CSC496: Deep learning in computer vision
Prof. Bei Xiao

Lecture 4: Lens, Linear filtering

Today's class

- Image Formation with pin-hole camera
- Lens
- Linear Filters

- Homework 1: A simple vision problem: Due Wed September 18th.

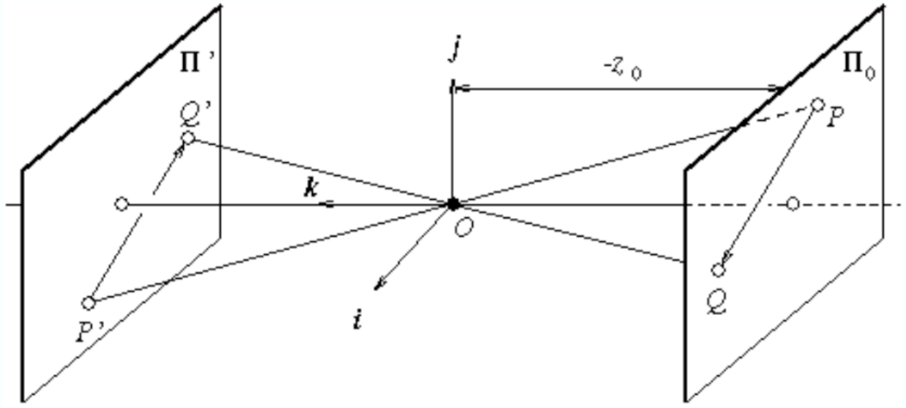
Classes forecast

- Linear filtering and image convolution September 13
- Spatial signal processing September 16
- Temporal signal processing September 23 (18^{th} no class)
- Statistical modeling of images September 26
- Onto machine learning September 30th

- Homework 1: A simple vision problem: Due Wed September 18^{th}.
- Homework 2: Pin-hole camera. Out on Tuesday, September 17^{th}.

Other projection models: Weak perspective

- Issue
  - perspective effects, but not over the scale of individual objects
  - collect points into a group at about the same depth, then divide each point by the depth of its group
  - Adv: easy
  - Disadv: only approximate



$$(x,y,z) \rightarrow \left(\frac{fx}{z_0}, \frac{fy}{z_0}\right)$$

Three camera projections

3-d point **2-d image position**

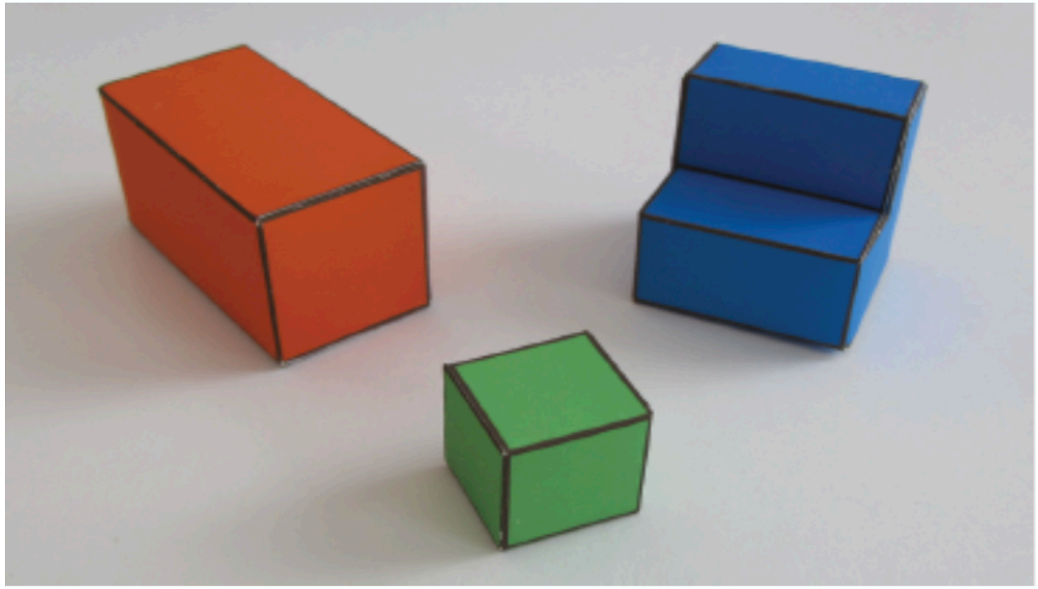
(1) Perspective: $(x, y, z) \rightarrow \left(\frac{fx}{z}, \frac{fy}{z}\right)$

(2) Weak perspective: $(x, y, z) \rightarrow \left(\frac{fx}{z_0}, \frac{fy}{z_0}\right)$

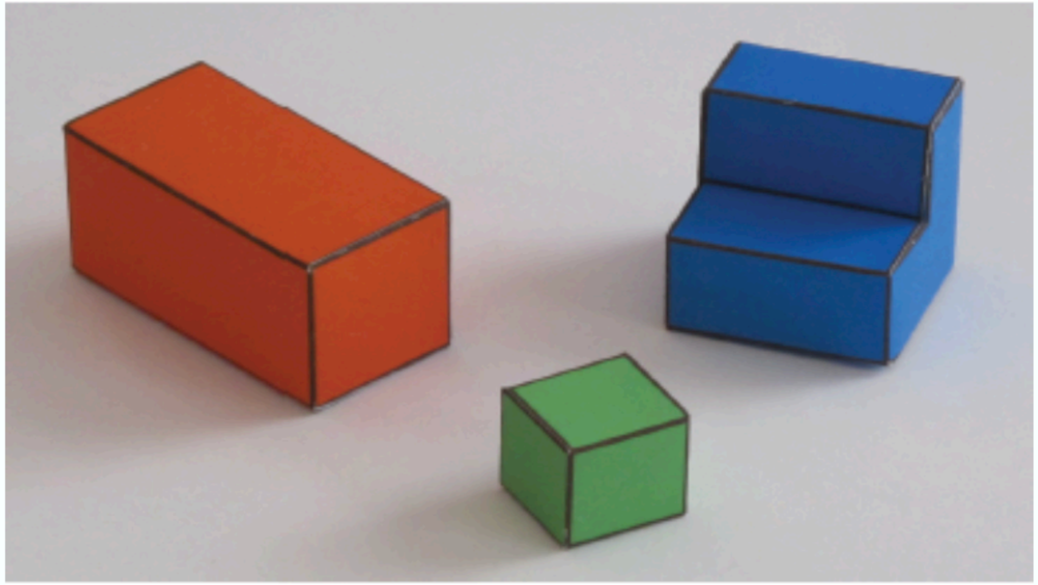
(3) Orthographic: $(x, y, z) \rightarrow (x, y)$

which is perspective, which orthographic?

Perspective projection

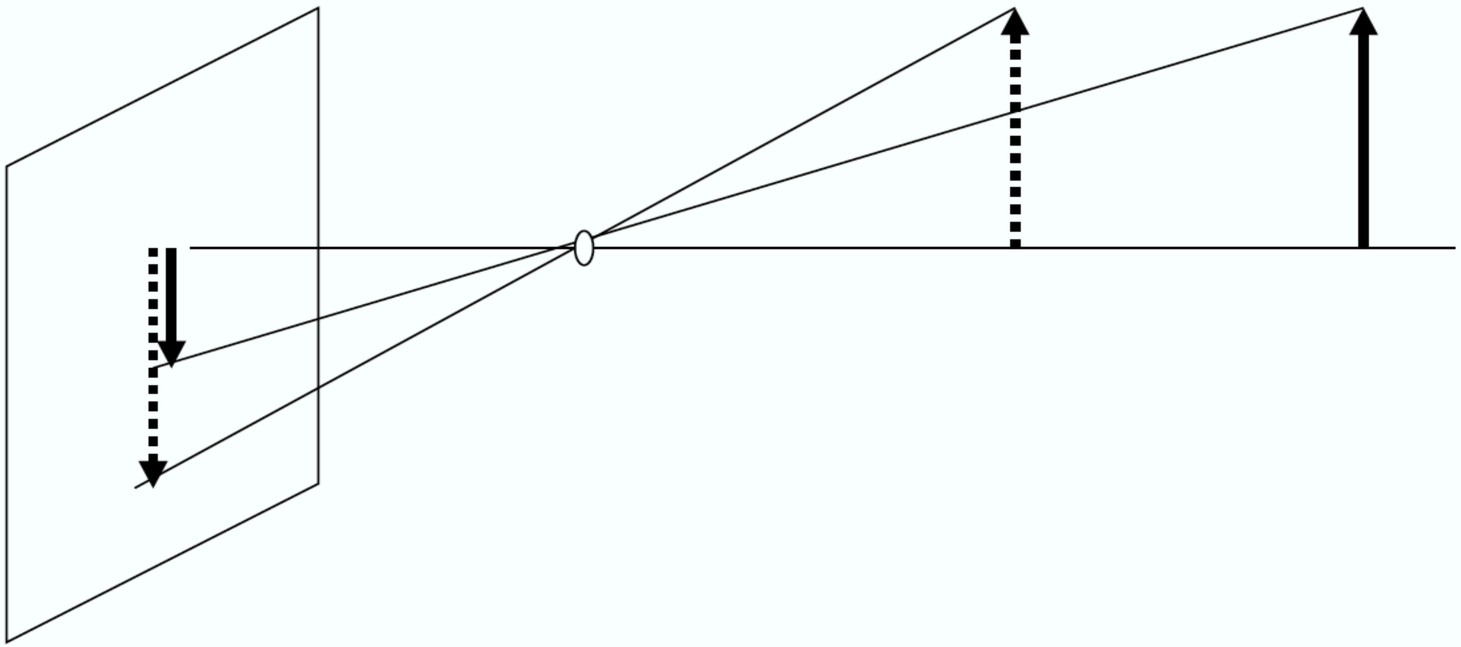


Parallel (orthographic) projection



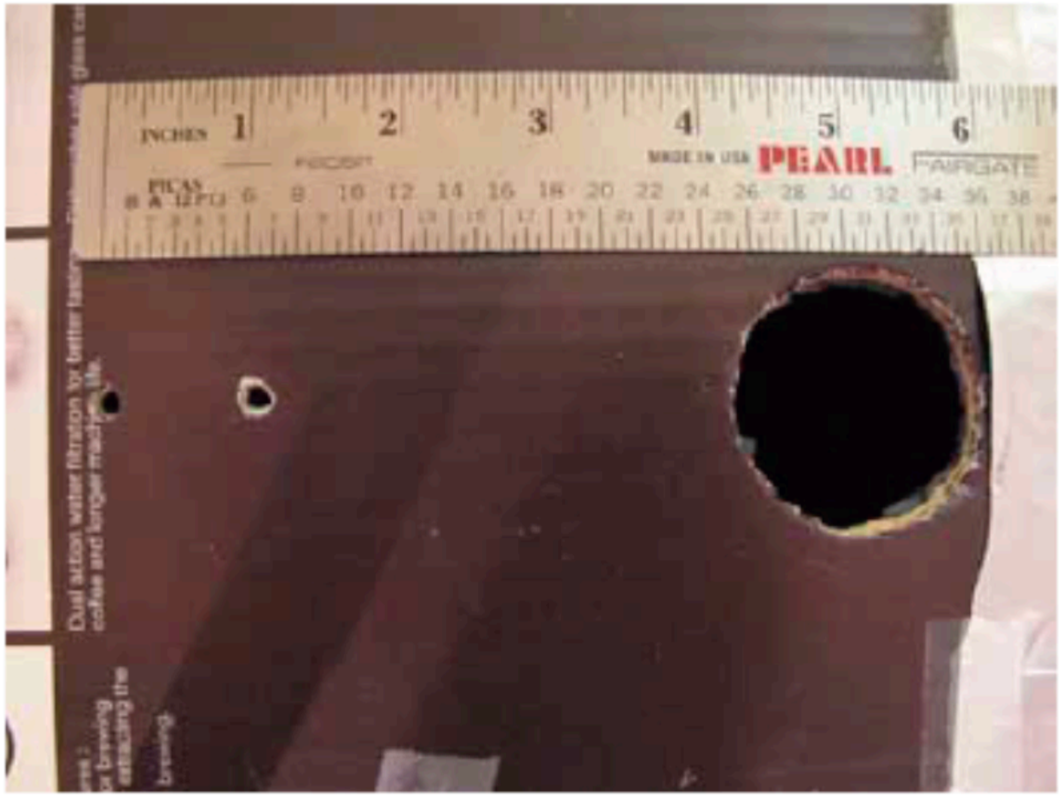
Example images from pinhole camera

Measuring distance

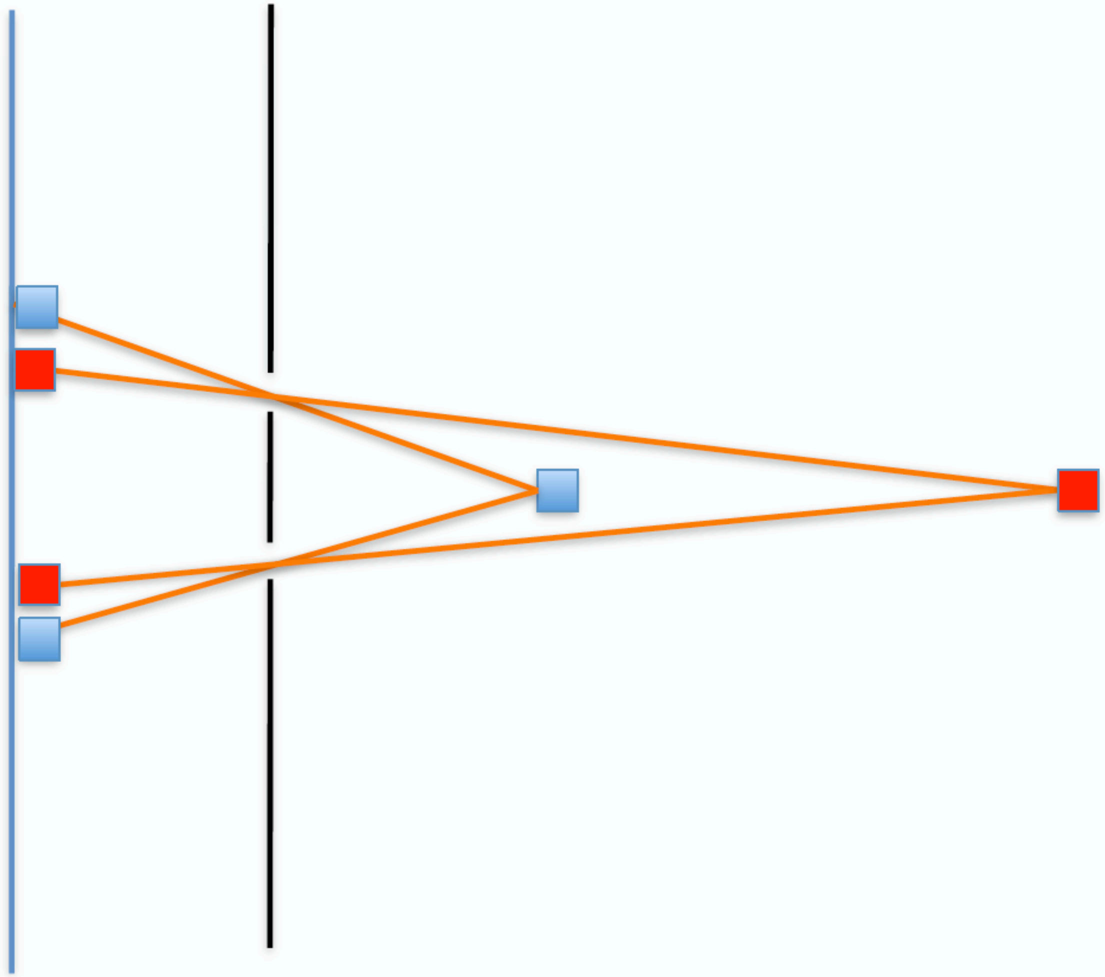


• Object size decreases with distance to the pinhole
• There, given a single projection, if we know the size of the object we can know how far it is.
• But for objects of unknown size, the 3D information seems to be lost.

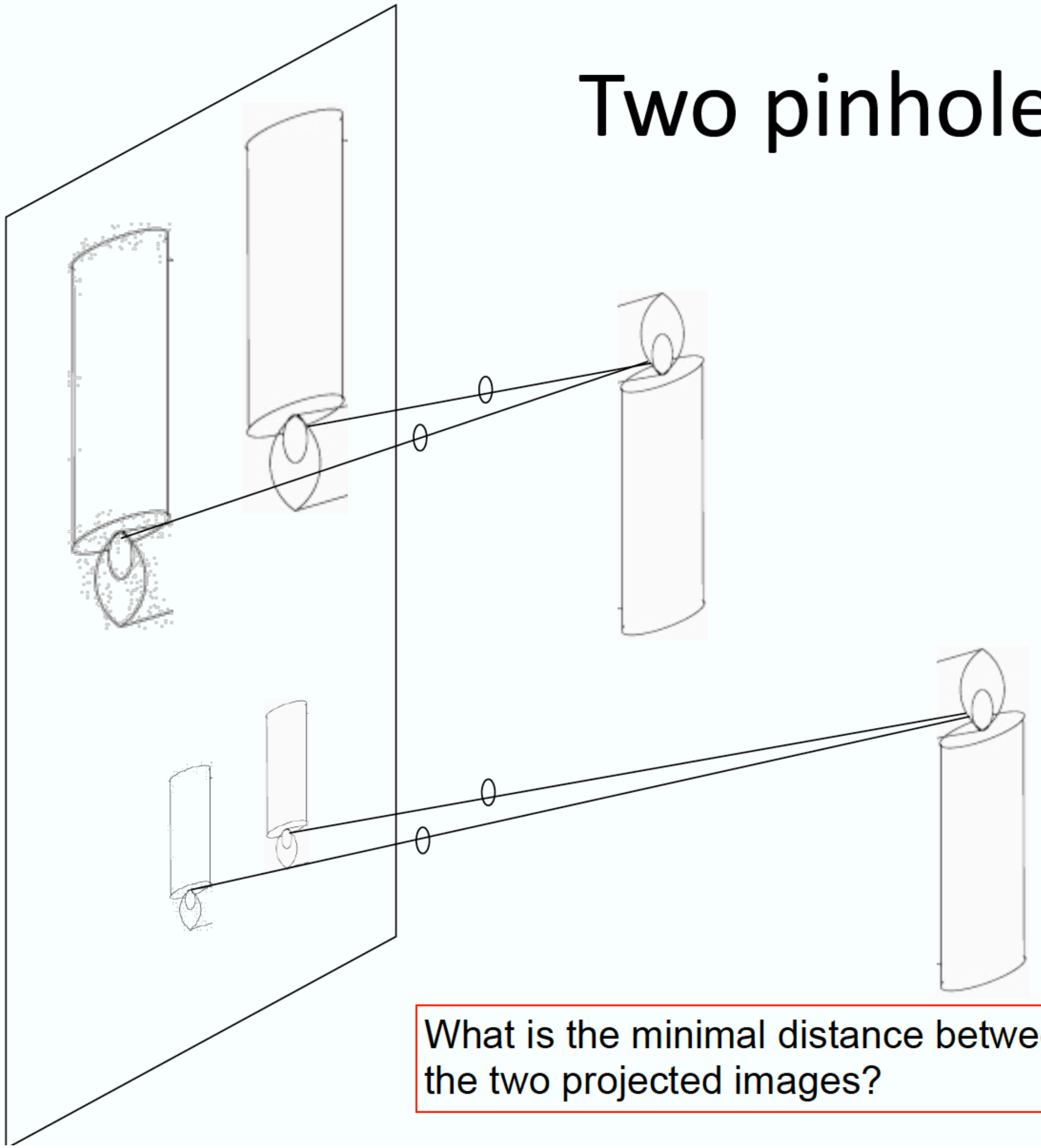
Playing with pinholes



Two pinholes



Two pinholes

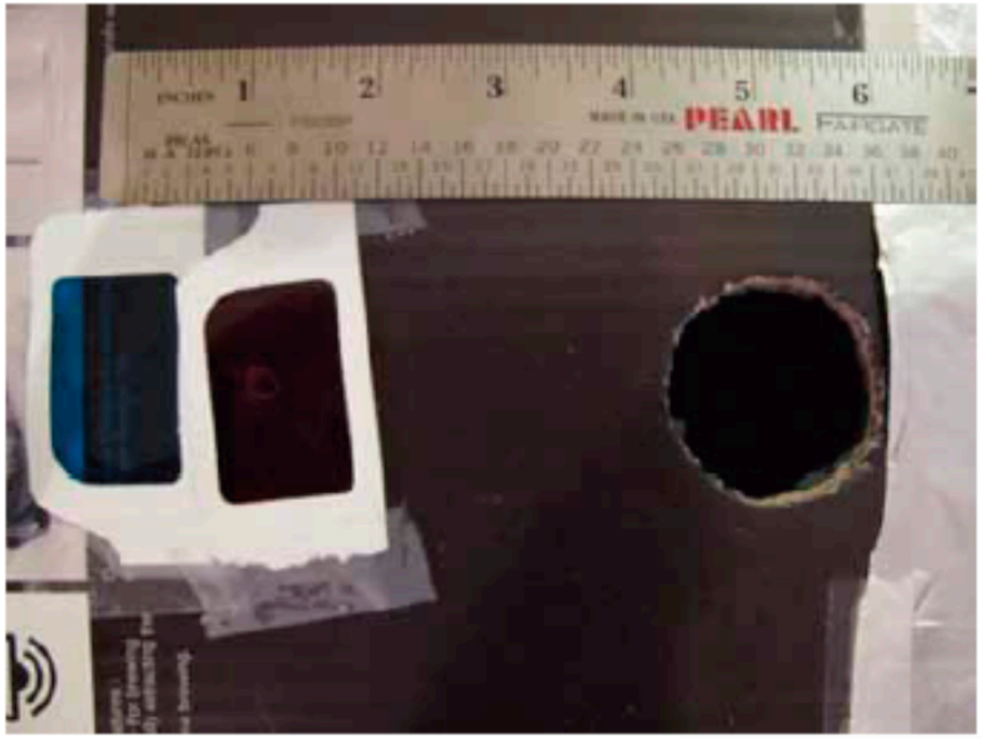


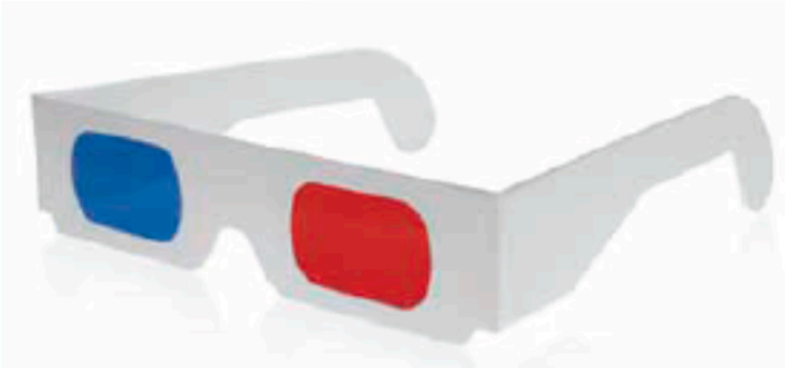
What is the minimal distance between the two projected images?

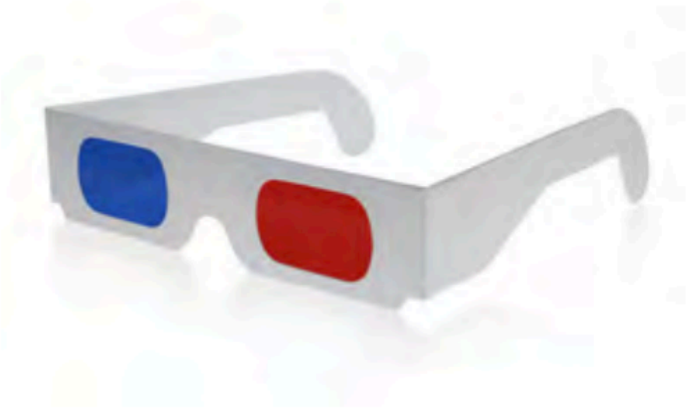
Anaglyph image

Anaglyph images are a particular method to produce 3D images. Anaglyph images are formed by superimposing a pair of stereo images into two different color channels. They provide a 3D effect when viewed with color glasses.

Anaglyph pinhole camera

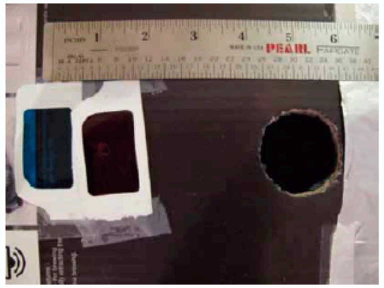






We used one of the glasses to get the filters and placed them in front of the two holes. Figure 3 shows two pictures of the two holes (top left) and the filters positioned in front of each hole (top right). On the bottom of that figure is a resulting 3D image, with the two views superimposed. That image should give you a 3D percept when viewed with anaglyph glasses.

Anaglyph pinhole camera



front of camera

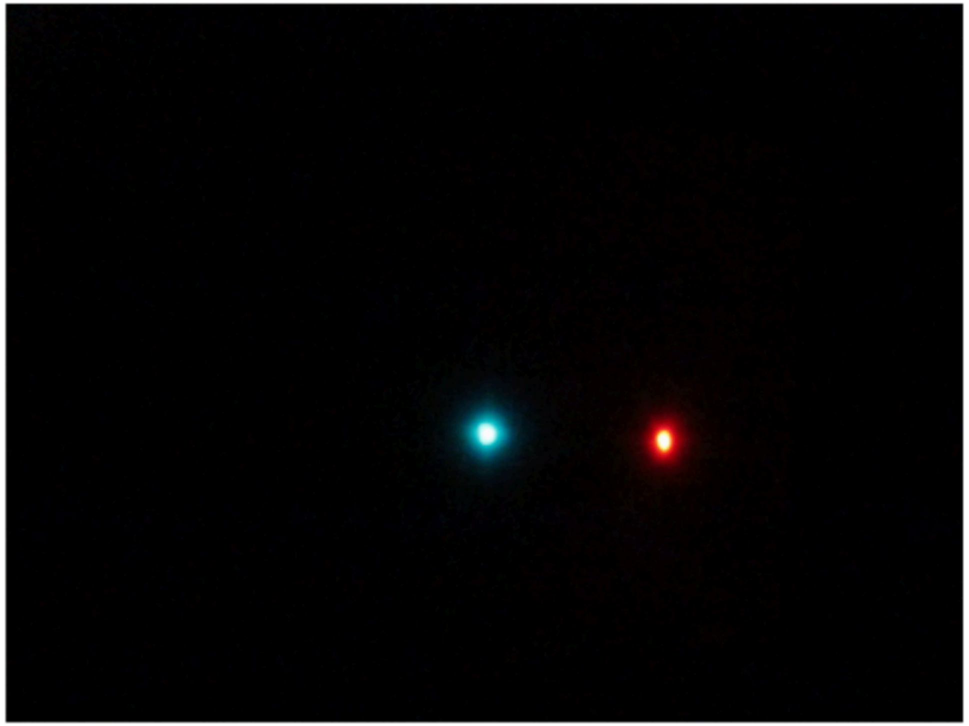
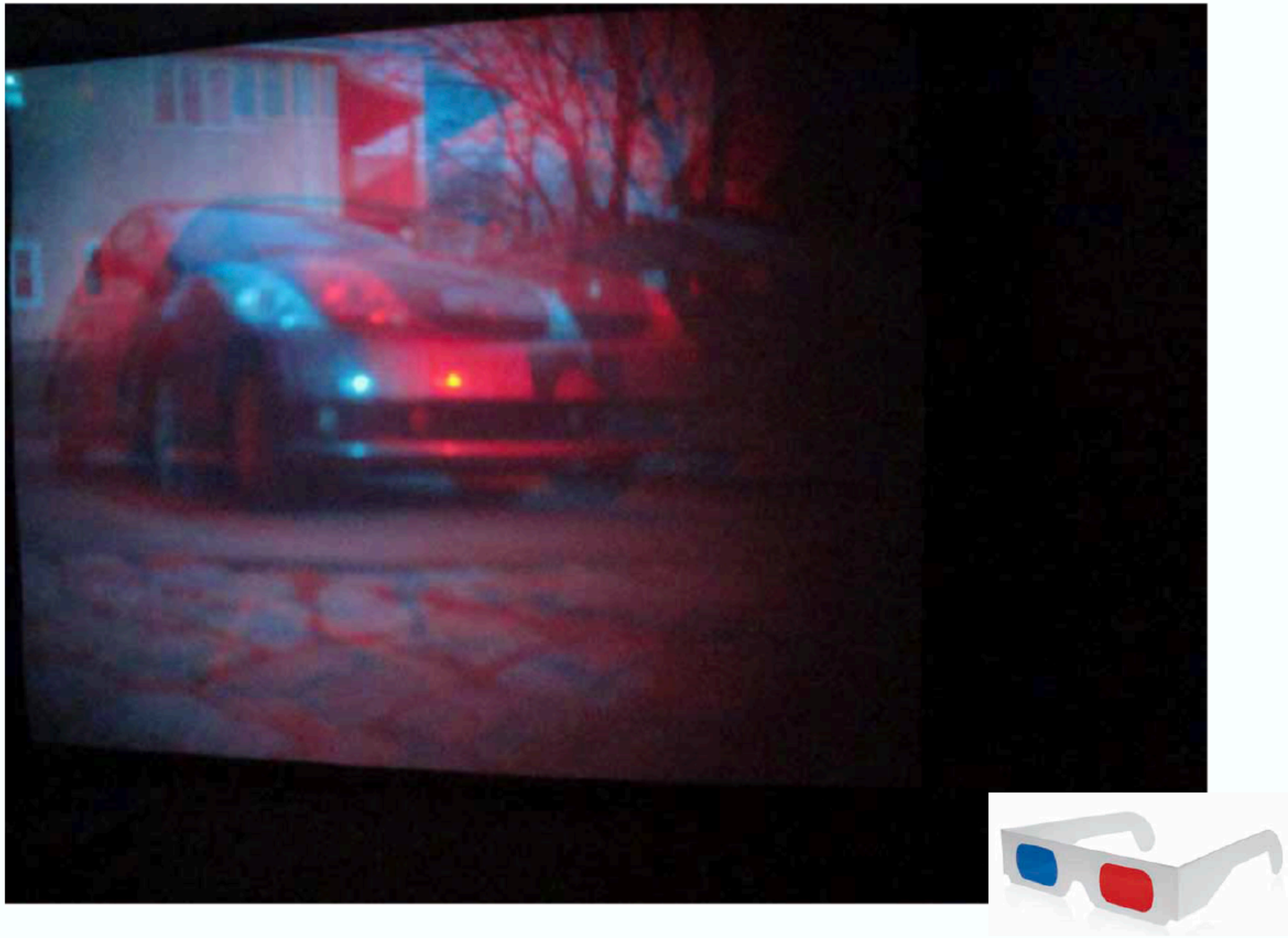


image of a point of light

Anaglyph pinhole camera



3D-Aware Scene Manipulation via Inverse Graphics

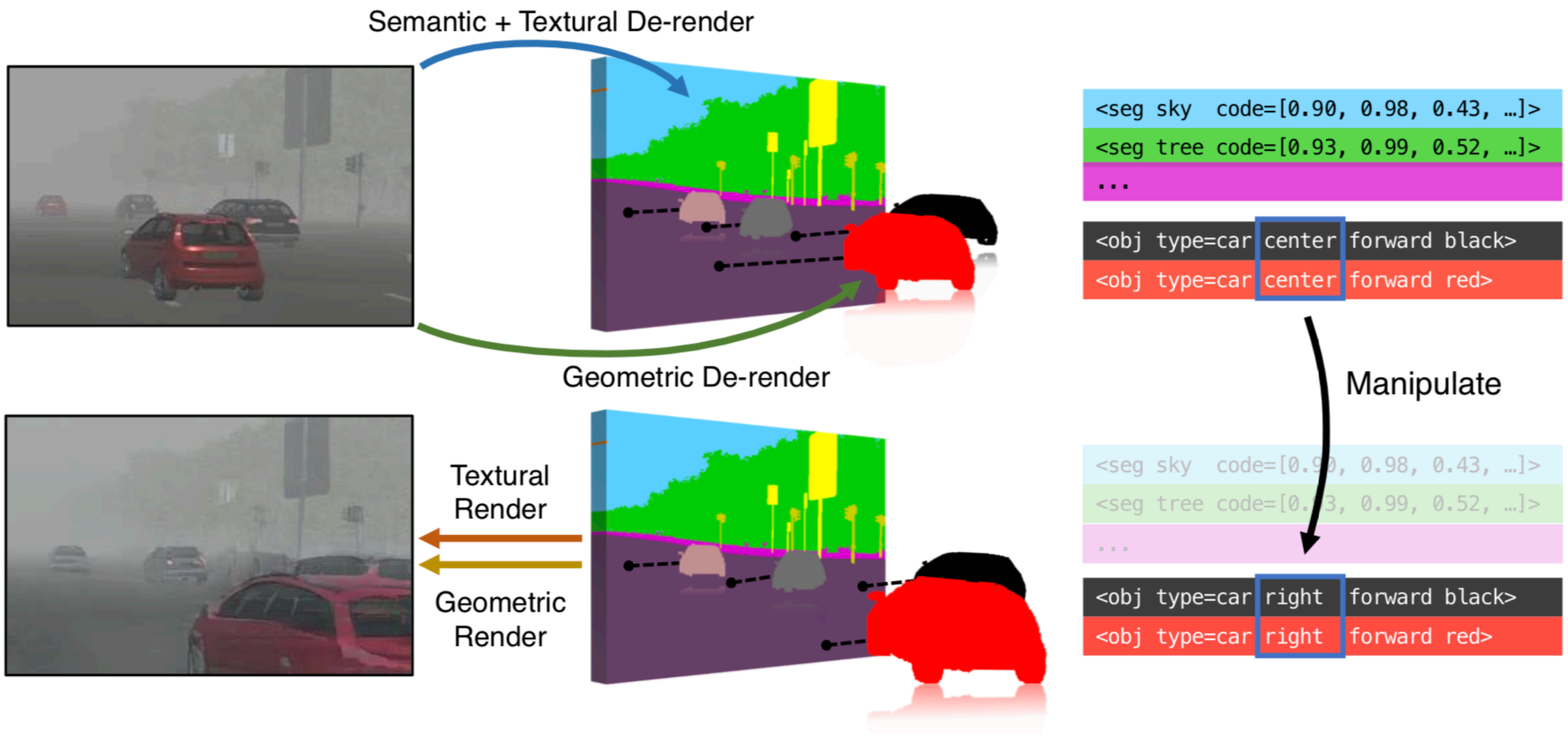


Figure 1: We propose to learn a holistic scene representation that encodes scene semantics as well as 3D and textural information. Our encoder-decoder framework learns disentangled representations for image reconstruction and 3D-aware image editing. For example, we can move cars to various locations with new 3D poses.

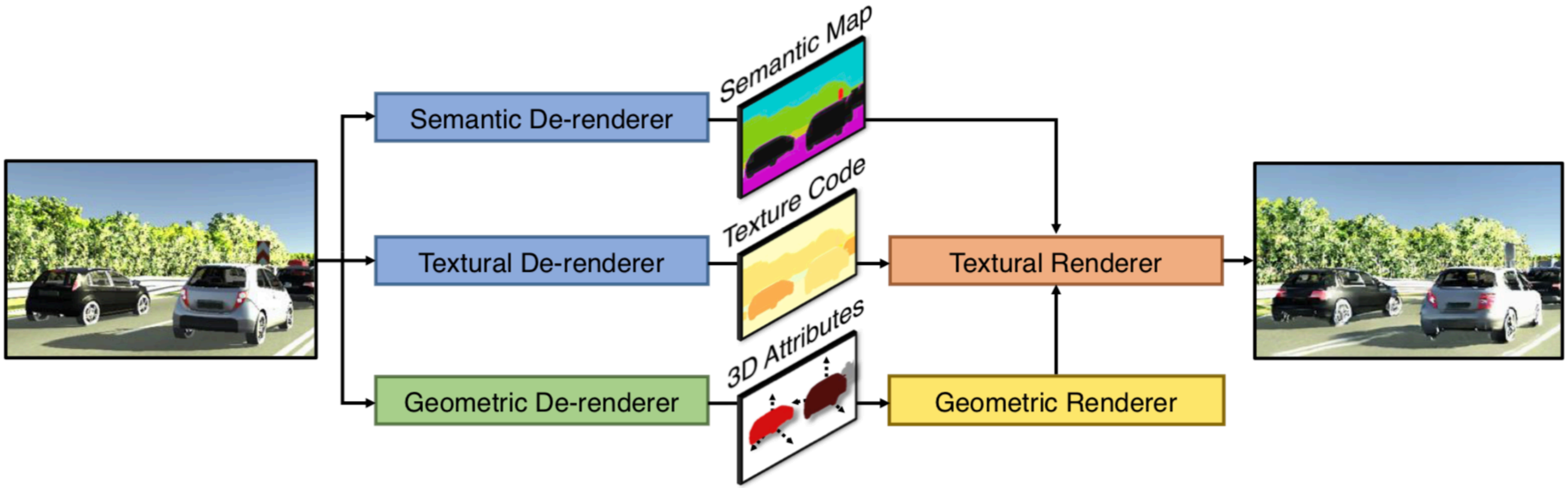
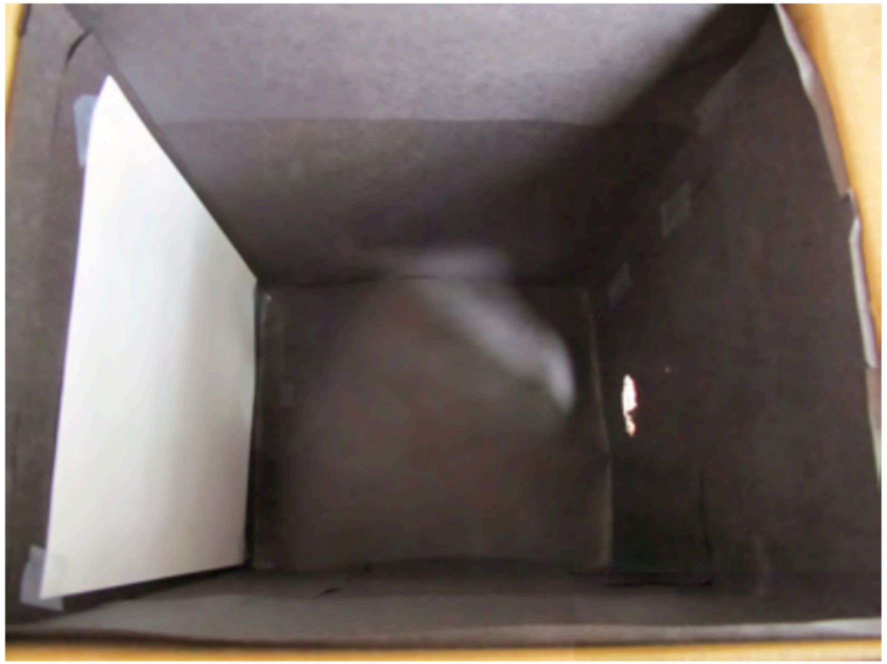


Figure 2: **Framework overview.** The de-renderer (encoder) consists of a semantic-, a textural- and a geometric branch. The textural renderer and geometric renderer then learn to reconstruct the original image from the representations obtained by the encoder modules.

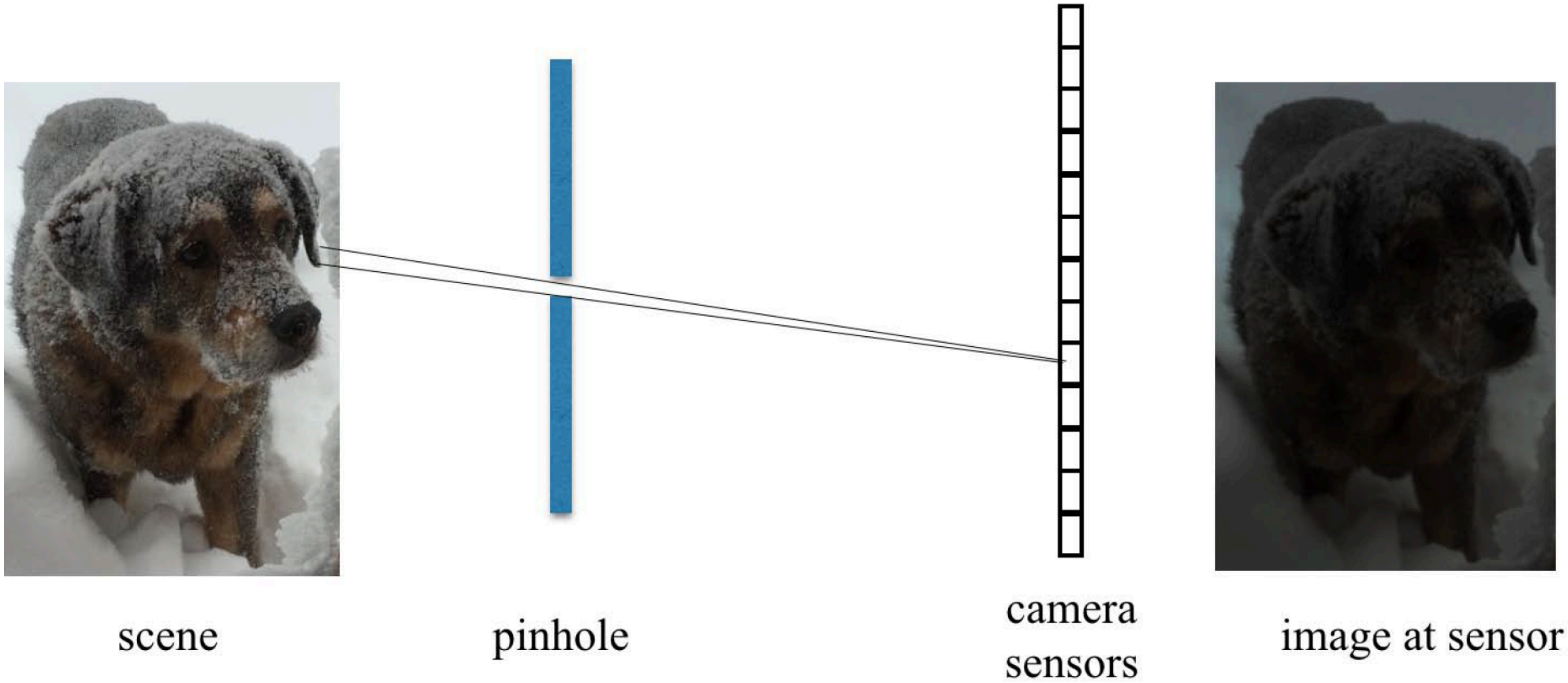
Problem set 2: Make your own pin-hole camera

- Build the camera: e.g. shoe box, make sure no light is leaking through.
- Decide where to put the hole: Make sure that the distance between the pinhole face and the screen face -- the focal length of your pinhole camera -- is not less than the minimum focus distance of your camera.
- Make pinkhole(s): I recommend cutting a larger hole in the cardboard box and attaching a thinner sheet of card stock, and having holes in the card stock.
- Cover the inside wall facing the pinhole with white copy paper.
- Create a hole for the digital camera

Problem set 2: Make your own pin-hole camera

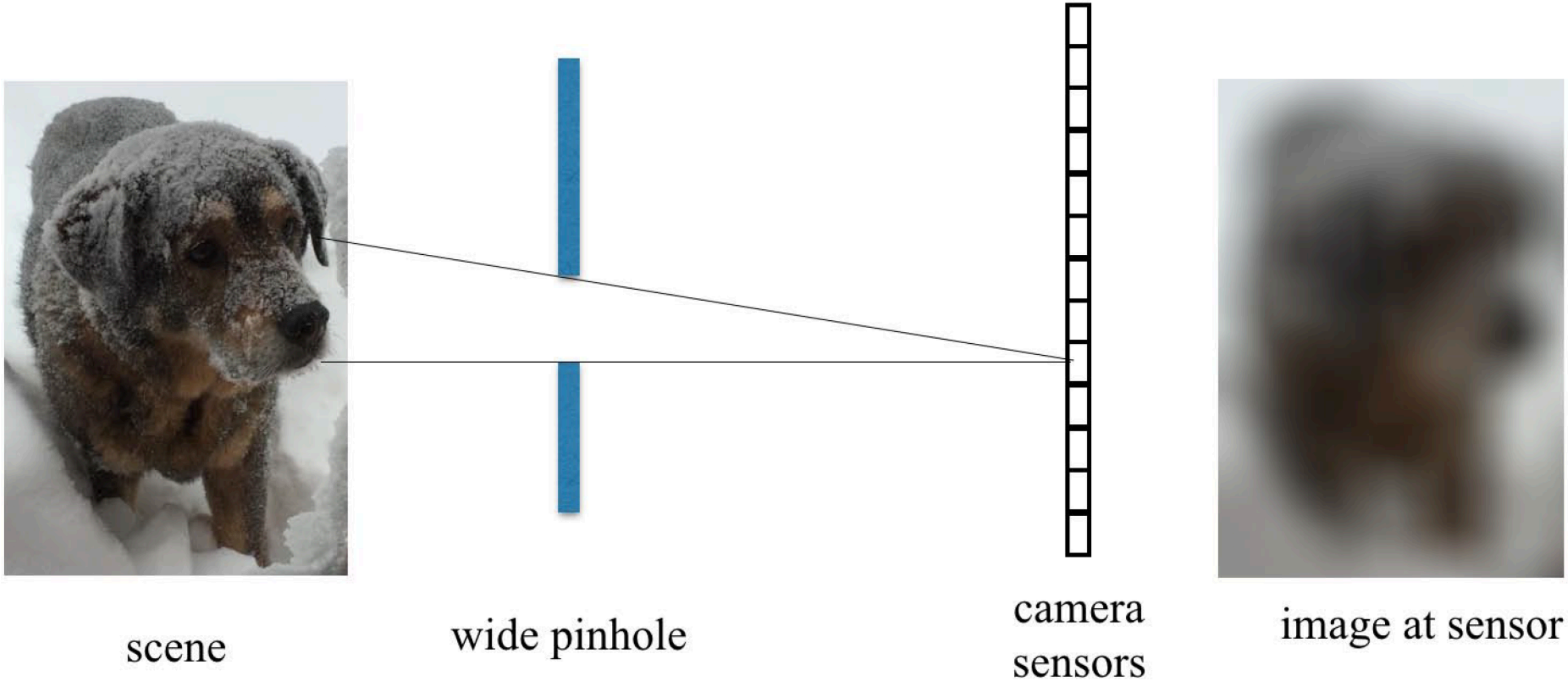


A problem: pinhole camera images are dark, or require long exposures

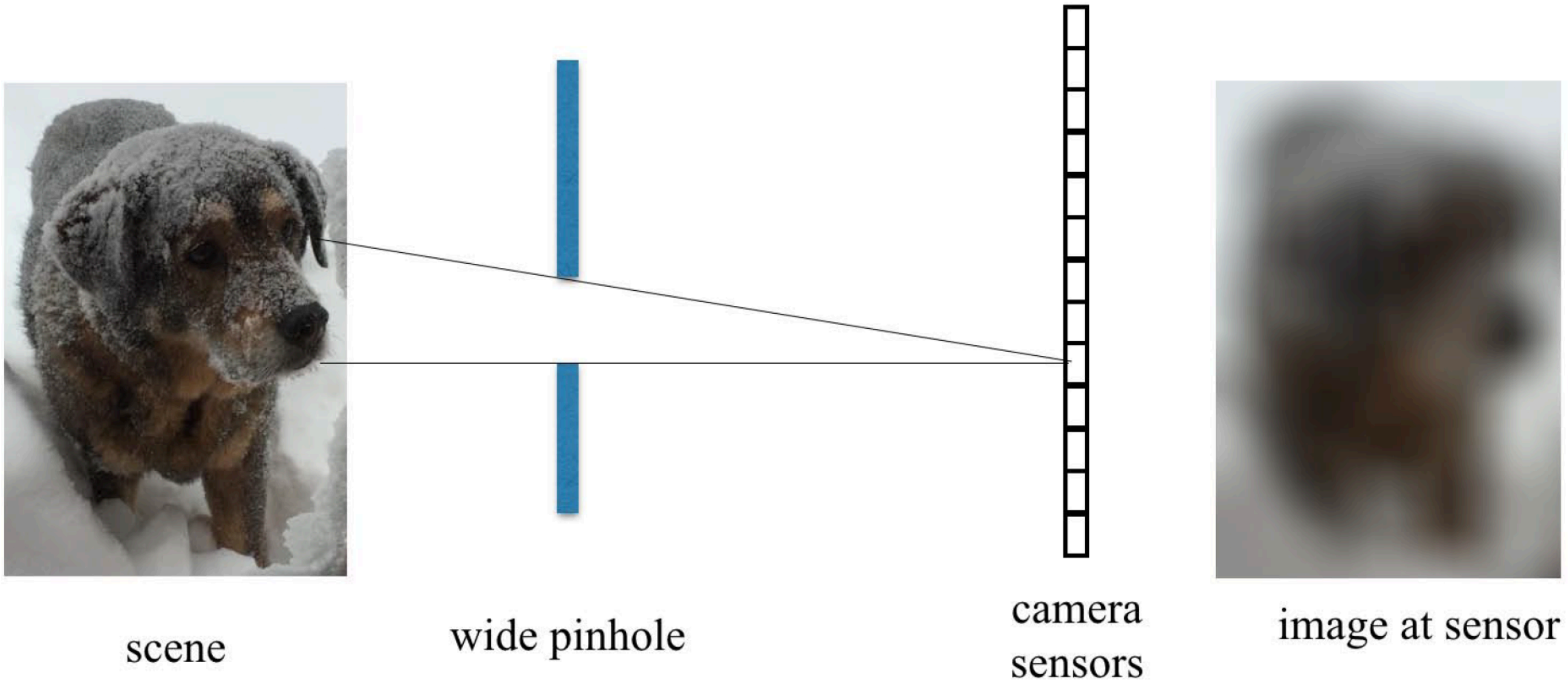


2019

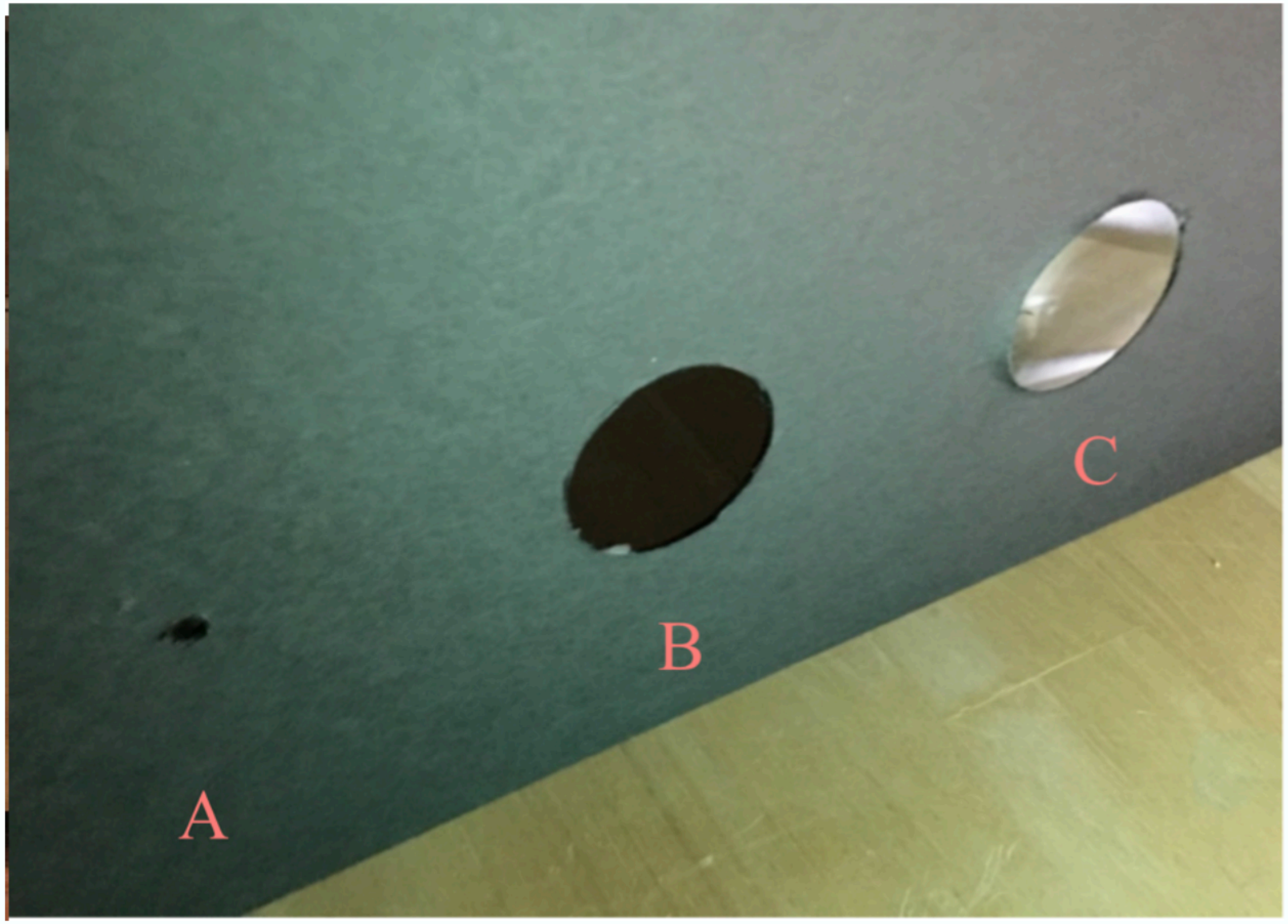
Large aperture gives a brighter image, but at the price of sharpness



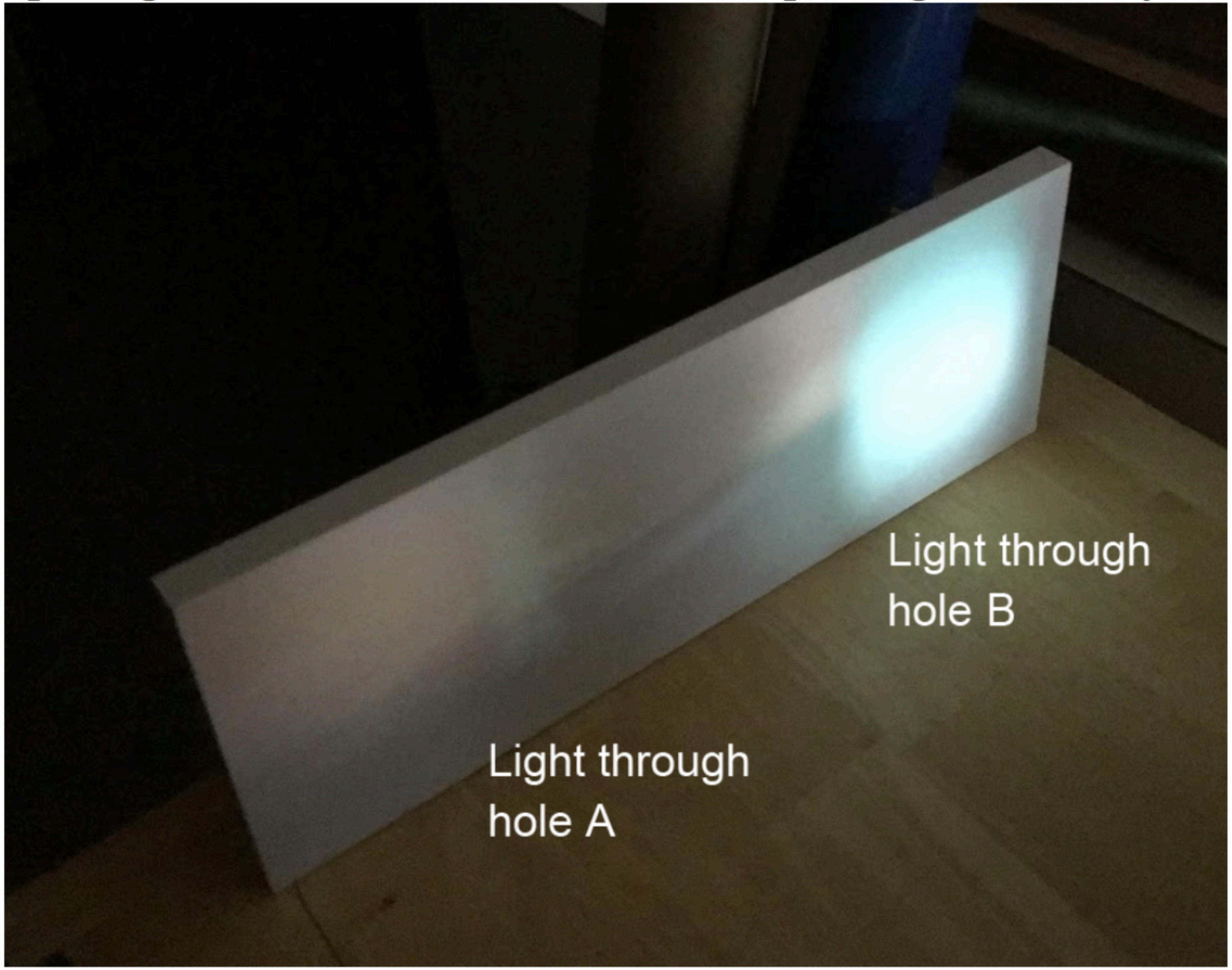
Large aperture gives a brighter image, but at the price of sharpness



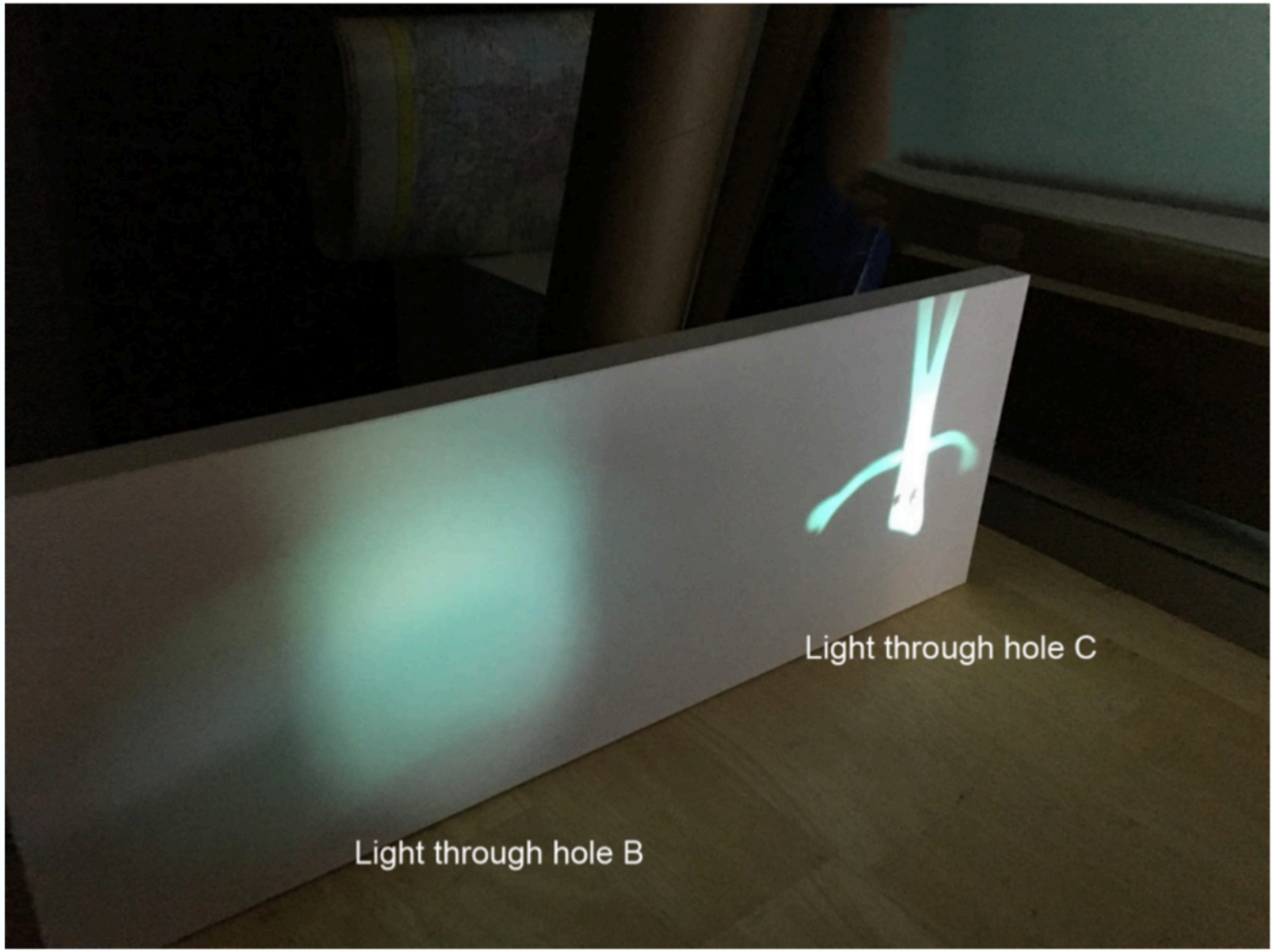
Let's try putting different occluders in betwee[n] the scene and the sensor plane



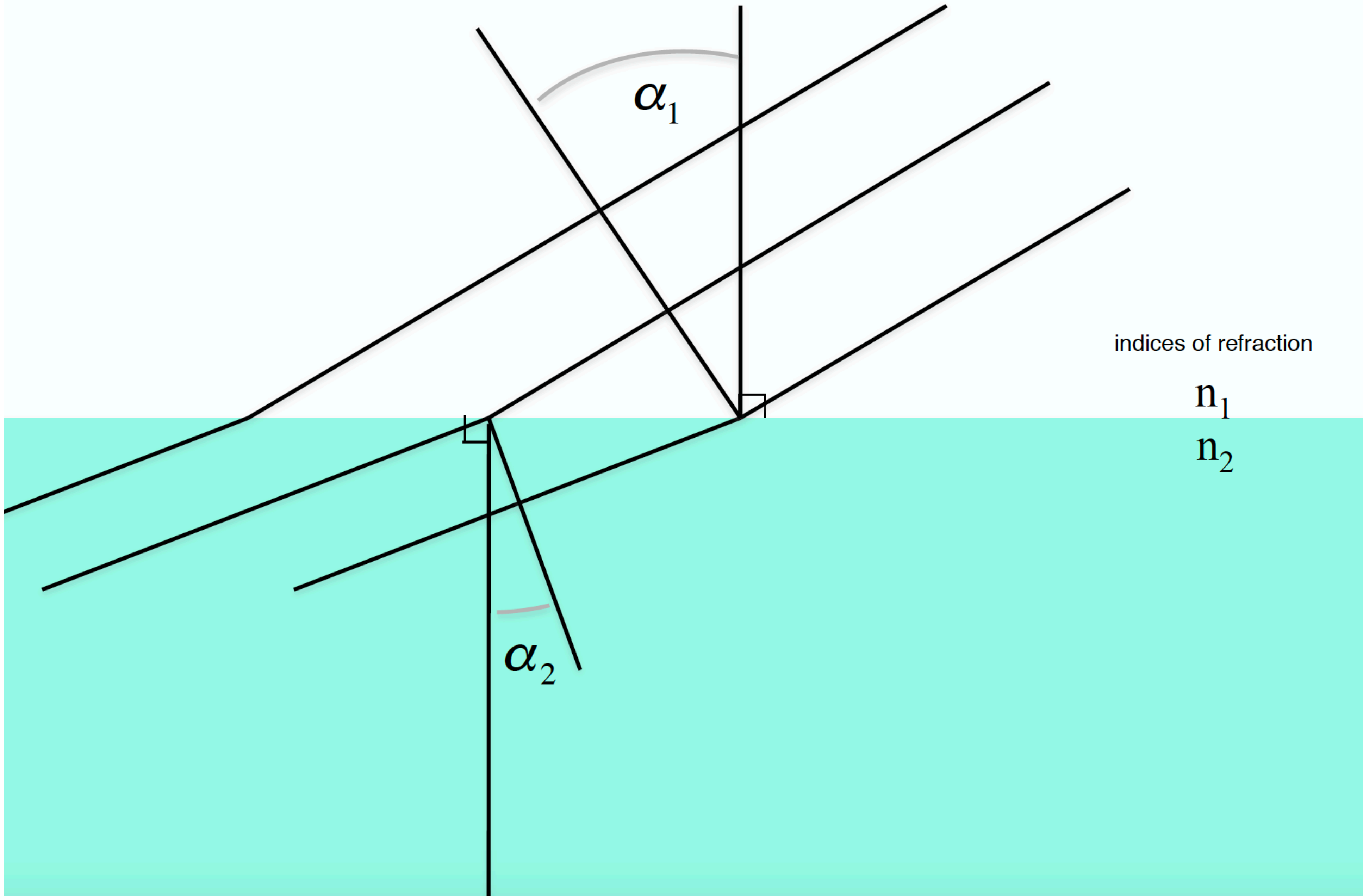
Influence of aperture size: with a small aperture, the image is sh
but dim. A large aperture gives a bright, but blurry image.



Images through large aperture, with and without lens present



Light at a material interface



Light at a material interface

$$\lambda_1 = \frac{c}{\omega\, n_1}$$ wavelength is speed/ freq

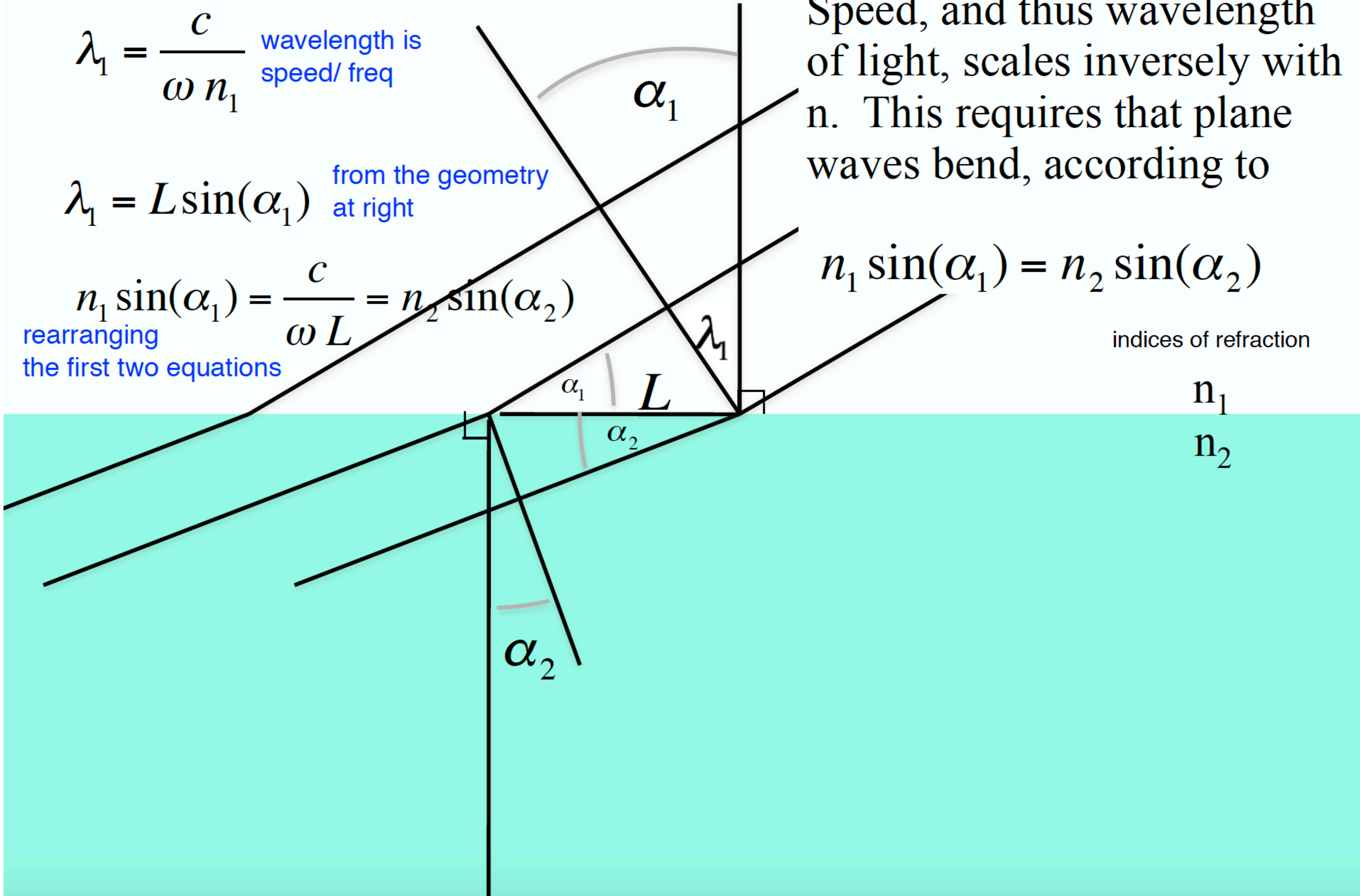
$$\lambda_1 = L\sin(\alpha_1)$$ from the geometry at right

$$n_1 \sin(\alpha_1) = \frac{c}{\omega\, L} = n_2 \sin(\alpha_2)$$

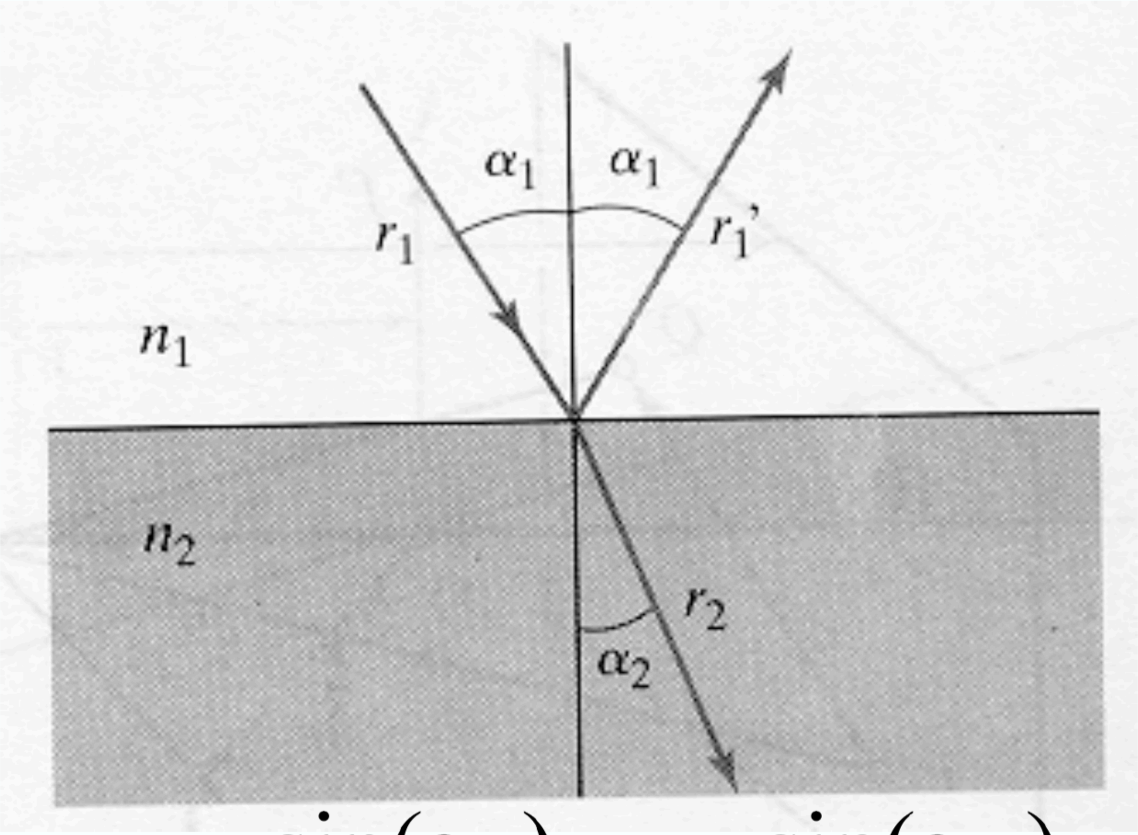
rearranging the first two equations

Speed, and thus wavelength of light, scales inversely with n. This requires that plane waves bend, according to

$$n_1 \sin(\alpha_1) = n_2 \sin(\alpha_2)$$



Refraction: Snell's law



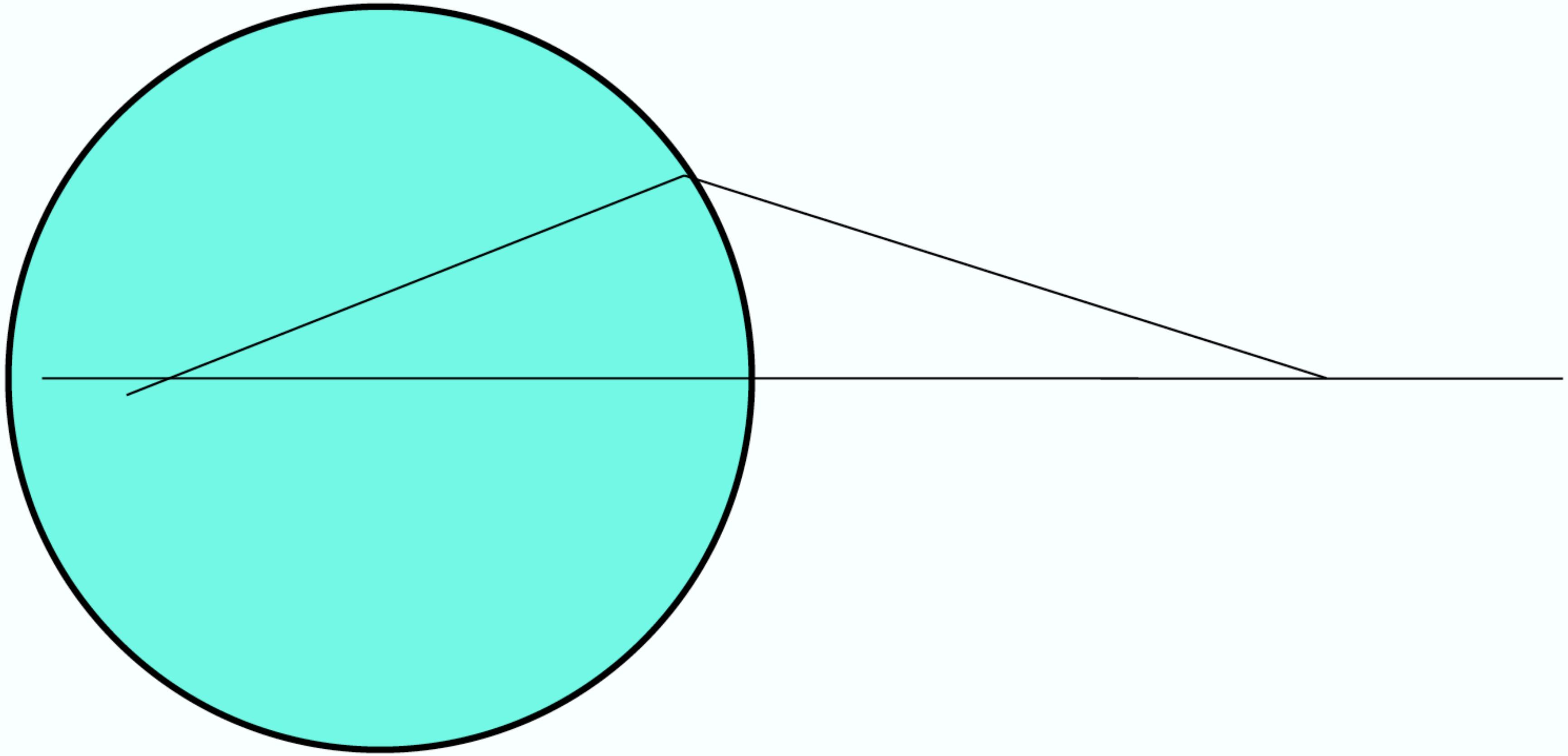
$$n_1 \sin(\alpha_1) = n_2 \sin(\alpha_2)$$

For small angles, $n_1\alpha_1 \approx n_2\alpha_2$

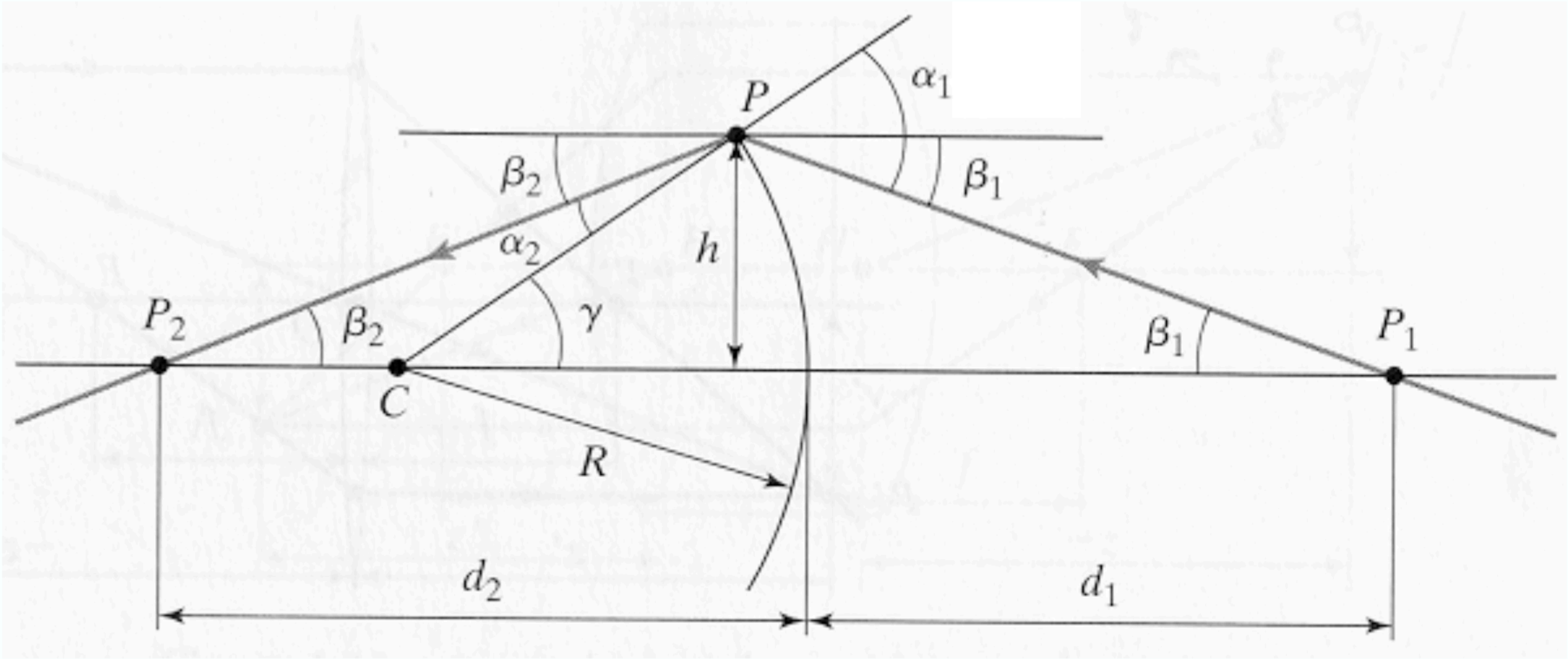
Snell’s law video

https://www.khanacademy.org/science/physics/geometric-optics/reflection-refraction/v/refraction-and-snell-s-law

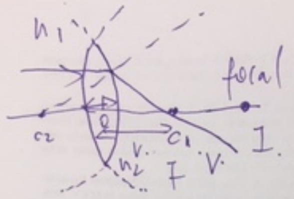
Spherical lens



For a spherical lens surface, we can define the relevant angles, apply Snell's law, and find an expression telling how the lens focusses light



Pause for white-boarding



Curve's surface refraction formula.

$$\frac{R.M}{v} - \frac{I.M}{u} = \frac{RM. - IM}{R.}$$

u → object dim

$$\frac{n_2}{v} - \frac{n_1}{\infty} = \frac{n_2 - n_1}{R_1} \quad (1)$$

$$\frac{n_1}{f} - \frac{n_2}{v} = \frac{n_1 - n_2}{R_2} \quad (2).$$

$$\frac{n_1}{f} - \frac{n_1}{\infty} = \frac{n_2 - n_1}{R_1} + \frac{n_1 - n_2}{R_1}$$

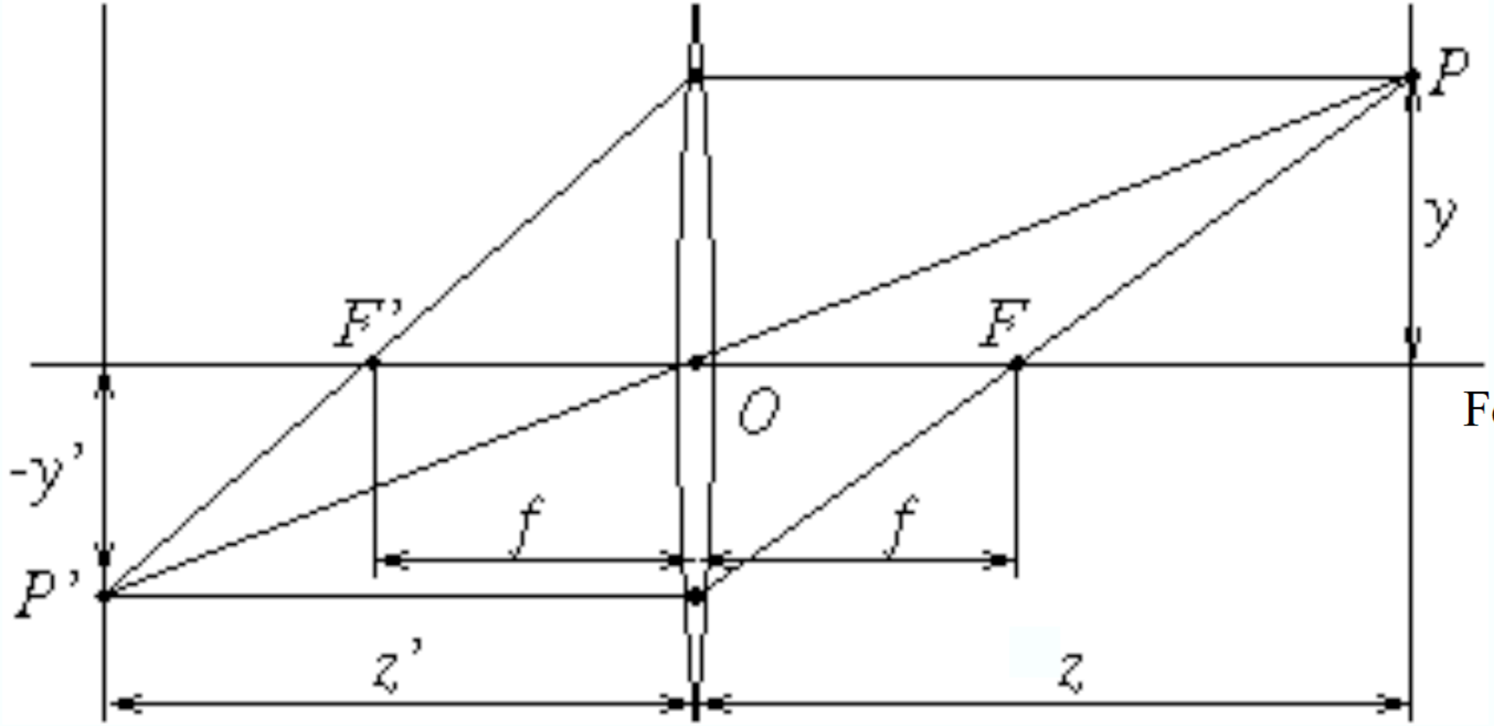
$$\frac{n_1}{f} = \frac{n_2 - n_1}{R_1} + \frac{n_1 - n_2}{R_2}.$$

$$\boxed{\frac{1}{f} = \frac{(n_2 - n_1)\left[\frac{1}{R_1} - \frac{1}{R_2}\right]}{n_1}}$$

lens's maker

$$= \left(\frac{n_2}{n_1} - 1\right)\left[\frac{1}{R_1} - \frac{1}{R_2}\right]$$

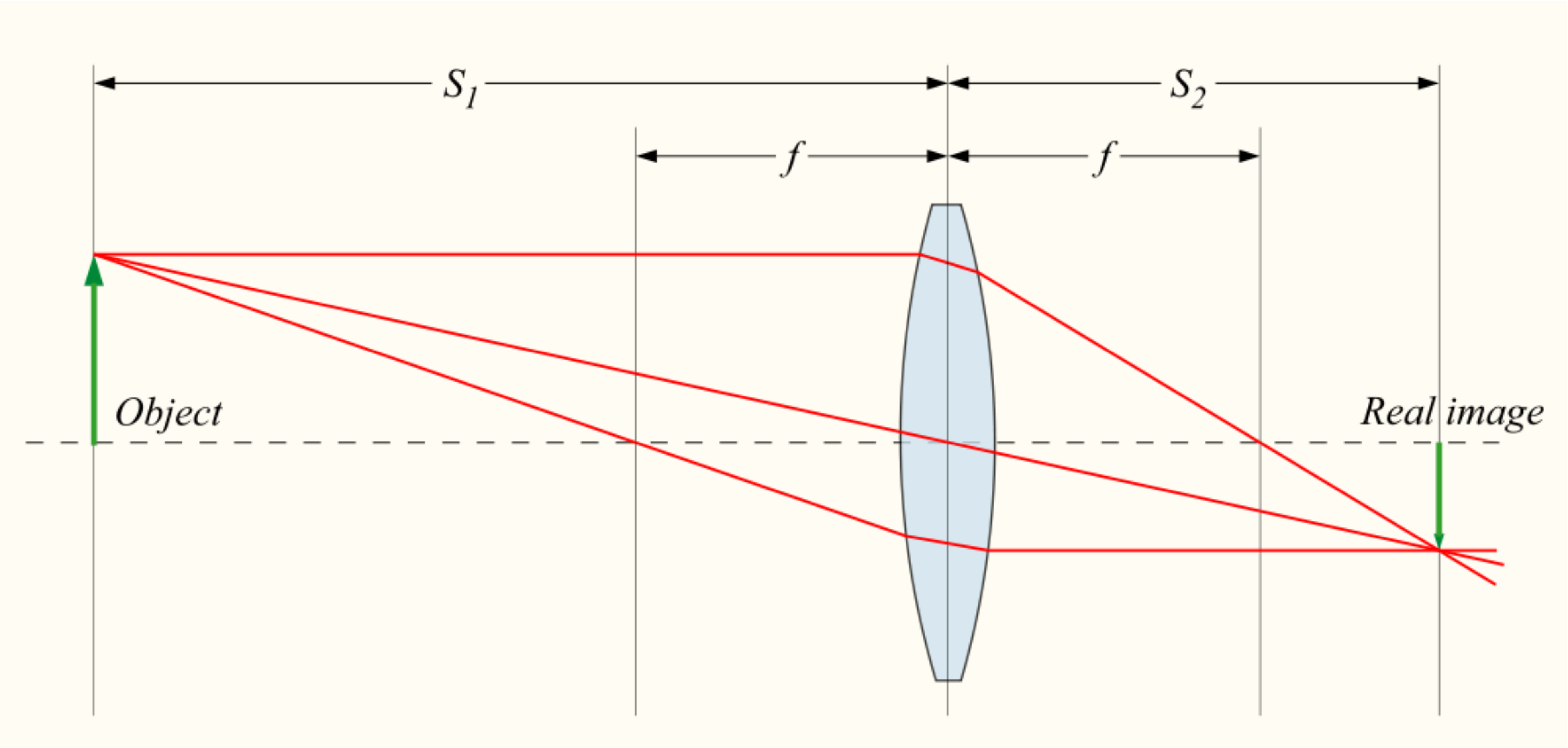
The thin lens, first order optics



Forsyth&Ponce

The lensmaker's equation:

$$\frac{1}{z'} + \frac{1}{z} = \frac{1}{f} \qquad f = \frac{R}{2(n-1)}$$

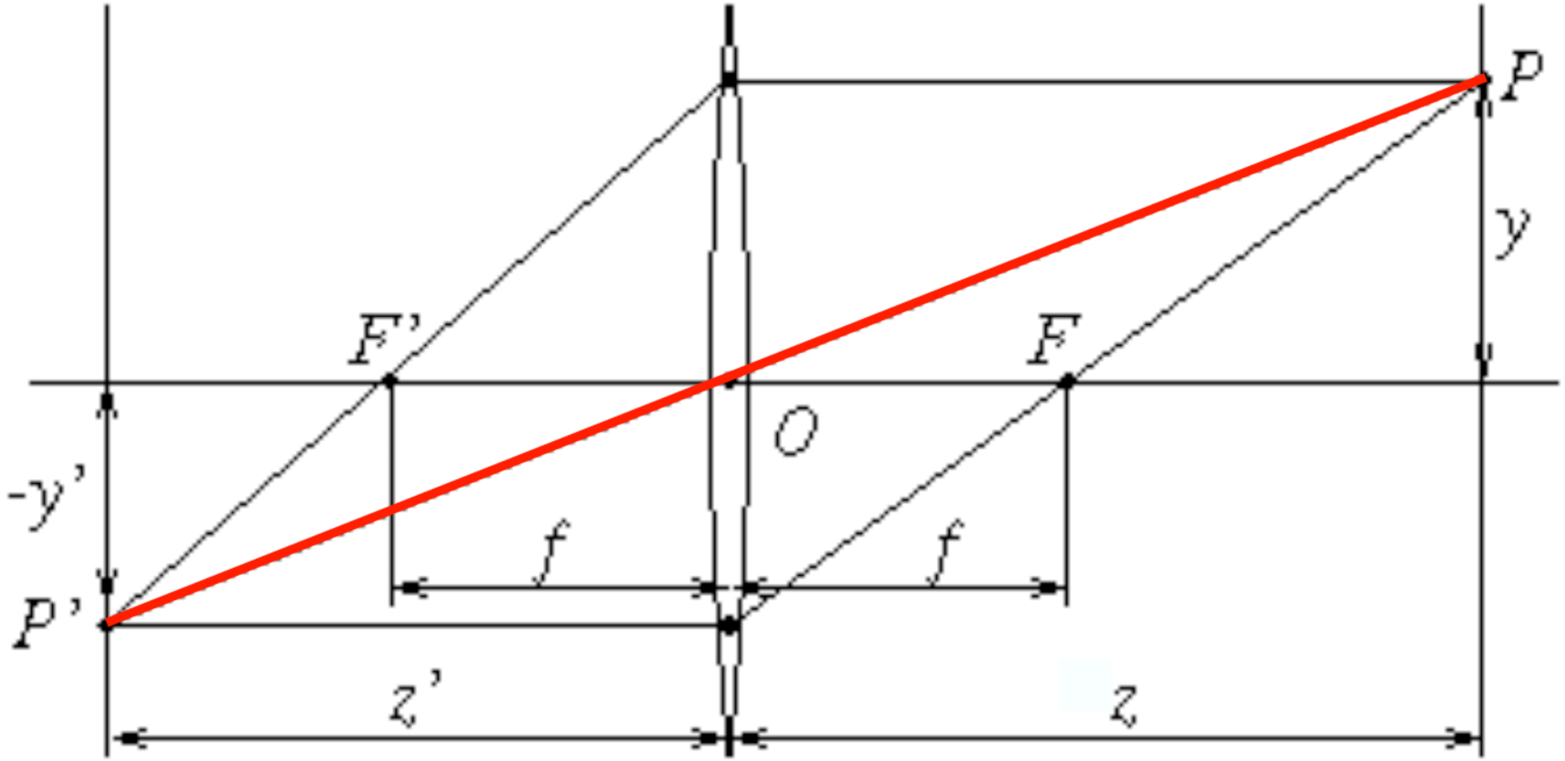


$$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}.$$

Len's maker's equation

How does the mapping from the 3-d world to the image plane compare for a lens and for a pinhole camera?

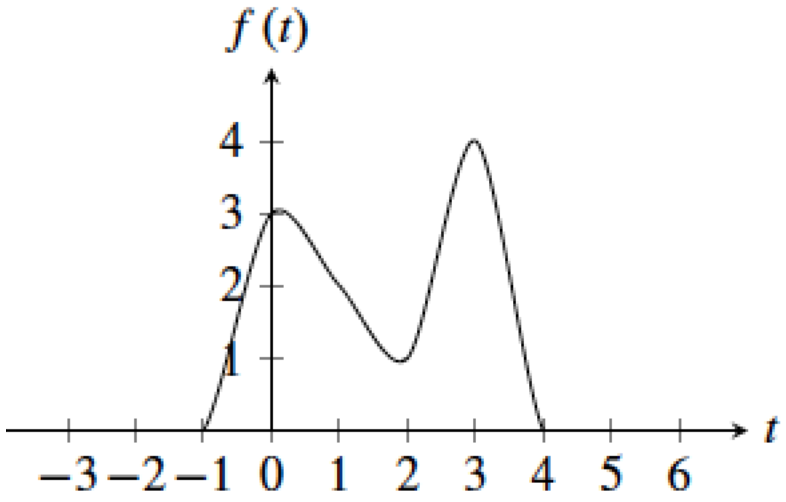
The perspective projection of a pinhole camera. But note that many more of the rays leaving from P arrive at P’



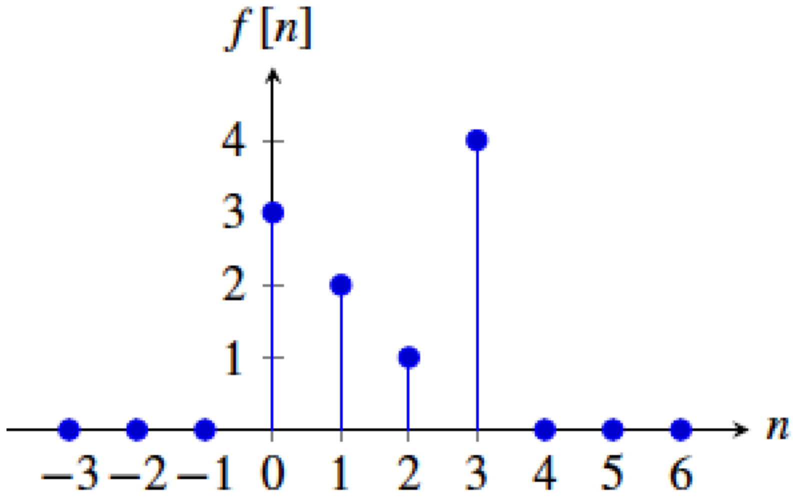
More on Lens maker's formula

- https://www.khanacademy.org/science/in-in-class-12th-physics-india/in-in-ray-optics-and-optical-instruments/in-in-refraction-at-curved-surfaces/v/lens-makers-formula

Signals and systems

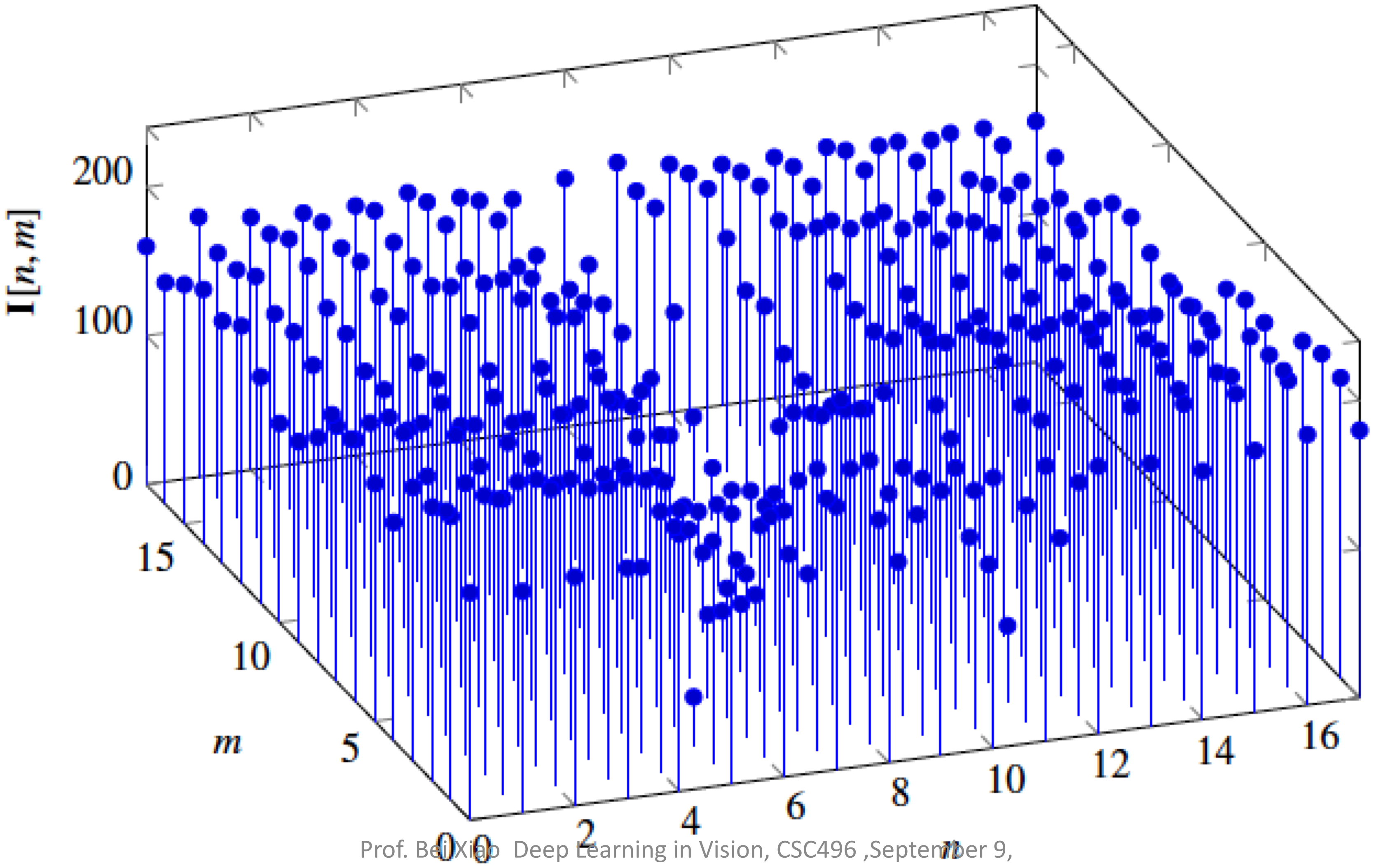


Time continuous signal

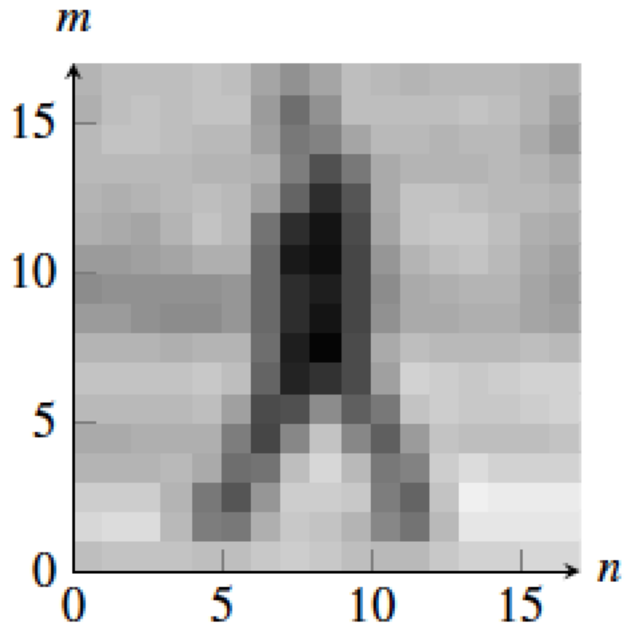


Time discrete signal

A 2D discrete signal



$$
\mathbf{I} = \left[\begin{array}{cccccccccccccccccc}
160 & 175 & 171 & 168 & 168 & 172 & 164 & 158 & 167 & 173 & 167 & 163 & 162 & 164 & 160 & 159 & 163 & 162 \\
149 & 164 & 172 & 175 & 178 & 179 & 176 & 118 & 97 & 168 & 175 & 171 & 169 & 175 & 176 & 177 & 165 & 152 \\
161 & 166 & 182 & 171 & 170 & 177 & 175 & 116 & 109 & 169 & 177 & 173 & 168 & 175 & 175 & 159 & 153 & 123 \\
171 & 174 & 177 & 175 & 167 & 161 & 157 & 138 & 103 & 112 & 157 & 164 & 159 & 160 & 165 & 169 & 148 & 144 \\
163 & 163 & 162 & 165 & 167 & 164 & 178 & 167 & 77 & 55 & 134 & 170 & 167 & 162 & 164 & 175 & 168 & 160 \\
173 & 164 & 158 & 165 & 180 & 180 & 150 & 89 & 61 & 34 & 137 & 186 & 186 & 182 & 175 & 165 & 160 & 164 \\
152 & 155 & 146 & 147 & 169 & 180 & 163 & 51 & 24 & 32 & 119 & 163 & 175 & 182 & 181 & 162 & 148 & 153 \\
134 & 135 & 147 & 149 & 150 & 147 & 148 & 62 & 36 & 46 & 114 & 157 & 163 & 167 & 169 & 163 & 146 & 147 \\
135 & 132 & 131 & 125 & 115 & 129 & 132 & 74 & 54 & 41 & 104 & 156 & 152 & 156 & 164 & 156 & 141 & 144 \\
151 & 155 & 151 & 145 & 144 & 149 & 143 & 71 & 31 & 29 & 129 & 164 & 157 & 155 & 159 & 158 & 156 & 148 \\
172 & 174 & 178 & 177 & 177 & 181 & 174 & 54 & 21 & 29 & 136 & 190 & 180 & 179 & 176 & 184 & 187 & 182 \\
177 & 178 & 176 & 173 & 174 & 180 & 150 & 27 & 101 & 94 & 74 & 189 & 188 & 186 & 183 & 186 & 188 & 187 \\
160 & 160 & 163 & 163 & 161 & 167 & 100 & 45 & 169 & 166 & 59 & 136 & 184 & 176 & 175 & 177 & 185 & 186 \\
147 & 150 & 153 & 155 & 160 & 155 & 56 & 111 & 182 & 180 & 104 & 84 & 168 & 172 & 171 & 164 & 168 & 167 \\
184 & 182 & 178 & 175 & 179 & 133 & 86 & 191 & 201 & 204 & 191 & 79 & 172 & 220 & 217 & 205 & 209 & 200 \\
184 & 187 & 192 & 182 & 124 & 32 & 109 & 168 & 171 & 167 & 163 & 51 & 105 & 203 & 209 & 203 & 210 & 205 \\
191 & 198 & 203 & 197 & 175 & 149 & 169 & 189 & 190 & 173 & 160 & 145 & 156 & 202 & 199 & 201 & 205 & 202 \\
153 & 149 & 153 & 155 & 173 & 182 & 179 & 177 & 182 & 177 & 182 & 185 & 179 & 177 & 167 & 176 & 182 & 180
\end{array}\right]
$$



A tiny person of 18 x 18 pixels

Signal / image space

Scalar product between two signals *f*, *g* :

$$\langle f, g \rangle = \sum_{n=0}^{N-1} f[n]\, g^*[n] = f^T g^*$$

L2 norm of *f* :

$$E_f = \|f\|^2 = \langle f, f \rangle = \sum_{n=0}^{N-1} |f[n]|^2 = f^T f^*$$

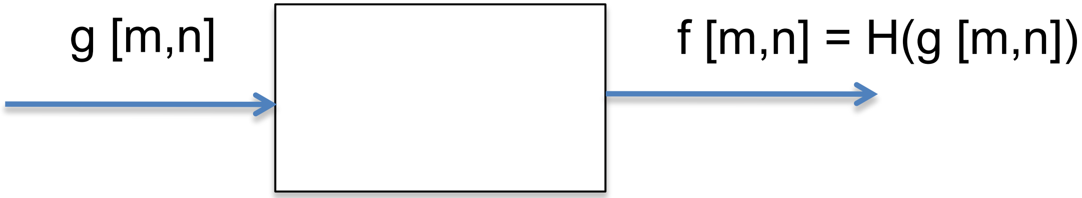
Distance between two signals *f*, *g* :

$$d_{f,g}^2 = \|f - g\|^2 = \sum_{n=0}^{N-1} |f[n] - g[n]|^2 = E_f + E_g - 2\langle f, g \rangle$$

Filtering

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

Linear filtering



For a filter to be linear, it has to verify:

$$f[m,n] = H(a[m,n] + b[m,n]) = H(a[m,n]) + H(b[m,n])$$

$$f[m,n] = H(C\,a[m,n]) = C\,H(a[m,n])$$

Linear filtering

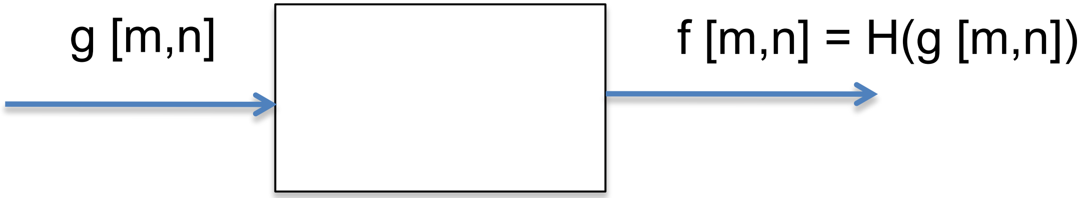
A linear filter in its most general form can be written as,
in 1D for a signal of length N:

$$f[n] = \sum_{k=0}^{N-1} h[n,k]\, g[k]$$

It is useful to make it more explicit by writing:

$$\begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[M] \end{bmatrix} = \begin{bmatrix} h[0,0] & h[0,1] & \dots & h[0,N] \\ h[1,0] & h[1,1] & \dots & h[1,N] \\ \vdots & \vdots & \vdots & \vdots \\ h[M,N] & h[M,1] & \dots & h[M,N] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ \vdots \\ g[N] \end{bmatrix}$$

Linear filtering



In 2D:

$$f[n,m] = \sum_{k,\,l=0}^{N-1,\,M-1} h[n,m,k,l]\,g[k,l]$$

Which can also be written in matrix form as in the 1D case:

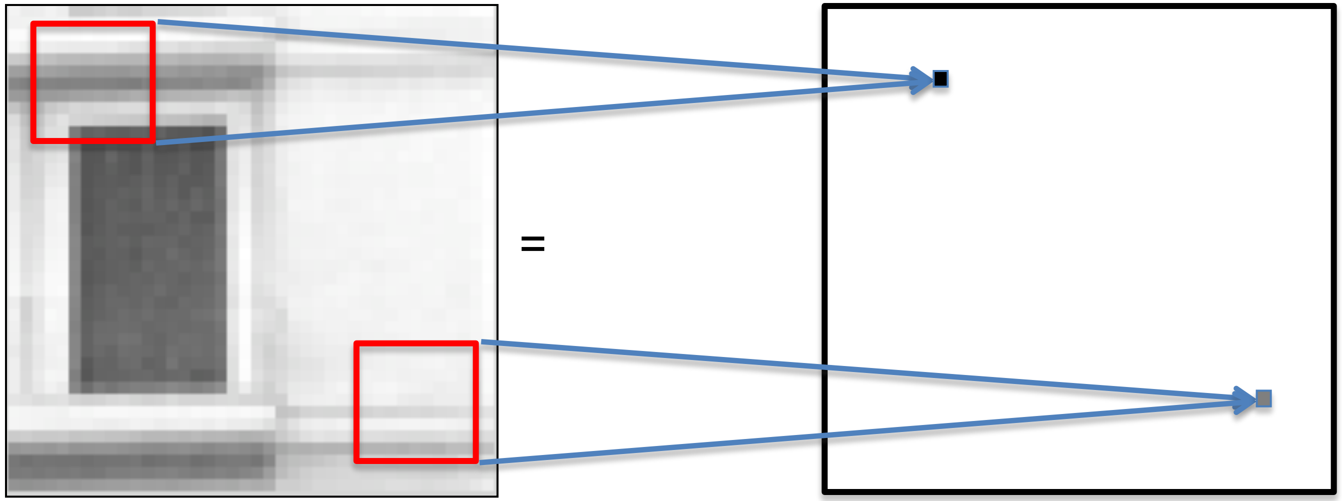
Credit picture: Fredo Durand

$$\begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[M] \end{bmatrix} = \begin{bmatrix} h[0,0] & h[0,1] & \dots & h[0,N] \\ h[1,0] & h[1,1] & \dots & h[1,N] \\ \vdots & \vdots & \vdots & \vdots \\ h[M,N] & h[M,1] & \dots & h[M,N] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ \vdots \\ g[N] \end{bmatrix}$$

Why should one pixel be treated differently than any another?

Credit picture: Fredo Durand

A translation invariant filter

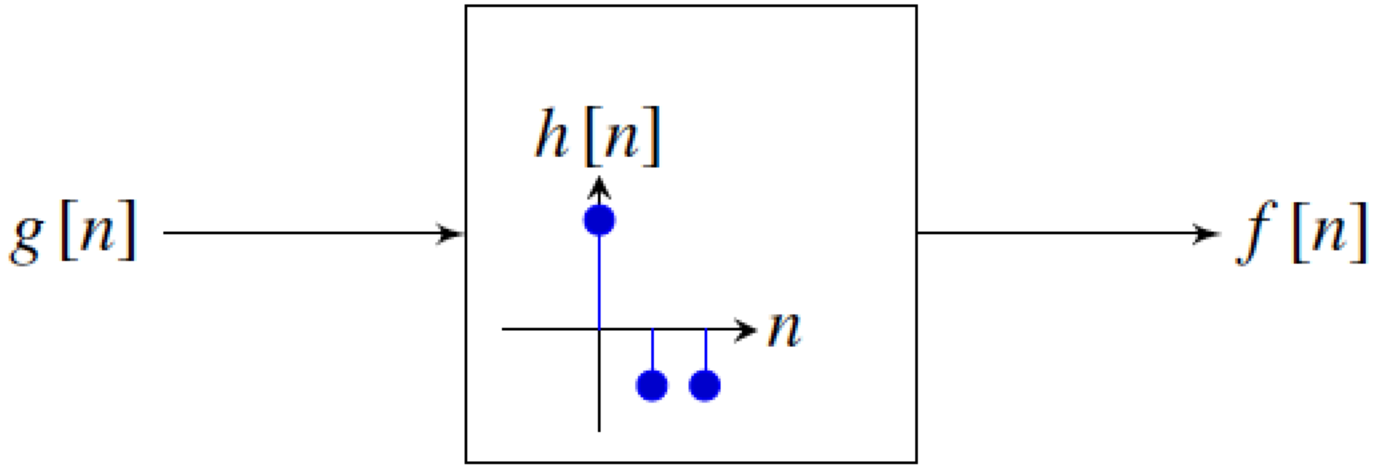


The same weighting occurs within each window

Convolution

$$f[n] = h \circ g = \sum_{k=0}^{N-1} h[n-k]\, g[k]$$

For example: h = [2, -1, -1]



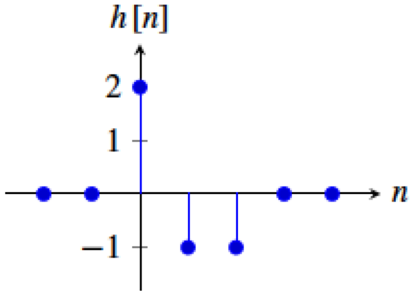
Pause for white-boarding

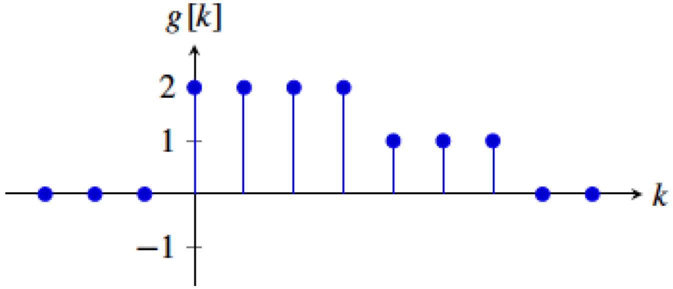
Convolution

In the 1D case, it helps to make explicit the structure of the matrix:

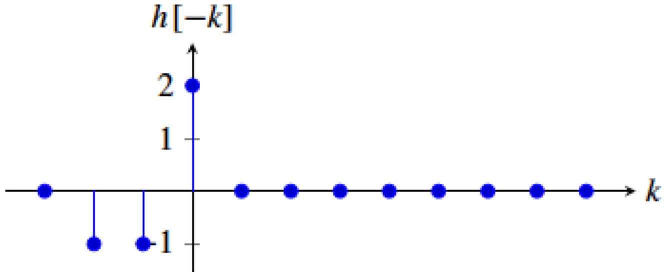
$$\begin{bmatrix} f[0] \\ f[1] \\ f[2] \\ \vdots \\ f[N] \end{bmatrix} = \begin{bmatrix} h[0] & h[-1] & h[-2] & \dots & h[-N] \\ h[1] & h[0] & h[-1] & \dots & h[1-N] \\ h[2] & h[1] & h[0] & \dots & h[2-N] \\ \vdots & \vdots & \vdots & \vdots & \\ h[N] & h[N-1] & h[N-2] & \dots & h[0] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ g[2] \\ \vdots \\ g[N] \end{bmatrix}$$

Convolution



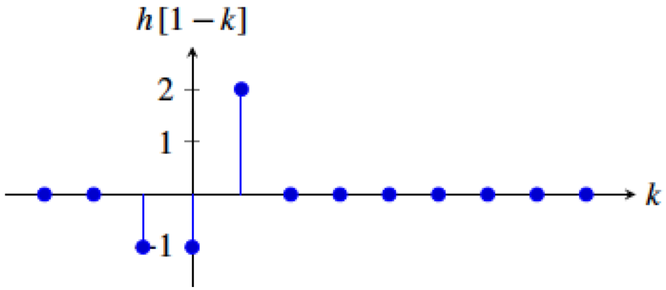


$f[0] = \sum_k h[-k]\, g[k]$



$f[0] = 4$

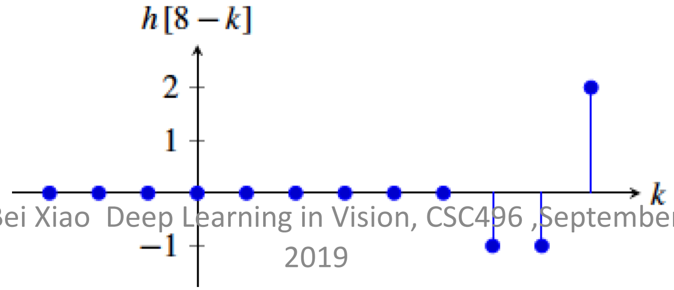
$f[1] = \sum_k h[1-k]\, g[k]$



$f[1] = 2$

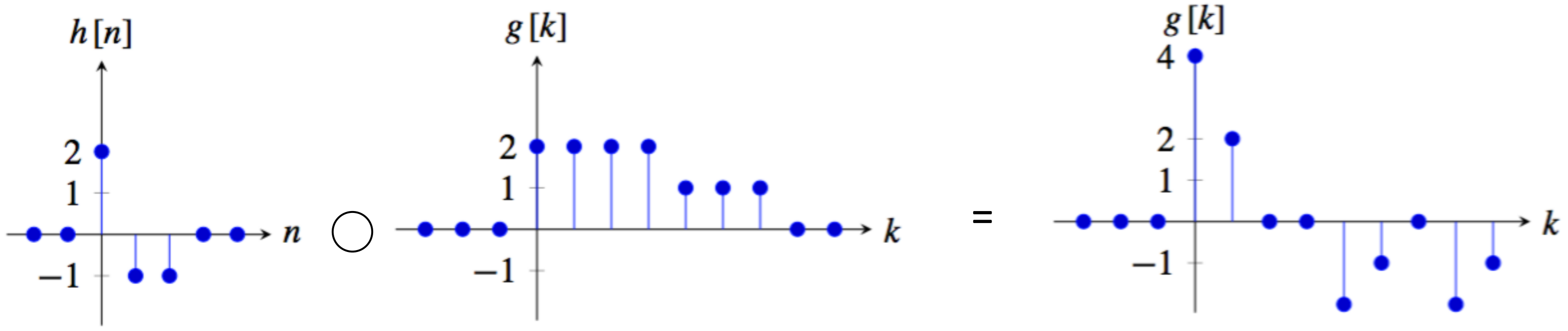
⋮

$f[8] = \sum_k h[8-k]\, g[k]$



$f[8] = -1$

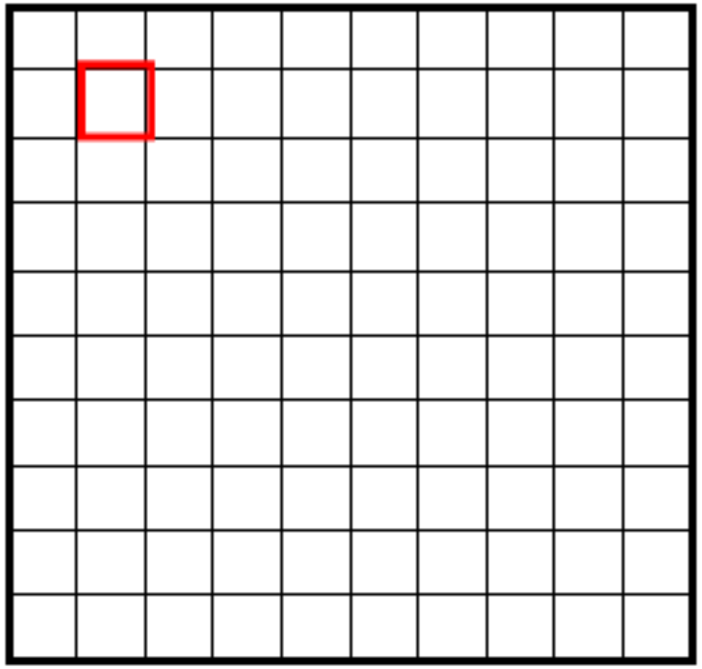
Convolution



2D convolution

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

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

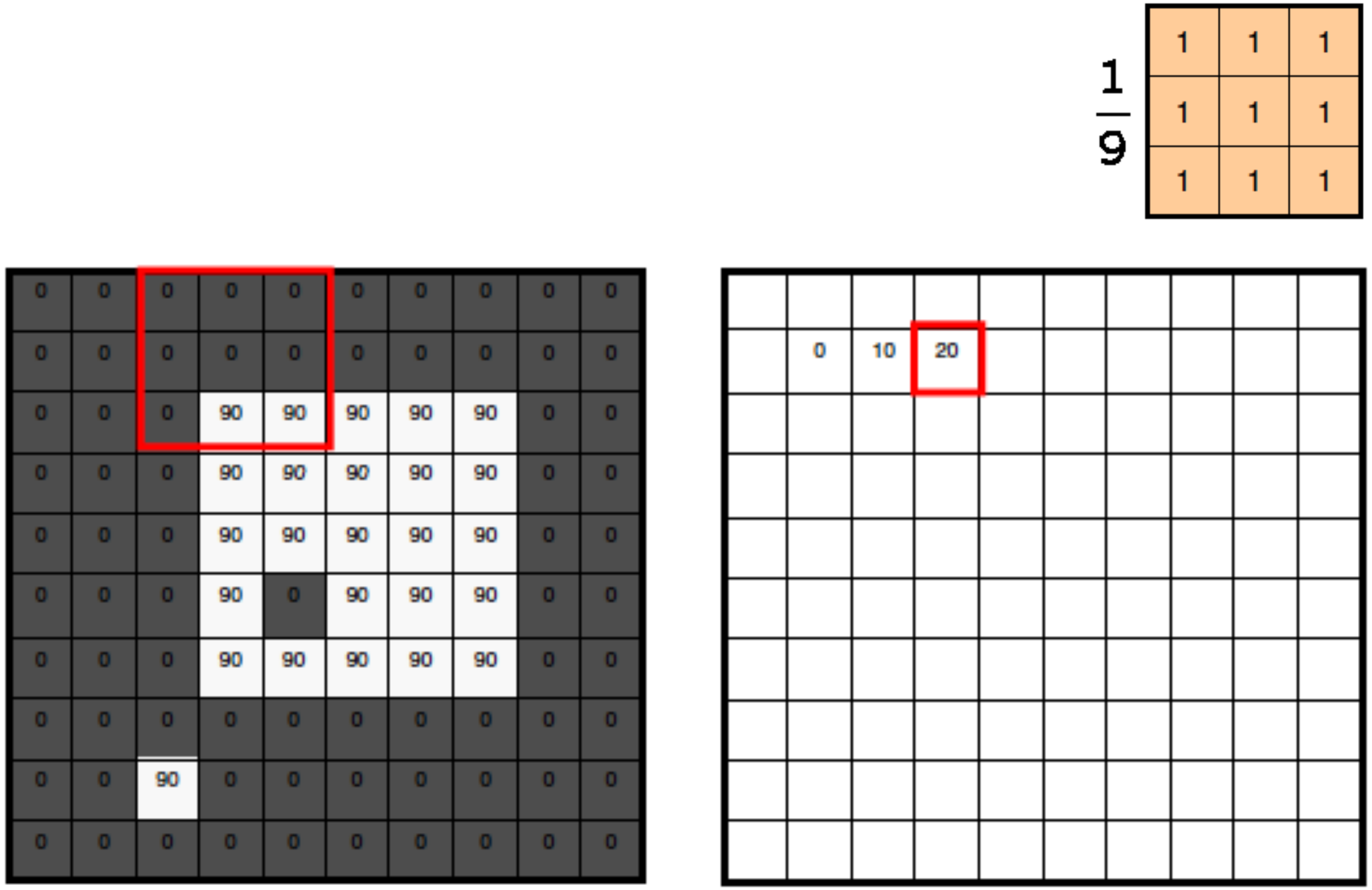


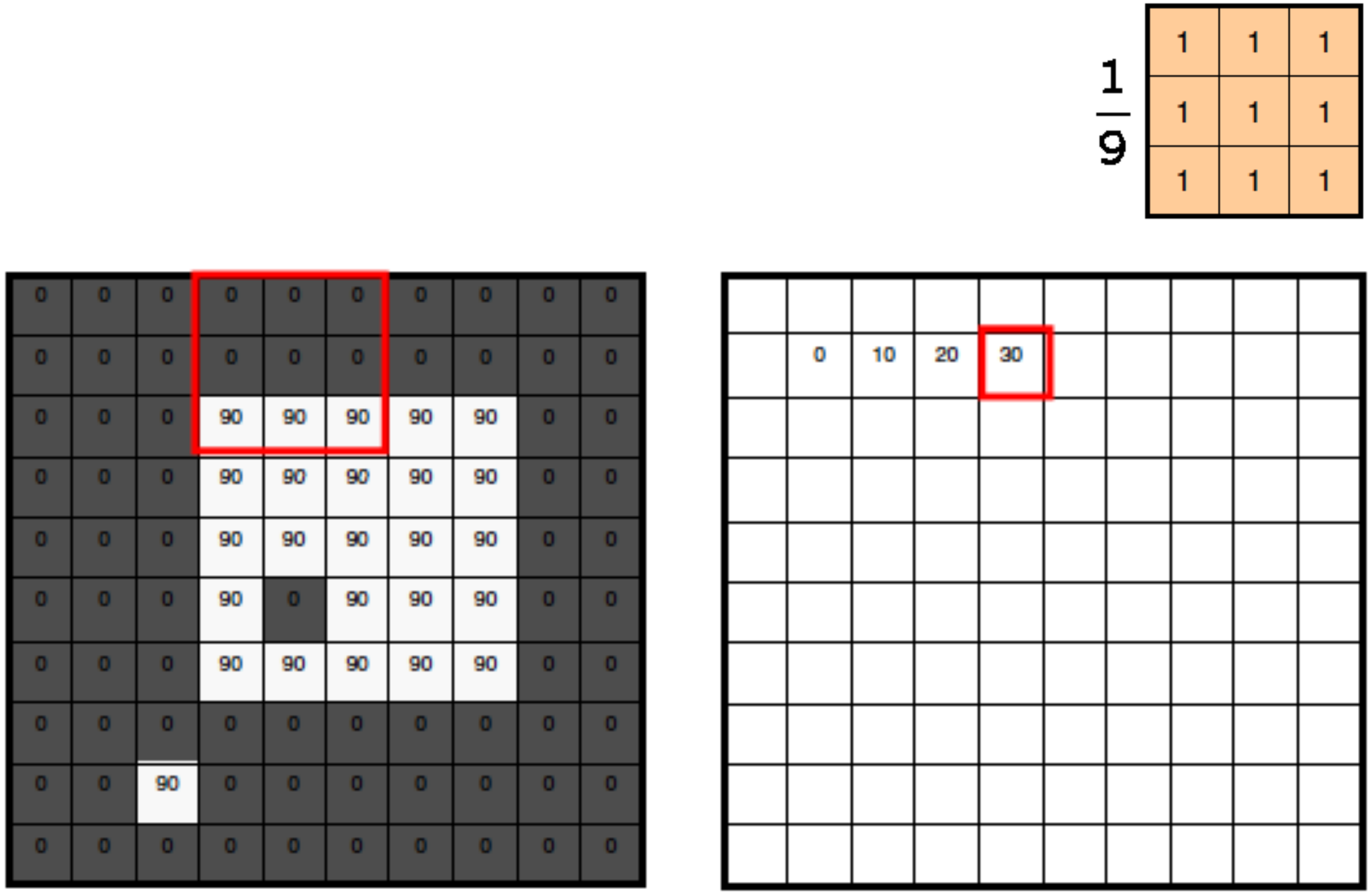
$$f[m,n] = h \circ g = \sum_{k,l} h[m-k, n-l]\, g[k,l]$$

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

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

	0	10							


1/9
1 1 1
1 1 1
1 1 1
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
0 10 20


1
9
1 1 1
1 1 1
1 1 1
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
0 10 20 30

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

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

	0	10	20	30	30				

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

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

	0	10	20	30	30				
				?					

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

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

	0	10	20	30	30				
						?			
				50					

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

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

	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	

What does it do?

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

2D convolution

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

m=0 1 2 …

111	115	113	111	112	111	112	111
135	138	137	139	145	146	149	147
163	168	188	196	206	202	206	207
180	184	206	219	202	200	195	193
189	193	214	216	104	79	83	77
191	201	217	220	103	59	60	68
195	205	216	222	113	68	69	83
199	203	223	228	108	68	71	77

g[m,n]

$\otimes$

-1	2	-1
-1	2	-1
-1	2	-1

h[m,n]

=

?	?	?	?	?	?	?	?
?	-5	9	-9	21	-12	10	?
?	-29	18	24	4	-7	5	?
?	-50	40	142	-88	-34	10	?
?	-41	41	264	-175	-71	0	?
?	-24	37	349	-224	-120	-10	?
?	-23	33	360	-217	-134	-23	?
?	?	?	?	?	?	?	?

f[m,n]

Properties of the convolution

Commutative

$$h[n] \circ g[n] = g[n] \circ h[n]$$

Associative

$$h[n] \circ g[n] \circ q[n] = h[n] \circ (g[n] \circ q[n]) = (h[n] \circ g[n]) \circ q[n]$$

Distributive with respect to the sum

$$h[n] \circ (f[n] + g[n]) = h[n] \circ f[n] + h[n] \circ g[n]$$

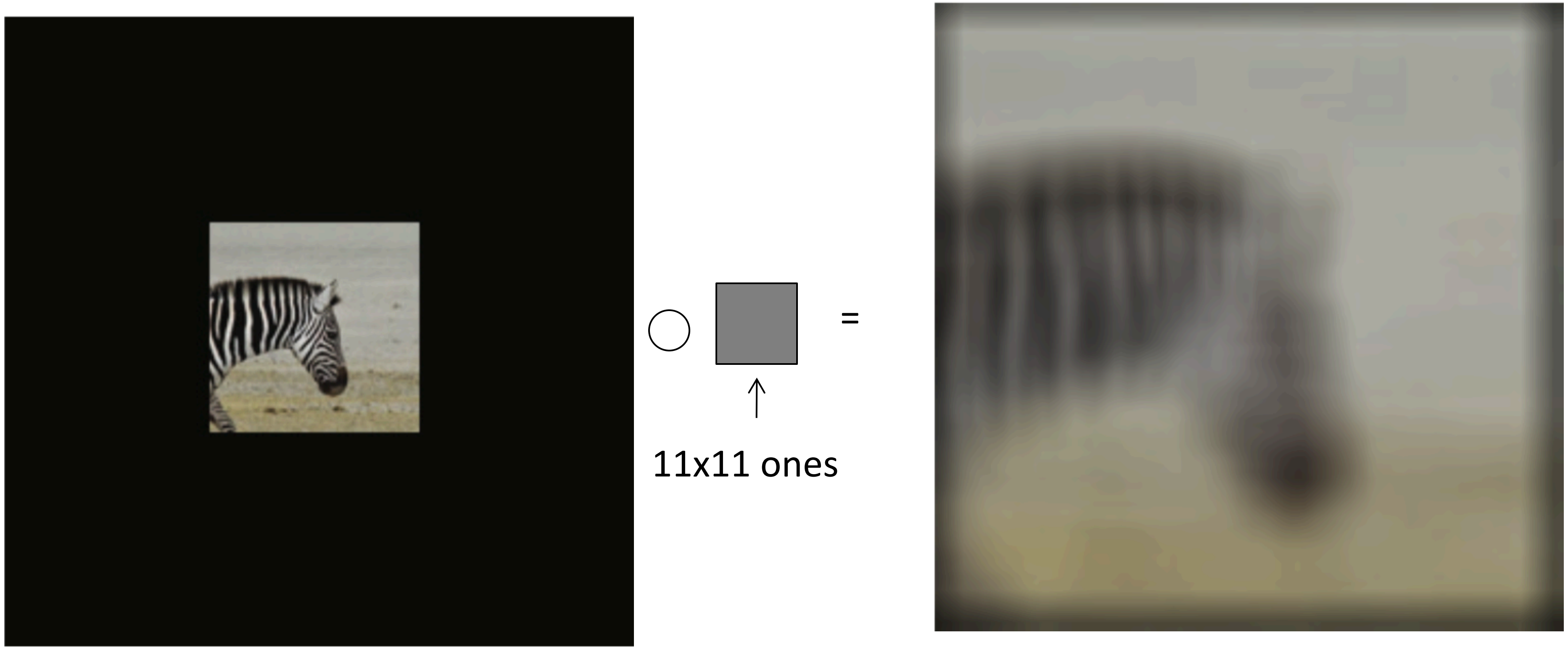
Shift property

$$f[n - n_0] = h[n] \circ g[n - n_0] = h[n - n_0] \circ g[n]$$

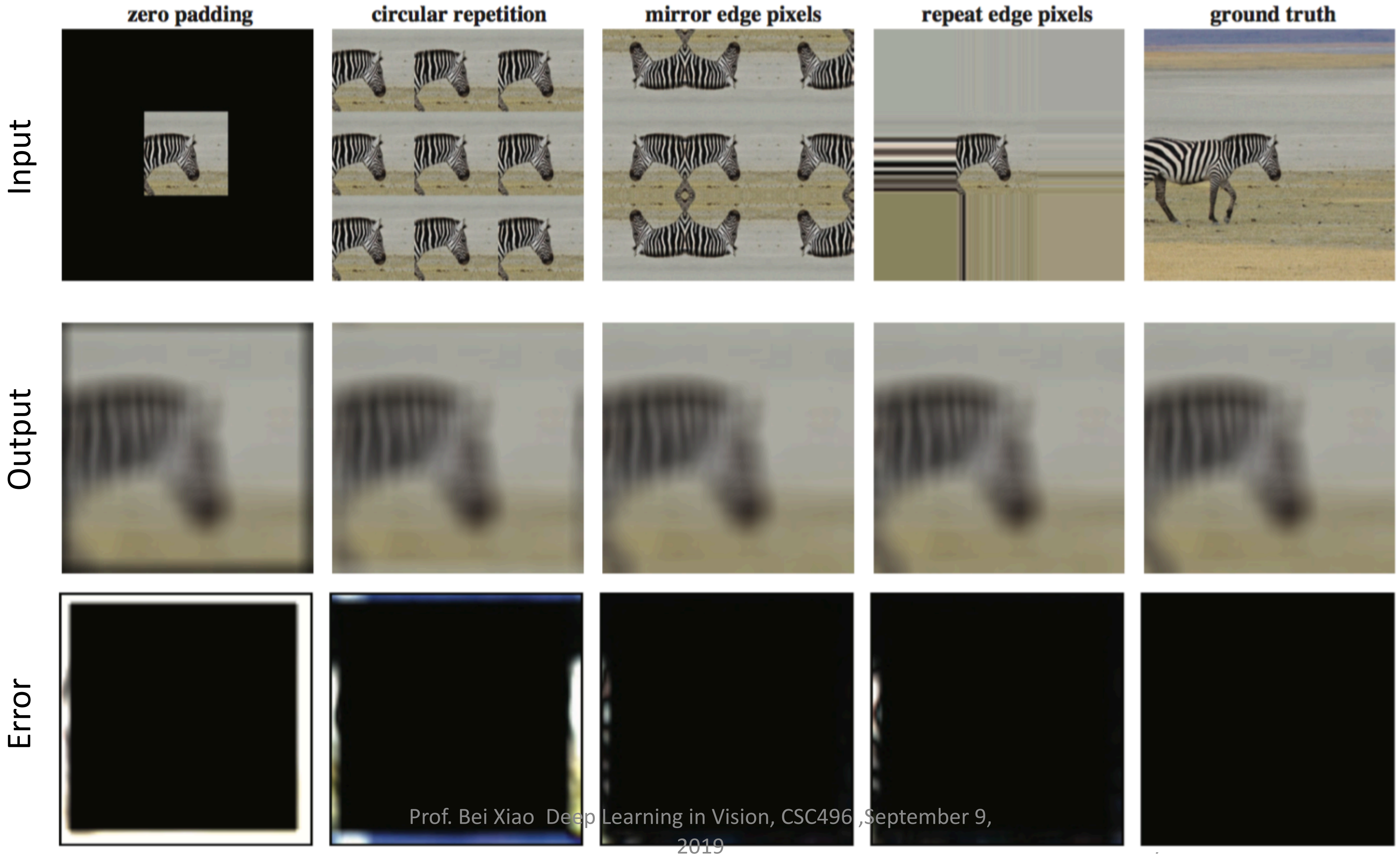
Handling boundaries

Handling boundaries

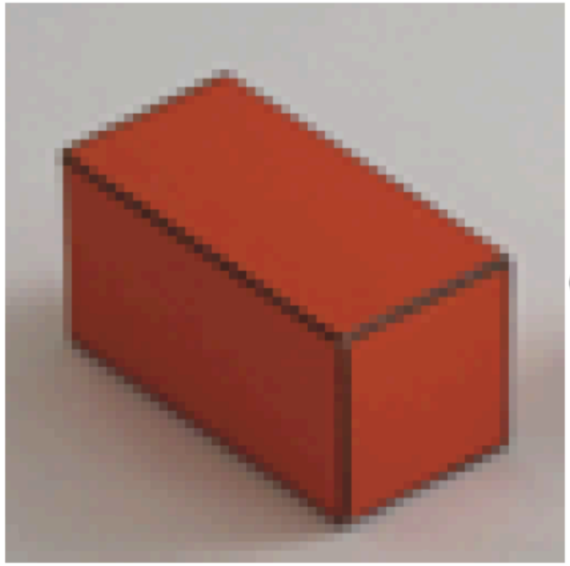
Zero padding



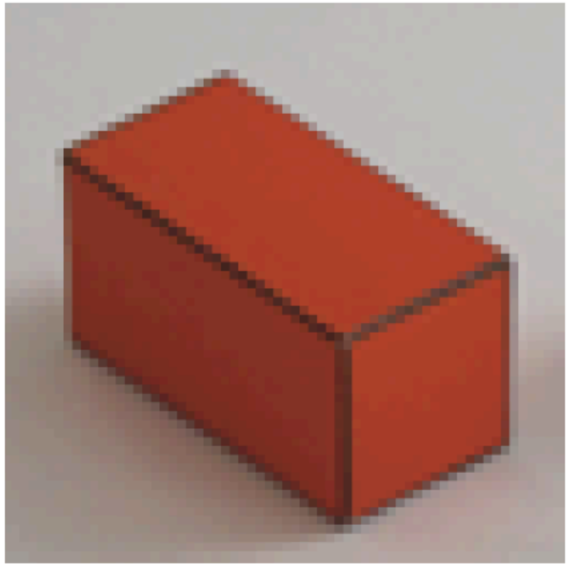
Handling boundaries



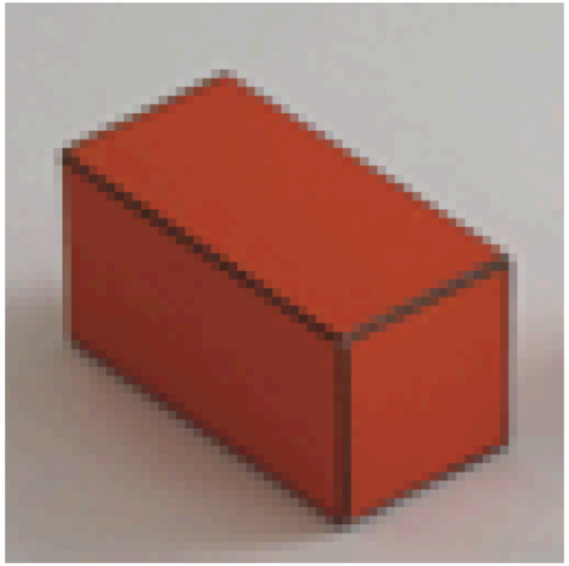
Examples



$$\circ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} =$$

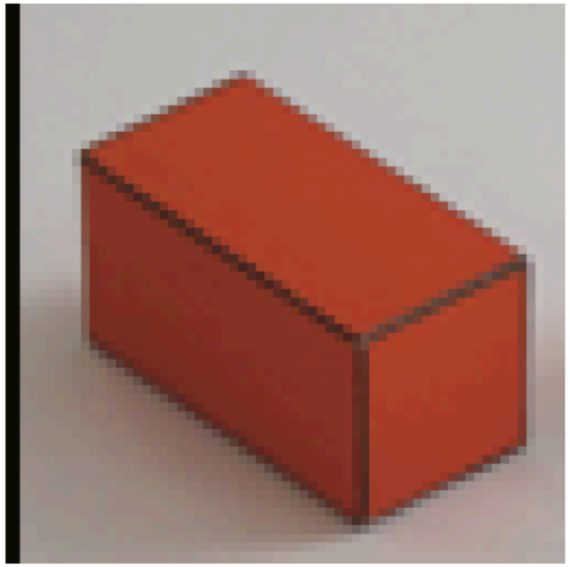


Examples



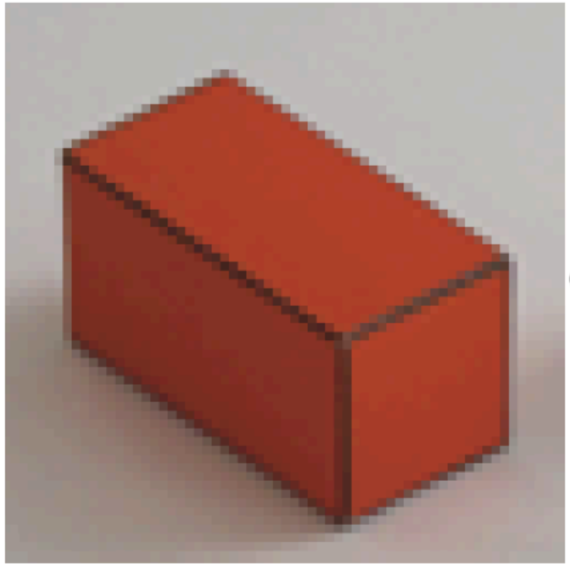
$$\circ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} =$$

2 pixels $\rightarrow$

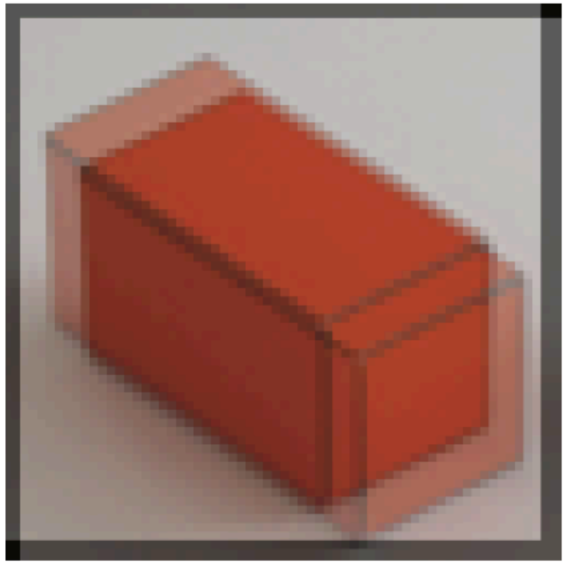


(using zero padding)

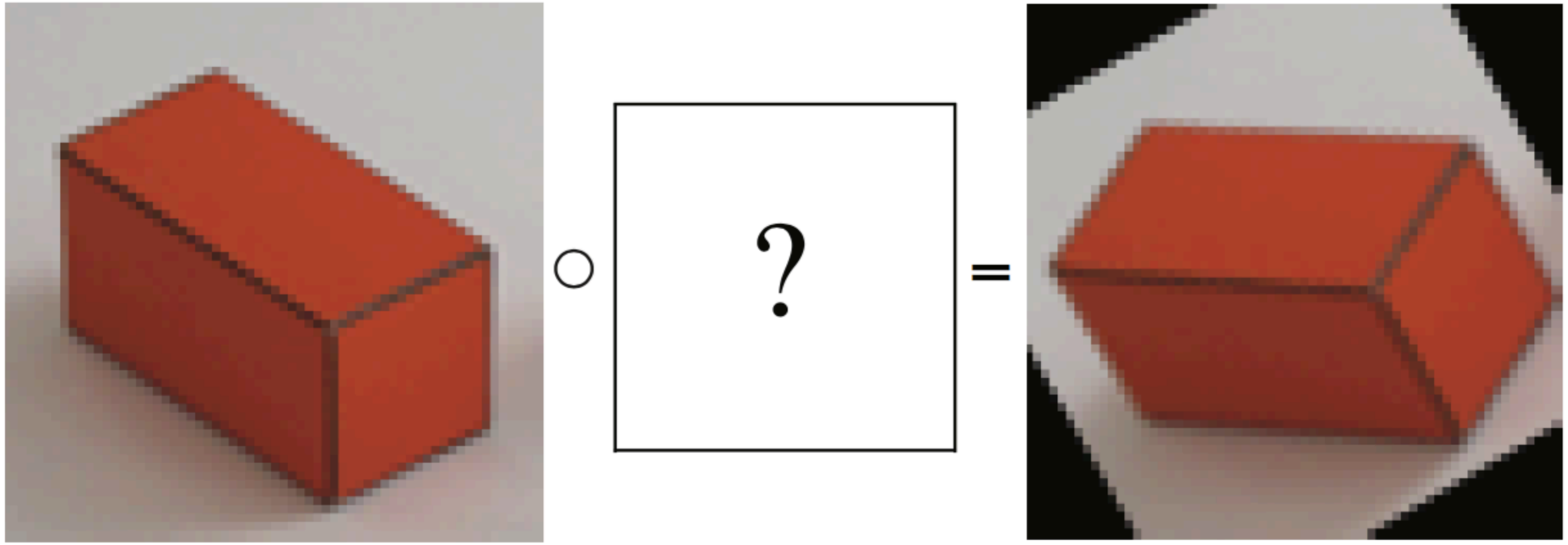
Examples



$$\circ \begin{bmatrix} .5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .5 \end{bmatrix} =$$



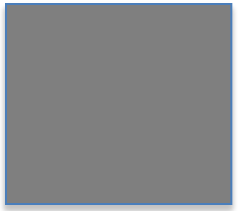
Examples



Rectangular filter

g[m,n]

$\otimes$



h[m,n]

=

f[m,n]

Rectangular filter

g[m,n]

$\otimes$

h[m,n]

=

f[m,n]

Rectangular filter

g[m,n]

$\otimes$

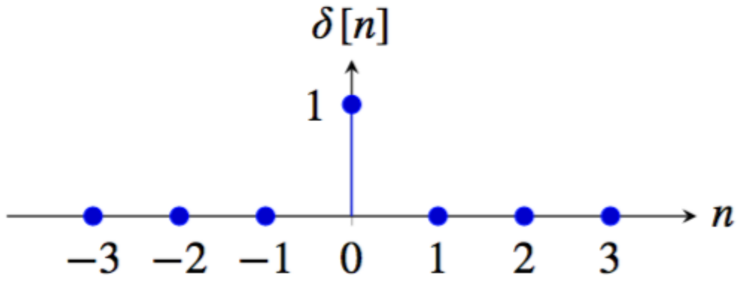
h[m,n]

=

f[m,n]

Important signals

The impulse



The result of convolving a signal *g*[n] with the impulse signal is the same signal:

$$f[n] = \delta \circ g = \sum_k \delta[n-k]\, g[k] = g[n]$$

Convolving a signal *f* with a translated impulse δ [*n* − *n*0] results in a translated signal:

$$f[n-n_0] = \delta[n-n_0] \circ f[n]$$

Important signals

Cosine and sine waves

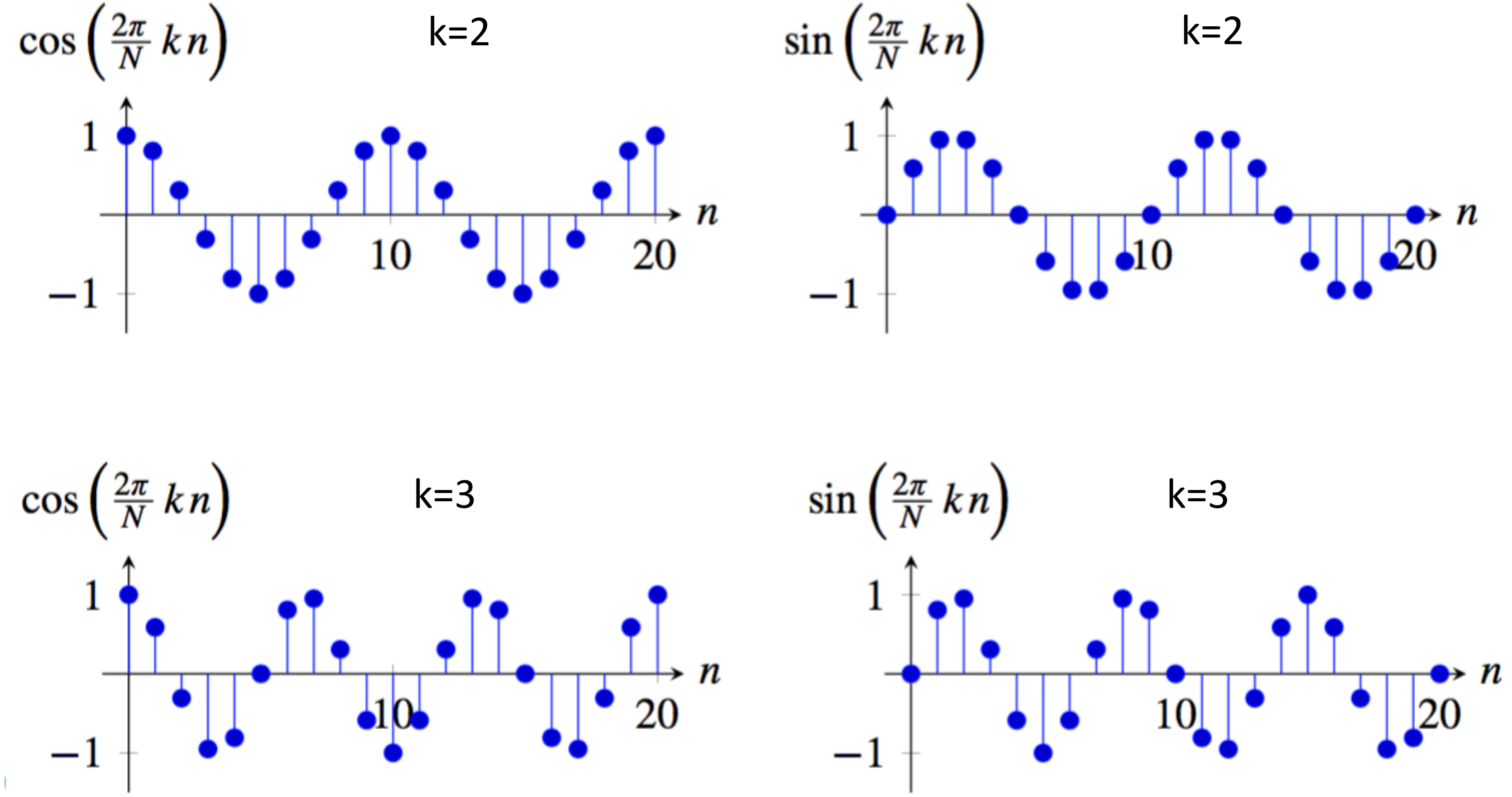
$$s(t) = A \sin(w\,t - \theta)$$

$$s_k[n] = \sin\left(\frac{2\pi}{N} k n\right) \qquad c_k[n] = \cos\left(\frac{2\pi}{N} k n\right)$$

$k \in [1, N/2]$ denotes the number of wave cycles that will occur within the region of support

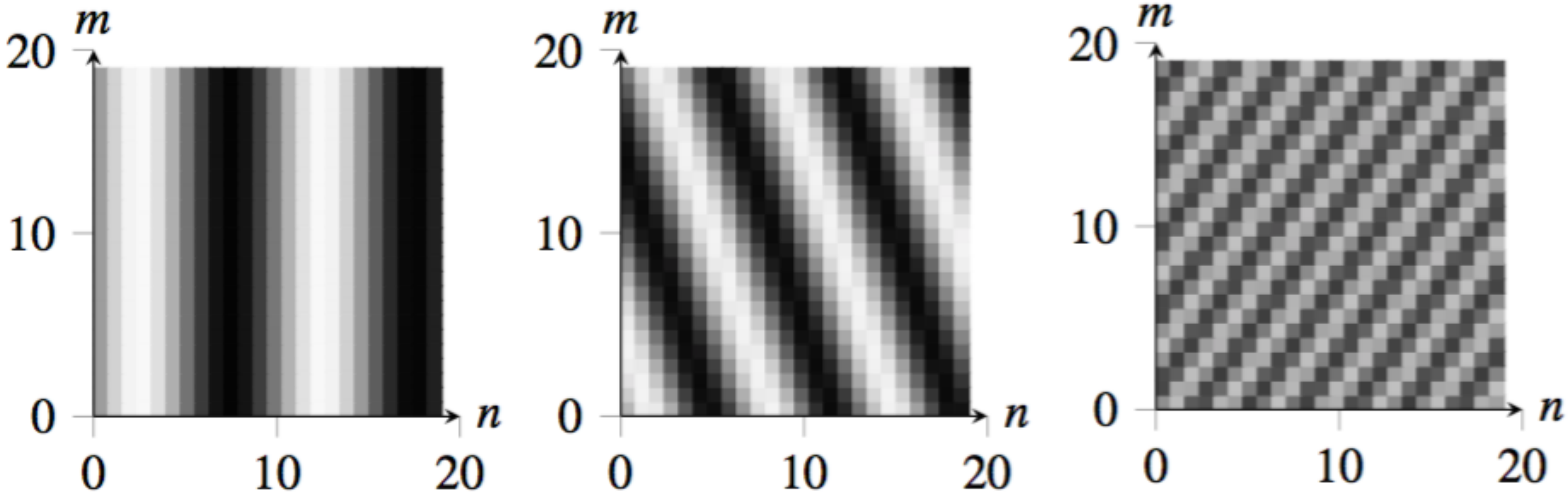
Important signals

Cosine and sine waves



Waves in 2D

$$s_{u,v}[n,m] = A\sin\left(2\pi\left(\frac{un}{N} + \frac{vm}{M}\right)\right) \qquad c_{u,v}[n,m] = A\cos\left(2\pi\left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$



u = 2, v = 0   u = 3, v = 1   u = 7, v = −5

Important signals

Complex exponential

$$s(t) = A \exp(j w t)$$

In discrete time (setting $A = 1$), we can write:

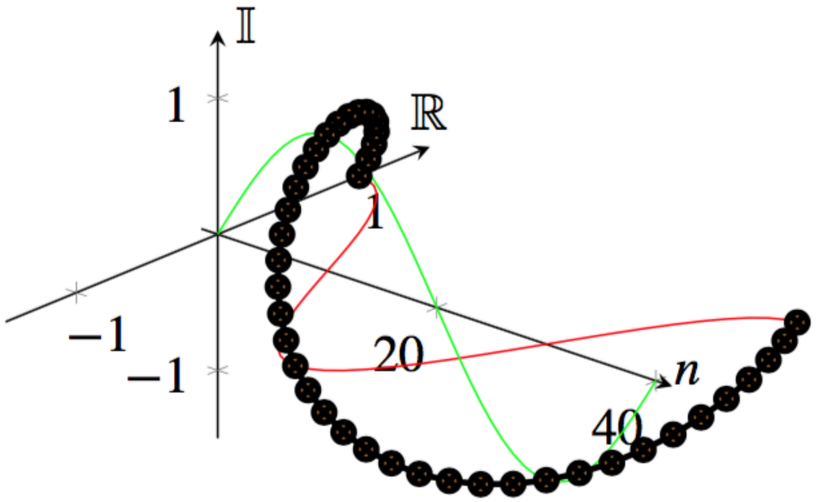
$$e_k[n] = \exp\left(j \frac{2\pi}{N} k n\right) = \cos\left(\frac{2\pi}{N} k n\right) + j \sin\left(\frac{2\pi}{N} k n\right)$$

And in 2D, the complex exponential wave is:

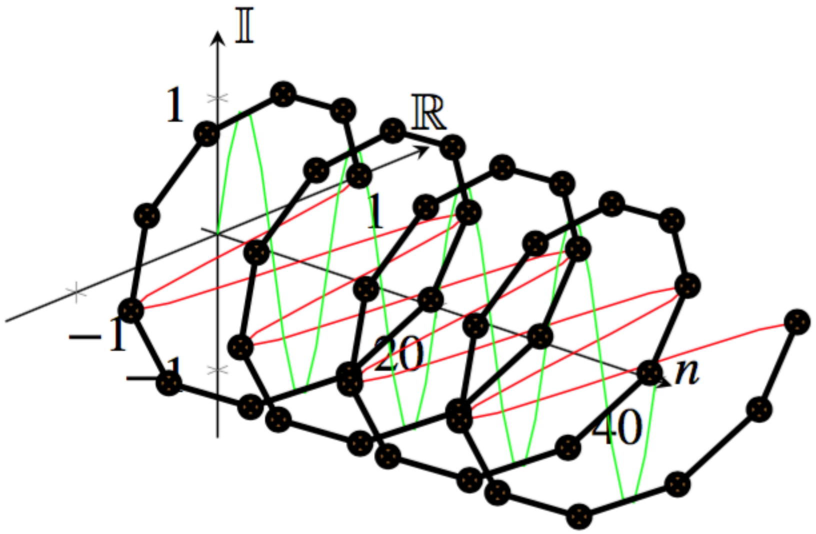
$$e_{u,v}[n,m] = \exp\left(2\pi j \left(\frac{u n}{N} + \frac{v m}{M}\right)\right)$$

Important signals

Complex exponential



$N = 40$, $k = 1$



$N = 40$, $k = 3$

Impulses, sine and cosine waves or complex exponentials form each an orthogonal basis for signals of length N

Take-home reading and next class

- Lecture 3 notes: cameras.pdf (uploaded on blackboards)
- Lecture 4 notes: SignalProcessing.pdf
- Linear filtering: page 111-120
- http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

- Next class: continue onto Image convolution