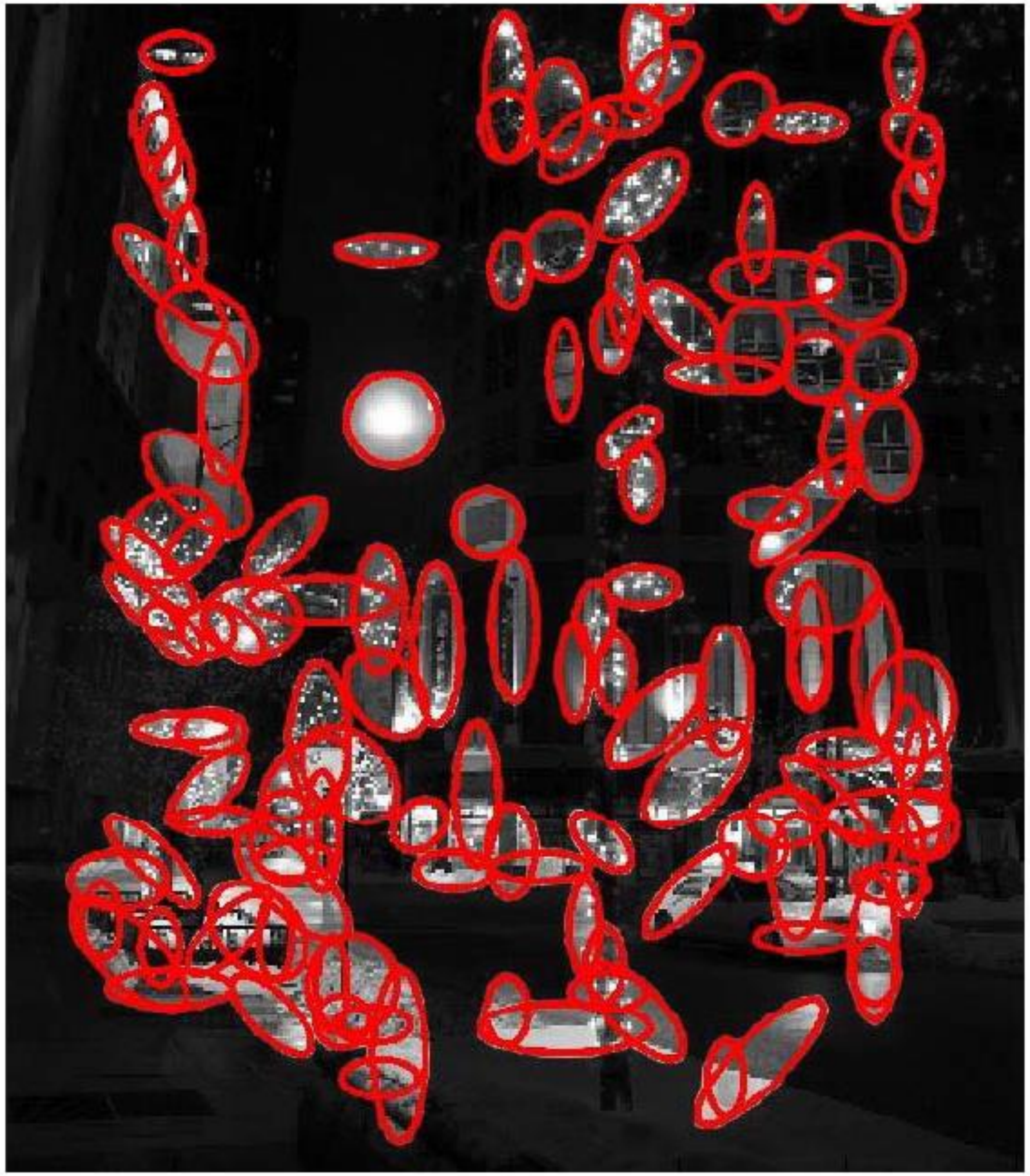
From the 3D to 2D



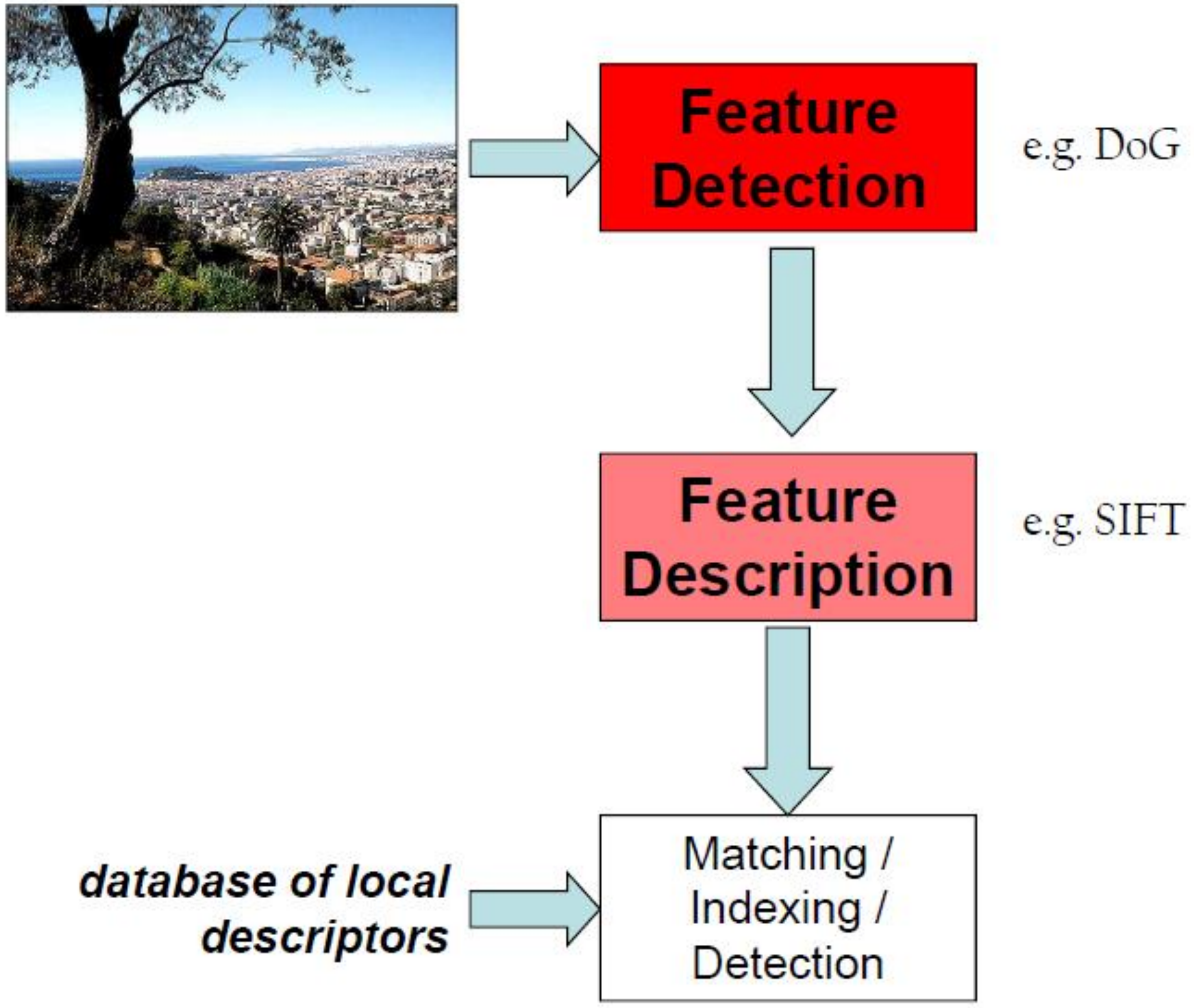
- Let's now focus on 2D
- Extract building blocks

Slide credit Fei Fei Li

Extract useful building blocks

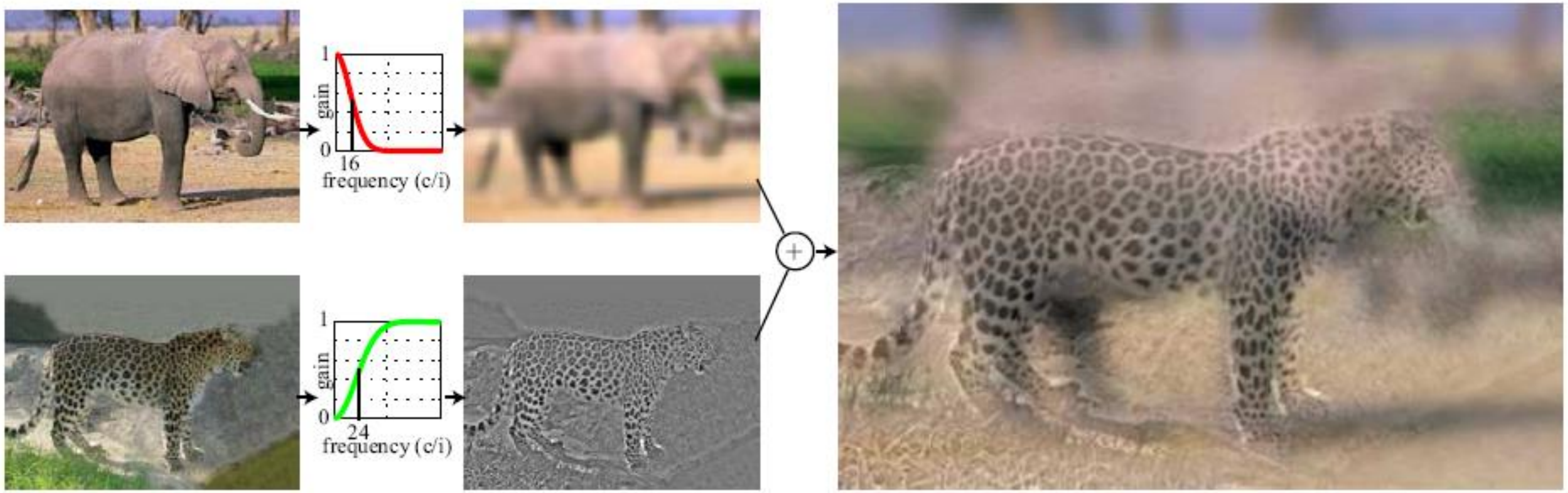


Slide credit Fei Fei Li



Slide credit Fei Fei Li

Hybrid Images



- A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

Upcoming classes: two views of filtering

- Image filters in spatial domain
  - Filter is a mathematical operation of a grid of numbers
  - Smoothing, sharpening, measuring texture

- Image filters in the frequency domain
  - Filtering is a way to modify the frequencies of images
  - Denoising, sampling, image compression

Image filtering (or convolution)

- Image filtering: compute function of local neighborhood at each position

- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Deep Convolutional Networks

Example: box filter

$$g[\cdot\,,\cdot\,]$$

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

Slide credit: David Lowe (UBC)

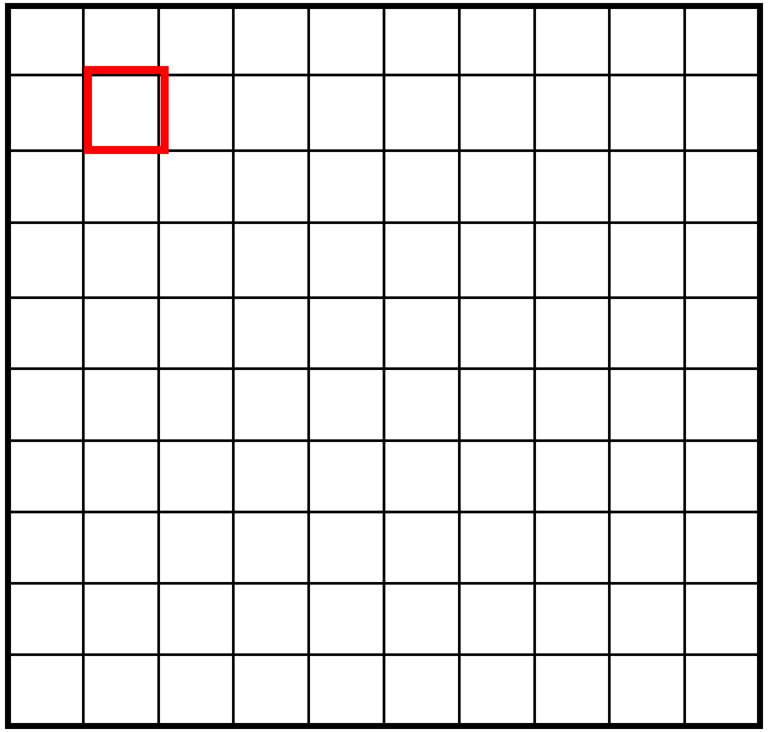
Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

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

$h[.,.]$



$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k, n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

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

$h[.,.]$

	0	10							

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

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

$h[.,.]$

	0	10	20						

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$f[.,.]$$

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

$$h[.,.]$$

	0	10	20	30					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot\,,\cdot\,]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot\,,\cdot\,]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
				?					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

$f[.,.]$

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

$h[.,.]$

	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	

$$h[m,n]=\sum_{k,l} g[k,l]\; f[m+k,n+l]$$

Credit: S. Seitz

Box Filter

What does it do?

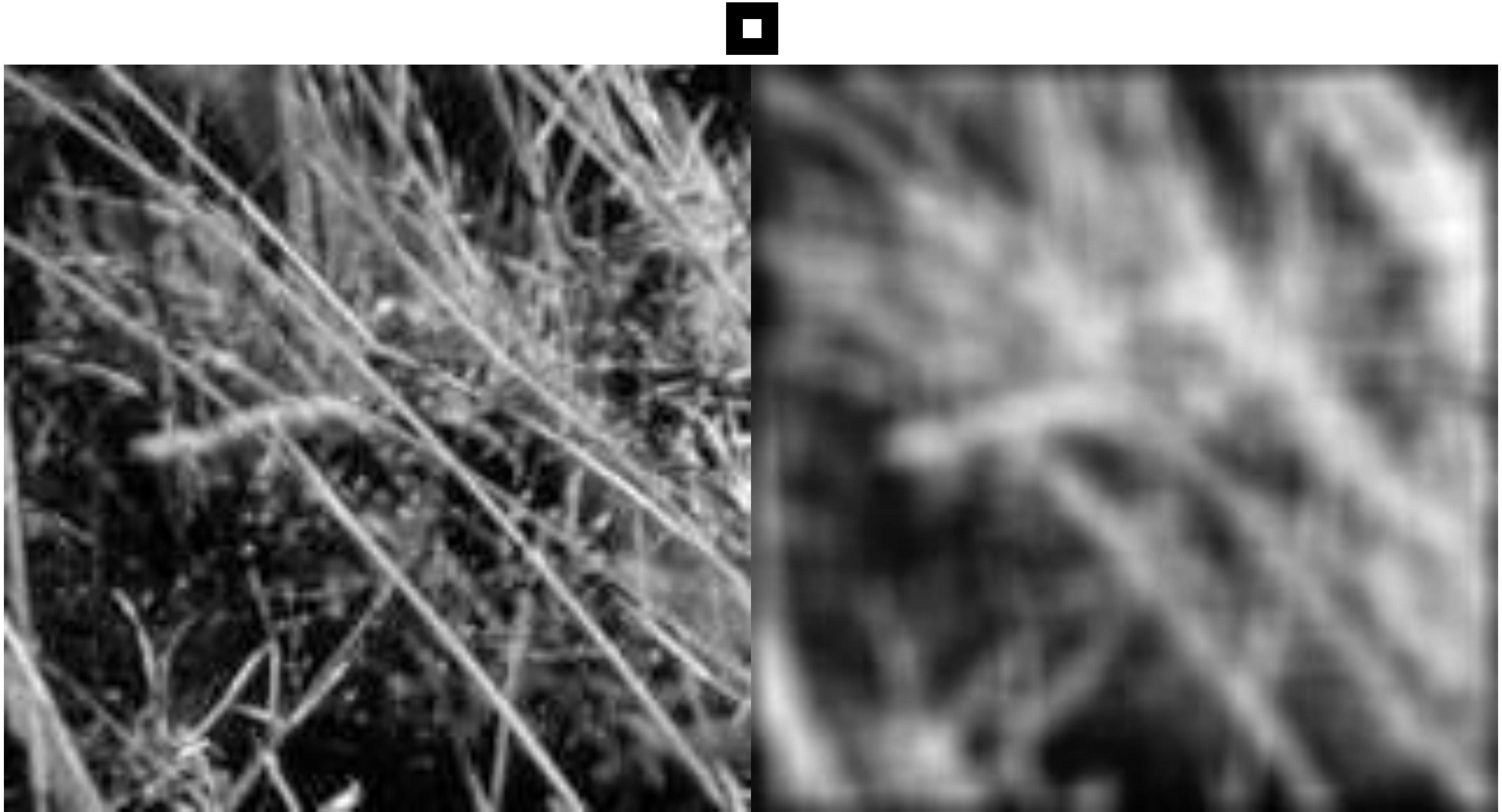
- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

$$g[\cdot\,,\cdot\,]$$

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

Slide credit: David Lowe (UBC)

Smoothing with box filter



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

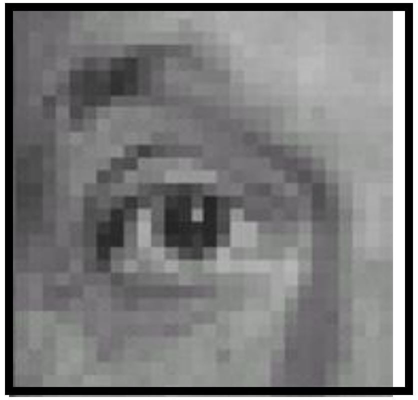
?

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

Practice with linear filters

Original

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

?

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

Original

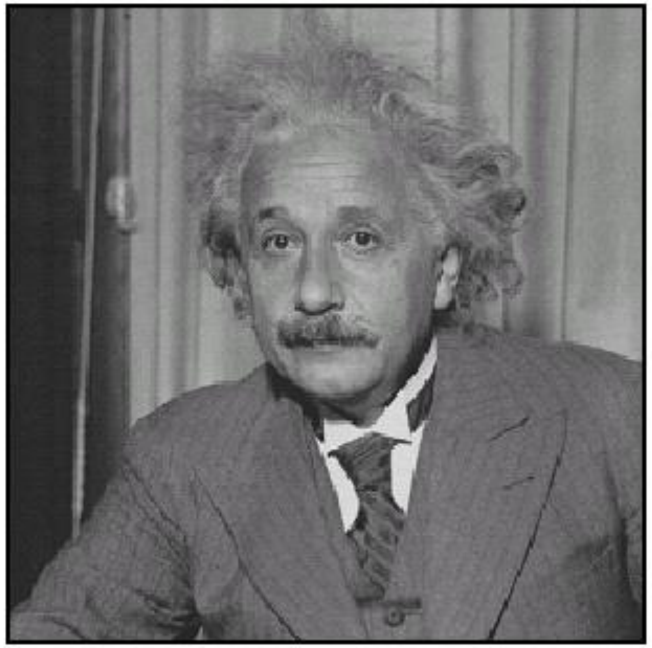
$$\begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Sharpening filter

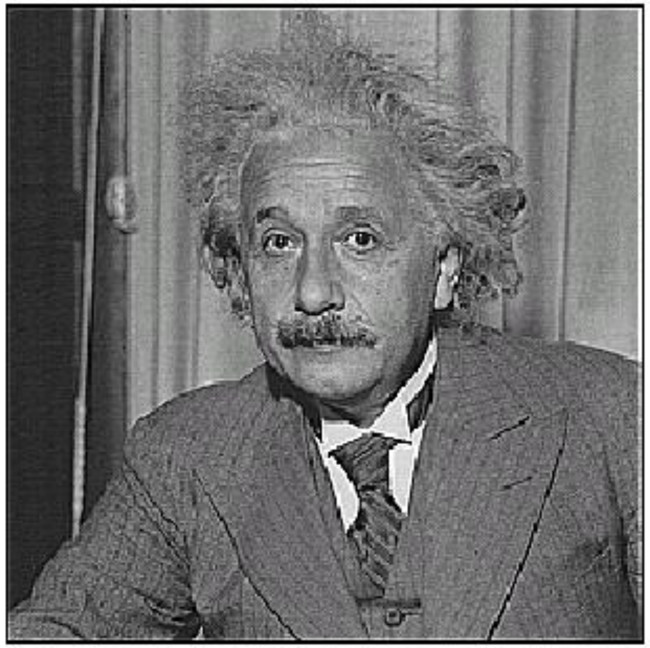
- Accentuates differences with local average

Source: D. Lowe

Sharpening



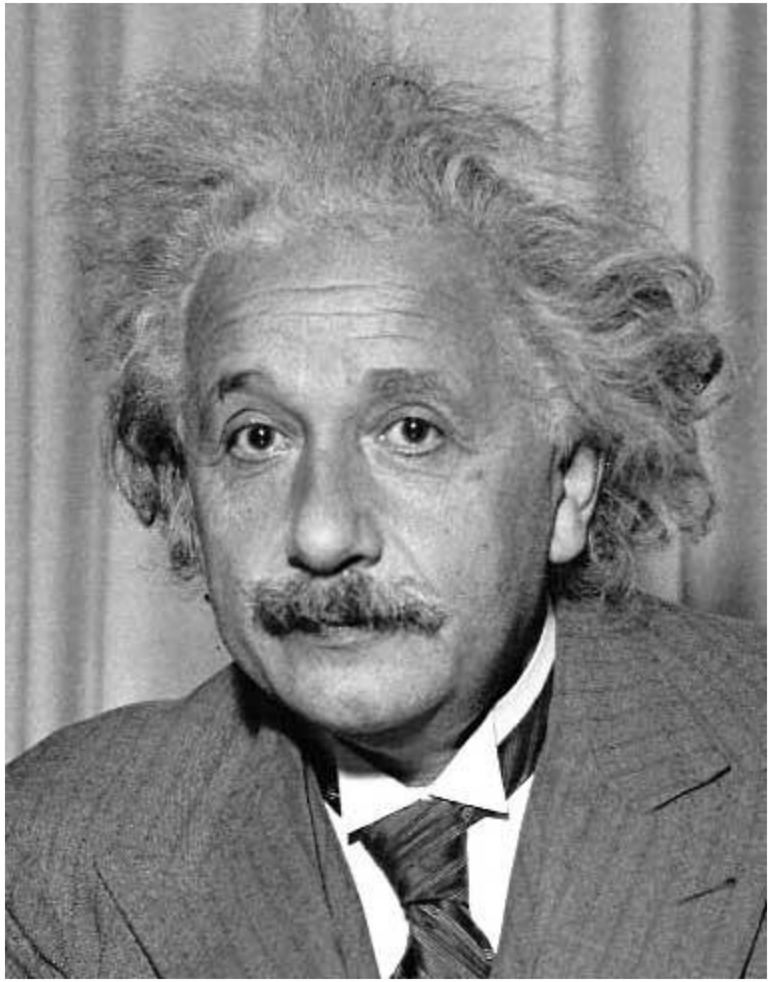
before



after

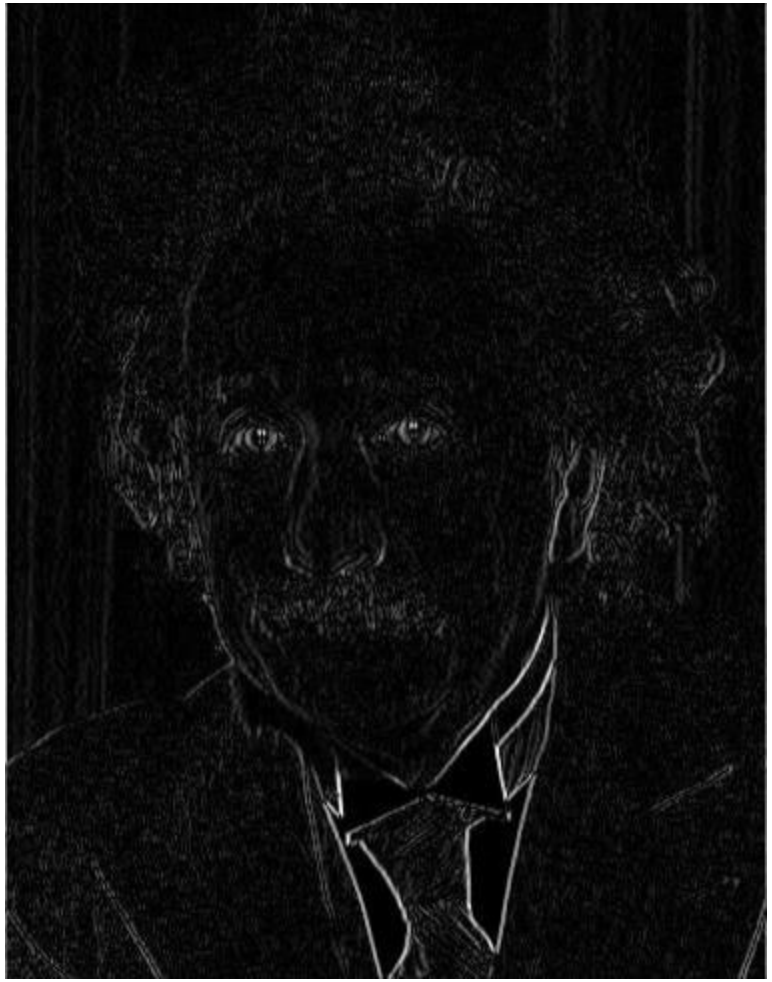
Source: D. Lowe

Other filters



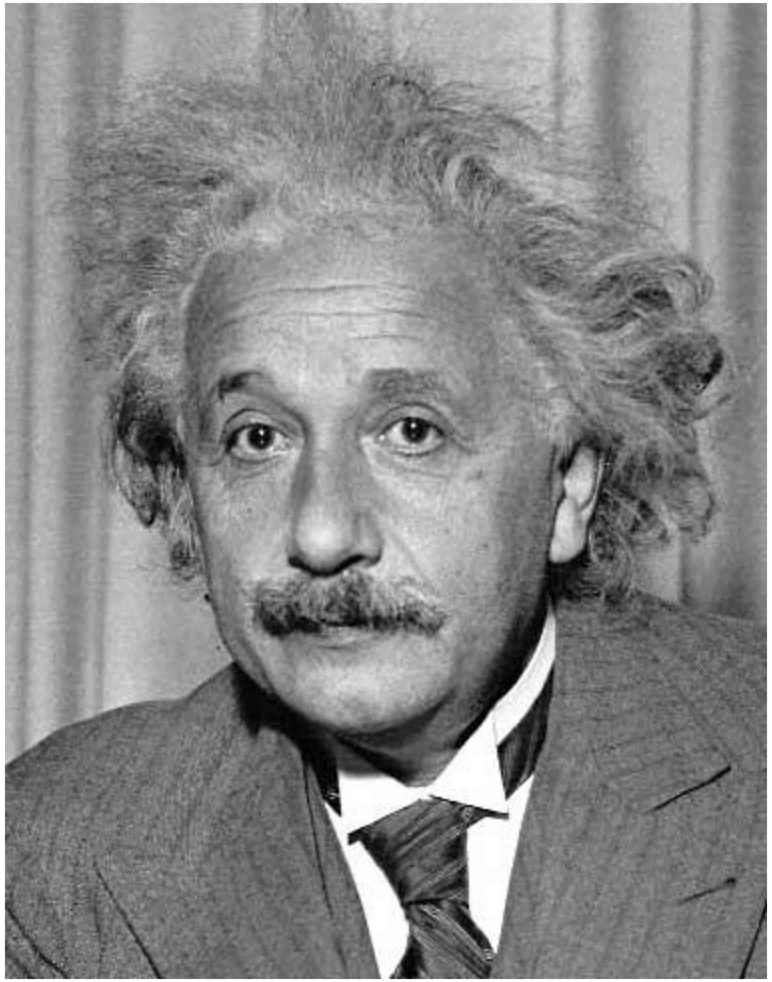
1	0	-1
2	0	-2
1	0	-1

Sobel



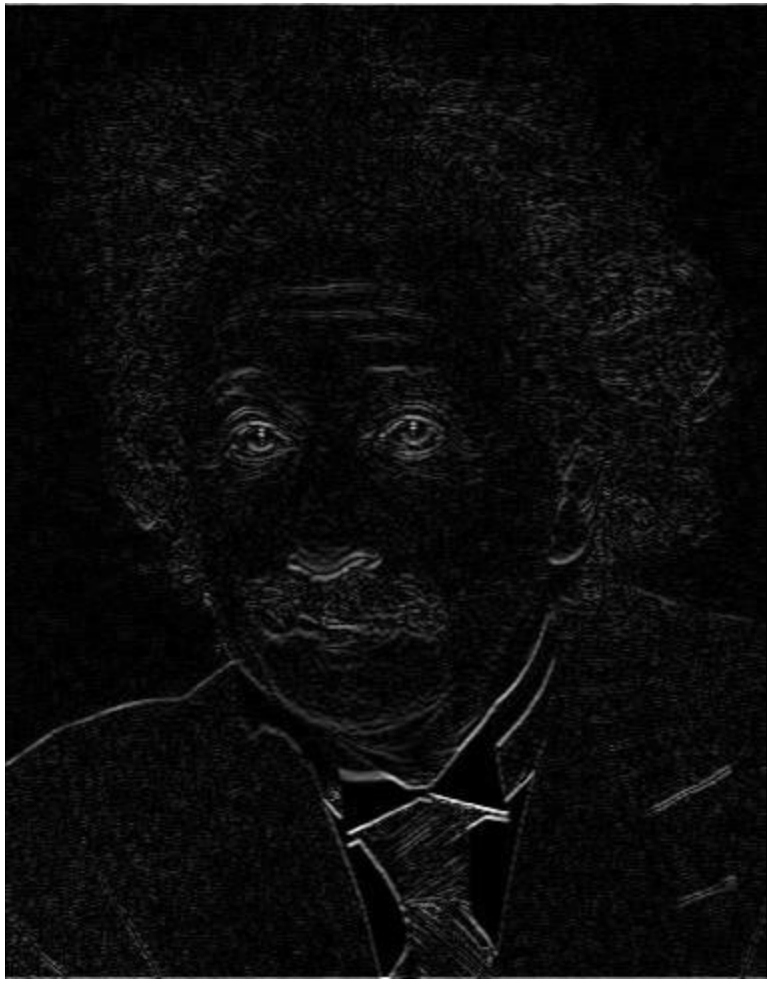
Vertical Edge
(absolute value)

Other filters



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

Sobel



Horizontal Edge
(absolute value)

Filtering vs. Convolution

```
f=filter, size k x l
I=image,  size m x n
```

- 2d filtering

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

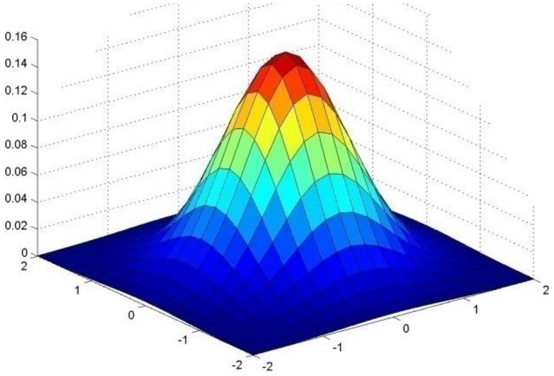
- 2d convolution

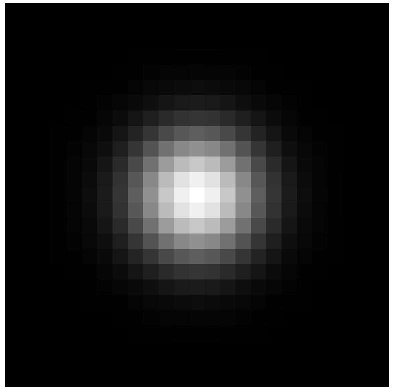
$$h[m,n] = \sum_{k,l} f[k,l]\, I[m-k, n-l]$$

In Python you can use https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html

Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness





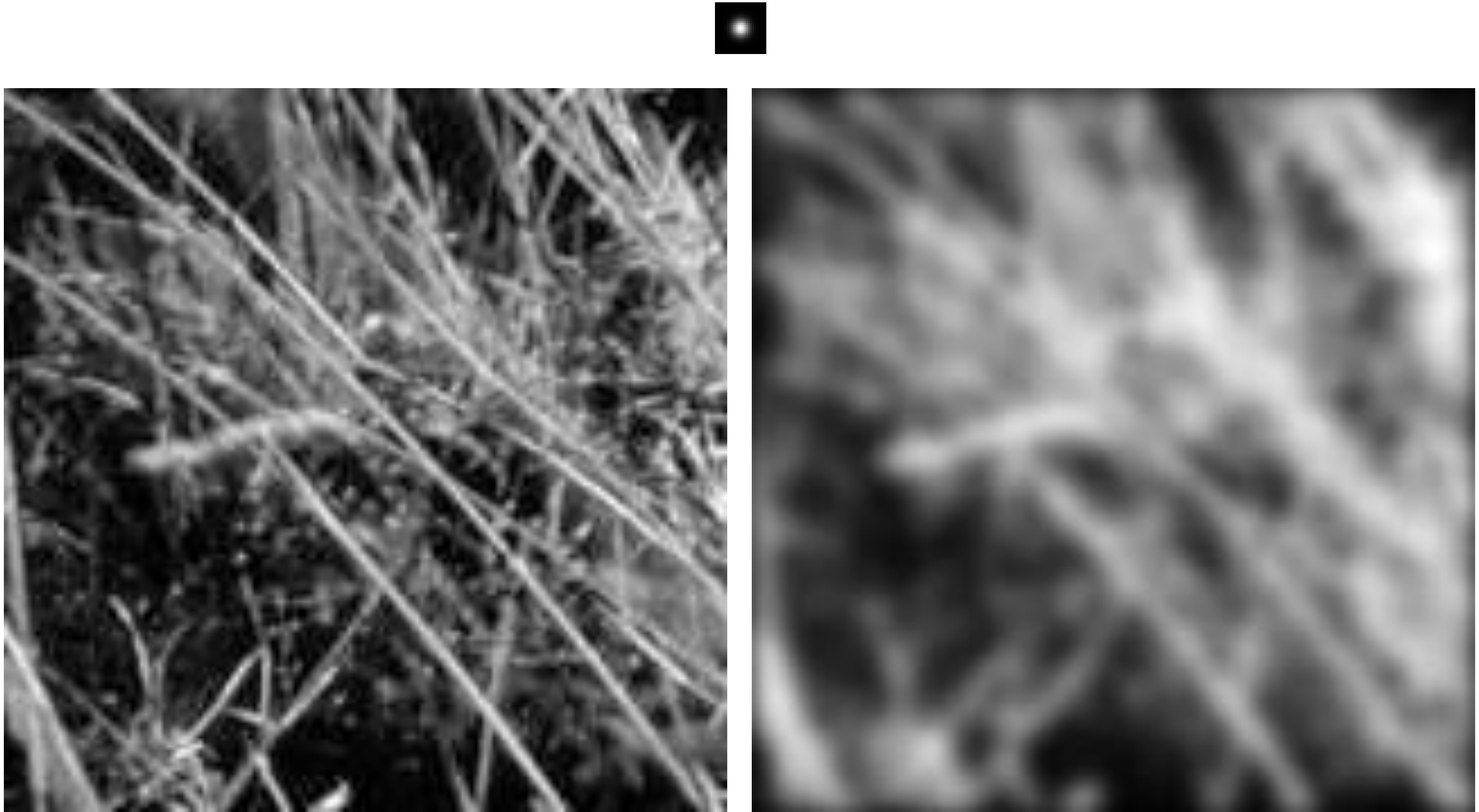
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1

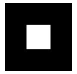
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

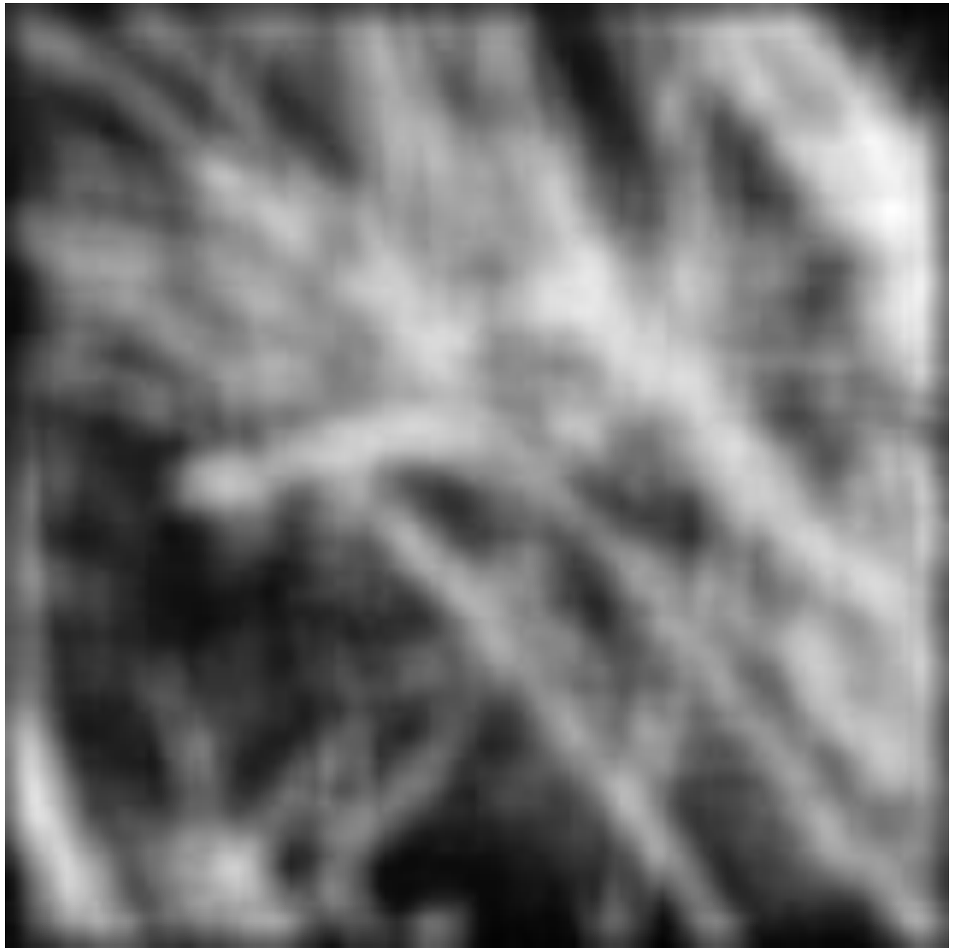
Slide credit: Christopher Rasmussen

Smoothing with Gaussian filter



Smoothing with box filter





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 σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  – Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

$$
\begin{aligned}
G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2 + y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right)
\end{aligned}
$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

Separability example

2D convolution
(center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

The filter factors
into a product of 1D
filters:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Perform convolution
along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

Followed by convolution
along the remaining column:

Source: K. Grauman

Separability

- Why is separability useful in practice?

Some practical matters

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 σ

Practical matters

- 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

Next class: Light and Color and Thinking in Frequency