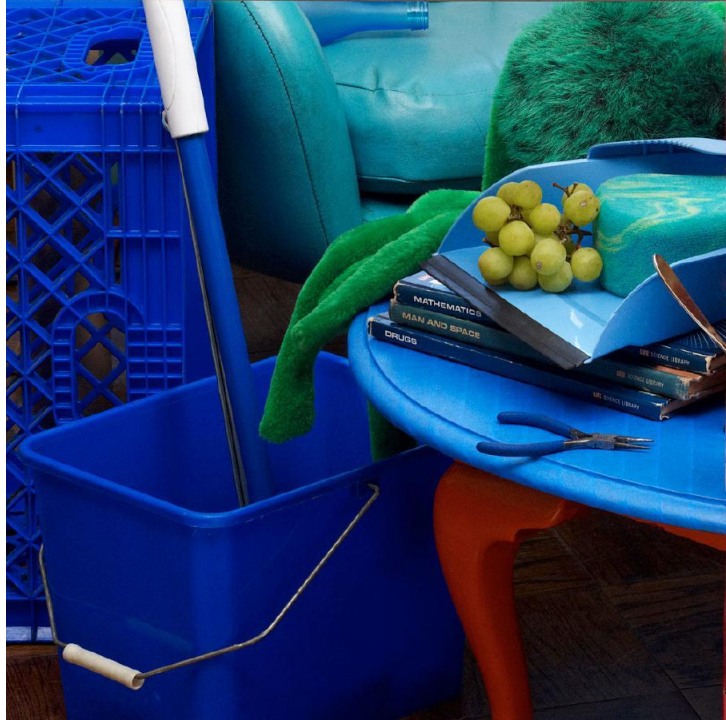










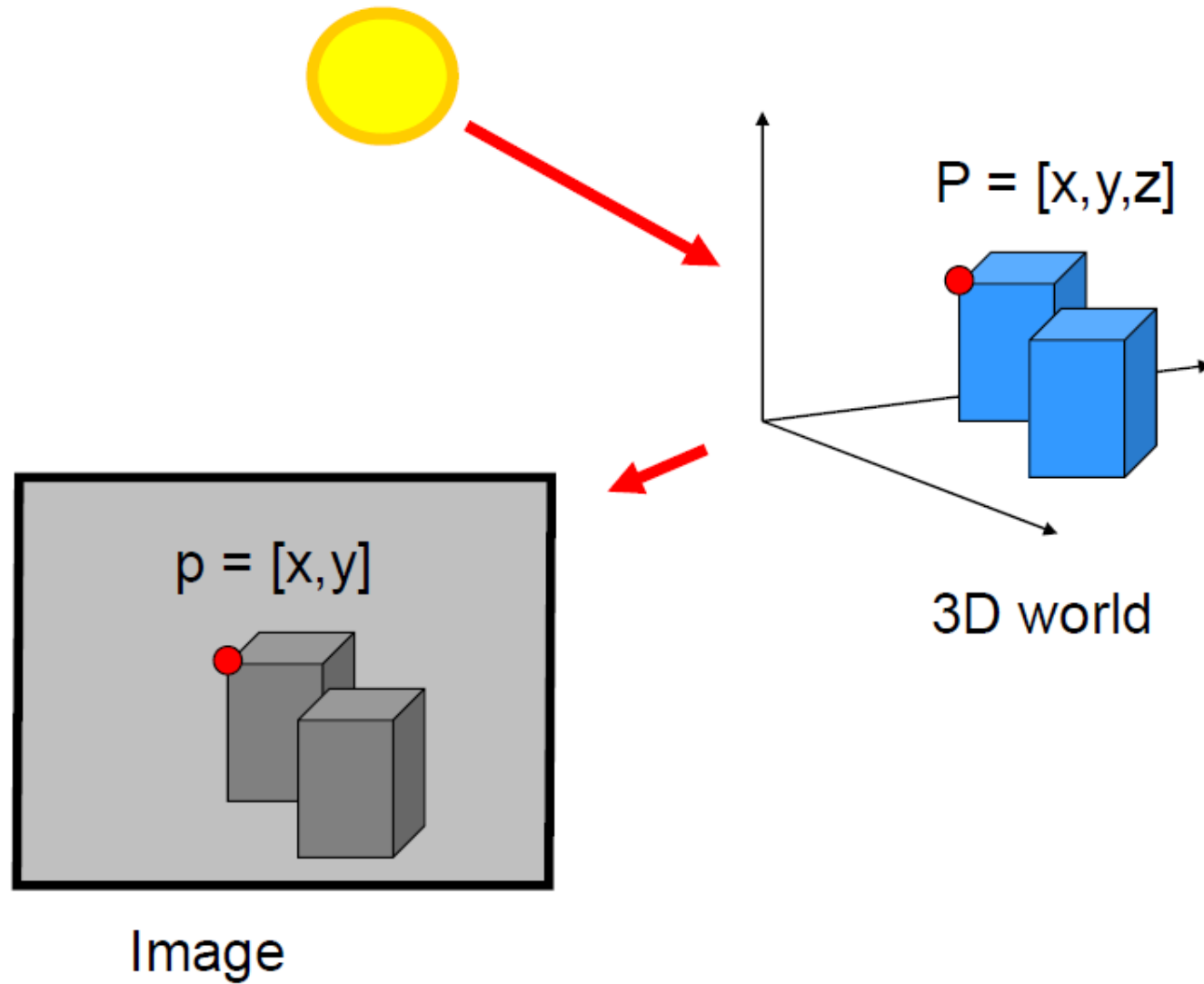
Welcome back!

- Optional project 0 is out.
- Project 1 will be out soon.
- Today
 - Image projection - Szeliski 2.1 (2.1.4 in particular)
 - Filtering – Szeliski 3.2

Image formation

2.1	Geometric primitives and transformations	36
2.1.1	2D transformations	39
2.1.2	3D transformations	43
2.1.3	3D rotations	45
2.1.4	3D to 2D projections	50
2.1.5	Lens distortions	62
2.2	Photometric image formation	64
2.2.1	Lighting	65
2.2.2	Reflectance and shading	66
2.2.3	Optics	73
2.3	The digital camera	78
2.3.1	Sampling and aliasing	82
2.3.2	Color	85
2.3.3	Compression	97
2.4	Additional reading	98
2.5	Exercises	99

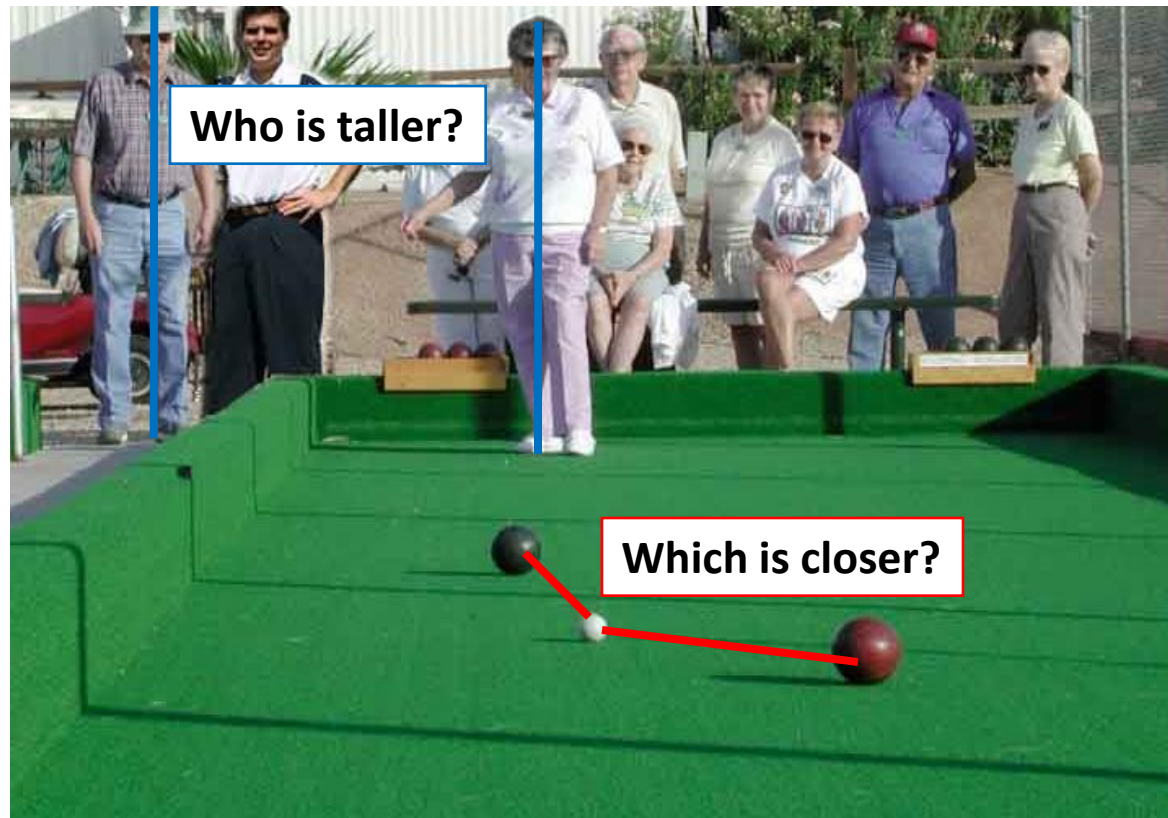
From the 3D to 2D



Projective Geometry

What is lost?

- Length



Length and area are not preserved

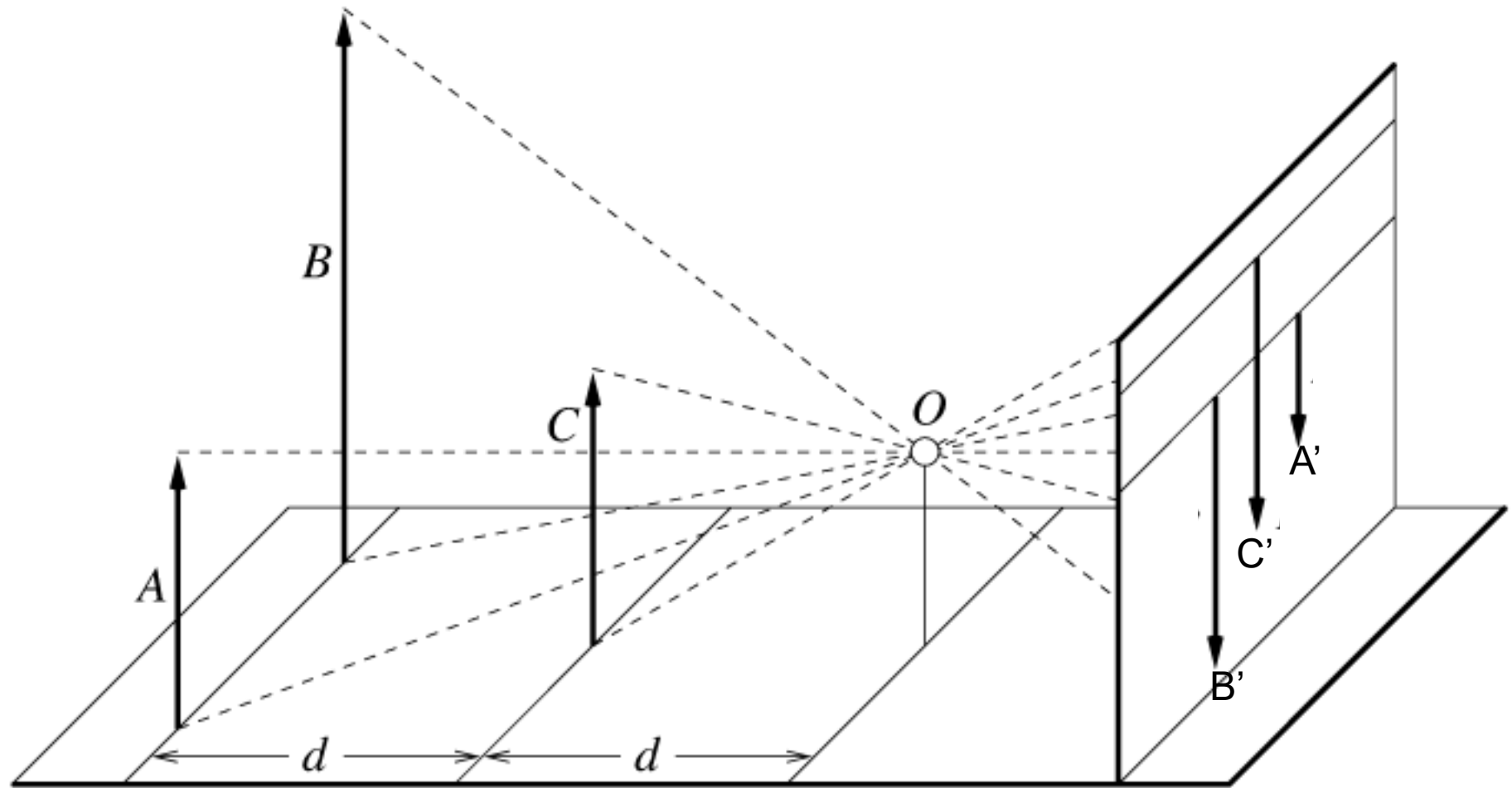
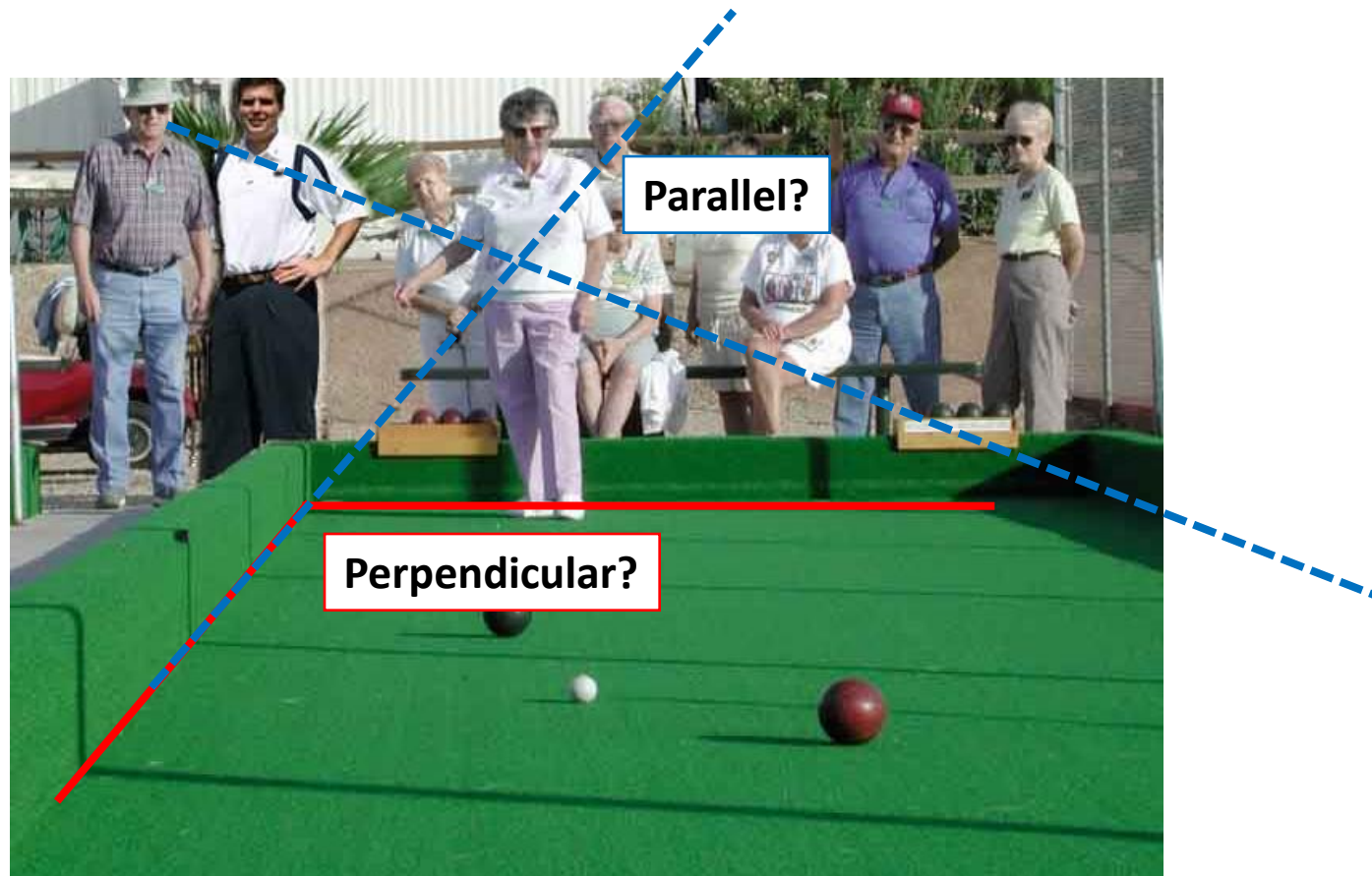


Figure by David Forsyth

Projective Geometry

What is lost?

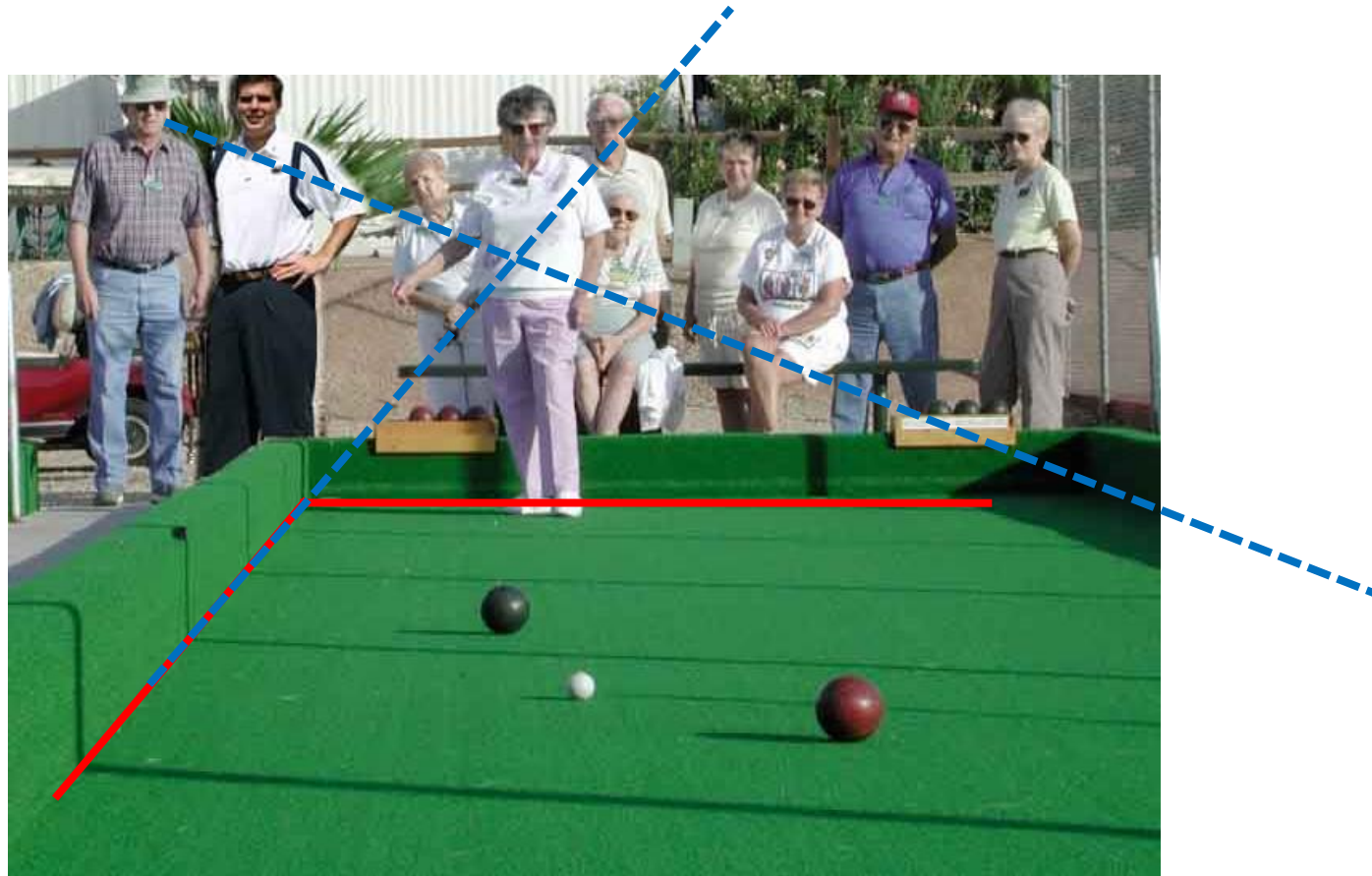
- Length
- Angles



Projective Geometry

What is preserved?

- Straight lines are still straight

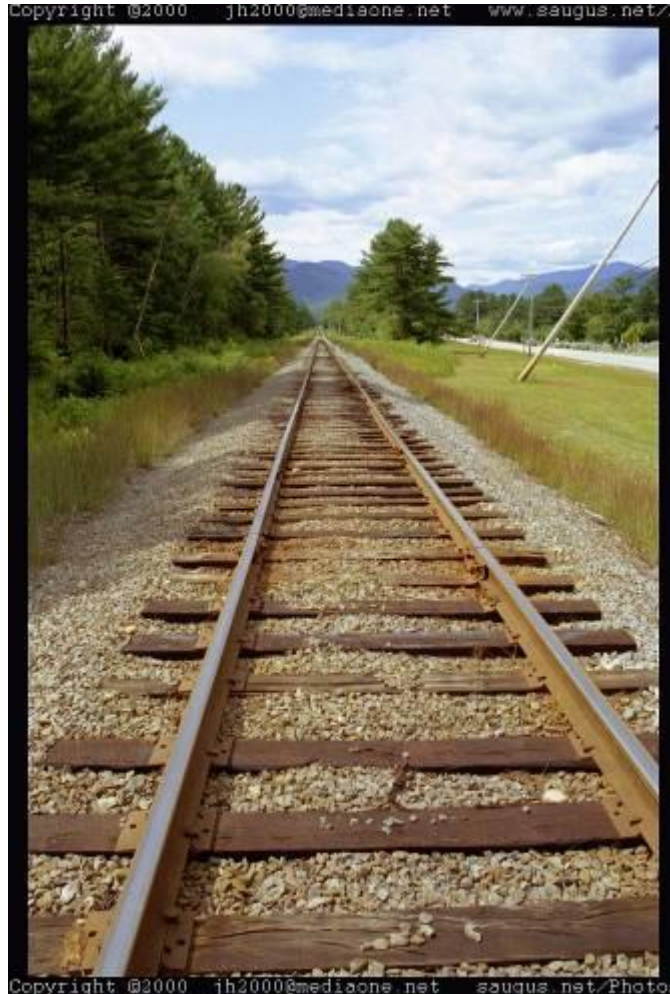


The pinhole camera model preserves straight lines, but real cameras might not

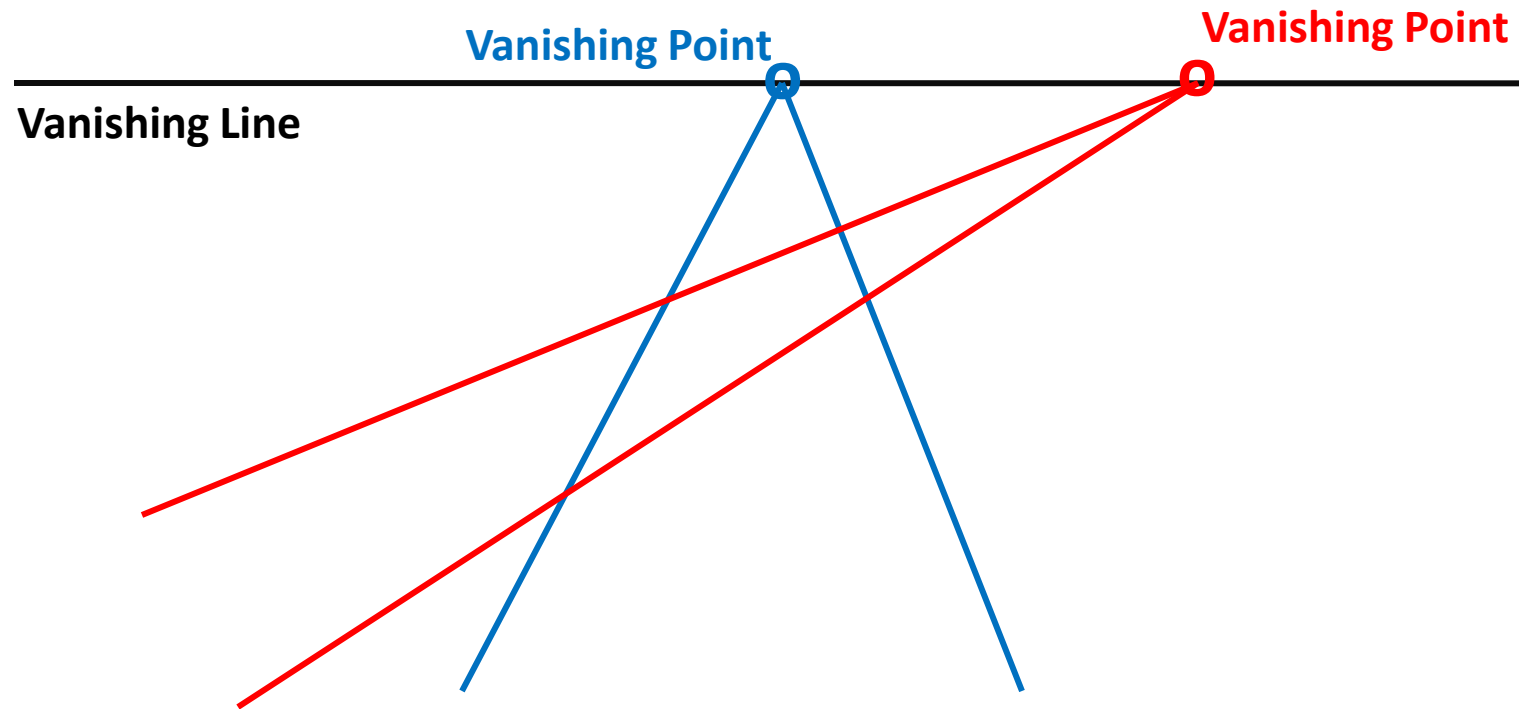


Vanishing points and lines

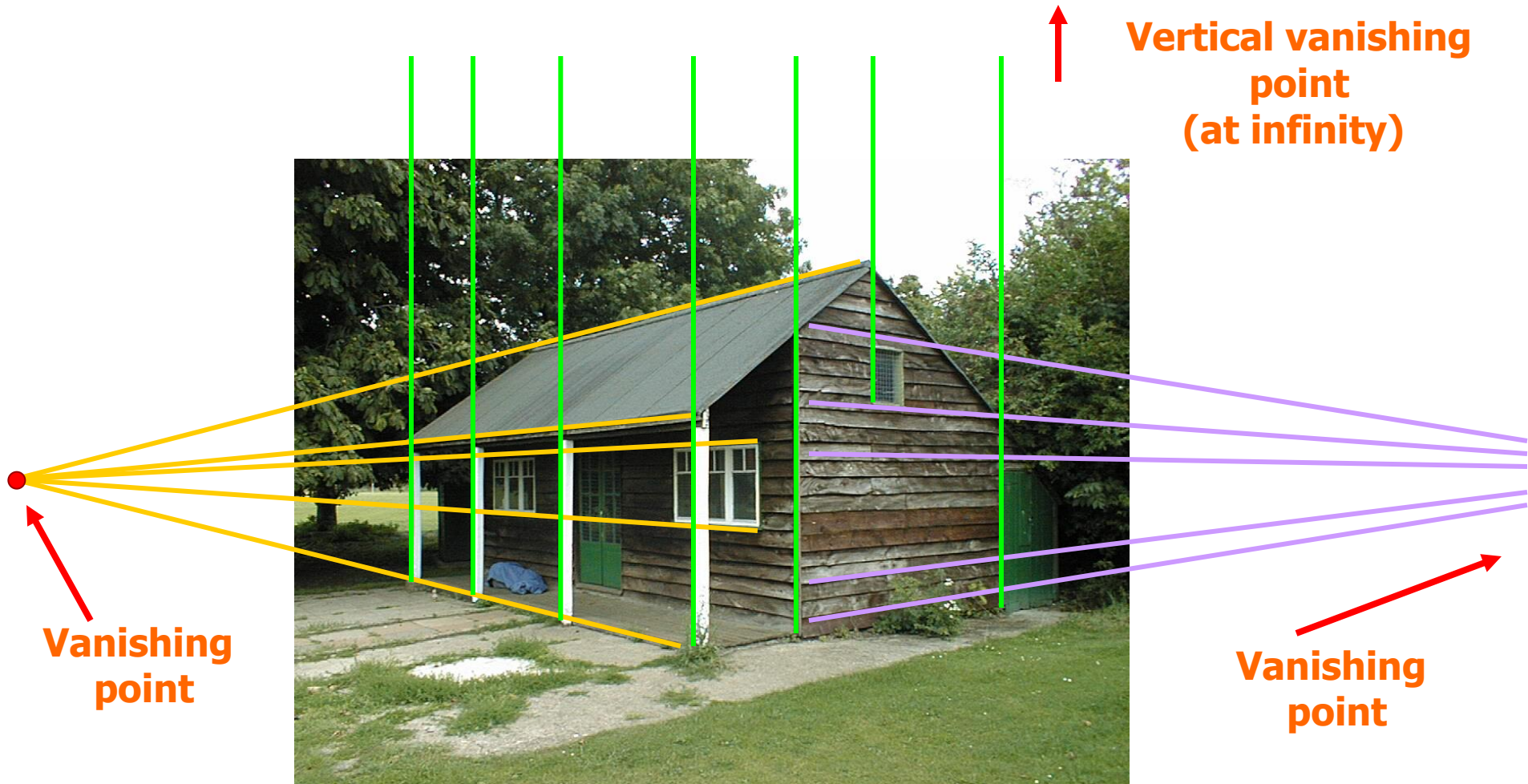
Parallel lines in the world intersect in the image at a “vanishing point”



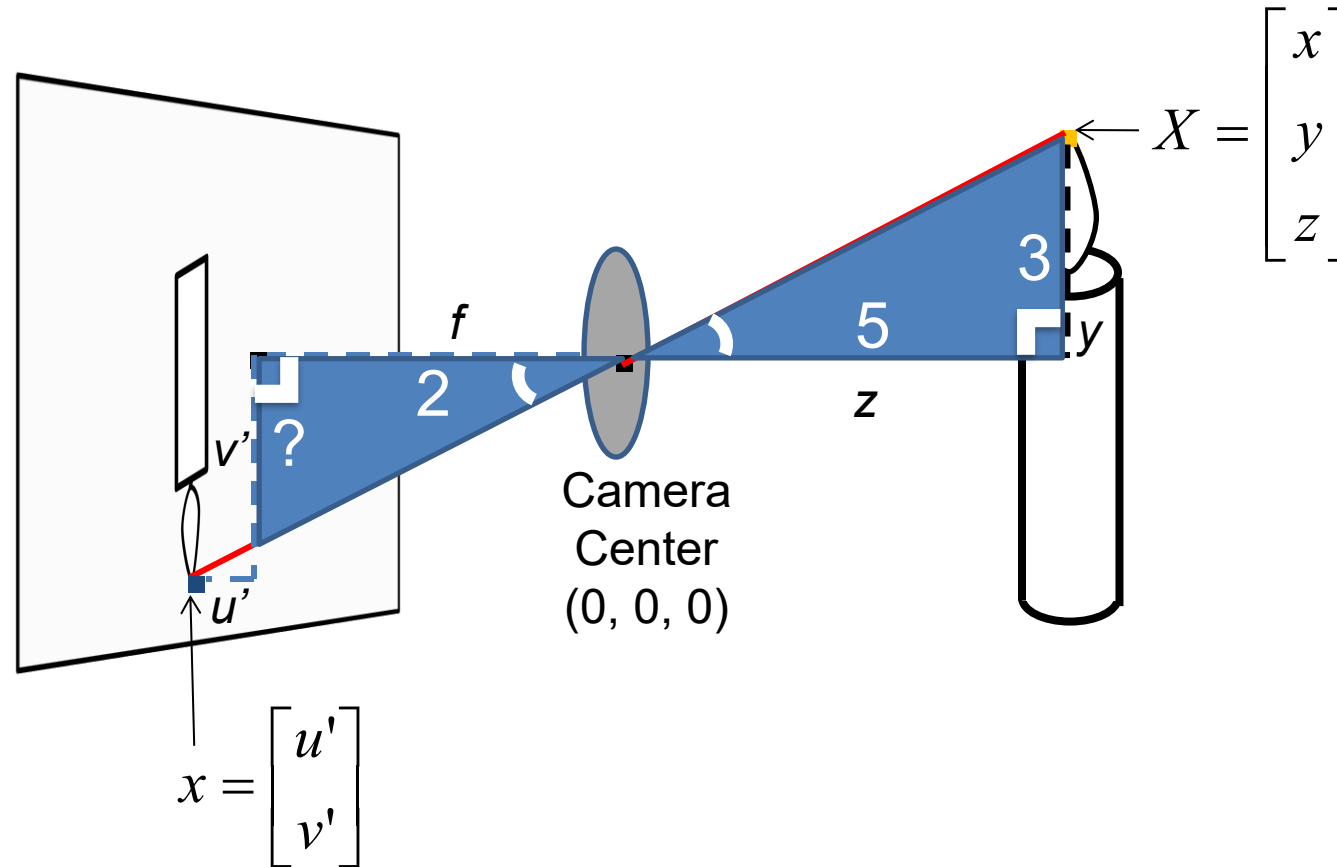
Vanishing points and lines



Vanishing points and lines



Projection: world coordinates $\rightarrow$ image coordinates



If $x = 2$, $y = 3$, $z = 5$, and $f = 2$

What are u' and v' ?

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

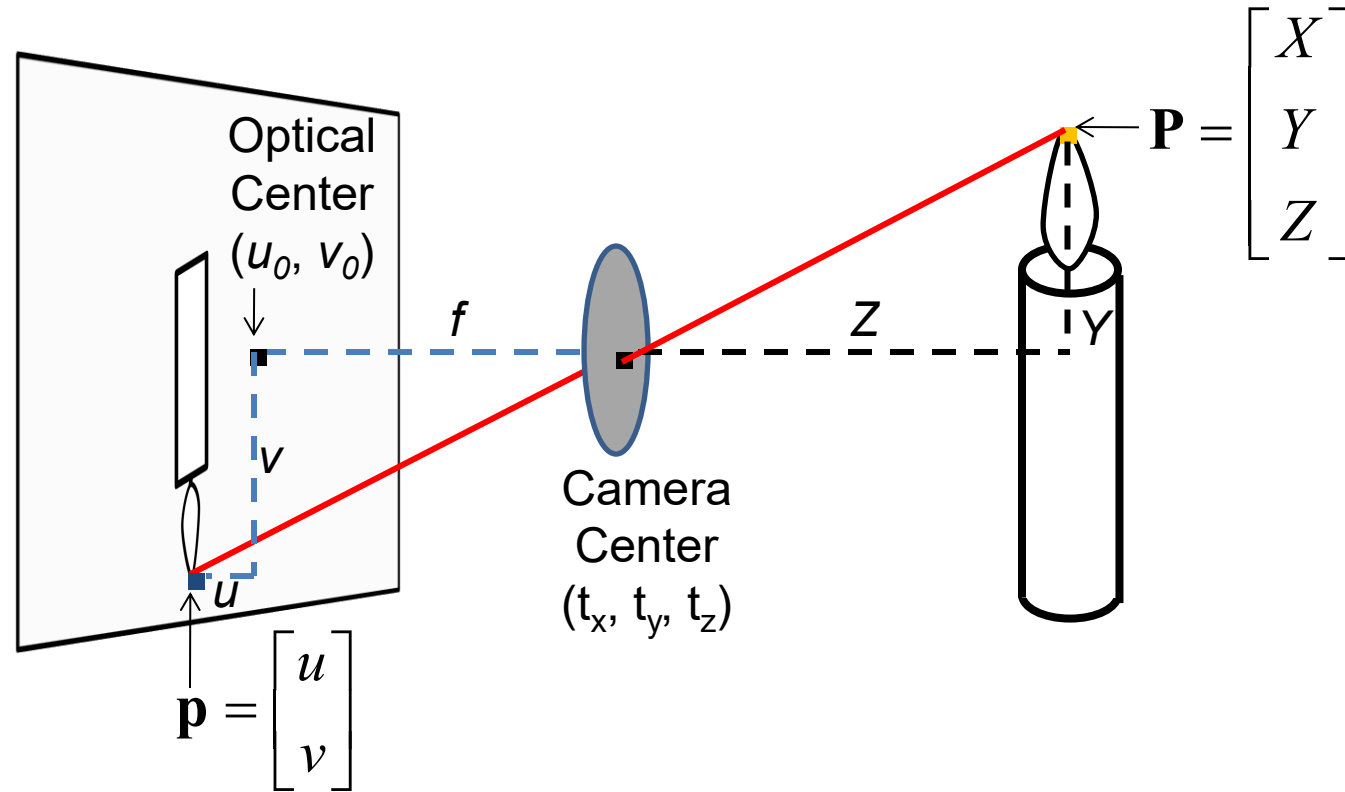
$$u' = -x * \frac{f}{z}$$

$$u' = -2 * \frac{2}{5}$$

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

$$v' = -3 * \frac{2}{5}$$

Projection: world coordinates $\rightarrow$ image coordinates



How do we handle the general case?

Interlude: why does this matter?

Relating multiple views



Photo Tourism

Exploring photo collections in 3D

Microsoft



(a)



(b)



(c)

Photo Tourism

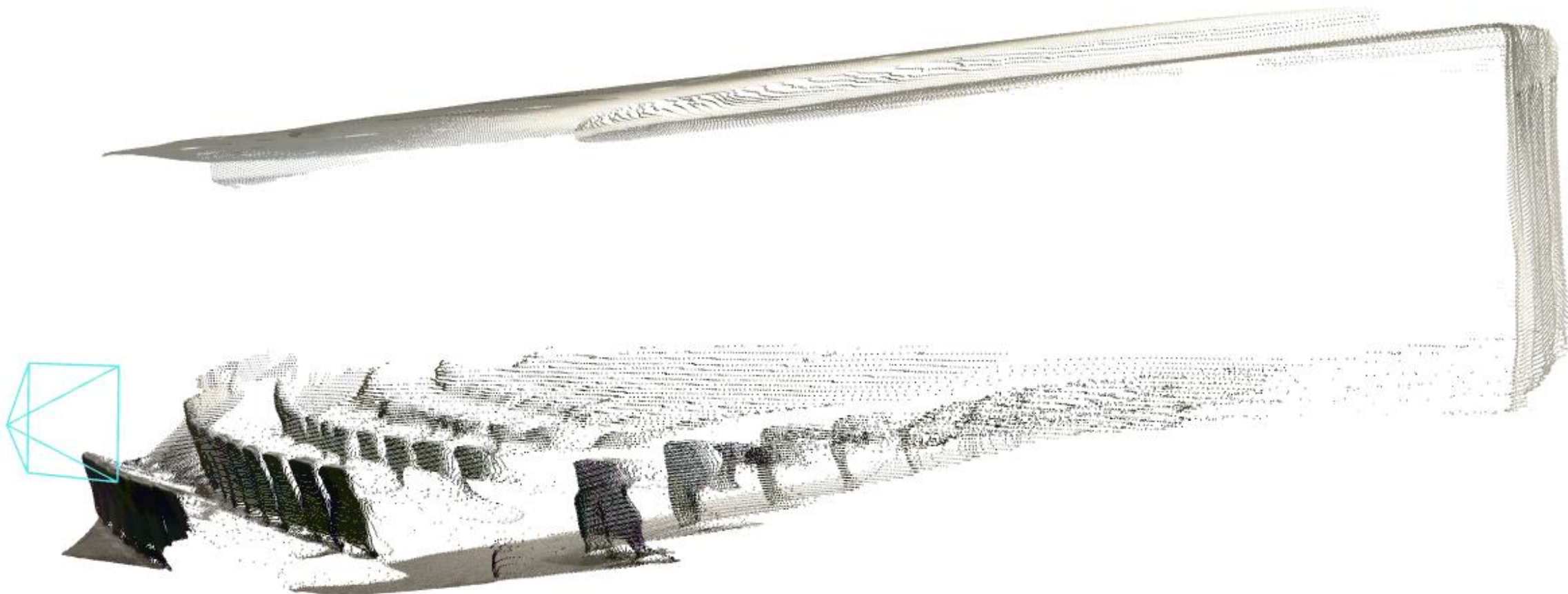
Exploring photo collections in 3D

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

SIGGRAPH 2006

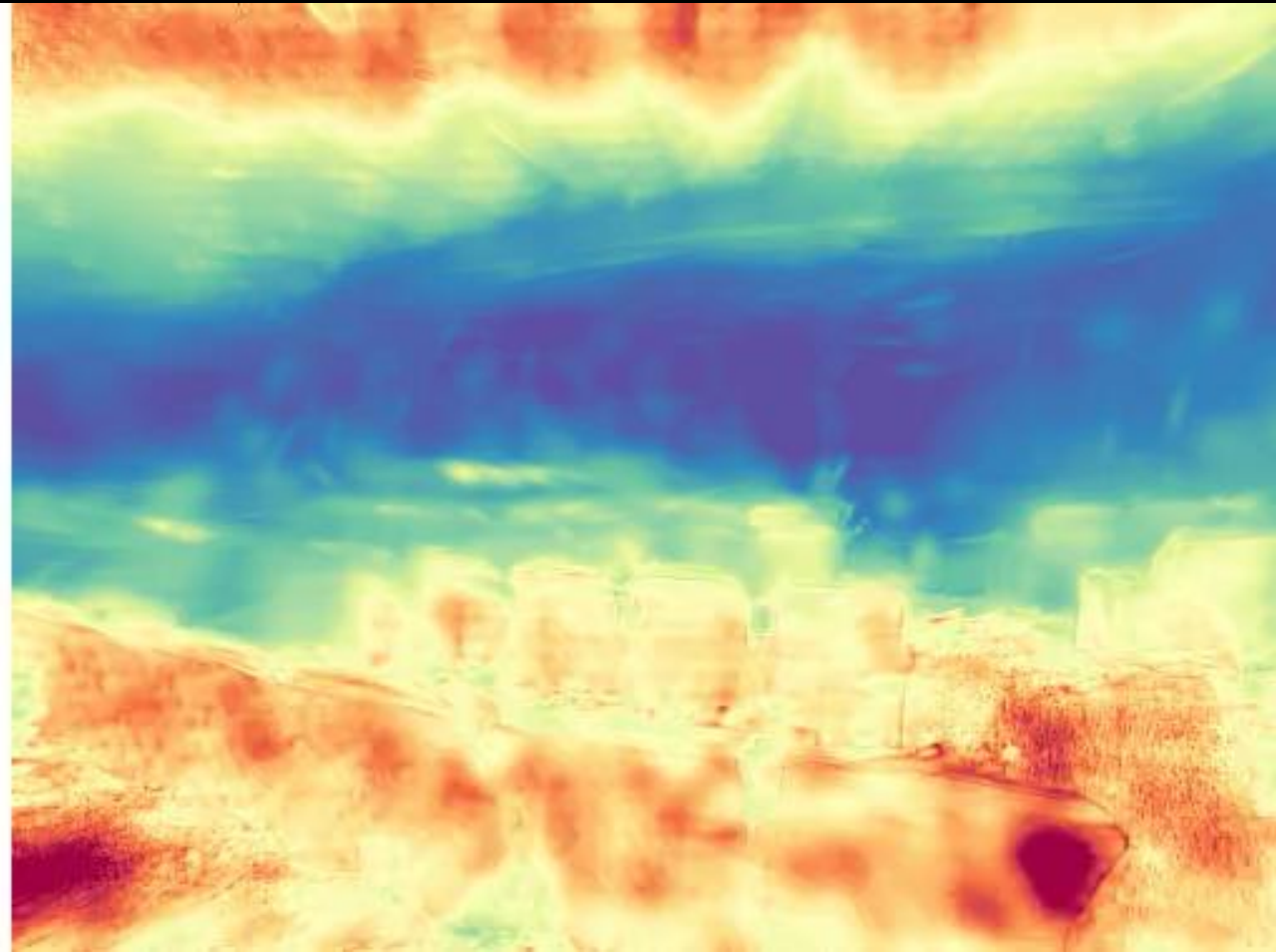


DepthAnything V3 Reconstruction

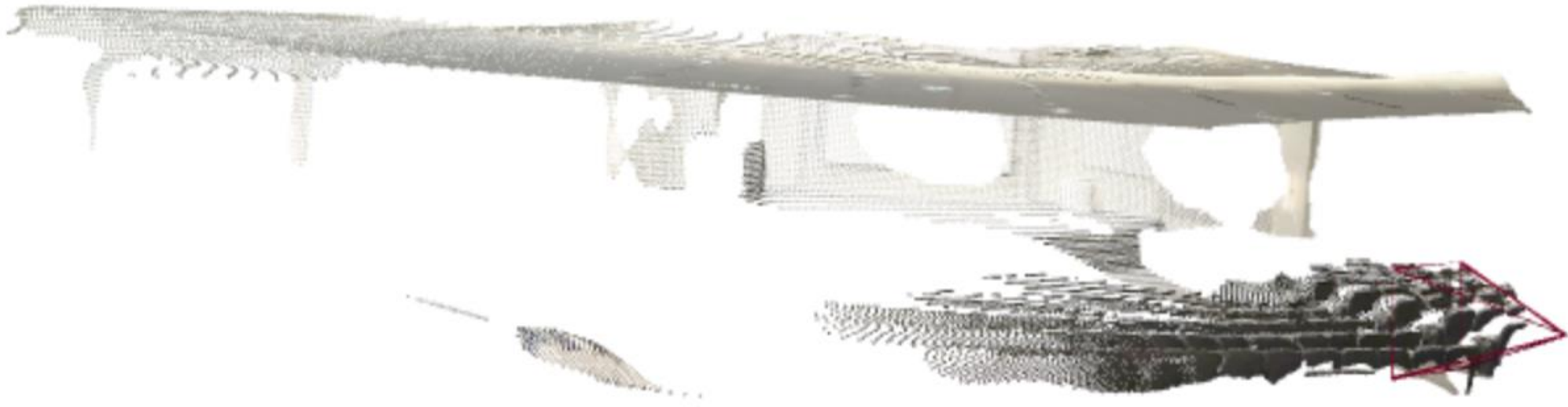


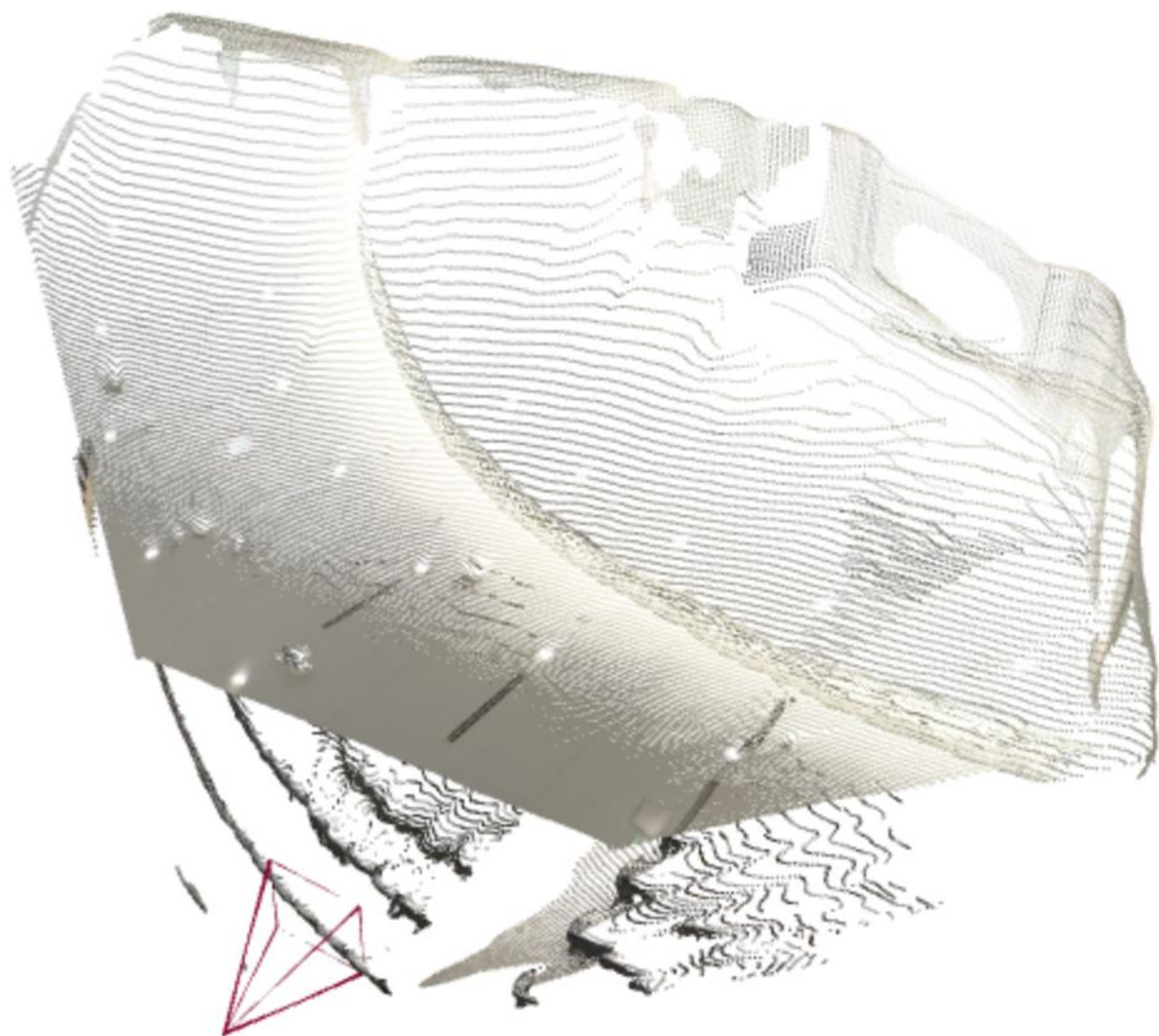
DepthAnything V3 Reconstruction





VGGT Reconstruction





3. Method

We introduce VGGT, a large transformer that ingests a set of images as input and produces a variety of 3D quantities as output. We start by introducing the problem in Sec. 3.1, followed by our architecture in Sec. 3.2 and its prediction heads in Sec. 3.3, and finally the training setup in Sec. 3.4.

3.1. Problem definition and notation

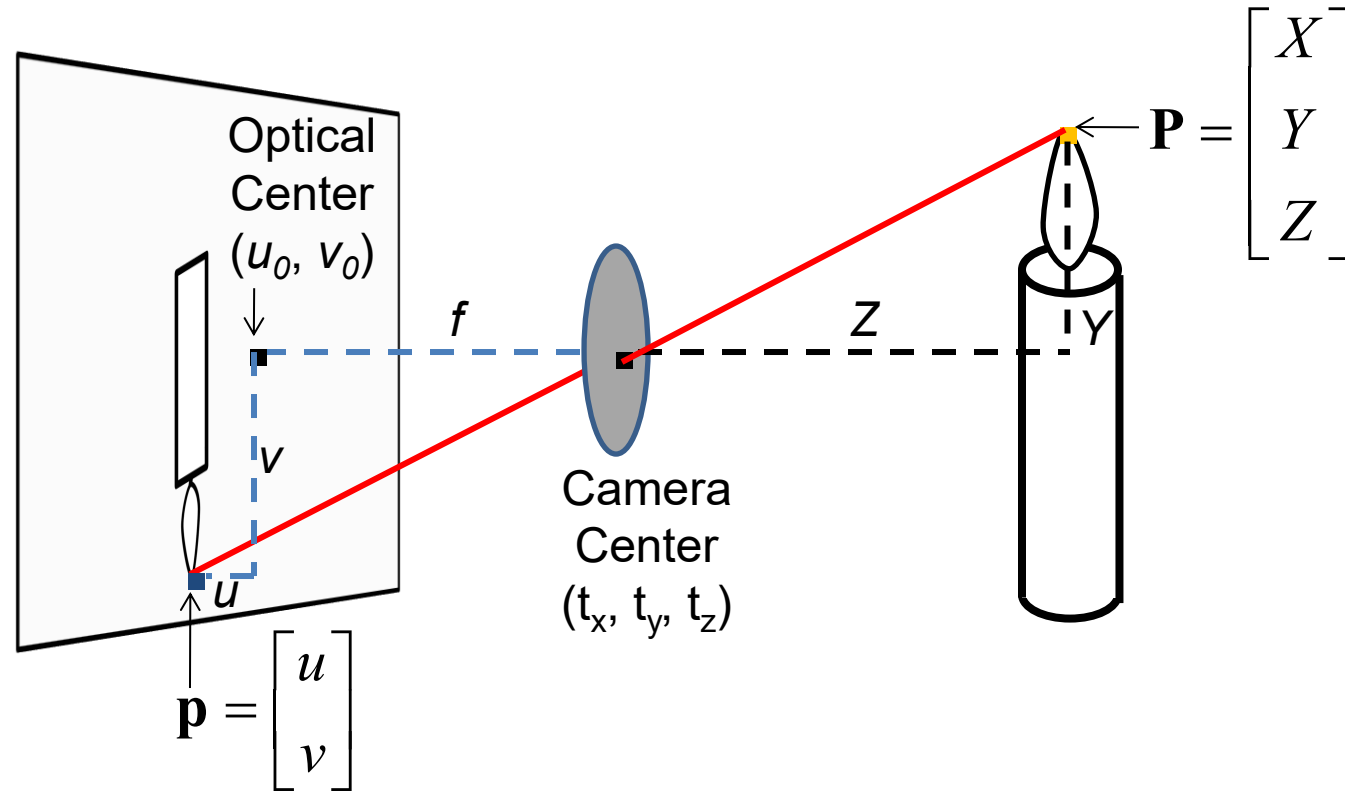
The input is a sequence $(I_i)_{i=1}^N$ of N RGB images $I_i \in \mathbb{R}^{3 \times H \times W}$, observing the same 3D scene. VGGT’s transformer is a function that maps this sequence to a corresponding set of 3D annotations, one per frame:

$$f((I_i)_{i=1}^N) = (\mathbf{g}_i, D_i, P_i, T_i)_{i=1}^N. \quad (1)$$

The transformer thus maps each image I_i to its camera parameters $\mathbf{g}_i \in \mathbb{R}^9$ (intrinsics and extrinsics), its depth map $D_i \in \mathbb{R}^{H \times W}$, its point map $P_i \in \mathbb{R}^{3 \times H \times W}$, and a grid $T_i \in \mathbb{R}^{C \times H \times W}$ of C -dimensional features for point tracking. We explain next how these are defined.

For the **camera parameters** $\mathbf{g}_i$, we use the parametrization from [125] and set $\mathbf{g} = [\mathbf{q}, \mathbf{t}, \mathbf{f}]$ which is the concatenation of the rotation quaternion $\mathbf{q} \in \mathbb{R}^4$, the translation vector $\mathbf{t} \in \mathbb{R}^3$, and the field of view $\mathbf{f} \in \mathbb{R}^2$. We assume that the camera’s principal point is at the image center, which is common in SfM frameworks [95, 125].

Projection: world coordinates $\rightarrow$ 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

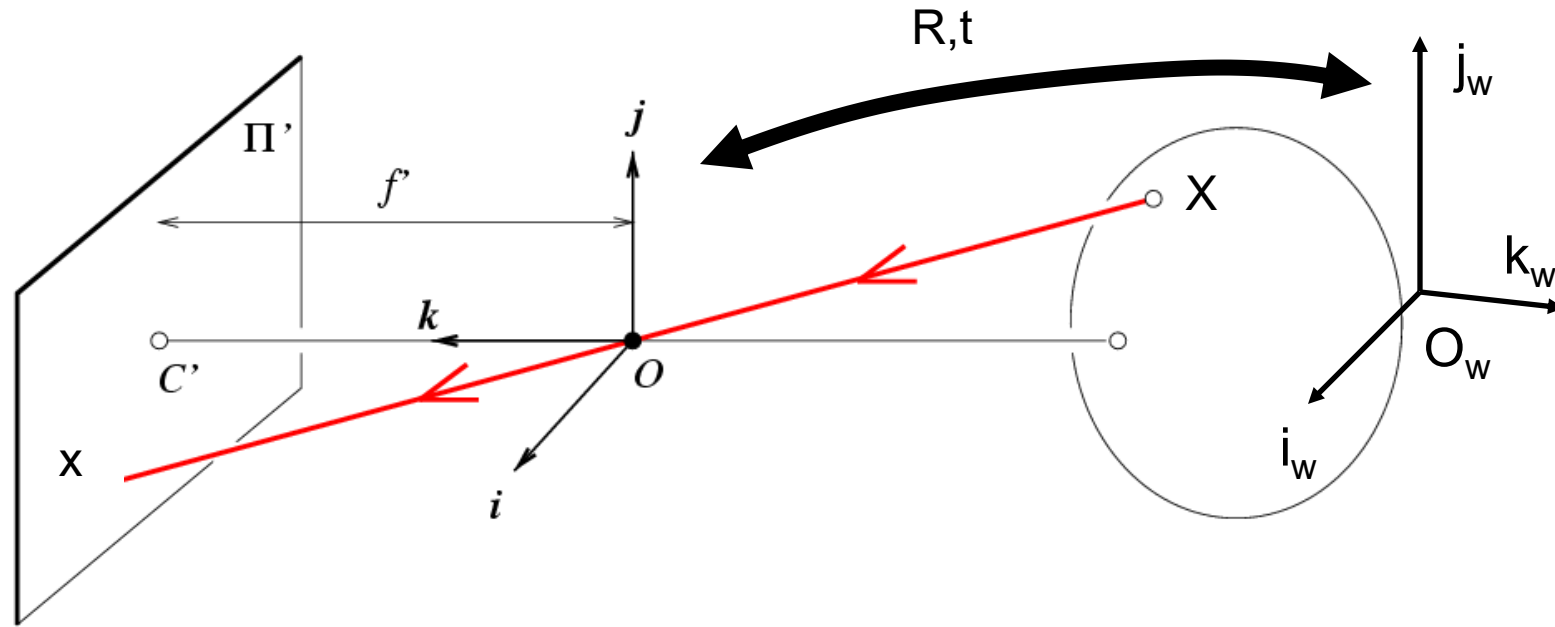
Invariant to scaling

$$k \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{kx}{kw} \\ \frac{ky}{kw} \\ \frac{kw}{kw} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \\ 1 \end{bmatrix}$$

Homogeneous Coordinates Cartesian Coordinates

Point in Cartesian is ray in Homogeneous

Projection matrix



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

$\mathbf{x}$: Image Coordinates: $(u, v, 1)$

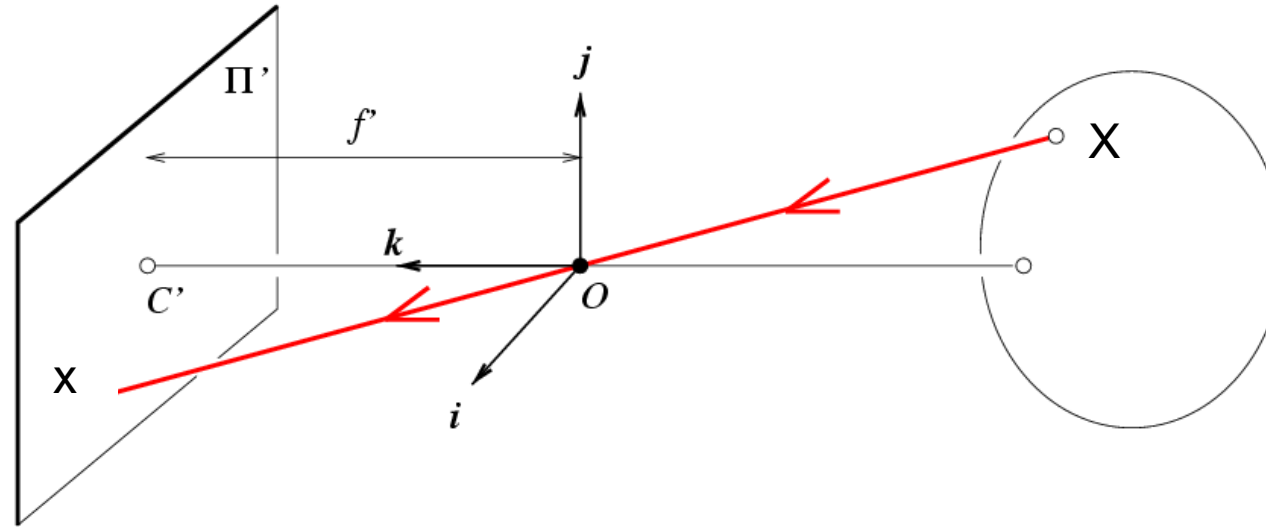
$\mathbf{K}$: Intrinsic Matrix (3×3)

$\mathbf{R}$: Rotation (3×3)

$\mathbf{t}$: Translation (3×1)

$\mathbf{X}$: World Coordinates: $(X, Y, Z, 1)$

Projection matrix



Intrinsic Assumptions

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

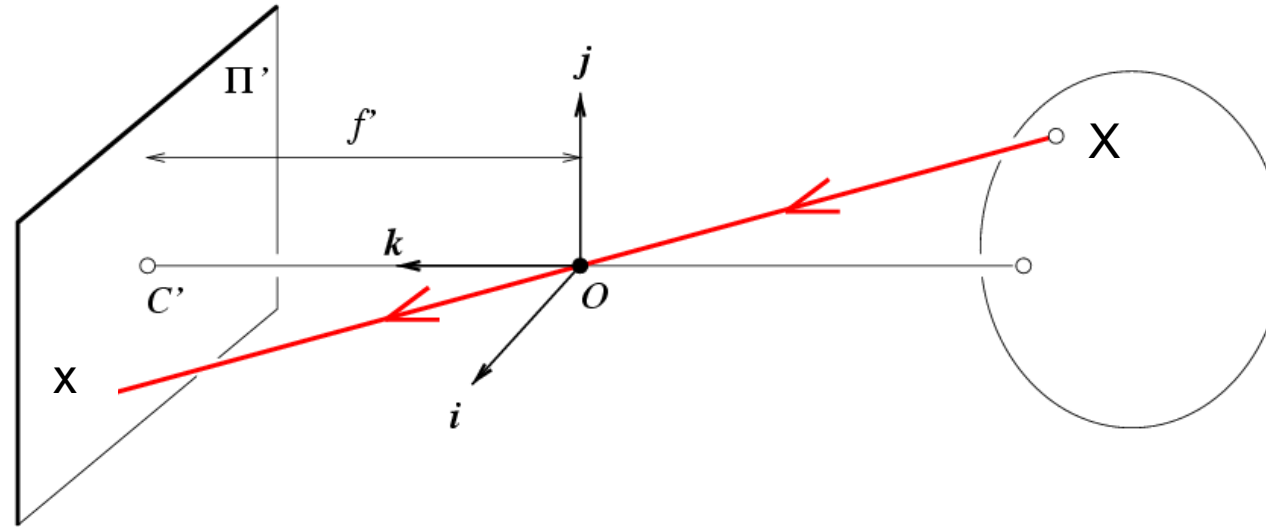
Extrinsic Assumptions

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

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

The matrix $\mathbf{K}$ is indicated by a red dashed line and a red arrow pointing to the first three rows of the matrix.

Projection matrix



Intrinsic Assumptions

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

Extrinsic Assumptions

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

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

Remove assumption: center pixel is (0,0)

Intrinsic Assumptions Extrinsic Assumptions

- Unit aspect ratio
- No skew

- 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

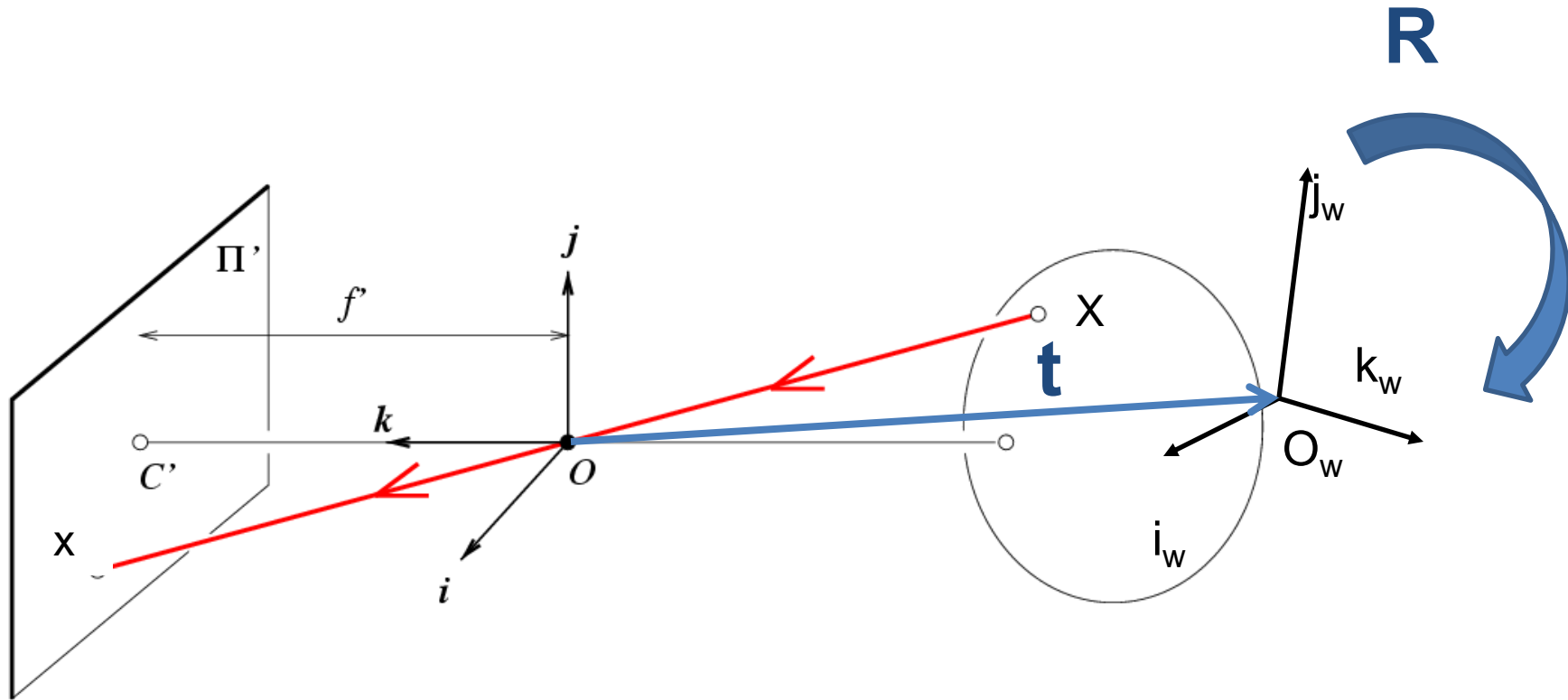
Intrinsic Assumptions Extrinsic Assumptions

- 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

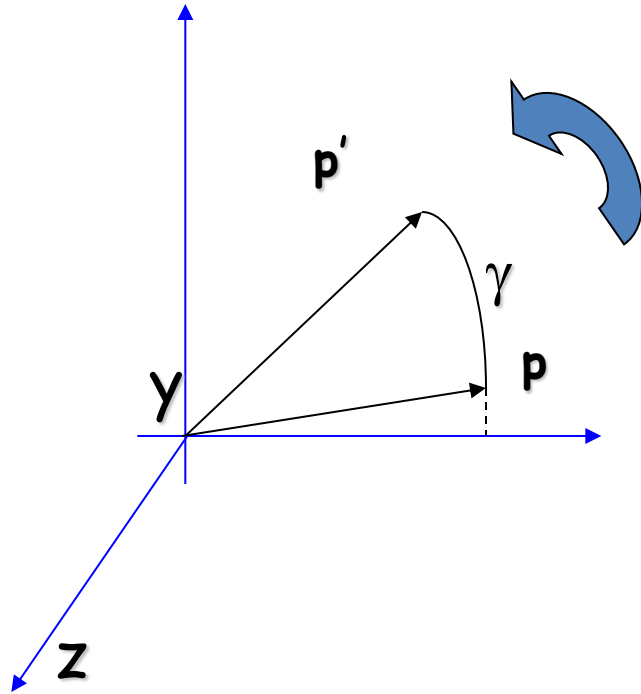
Intrinsic Assumptions Extrinsic Assumptions

- 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

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



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

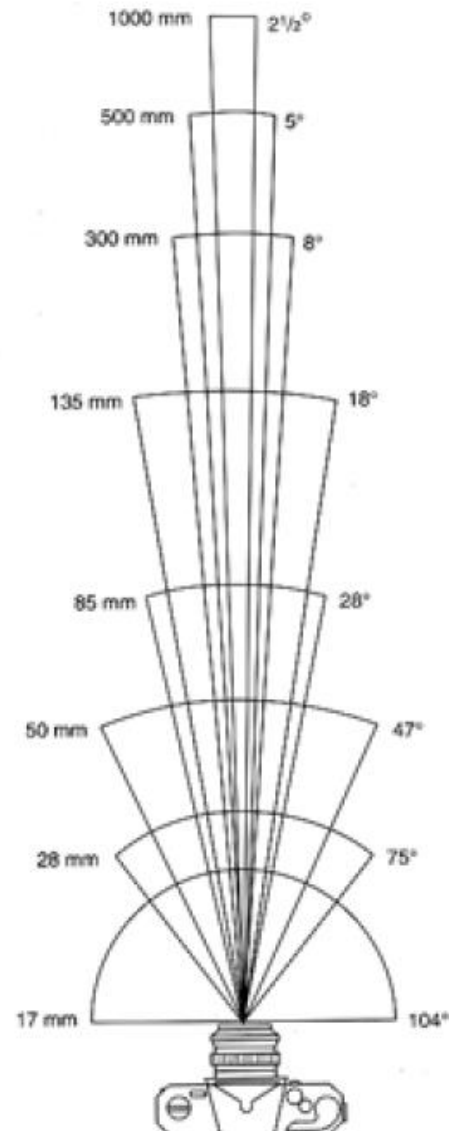
Degrees of freedom

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



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

Field of View (Zoom, focal length)



17mm



28mm



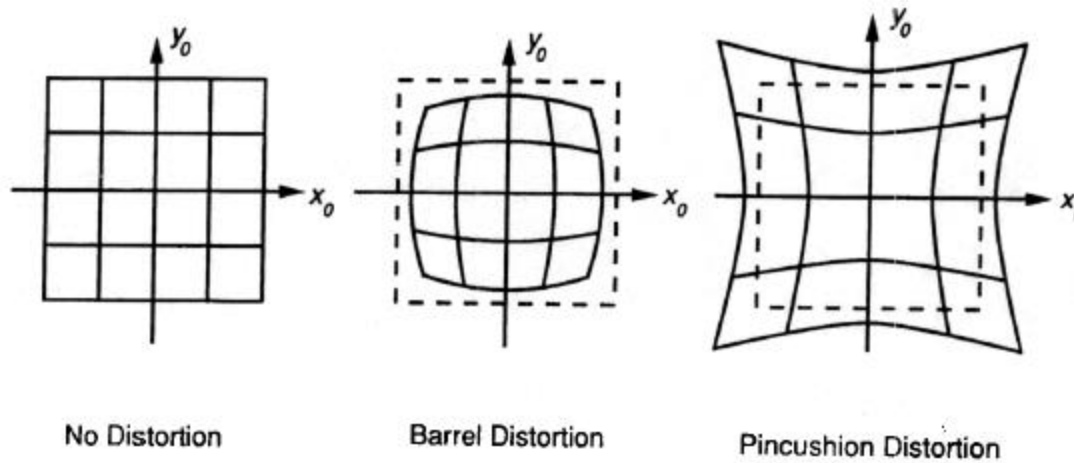
50mm



85mm

From London and Upton

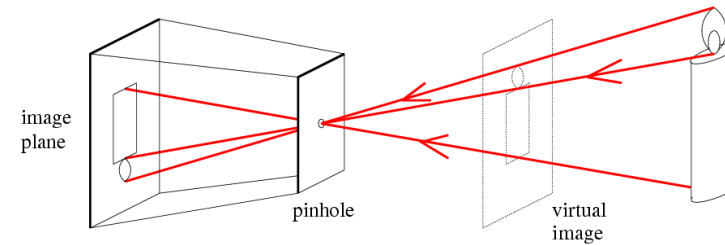
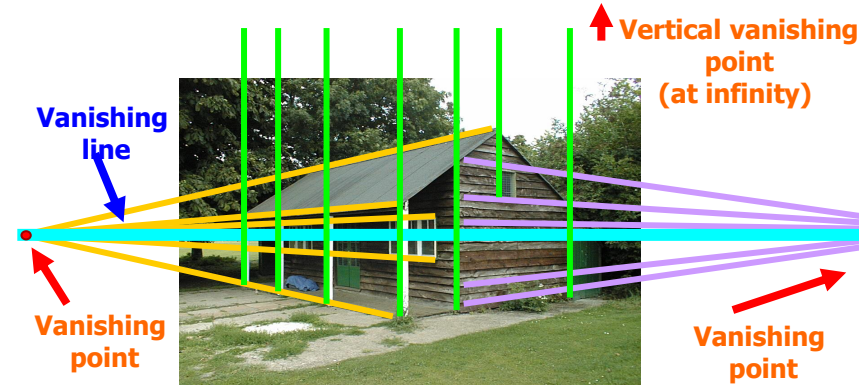
Beyond Pinholes: Radial Distortion



Corrected Barrel Distortion

Things to remember

- 3D -> 2D causes some weirdness in geometric relationships
- Pinhole camera model and camera projection matrix
- Homogeneous coordinates



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

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

Reminder: read your book

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

New Topic: Image Filtering



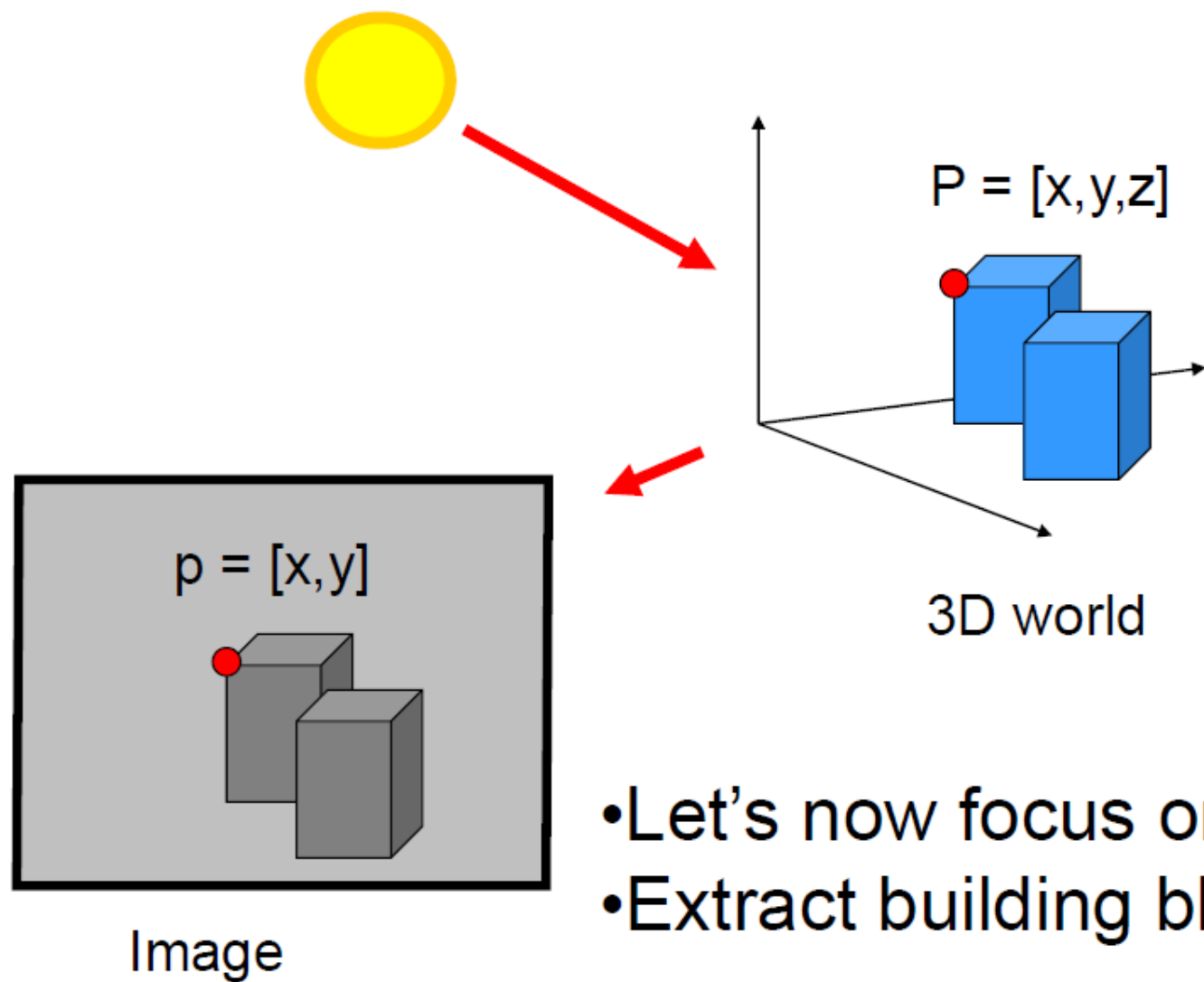
Computer Vision

James Hays

Many slides by Derek Hoiem

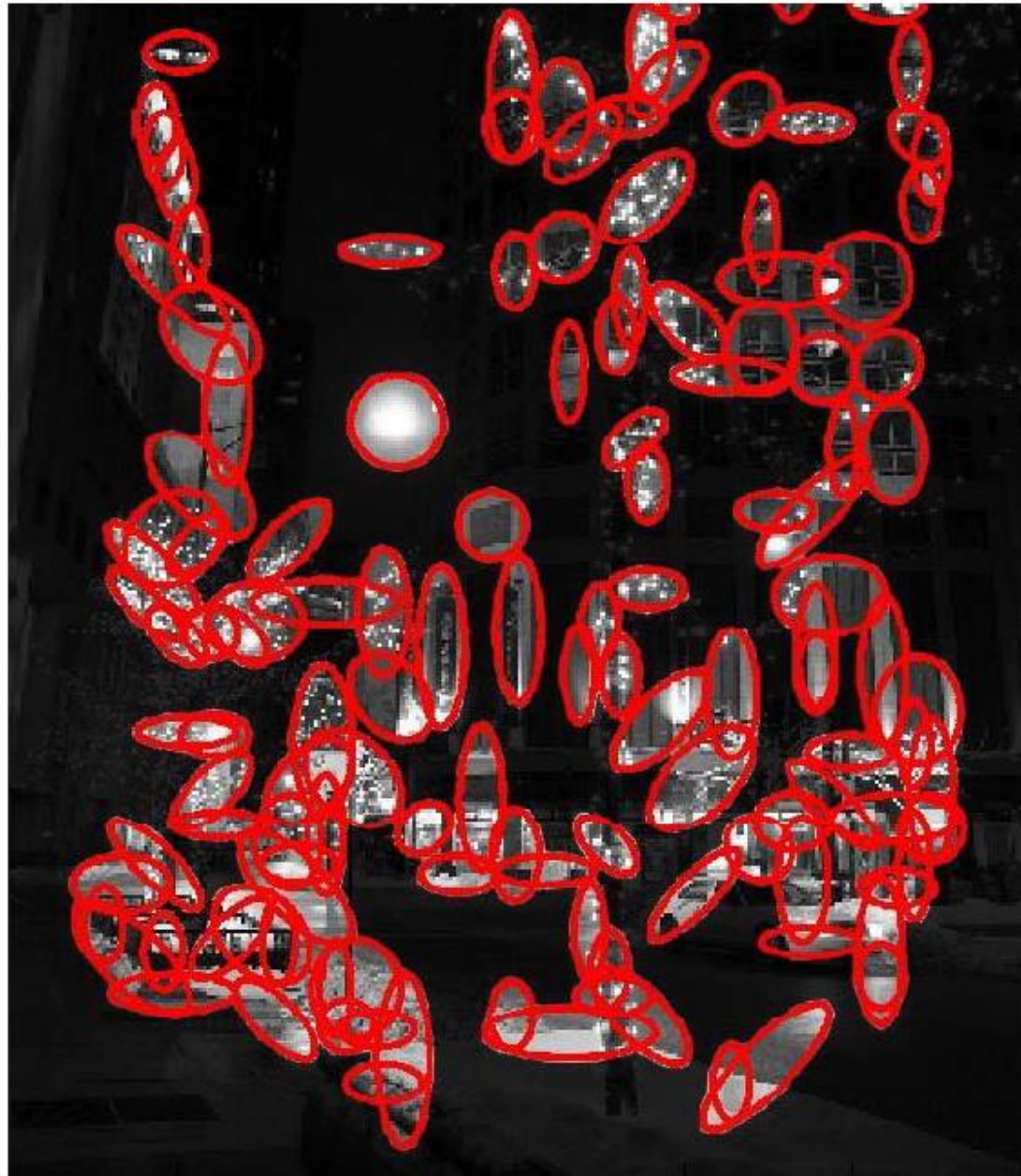


From the 3D to 2D



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

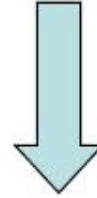
Extract useful building blocks





**Feature
Detection**

e.g. DoG

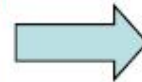


**Feature
Description**

e.g. SIFT

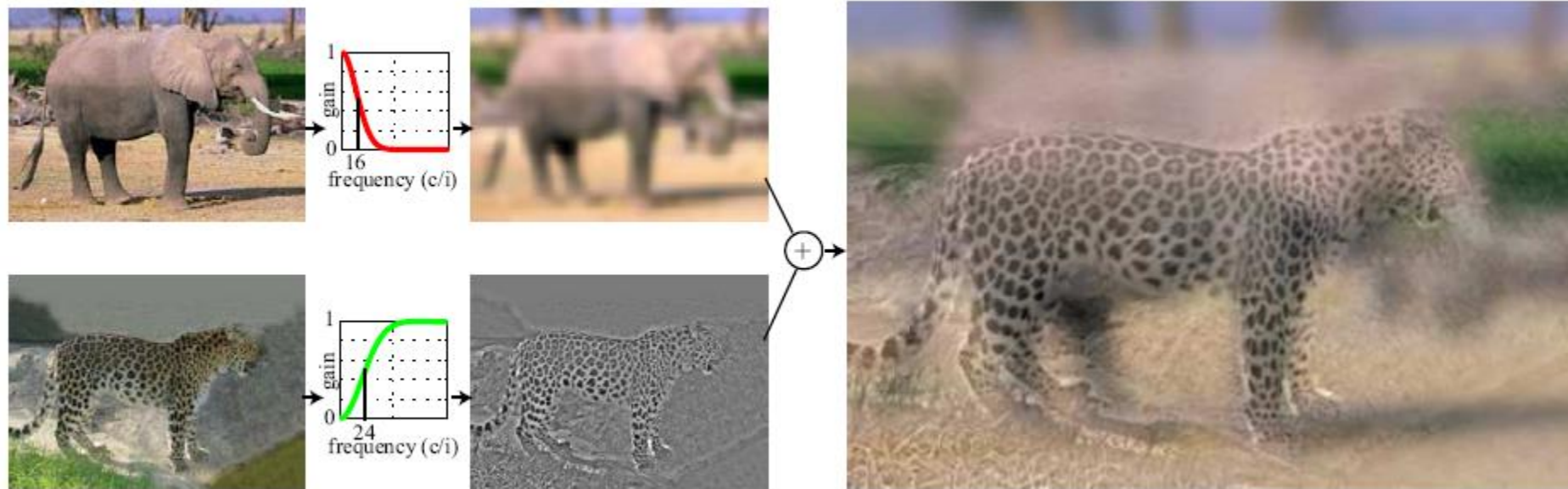


*database of local
descriptors*



Matching /
Indexing /
Detection

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 a simple function of local neighborhood at each position
- Really important!
 - Manipulate images
 - Denoise, resize, increase contrast, remove high frequencies, etc. (Project 1)
 - Extract information from images
 - Texture, edges, distinctive points, etc. (Project 2)
 - Deep Convolutional Networks (Project 3 and 4)

Example: box filter

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

1	1	1
1	1	1
1	1	1

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	?							

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10							

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10	20						

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10	20	30					

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10	20	30	30				

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10	20	30	30				

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10	20	30	30				
						?			
				50					

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

Image filtering

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

$$f[\cdot, \cdot]$$

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

$$h[\cdot, \cdot]$$

	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	

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

Box Filter

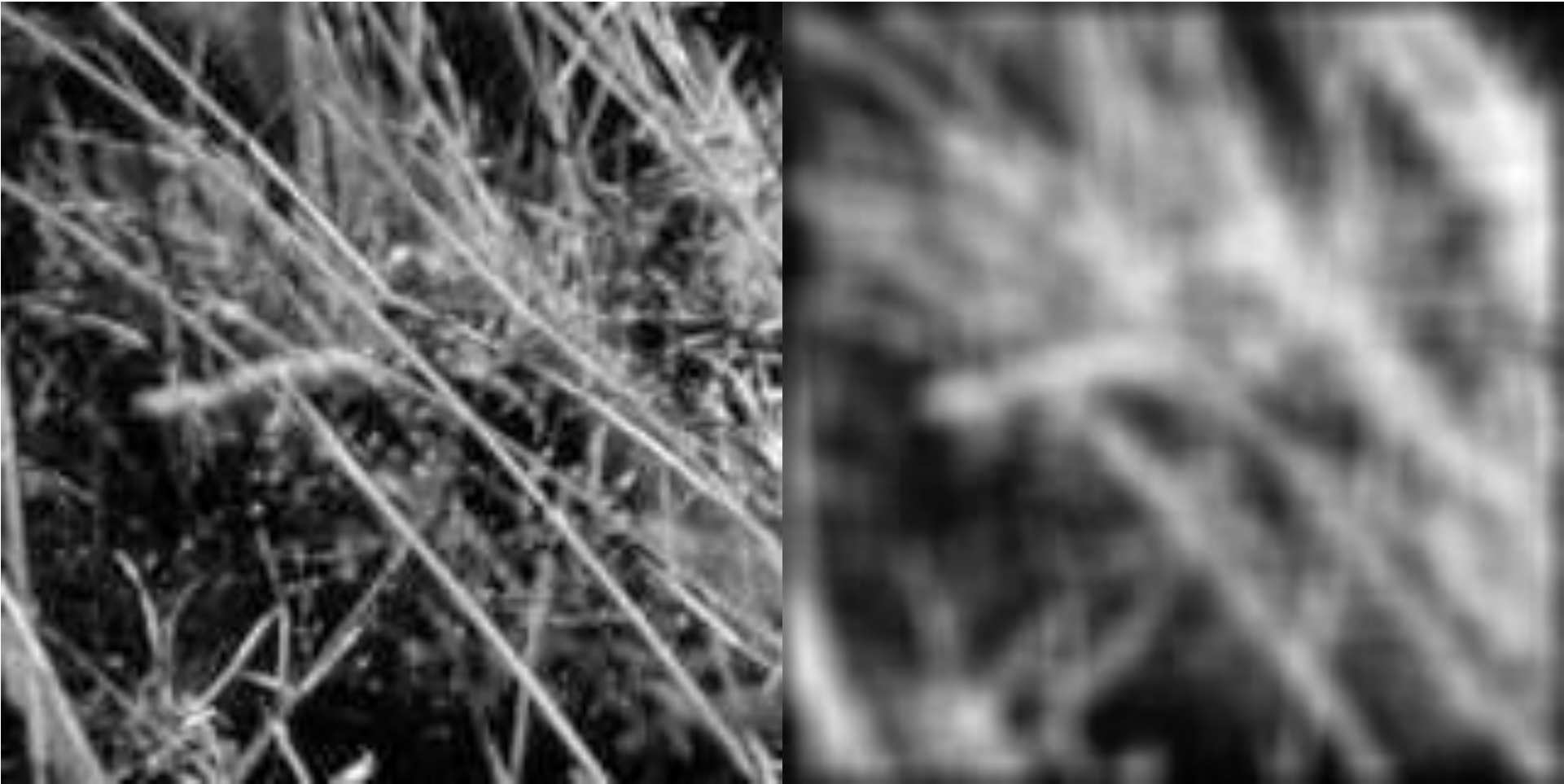
What does it do?

- Returns the average of each neighborhood
- Smooths an image (removes sharp features, removes some high spatial frequency features)

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

1	1	1
1	1	1
1	1	1

Smoothing with box filter



Practice with linear filters



Original

0	0	0
0	1	0
0	0	0

?

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

?

Practice with linear filters



Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

—

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

?

(Note that filter sums to 1)

Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

−

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1



Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

-

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

+

0	0	0
0	1	0
0	0	0

-1/9	-1/9	-1/9
-1/9	8/9	-1/9
-1/9	-1/9	-1/9



Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

-

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

+

0	0	0
0	1	0
0	0	0

-1/9	-1/9	-1/9
-1/9	8/9	-1/9
-1/9	-1/9	-1/9

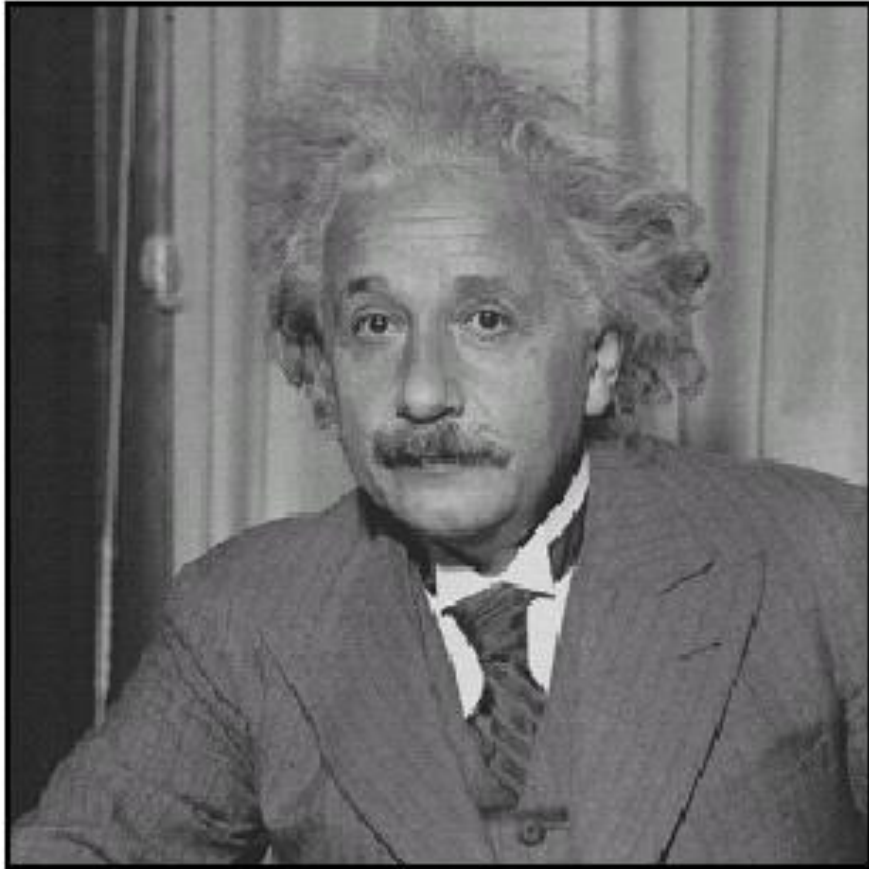
-1/9	-1/9	-1/9
-1/9	17/9	-1/9
-1/9	-1/9	-1/9



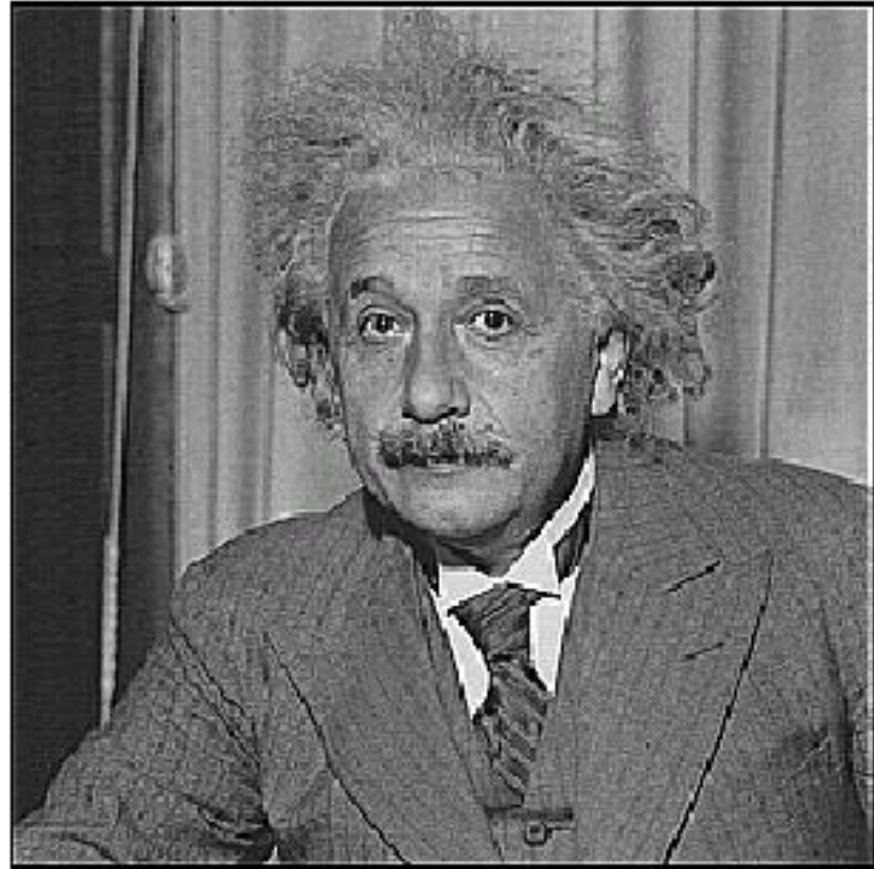
Sharpening filter

- Accentuates differences w.r.t. local average

Sharpening

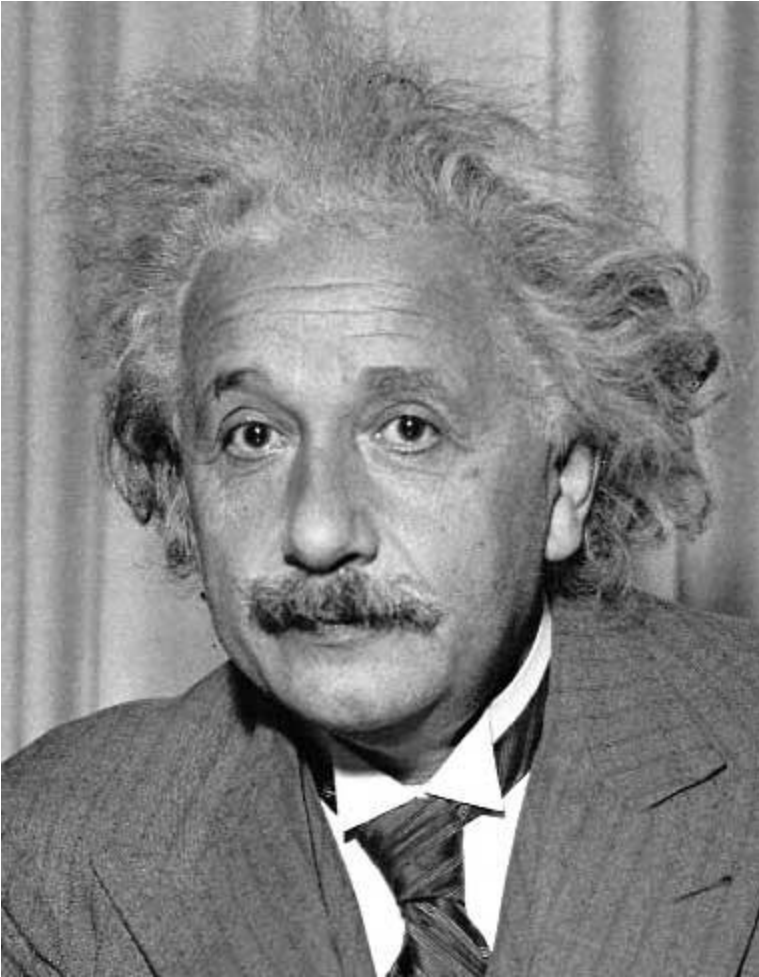


before



after

Other filters



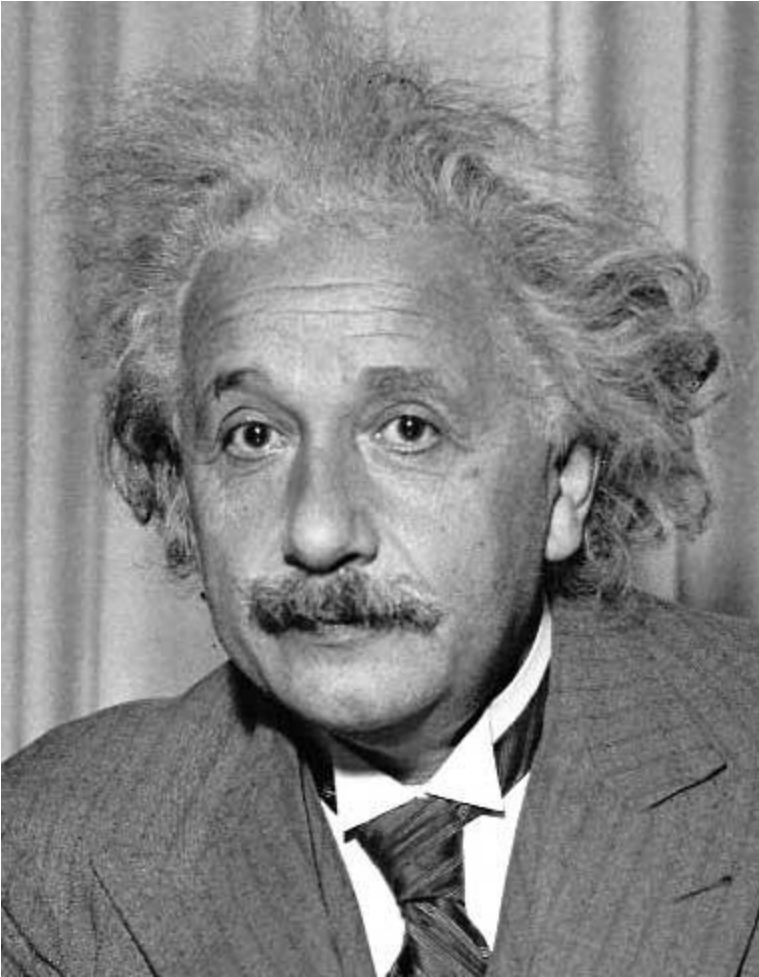
1	0	-1
2	0	-2
1	0	-1

Sobel



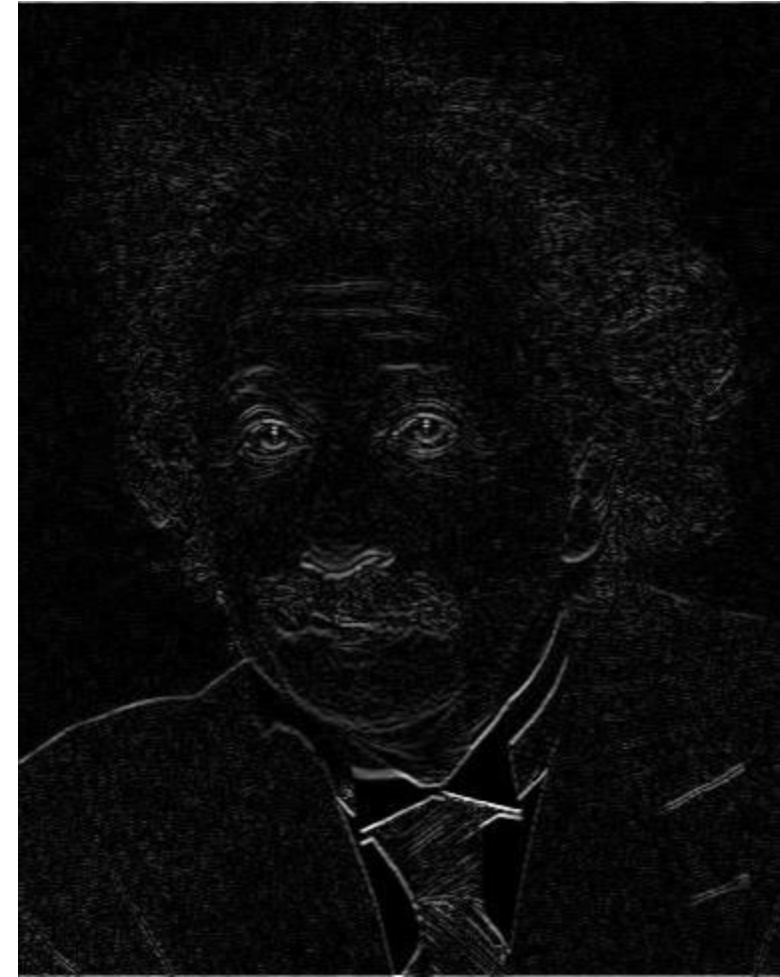
Vertical Edge
(absolute value)

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`

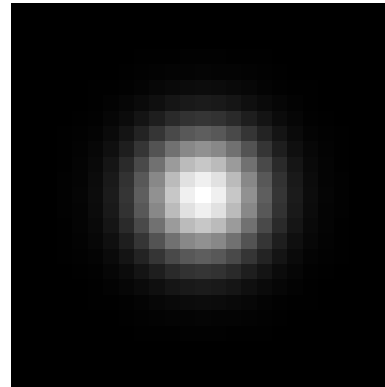
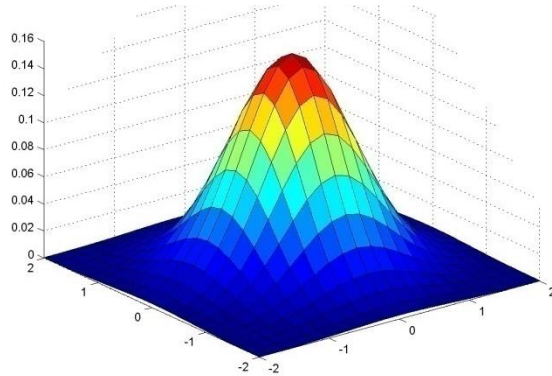
`I=image, size m x n`

- 2d filter “Many readers will skim over formulas on their first reading of your exposition. Therefore, your sentences should flow smoothly when all but the simplest formulas are replaced by “blah” or some other grunting noise.”
- 2d convolution - Donald Knuth.

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

Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness

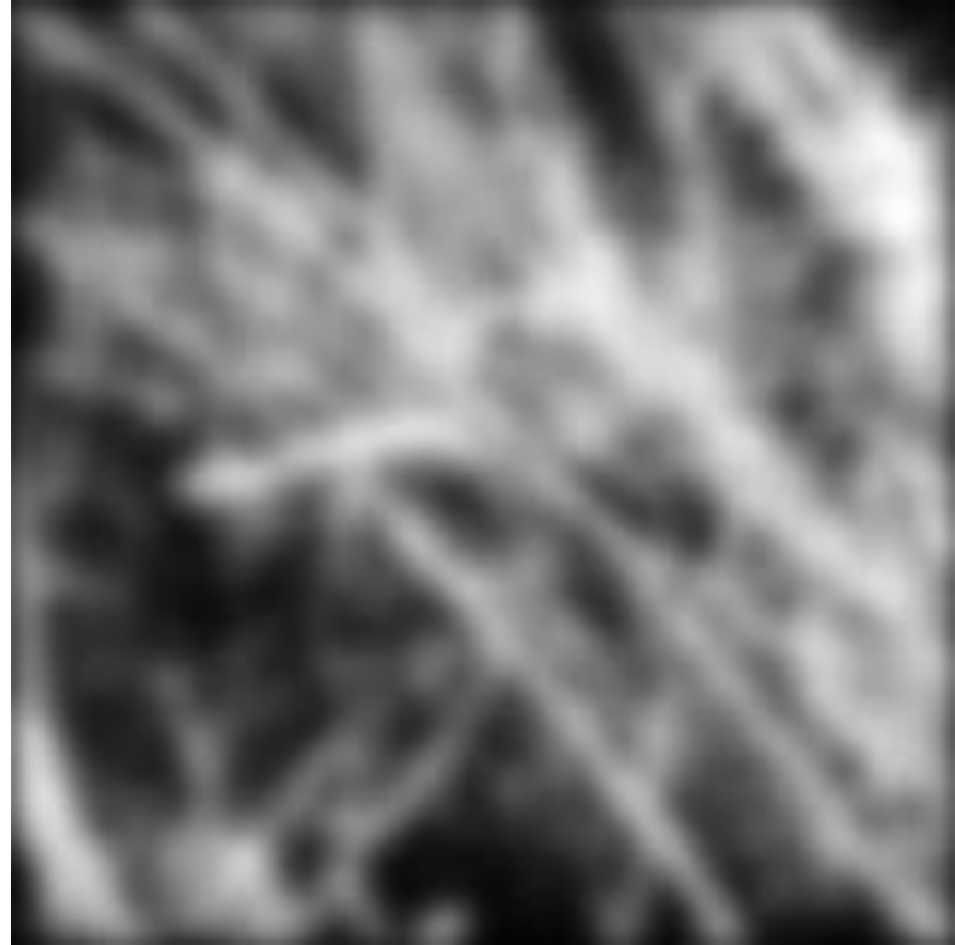


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

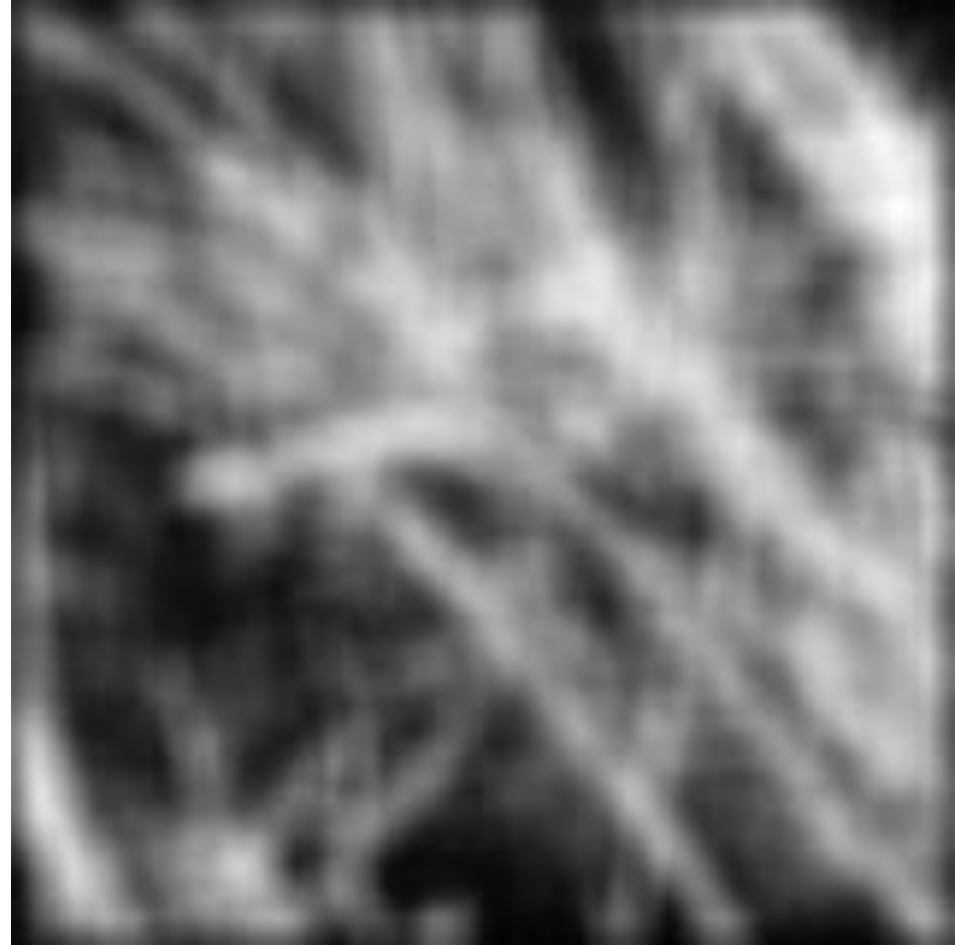
5 x 5, $\sigma = 1$

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

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

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

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:

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 3σ

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



Next class: Light and Color and Thinking in Frequency

