ComputerVision I

Digital Images and Camera Models

Xosé M. Pardo

Escola Técnica Superior de Enxeñaría, USC

Academic year 2024/25

MASTER IN ARTIFICIAL INTELLIGENCE

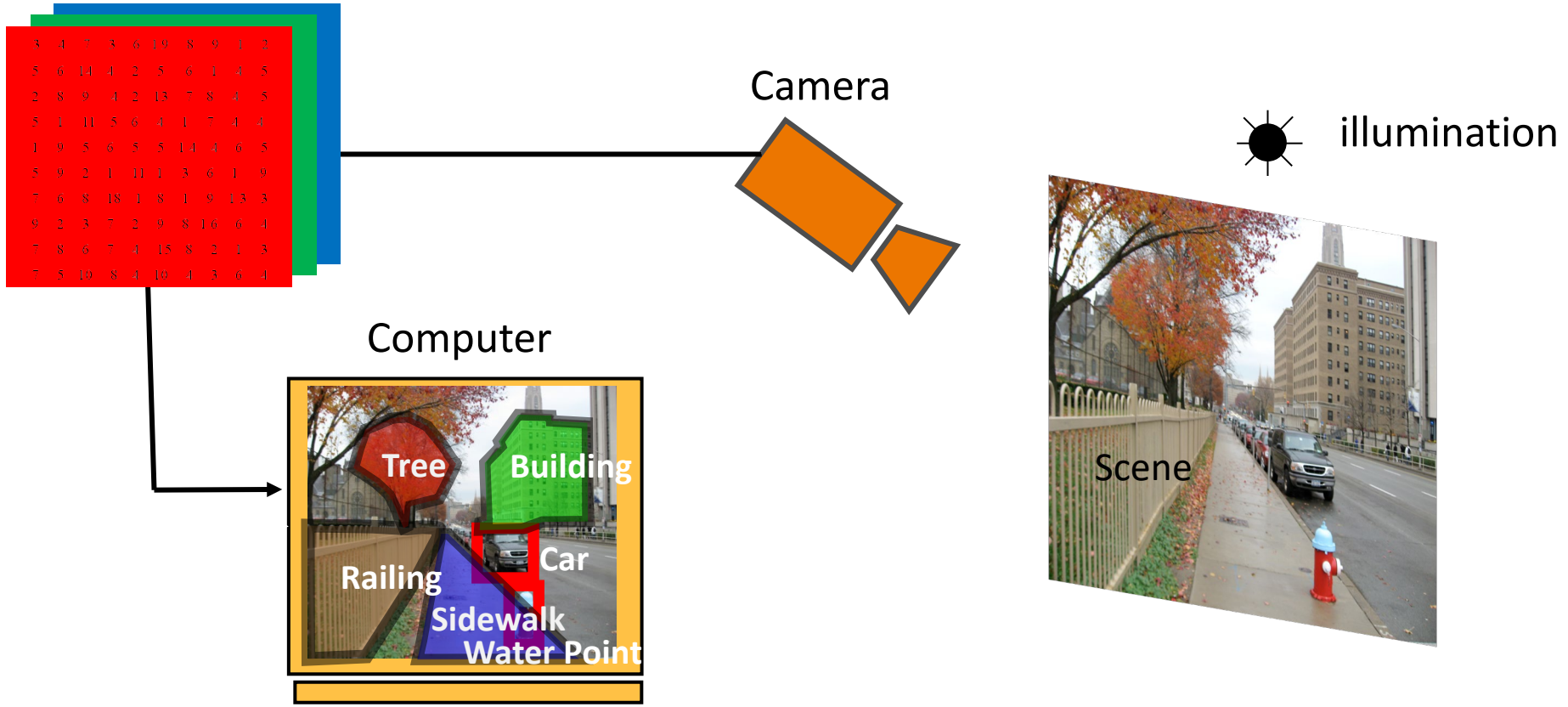
USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

UNIVERSIDADE DA CORUÑA

Universidade Vigo

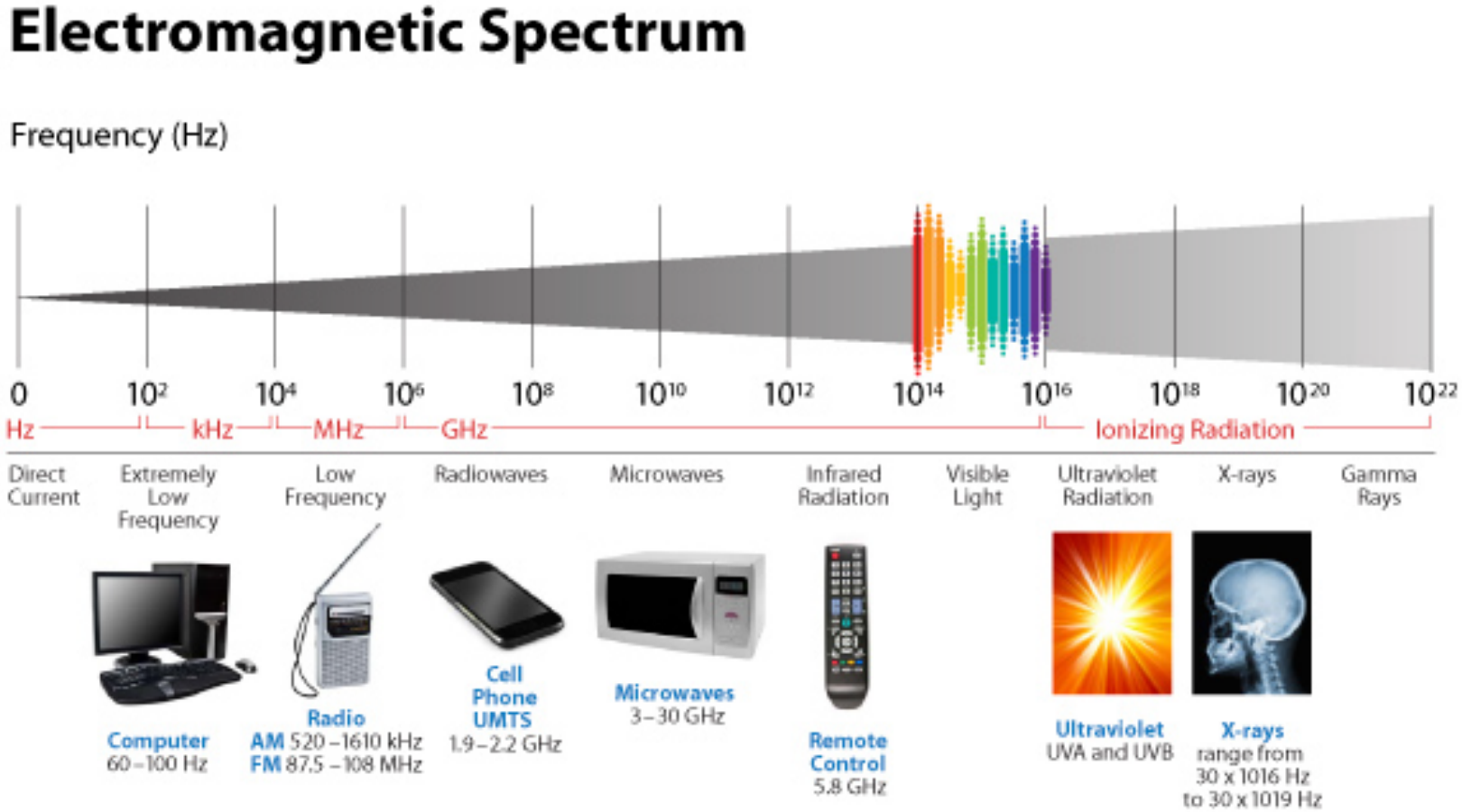
Computer Vision

- Computer vision problem:
  - Explain the mechanisms which transform a matrix/tensor of light intensity measurements into object-level representations, which can be stored in memory and processed by other cognitive systems.



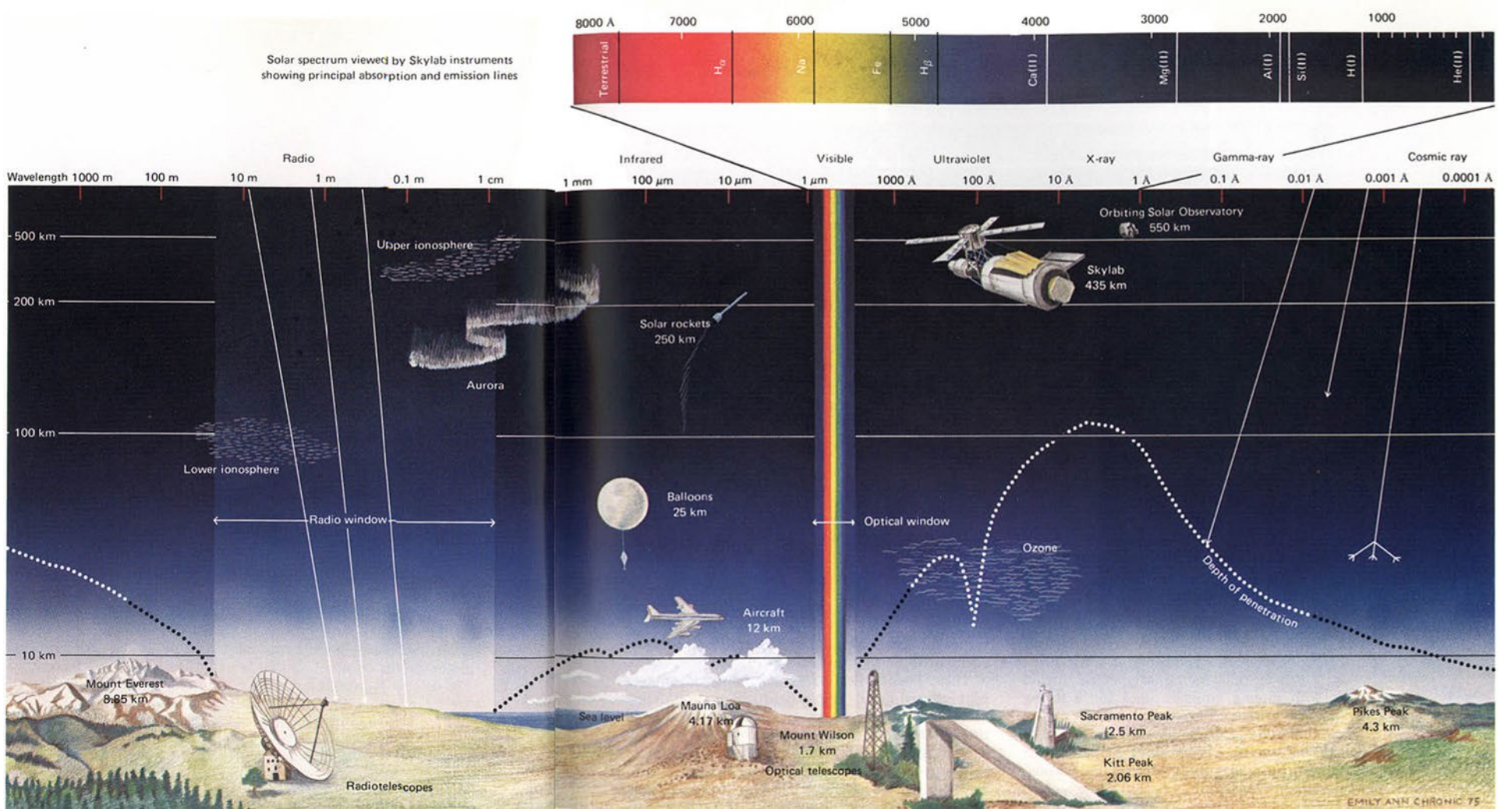
Vision and Electromagnetic Waves

- Vision: perception of distances, shapes, objects, and motion, without direct physical contact
  - Interaction through electromagnetic waves.
  - Radiation in visible spectrum: the first EM waves to interact with energy levels within atoms.



Solar Radiation Spectrum

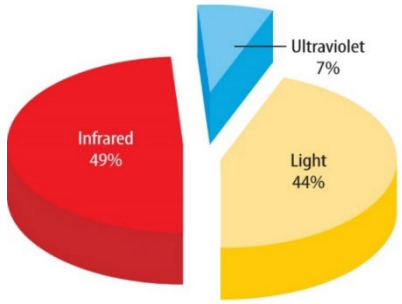
- **Light waves** are produced by sun incessantly.
- Visible sunlight is but one part of the total radiation Earth receives from the Sun.

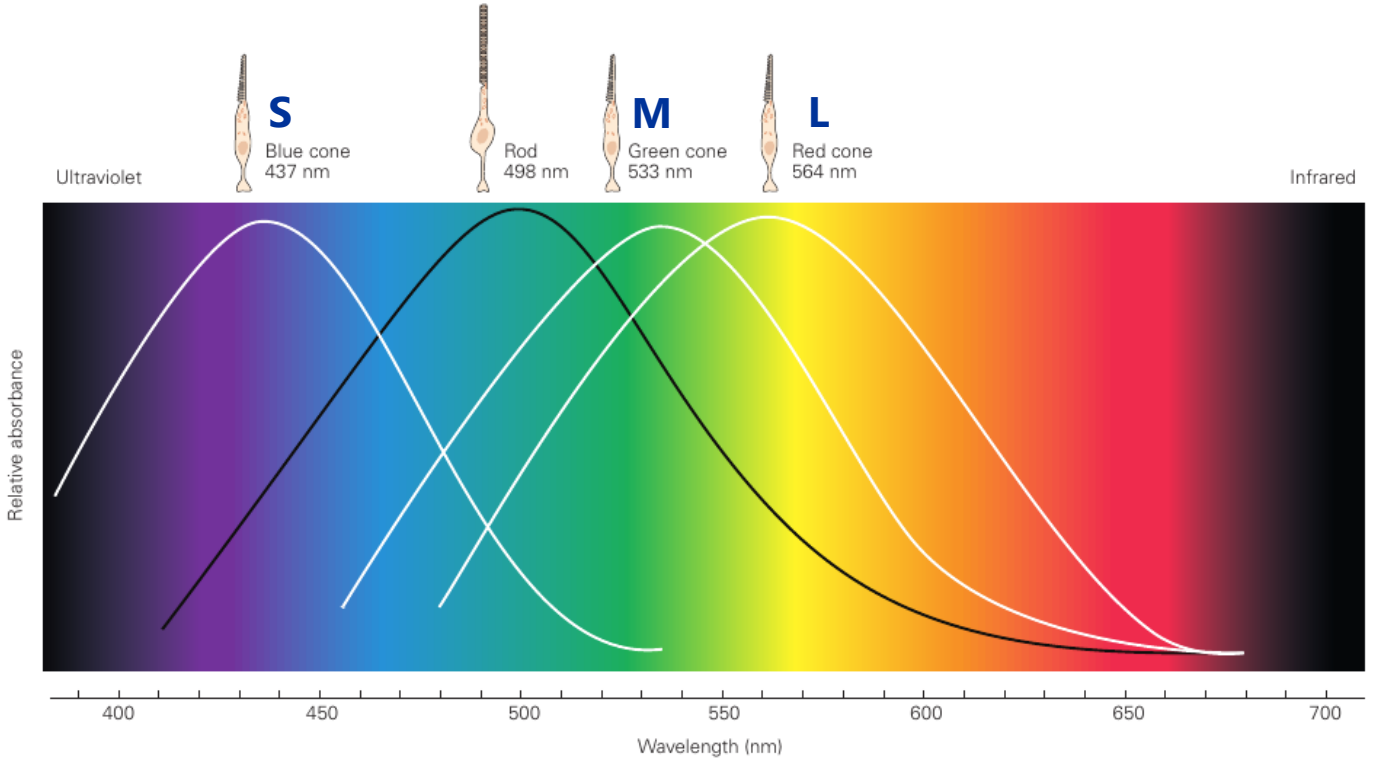


Human Visual System

- Sensitivity of eye sensors to EM radiation:
  - Our eyes see only a narrow band of wavelengths from about 400 to 700 nm, (violet to red of visible spectrum).
  - Spectral absorption of rods and cones:
    - Adapted to main radiaton source

EM radiation that reaches the Earth from the Sun:

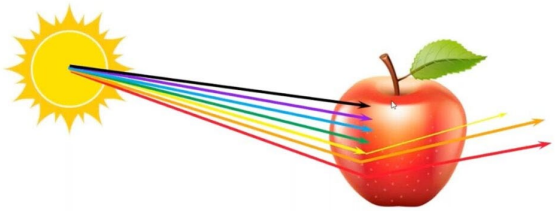




Vision and Electromagnetic Waves

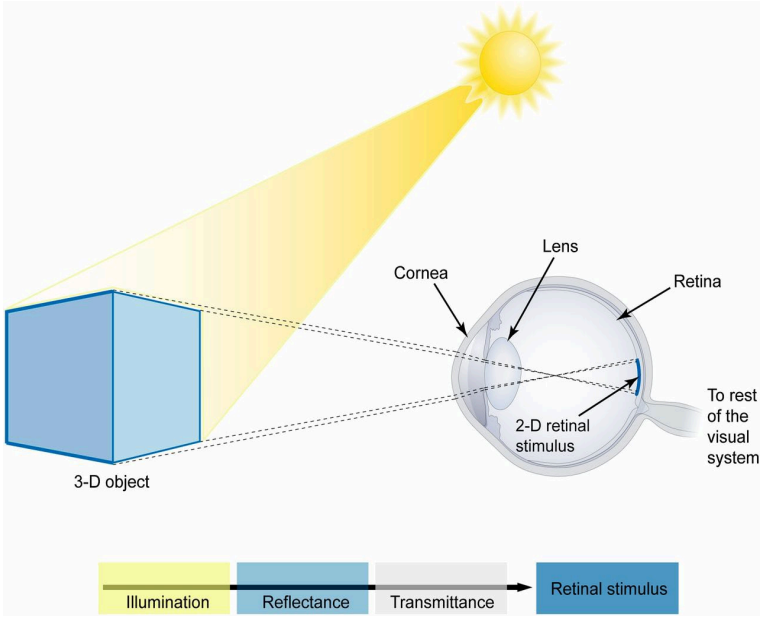
- Vision: perception of distances, shapes, objects, and motion, without direct physical contact:
  - **Visible light itself cannot be seen** – it is only the relationship of light and object that we can perceive.
    - What we see is the result of a particular series of wave frequencies striking a variety of objects of different substances and densities.
    - Objects, depending upon their own qualities, absorb and reflect different aspects of these incoming “light” waves, reflecting certain colors that we perceive through receptors in our eyes.

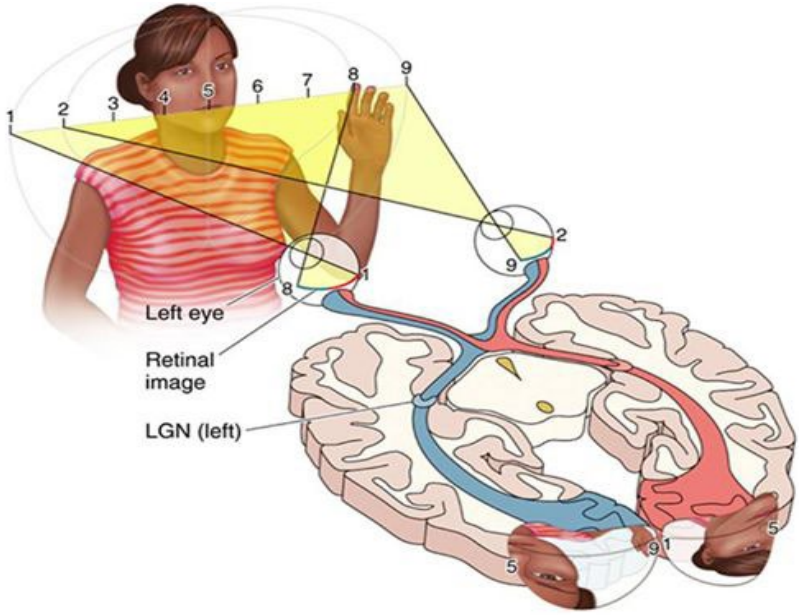
The Human Visual System (HVS) only perceives light-object interaction



Human Visual System

- Vision begins with light passing through the cornea and the lens, and reaching a sheet of photoreceptors in retina.
- Image on the retina is:
  - Reversed: Objects above the center project to the lower part and vice versa.
  - Result of by comparing the amount of light striking any small region of the retina with the amount of surrounding light
- Information from the retina (electrical signals) is sent via optic nerve to other parts of the brain, which ultimately process the image and allow us to see.





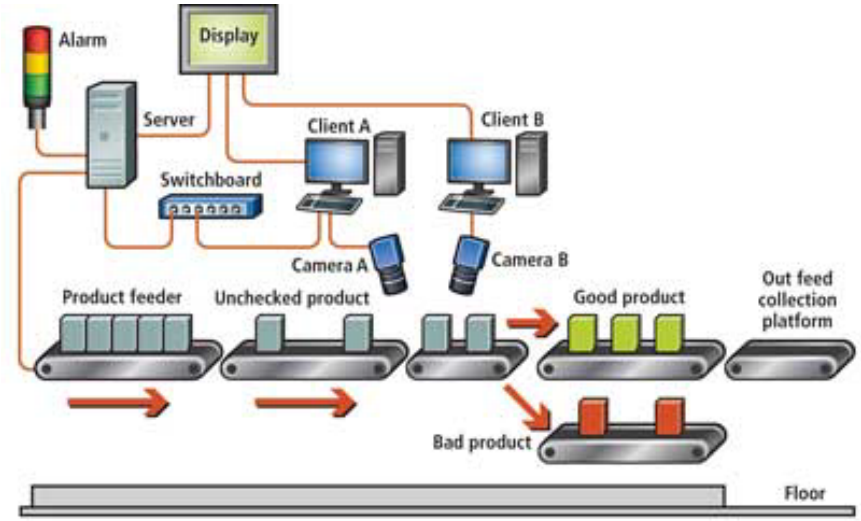
Human Visual System

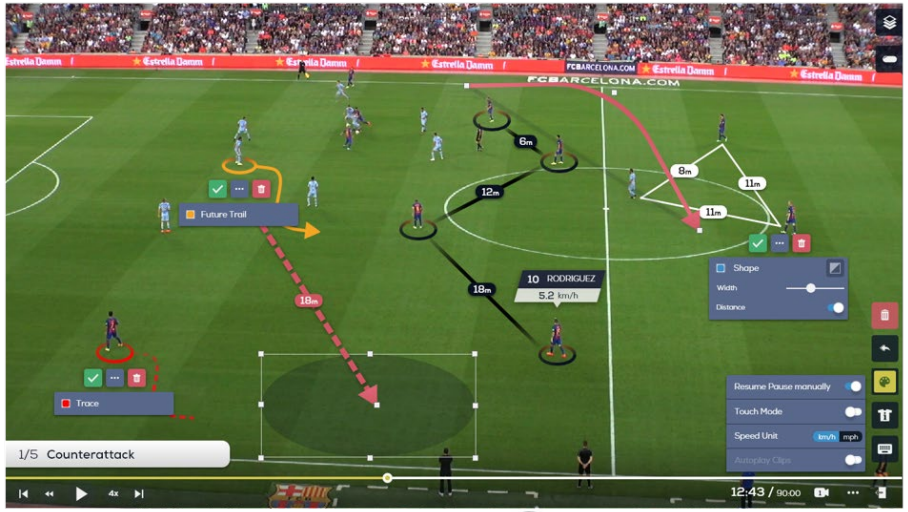
Sensory Data Collection	
Sensory System	Bits/sec
Vision	10M
Touch	1M
Hearing	0,1M
Smell	0,1M
Taste	1K

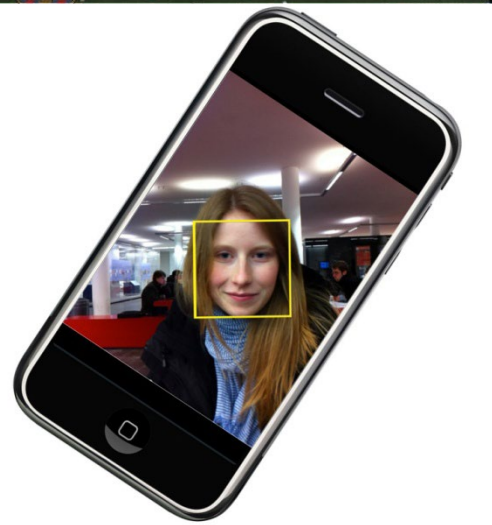
Intelligent or conscious processing speed	
SVH	50 bits/sec

Unconscious pre-processing High speed	
High speed	100 G operations/sec
High level of compression	from MBs to few bits

Computer Vision



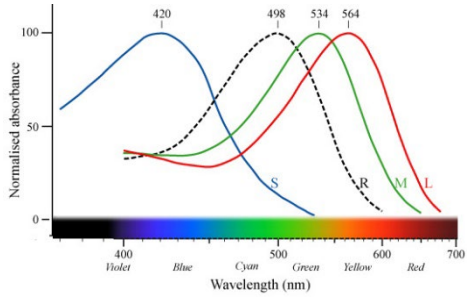


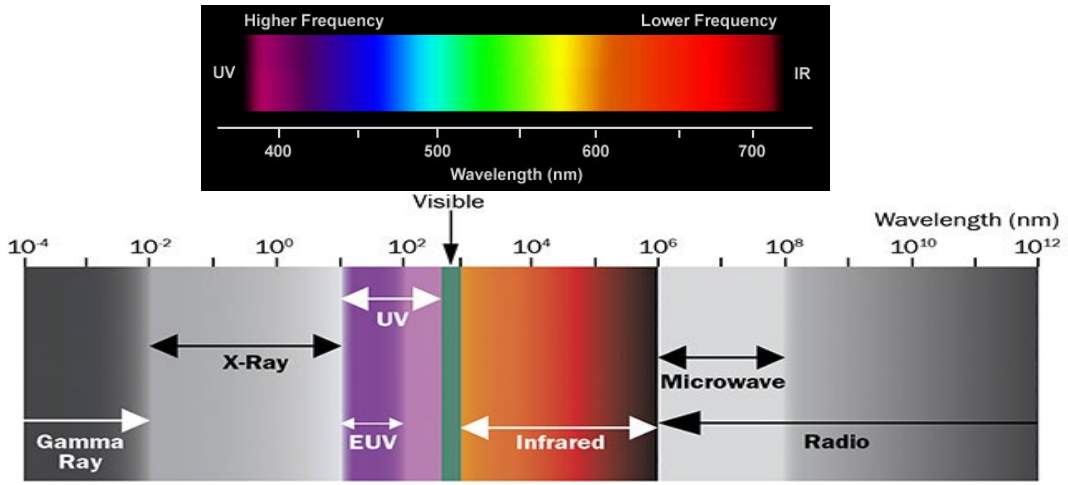


Digital Images

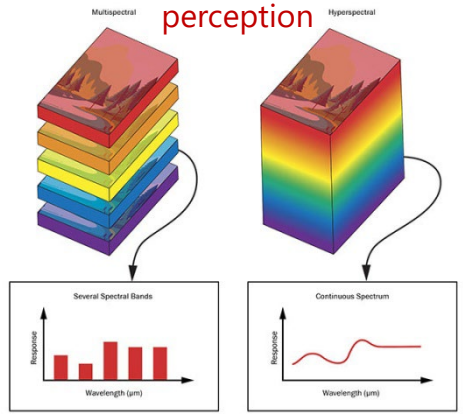
- A digital image ***I*** is a tensor of intensity of reflected electromagnetic radiation of size CxHxW:
  - C is the number of channels
  - H and W are the sizes of the 2D image
  - If C==1, *I*(*x,y*) gives the gray-level at position (*x, y*)
  - If C=3, color image, *f(x,y)=[r(x,y),g(x,y), b(x,y)]*
  - If C>3 multi/hyper-spectral images

C==1 and C==3 are related to human perception





C>3 are not related to human perception



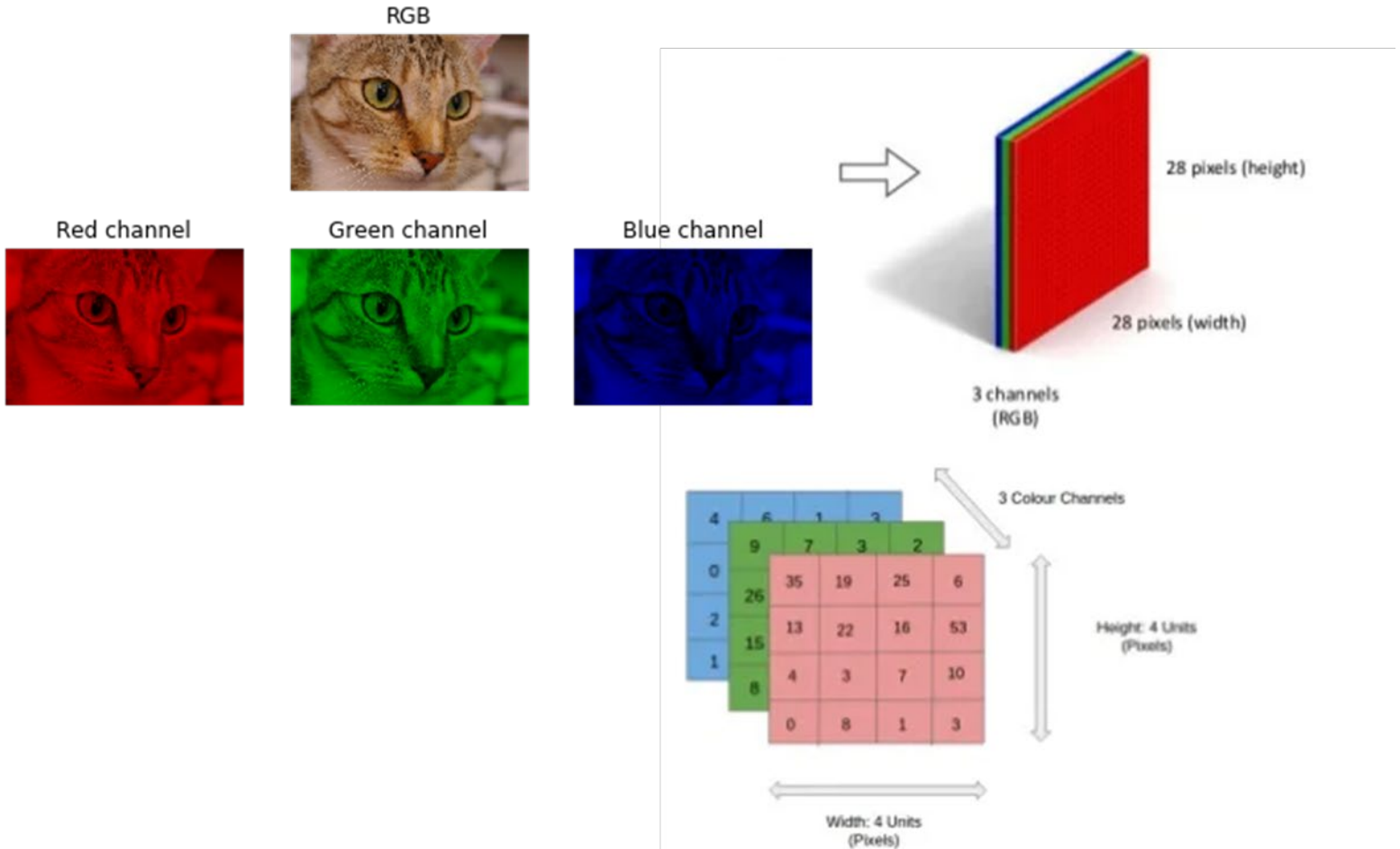
Digital Images

- Grayscale images: C==1 (2D matrix):
  - 1 Bit to represent a binary value in {0,1}: binary images
  - 1 Byte to represent a value in [0,255]: gray-level images
  - >=2 Bytes to get higher contrast resolution


```
[[76 77 77 78 78 78 79 79 78 79 80 80 78 78 79 80 79 79 79 79]
 [76 77 77 78 78 78 79 79 78 78 79 79 78 78 78 79 80 80 80 80]
 [77 77 77 78 78 78 79 79 78 79 79 79 78 78 79 79 80 80 80 80]
 [77 77 77 78 78 78 79 79 79 80 80 80 79 79 80 80 80 80 80 80]
 [77 77 78 78 78 78 79 79 79 80 81 81 80 80 81 82 80 80 80 80]
 [77 77 78 78 78 78 79 79 79 79 80 80 80 80 80 81 81 81 81 82]
 [77 77 78 78 78 79 79 79 79 80 80 80 80 80 81 81 82 82 82 82]
 [77 77 78 78 79 79 80 79 80 80 81 81 81 81 81 82 81 81 81 81]
 [77 79 80 79 78 77 79 81 80 80 80 79 79 80 81 81 80 81 81 81]
 [77 78 79 79 78 78 80 81 80 80 80 80 80 81 81 81 81 81 81 81]
 [77 78 79 79 79 79 81 82 80 81 81 81 81 81 81 81 82 82 82 82]
 [77 78 79 79 79 79 81 82 80 80 81 82 82 82 81 81 82 82 82 82]
 [79 79 80 80 79 79 81 81 80 80 81 82 82 81 81 81 82 82 82 82]
 [80 80 80 80 80 80 81 81 80 81 81 81 81 81 81 81 82 82 82 82]
 [79 79 79 80 80 80 81 81 81 81 81 81 81 82 82 82 82 82 82 82]
 [79 79 79 80 80 81 82 82 82 82 82 81 81 82 83 83 82 82 83 83]
 [82 81 82 83 83 81 80 81 82 82 83 83 82 83 83 83 82 83 83 83]
 [82 81 81 83 83 81 80 81 82 81 81 81 82 83 82 82 82 83 83 83]
 [81 81 81 82 82 81 81 81 83 82 80 81 83 84 83 81 82 82 83 83]
 [81 81 82 83 83 82 82 81 84 83 82 83 84 84 83 83 82 82 83 83]]
```

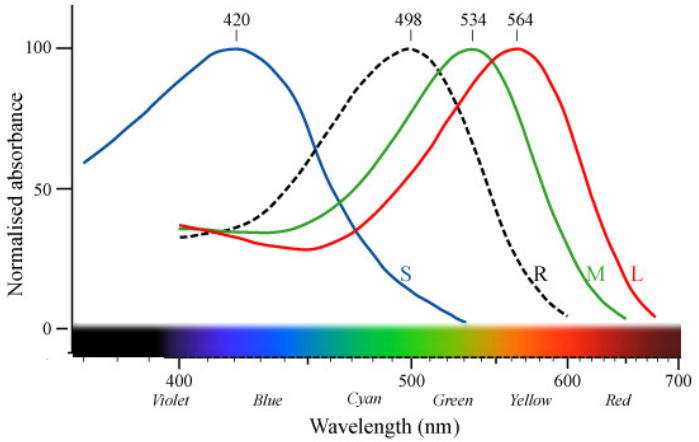
Digital Images

- Color images: C==3: (3D tensor):
  - 1 Byte per channel to represent a value in [0,255]
  - ~16M colors



Color Spaces

Light: "visible electromagnetic radiation" with wavelenghts between 740 and 380 nano meters (nm).



Light is not a single point in this range, but a **combination of several reflected frequencies**.

- An accurate color model **should be defined with a long** vector of integers.
- Why do we use only three (R,G,B)?
  - We only model the human color representation.

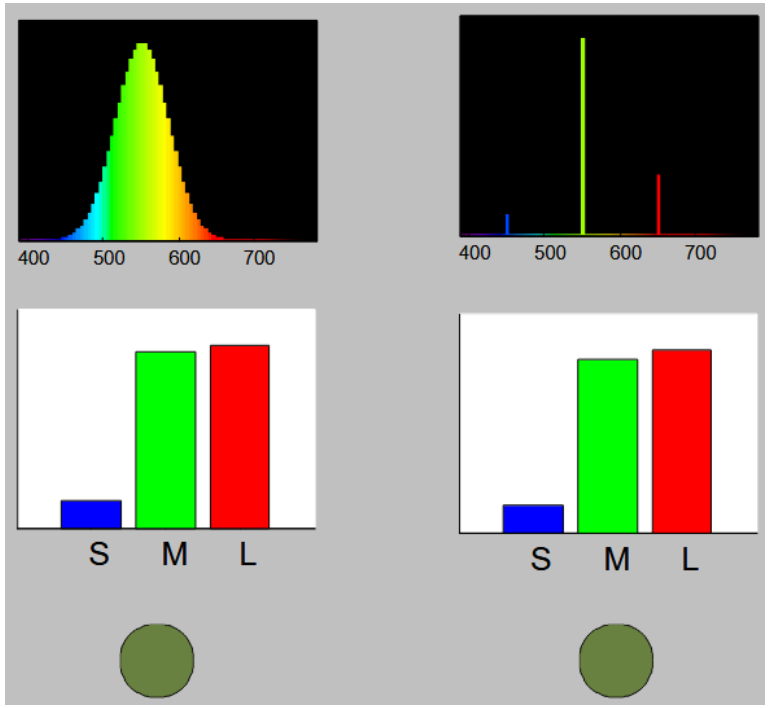
Color Spaces

Human brain manages lesser number of colors than the number of possible spectral combinations.

- Two different spectra that produce the same L, M and S cone responses will look the same.
  - These pairs are called ‘metamers’
  - This explains why we only need three primaries to produce a variety of colors

A color is the variety of spectra that produces the same excitation in cones.

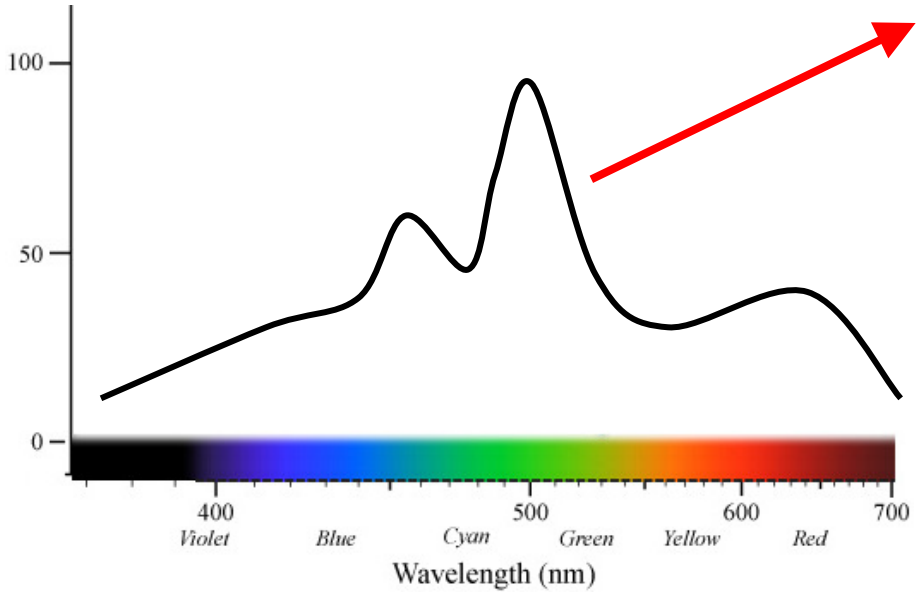
Color is a creation of our brain, not an intrinsic property of light.



Color Spaces

- **Color model:** mathematical model that describes the representation of colors by means of tuples of numbers.
- **Color space**: all the colors that can be represented by these tuples.

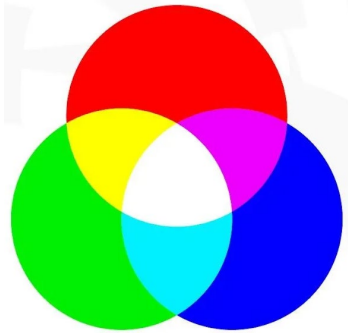
- Given a reflected visible spectrum, the color model allows us to extract the corresponding color tuple



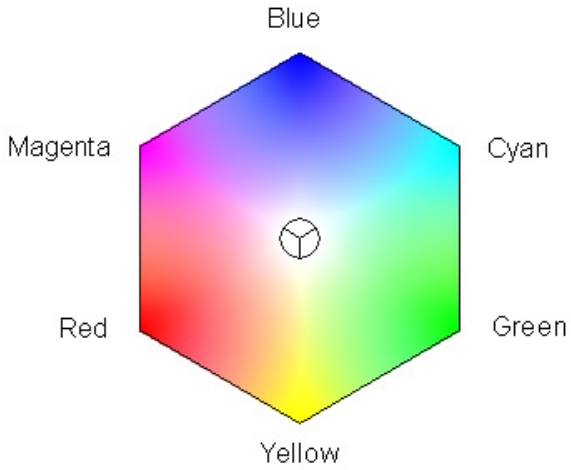
RGB: (14, 26, 17)

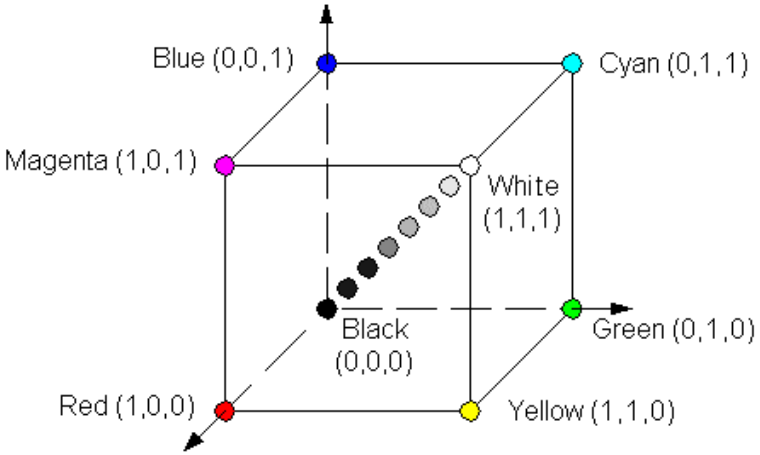
Color Spaces

- RGB model
  - Additive: red, green and blue colors are added together, according to their relative intensities, to produce a broad range of colors
    - Ideal for hardware
    - It is the most commonly used in digital imaging



  - **RGB** space has the shape of a **cube of side** 1.
    - (R=0,G=0,B=0): **black**, (R=1,G=1,B=1) **white**.
  - Drawbacks:
    - Correlated channels
    - Not perceptual

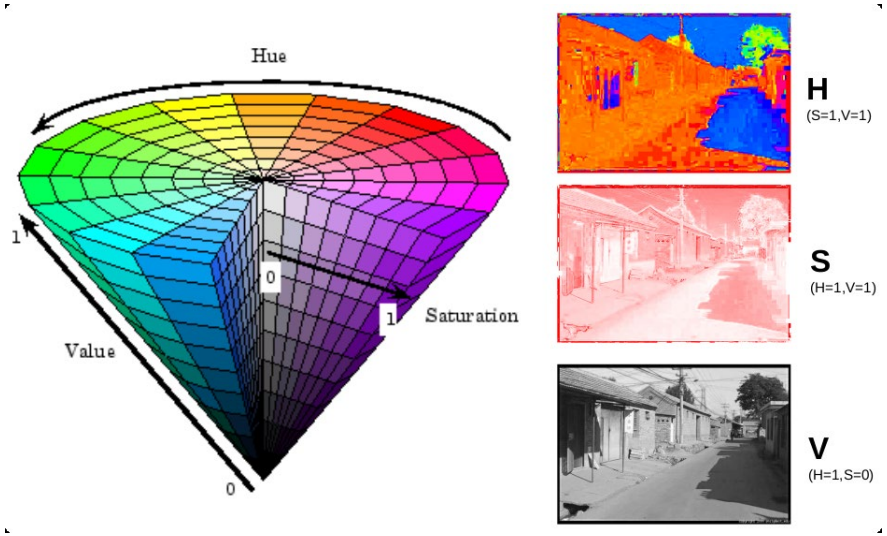




Color Spaces

- HSV model:
  - Differences between values of two HSV color vectors are correlated with human perceived chromatic differences.
  - **Value (V)**: controls the brightness (quality of being darker or lighter).
  - **Saturation (S)**: controls the amount of color used (difference between a color and a gray with the same V).
  - **Hue (H)**: specifies the angle of the color on the RGB color circle. A 0° hue results in red, 120° results in green, and 240° results in blue.

HSV is just mapping a particular RGB color space; it is a different coordinate system describing the original RGB space.



Color Spaces

- Grayscale model
  - The grayscale model uses only one component, lightness, which is measured in values ranging from 0 to 255.
  - Grayscale correspond to colors with equal values of the red, green, and blue components.
- Advantages:
  - Has only one channel and demand less computational resources.
  - Color is of limited benefit in many applications.
- Conversion from RGB to grayscale image:
  - Simple mean:
    - G= (R+G+B)/3
  - According to channel luminosity perception:
    - G= 0.2126R + 0.7152G + 0.0722B
  - According to global luminance:
    - G= 0.3R + 0.59G +0.11B

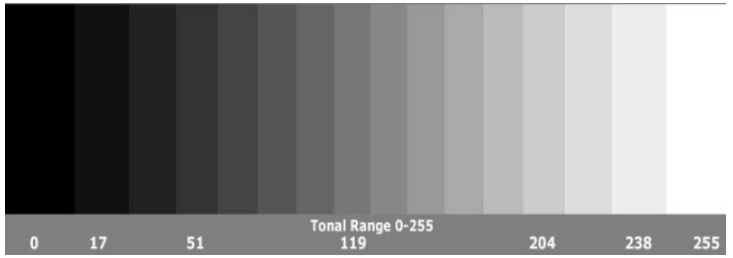
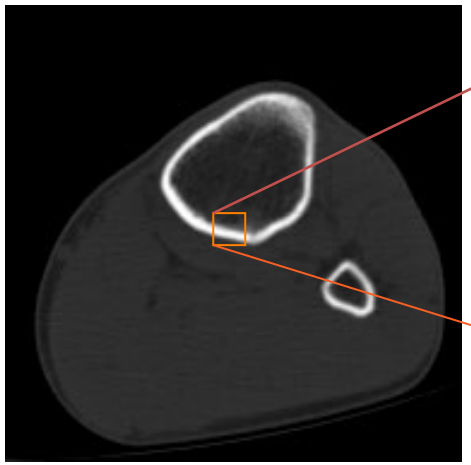


Image Sampling and Quantization



j →

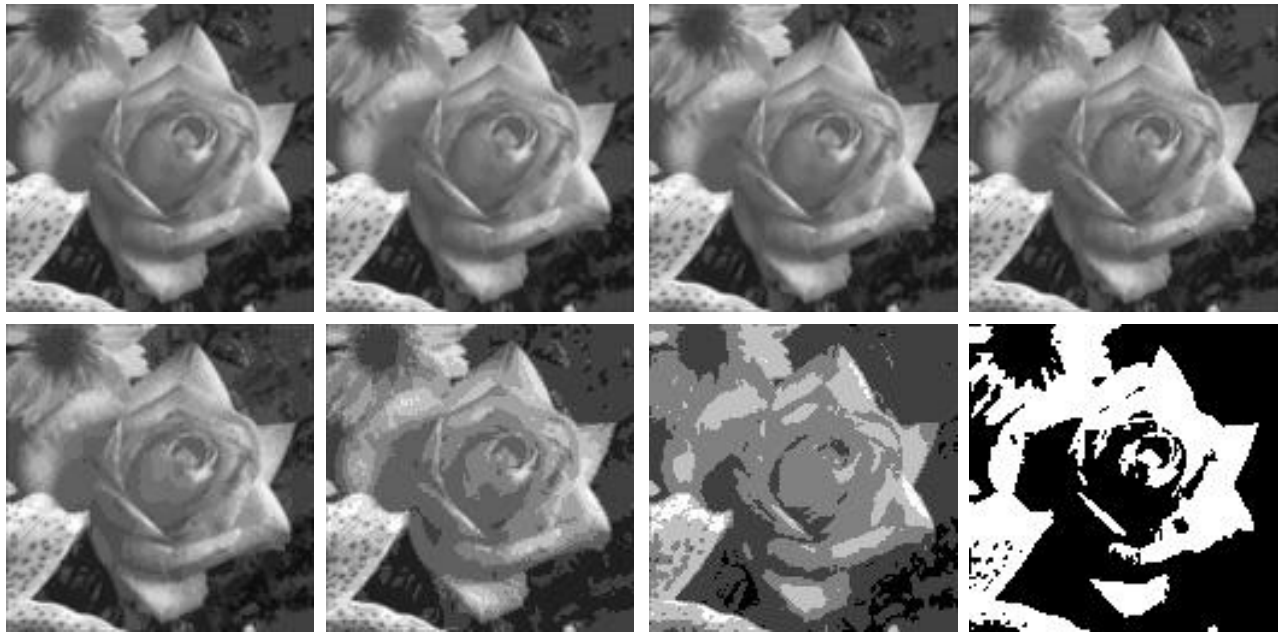
i ↓

54	44	41	40	39	36	31	38	48	52	56
62	50	47	52	51	44	38	47	55	55	59
113	94	79	78	73	66	63	64	68	68	72
172	155	131	110	95	88	86	82	81	82	85
223	220	201	182	167	153	136	119	106	95	92
251	250	245	241	238	230	214	200	184	162	149
228	239	243	246	251	252	249	245	241	232	224
128	152	175	194	213	222	231	247	254	255	254
55	64	73	86	105	124	146	176	195	213	227
49	49	48	49	50	57	69	85	101	119	142
48	47	51	53	51	48	49	52	56	59	66

- **Digital image:** function **I**(i,j) over the 2D discrete space, which also takes discrete values:
  - **Spatial Sampling**: regular grid
  - **Quantization:** bounded range of integer values

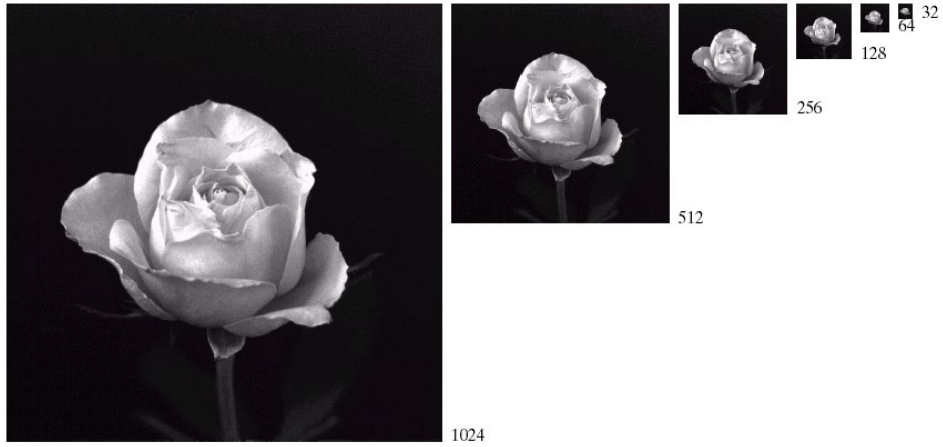
Image Quantization

Different quantization levels (contrast resolution): 256..2

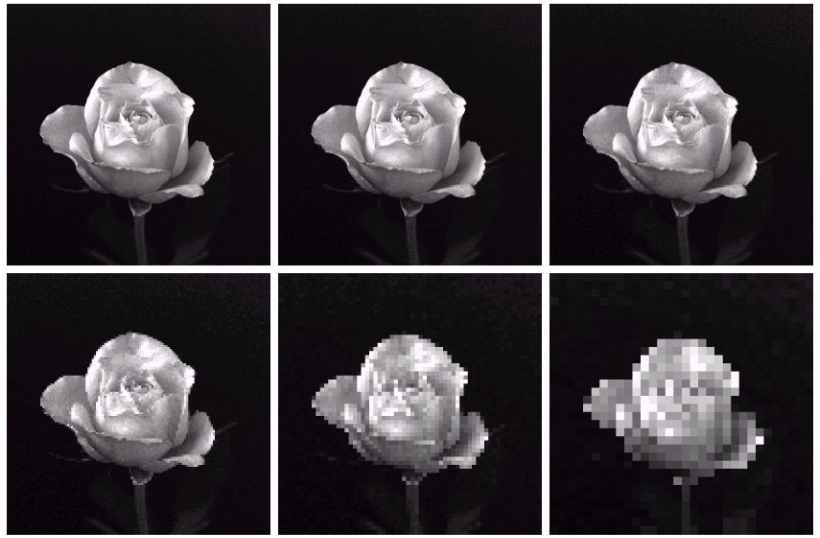


Spatial Sampling

Different spatial sampling (spatial resolution)



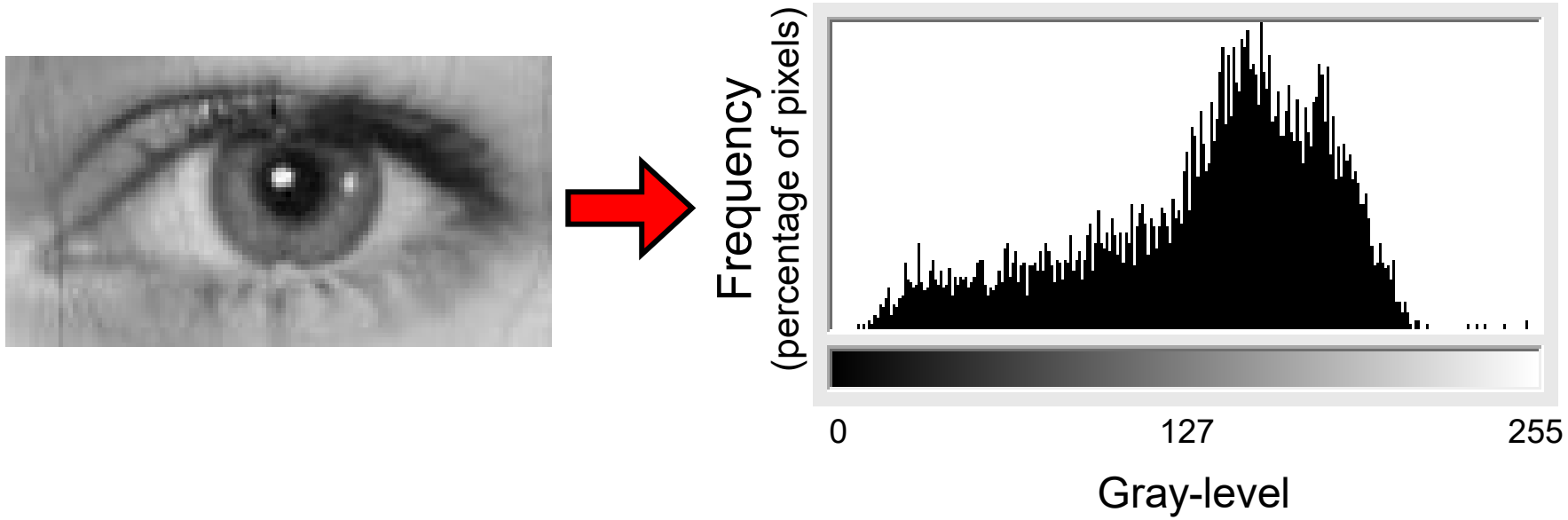
1024x1024 .. 32x32 pixels



Same image size, but different pixel sizes, to compare…

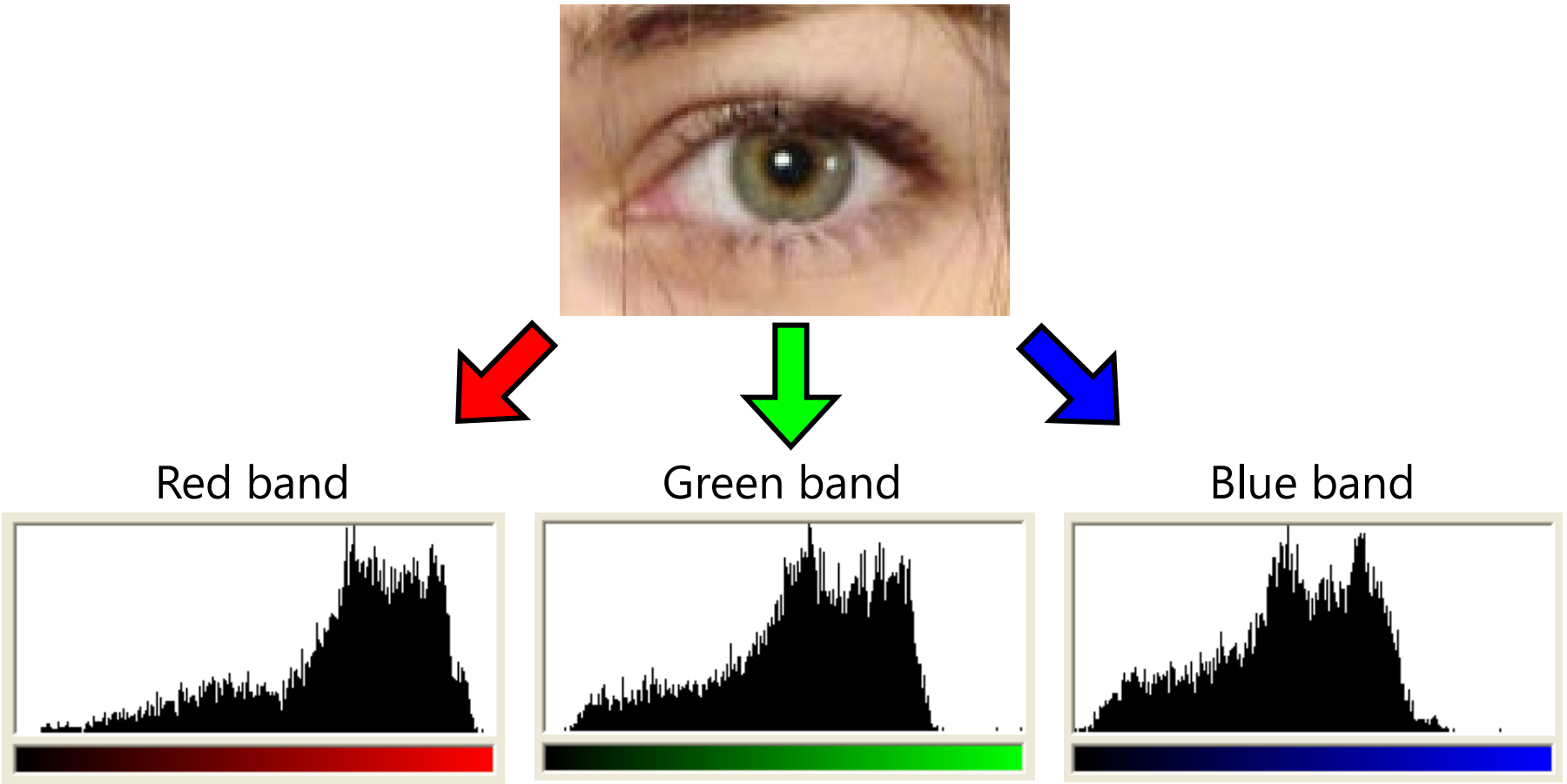
Histograms

- **Histogram**: graphic representation of a frequency distribution.
- **Image Histogram**: represents the frequency of each gray-level (or color channel) in an image.



Histograms

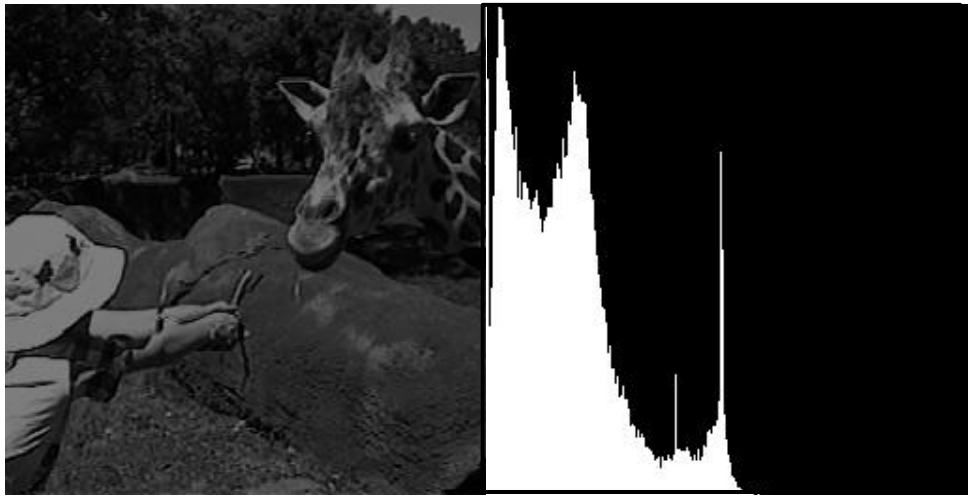
Color Histogram



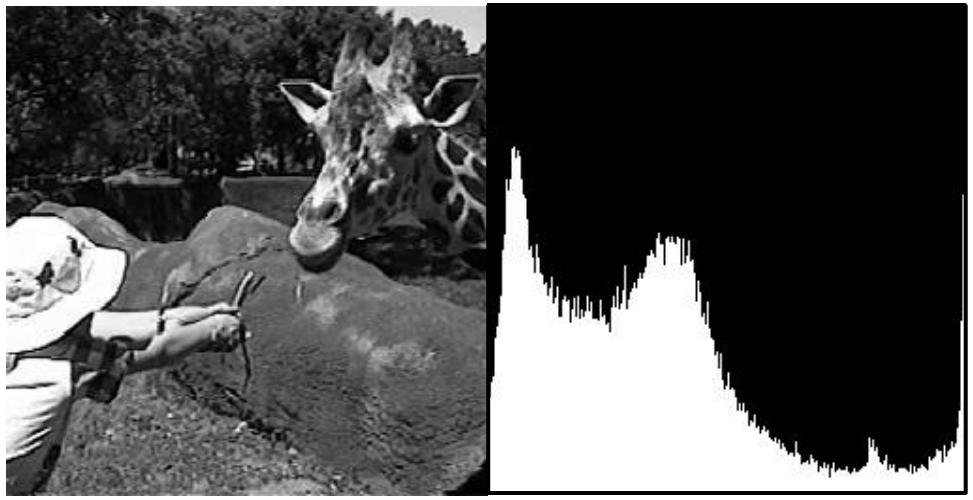
Histograms

Bright

Dark



High contrast



Low contrast

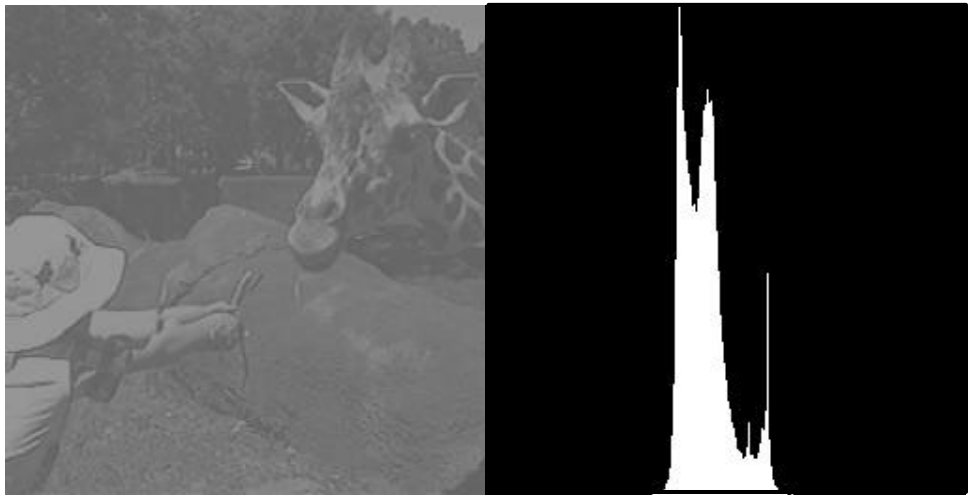


Image Formation

Camera: projection 3D-> 2D

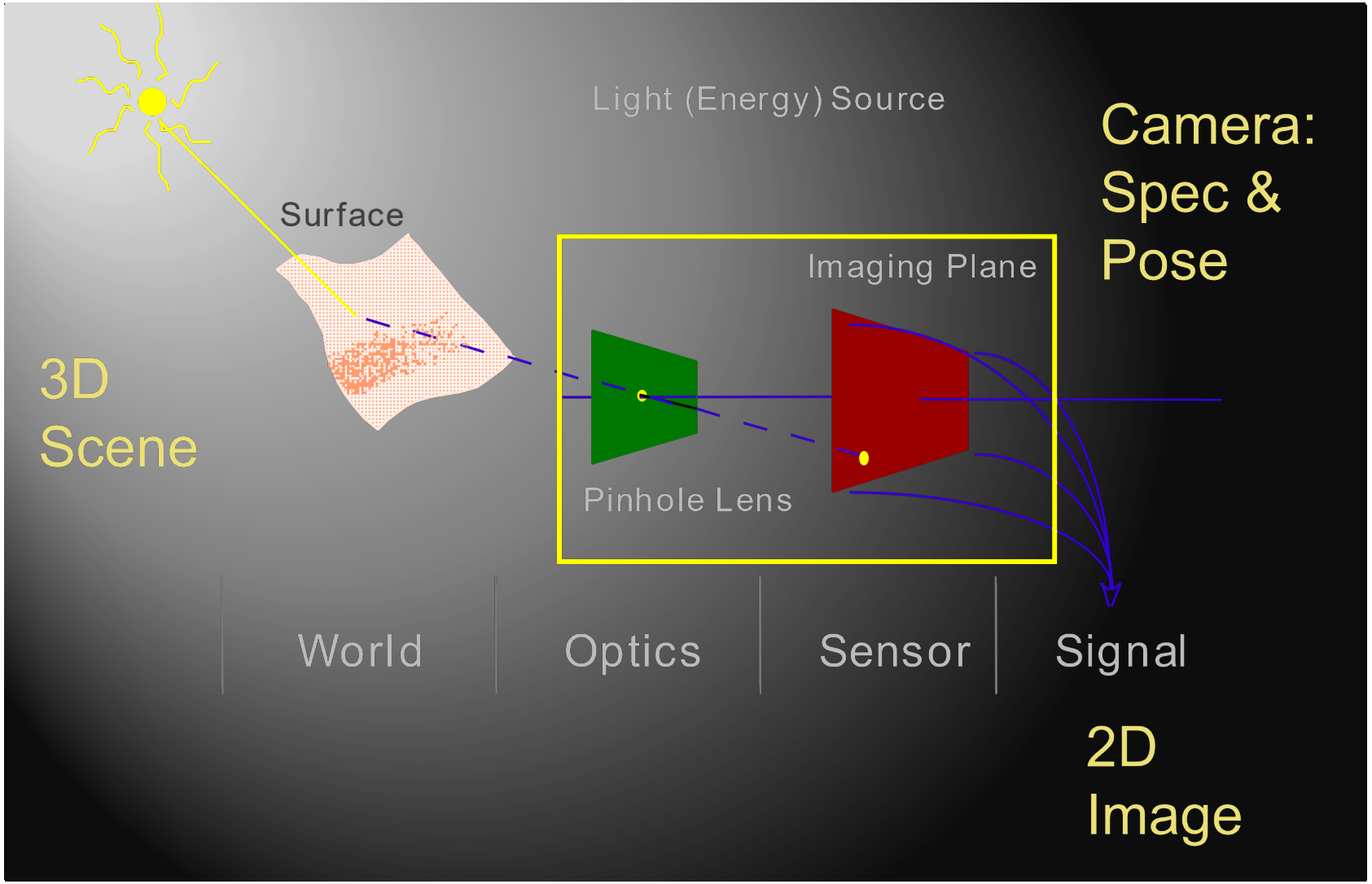


Image Formation

- World: reality
- Optics: focus light from world on sensor
- Sensor: converts light to electrical energy
- Signal: representation of incident light as continuous electrical energy
- Digitizer: converts continuous signal to discrete signal
- Digital Representation: representation of reality in computer memory

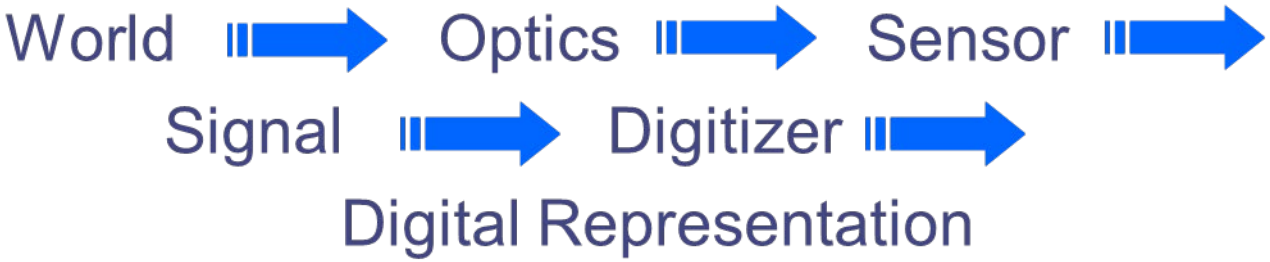


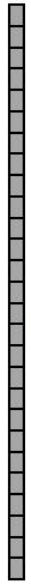
Image Formation

Camera Models

- Two models are commonly used:
  - Pin-hole camera
  - Thin lenses
- **Pin-hole** model: basis for most graphics and vision devices
  - Derived from physical construction of early cameras
  - Mathematics is very straightforward
- **Thin lens** model: first of the lens models
  - Lens gathers light over area and focuses on image plane
  - Mathematical model for a physical lens

Image Formation

real-world object



digital sensor (CCD or CMOS)

Image Formation

- All scene points contribute to all sensor pixels
- What does the image on the sensor look like?

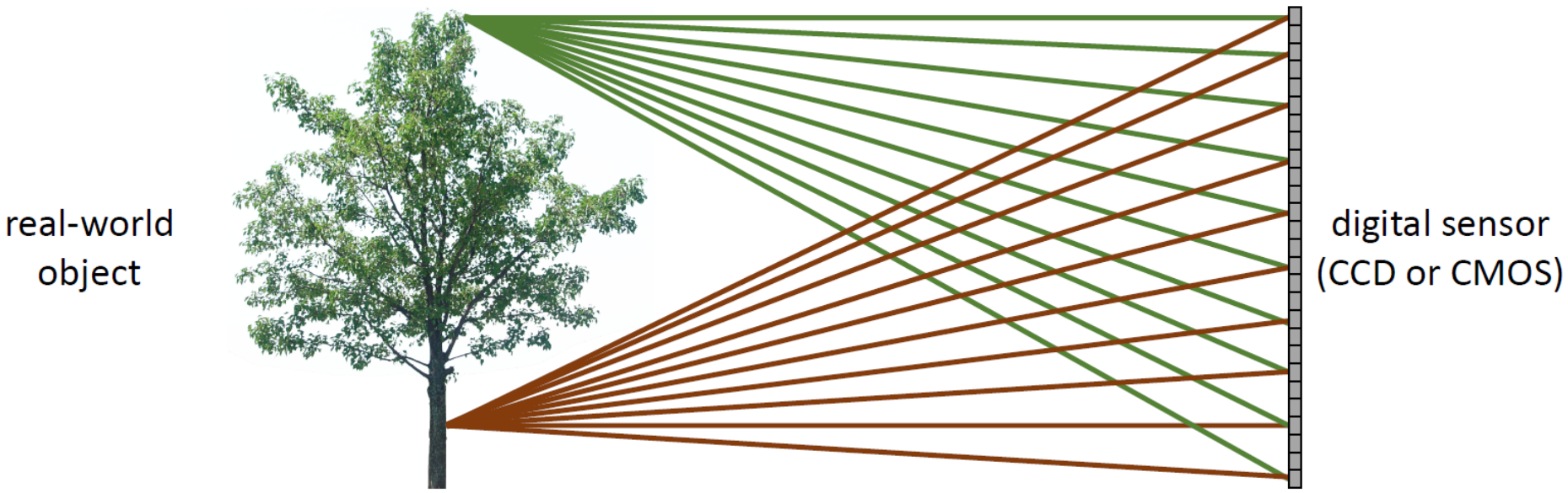
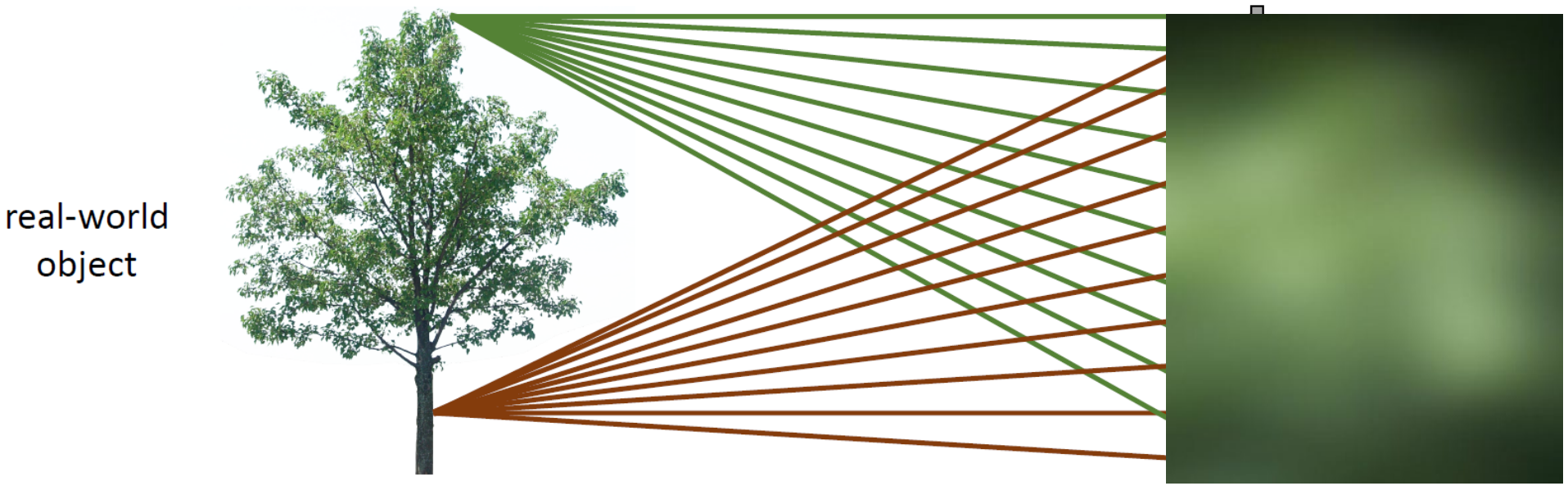


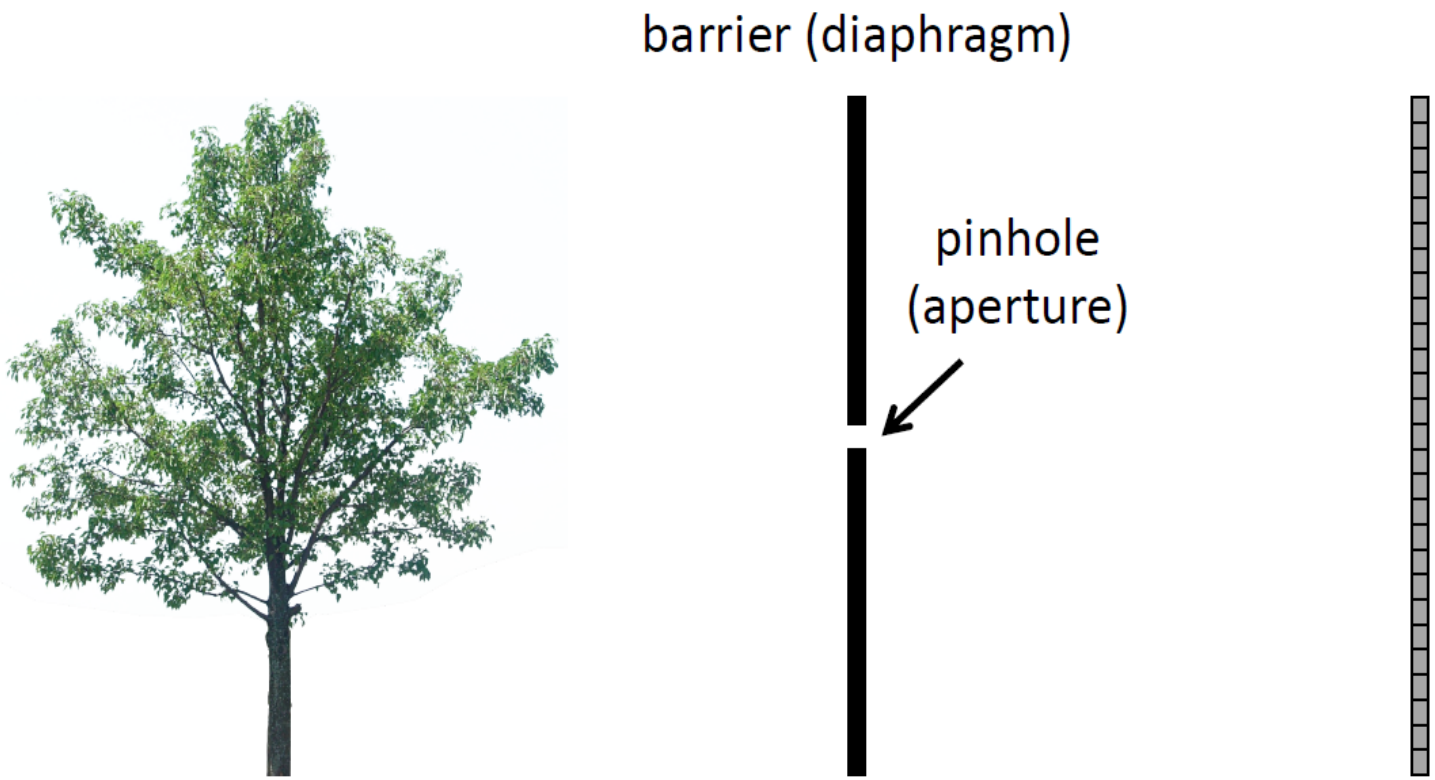
Image Formation

- All scene points contribute to all sensor pixels
- What does the image on the sensor look like?



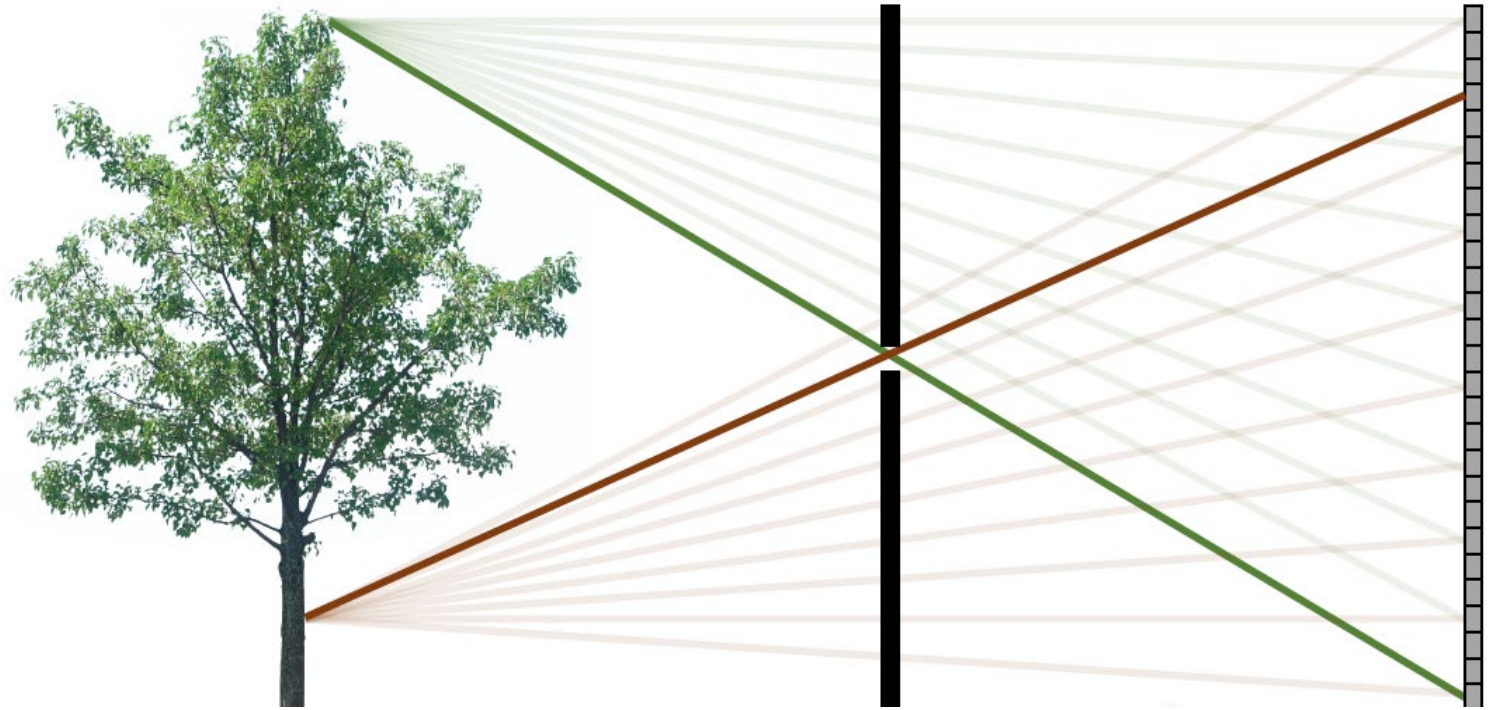
Pin-hole Model

- Bijective (one-to-one) relationship between image pixels and object surface points



Pin-hole Model

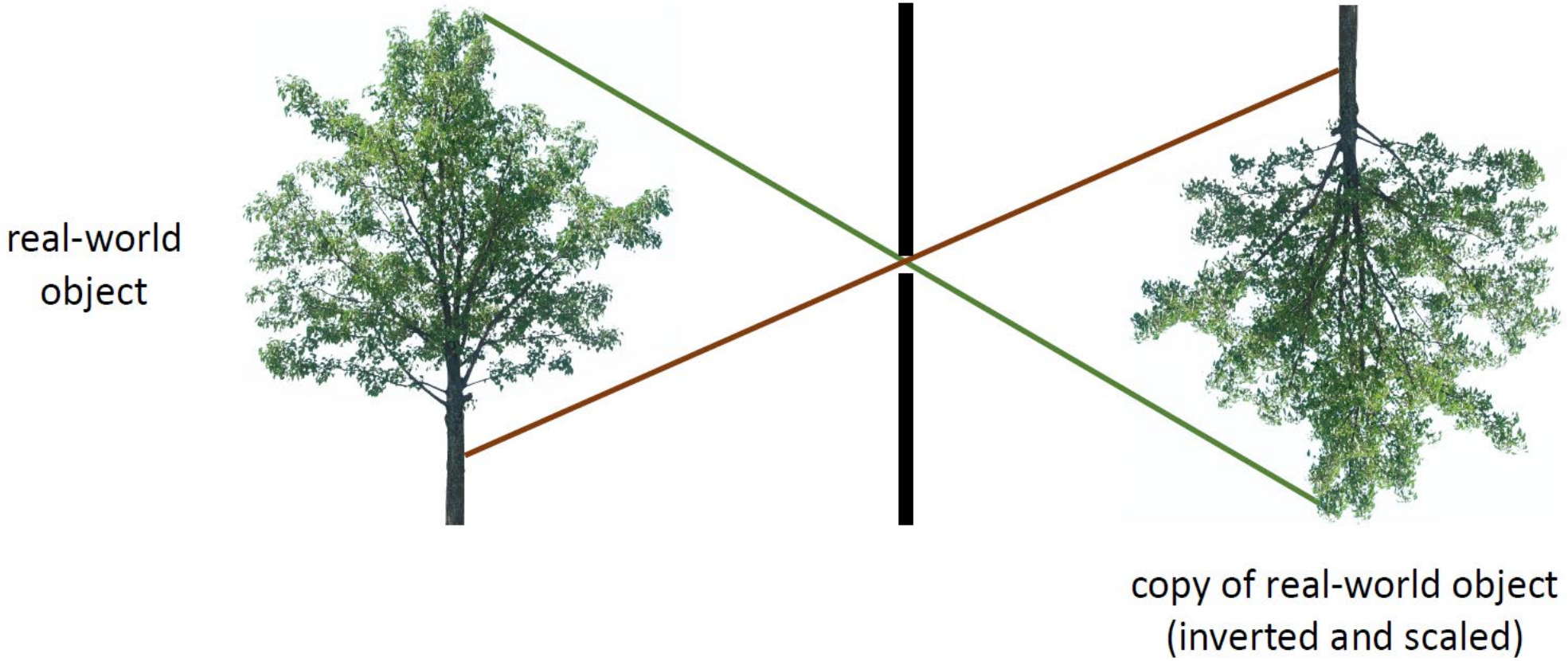
- Most rays are blocked
- Each scene point contributes to only one sensor pixel



- What does the image on the sensor look like?

Pin-hole Model

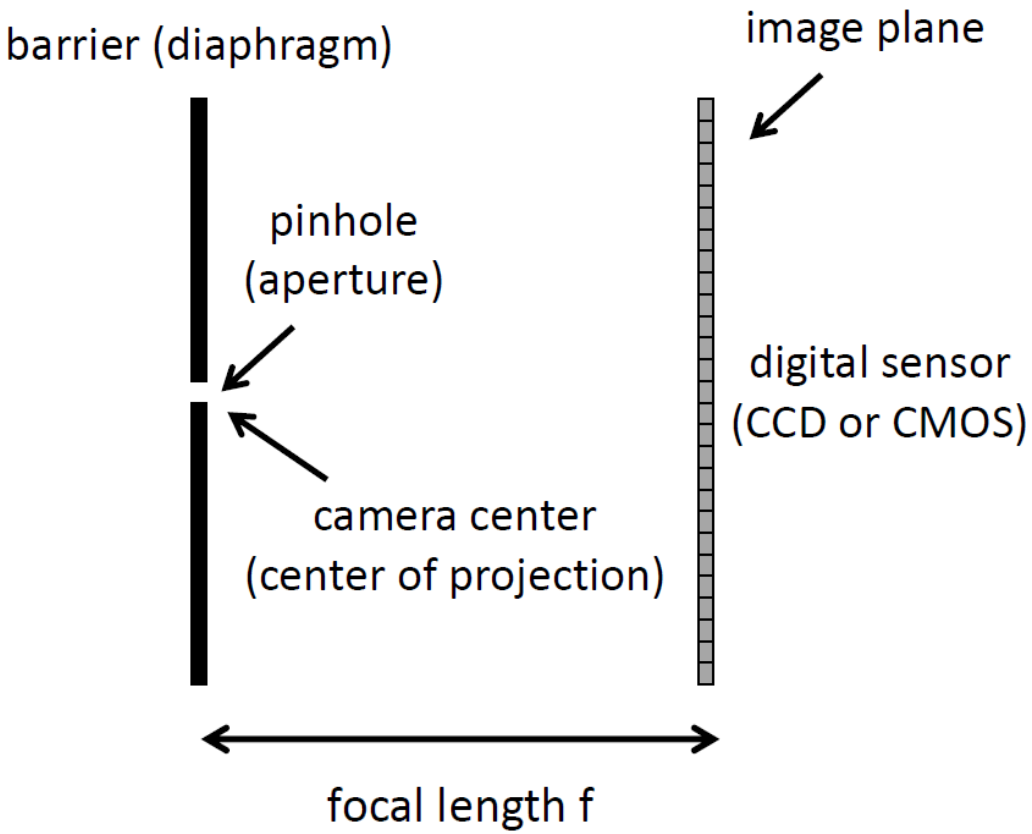
- Inverted and scaled object projection



Pin-hole Camera

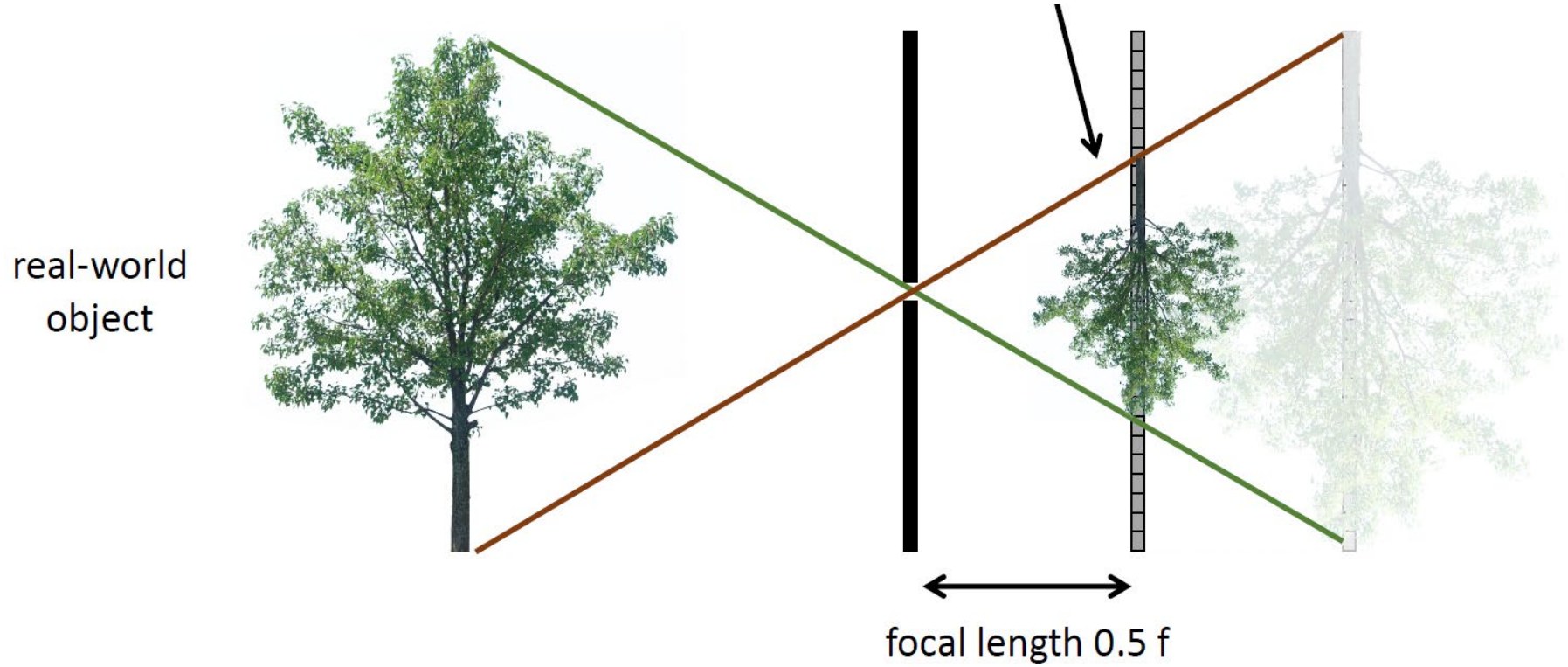
- Camera parameters:
  - Focal length
  - Aperture

real-world object



Pin-hole Camera

- Object image size depends on
  - Focal length (proportional)
  - Distance from real object to camera (inversely proportional)



Pin-hole Camera

- What happens as the diameters of the pin-hole increases?
- Object projection becomes blurrier….

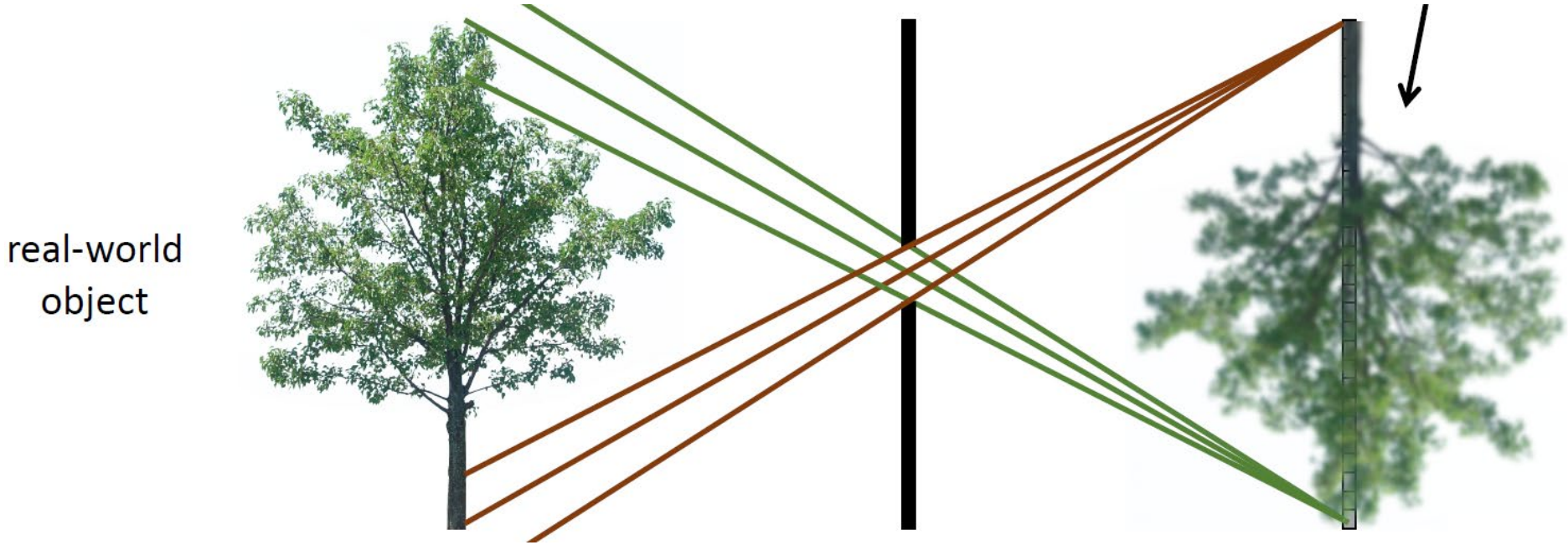
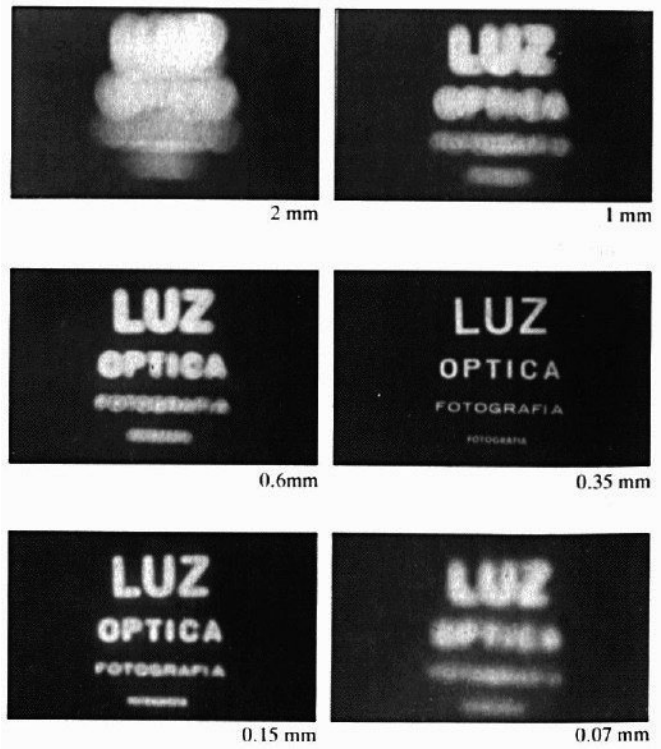


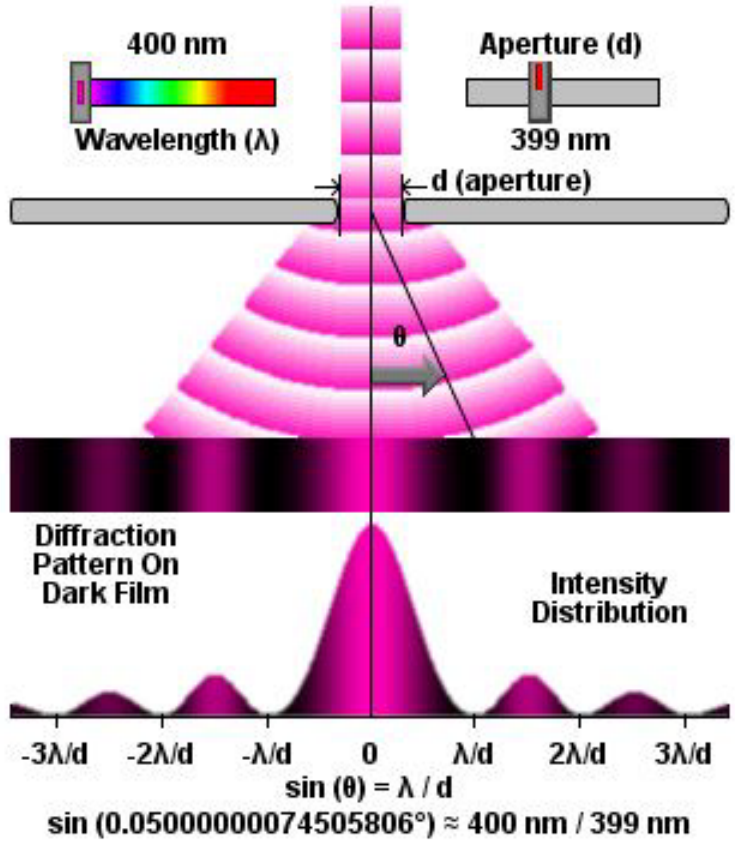
Image Formation

Camera models

- Why not make the aperture as small as possible?
  - Less light would pass
  - Diffraction issues ...



- Diffraction: spreading out of a wave front when it passes through an aperture which size is of same order of magnitude as wavelength



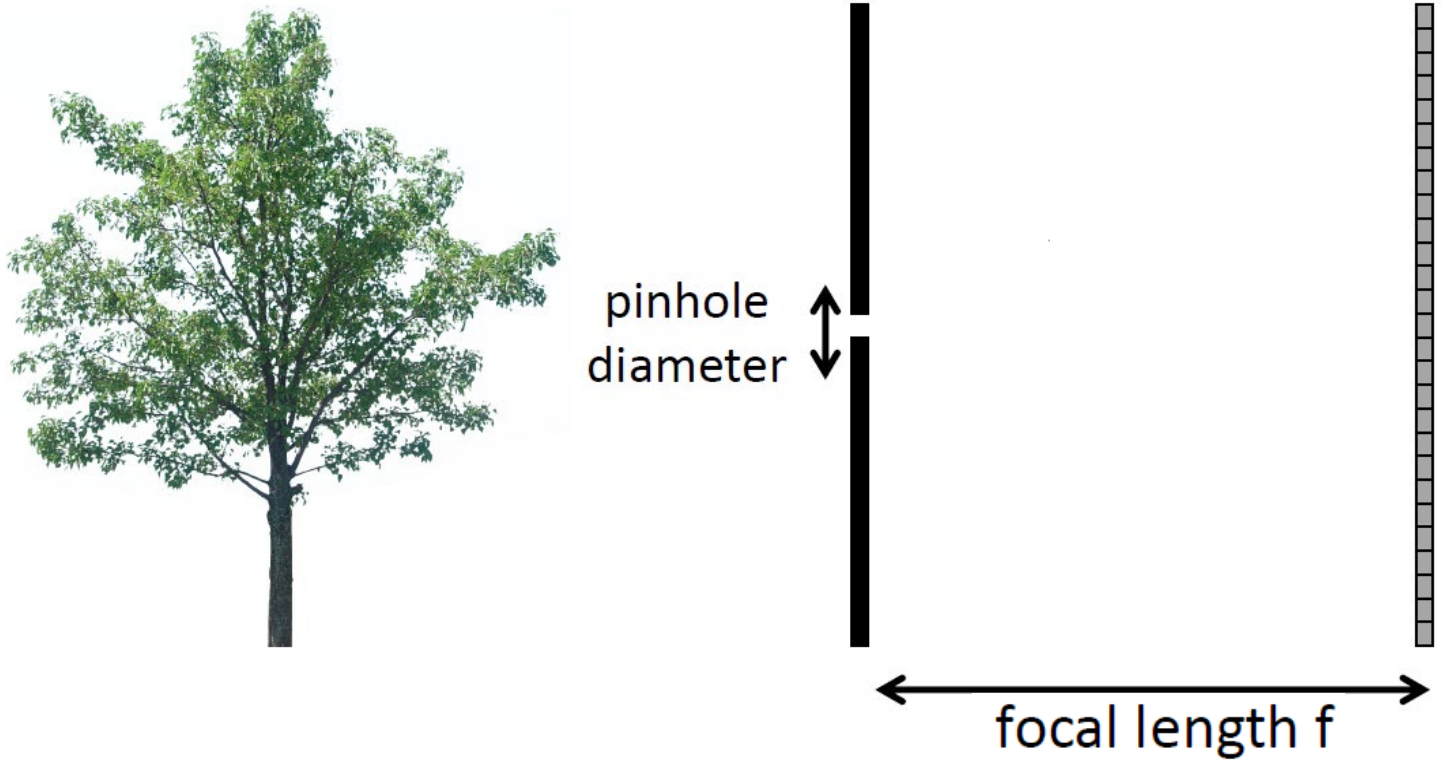
Pin-hole Camera

Light efficiency

- What is the effect of doubling the pinhole diameter?
  - 2x pinhole diameter → 4x light $\pi r^2 \rightarrow \pi(2r)^2$ **double hole radius, so 4 times more incident light**

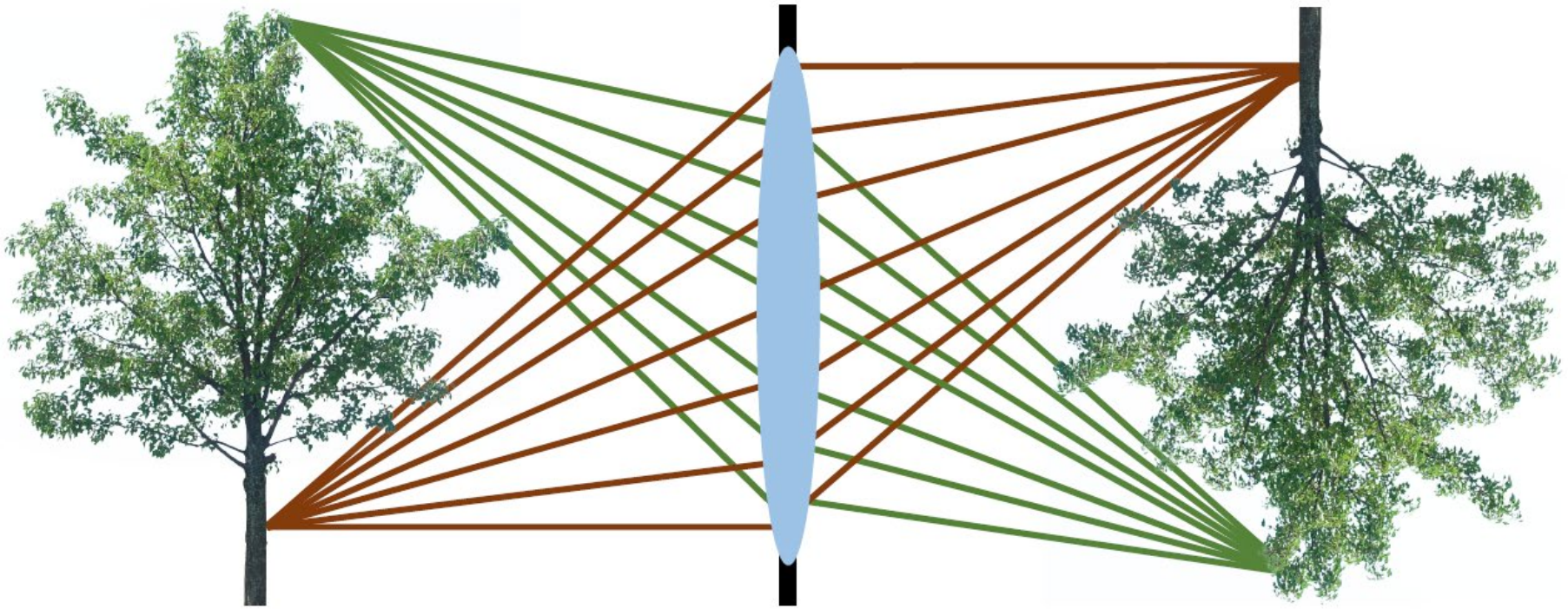
- What is the effect of doubling the focal length?
  - 2x focal length → 1/4x light

double f, then 4 times of spreading area, so 1/4 of incident light per pixel



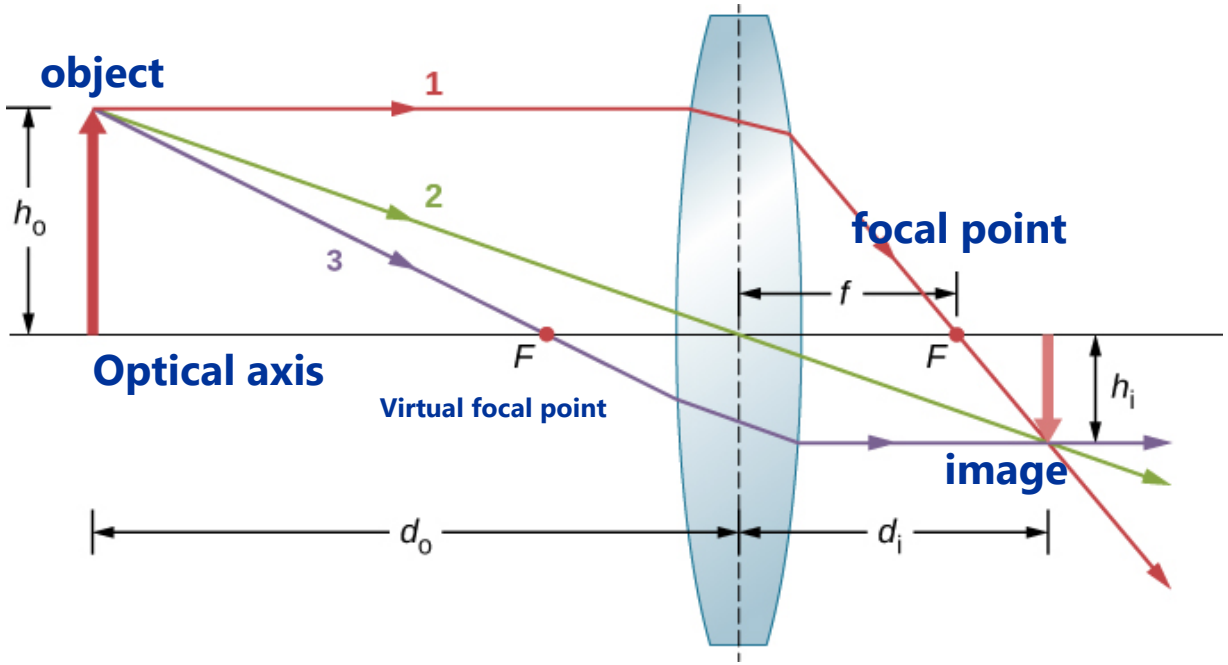
Lens Camera

- Lens: optical device which focuses light on the sensor.
  - Increase the aperture to capture more light in a larger area
  - Focuses light reflections from the same surface area in the same image pixels.
  - All in all, it improves resolution



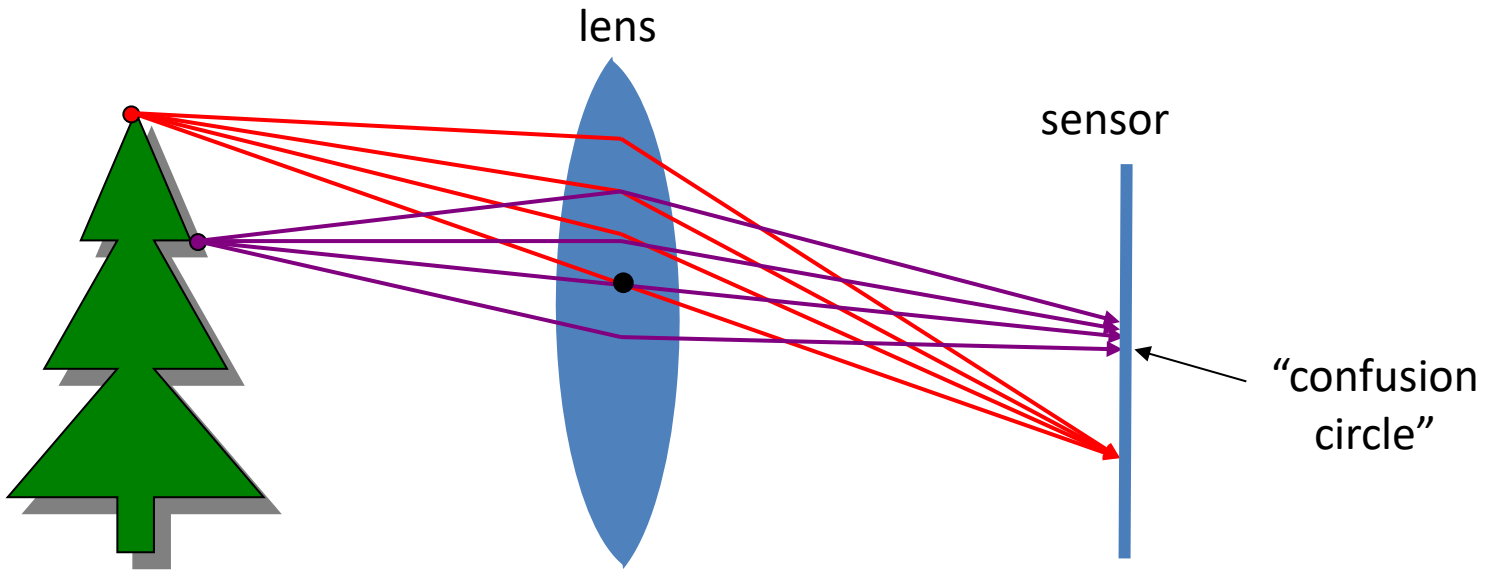
Lens Camera

- The three basic light rays used in ray tracing:
  - A ray which leaves the object parallel to the optical axis, is refracted to pass through the focal point (1).
  - A ray which passes through the lens's center is undeflected (2).
  - A ray passing through the (virtual) focal point is refracted to end up parallel to the optical axis (3).



Lens

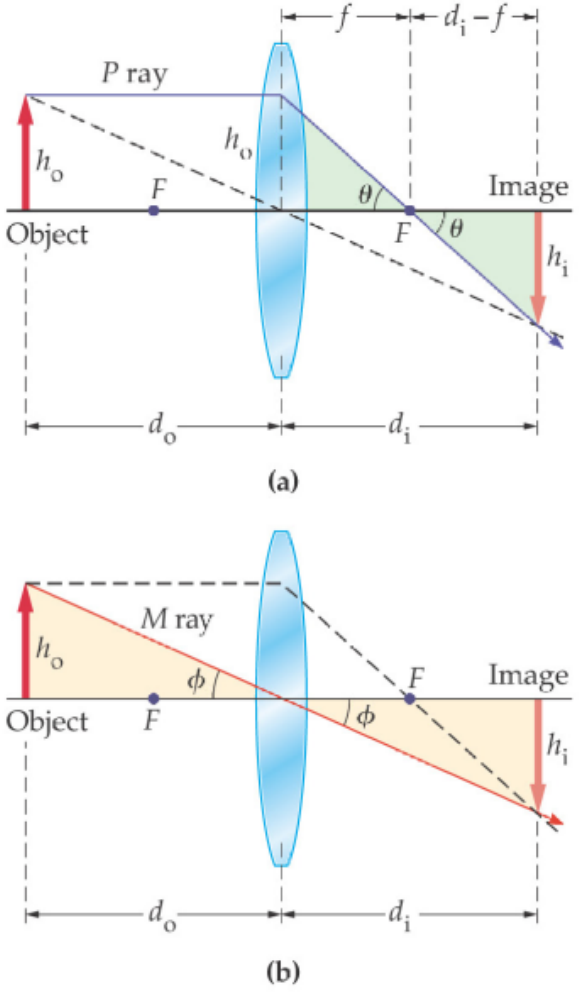
- There is a specific distance at which objects are "in focus"
- Other points in the scene are projected in a "circle of confusion" in the image.



- Reflected light from surface areas that are in focus projects in (around) the same cell in sensor. In particular rays of type (1) and (2)….

Lens

- Points on object surface that satisfy the thin lens equation are in focus



$$\frac{h_0}{f} = \frac{-h_i}{d_i - f}$$

$$\frac{h_o}{d_o} = \frac{-h_i}{d_i}$$

Thin Lens Equation

$$\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f}$$

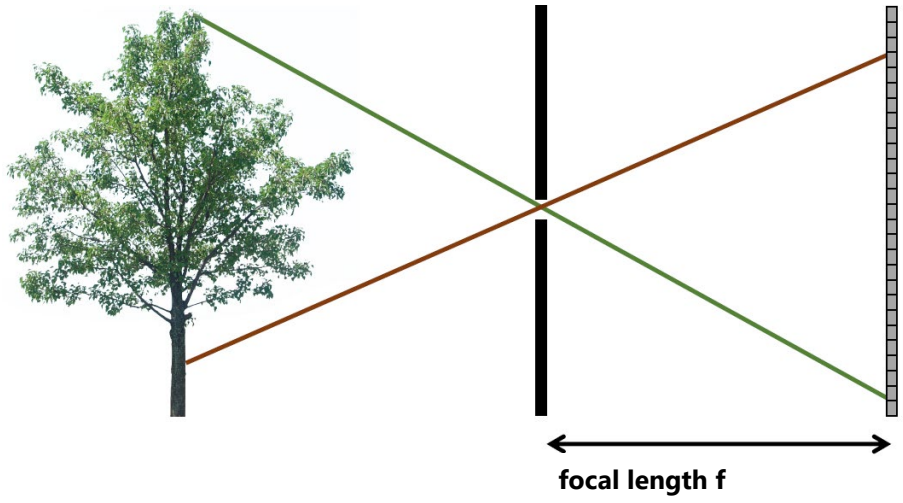
Magnification

$$m = -\frac{d_i}{d_o} = \frac{h_i}{h_o}$$

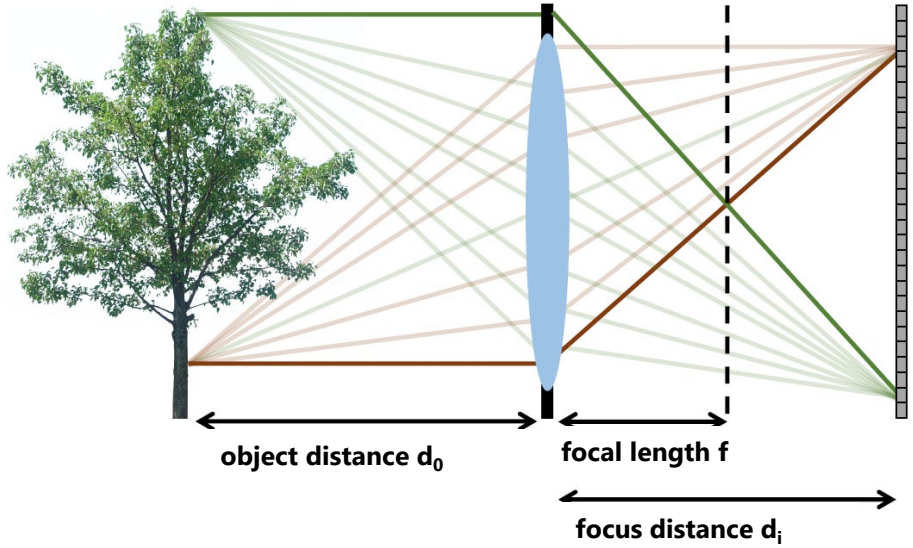
In converging lens with real object at the side of the lens where rays come from, and images in the opposite side, all values are positive except h_i. Negative values of m indicate invertion.

Image Formation

- In a pinhole camera, focal length is distance between aperture and sensor



- In a lens camera, focal length is distance where parallel rays intersect



- We can derive properties and descriptions that hold for both camera models if:
  - We use only central rays.
  - We assume the lens camera is in focus.
  - We assume that the focus distance of the lens camera is equal to the focal length of the pinhole camera

Image Formation

Perspective Projection

- Let's have a look at the camera model without considering its pose:
  - 3D World projected to 2D Image (No direct depth information)
  - Inverted image, reduced size
  - Center of the lens (center of projection): point through which all rays pass.
  - Z axis coincides with the optical axis (also called the central projection ray).
    - Image plane is located at Z = -f, lens at Z = 0, and virtual image at Z=f.
    - Z is also the distance to an object as measured along the optical axis. The X and Y axis lie in the image plane.

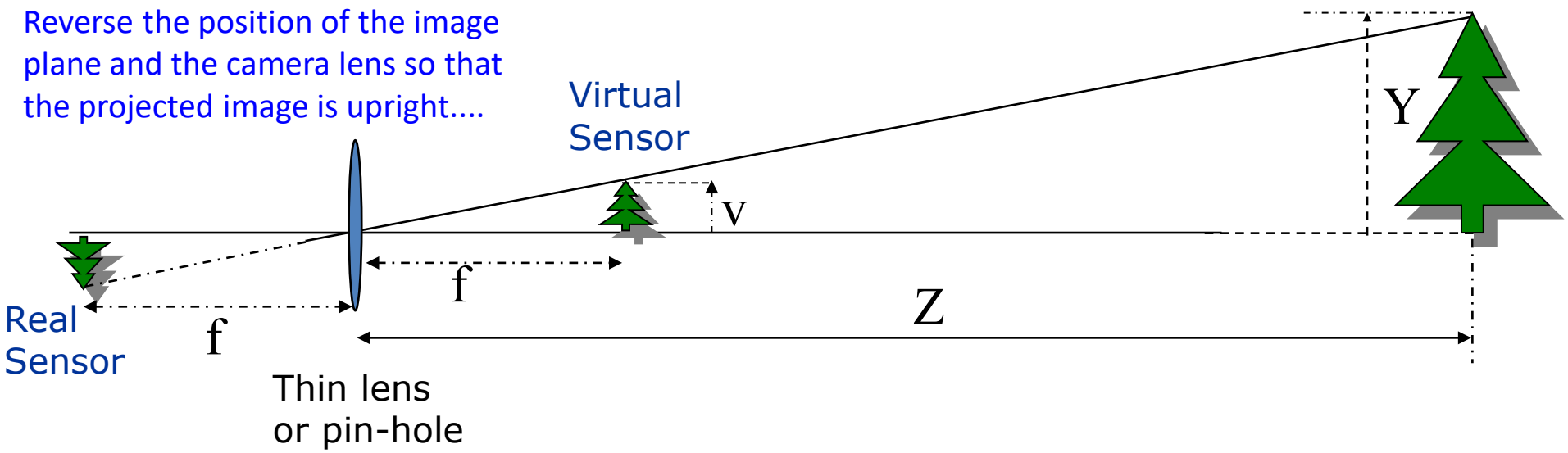


Image Formation

Perspective Projection

- Given a point **P** in 3D space, project it onto the image plane z= f at **p**
  - What is the equation for image coordinate **p** in terms of **P**?

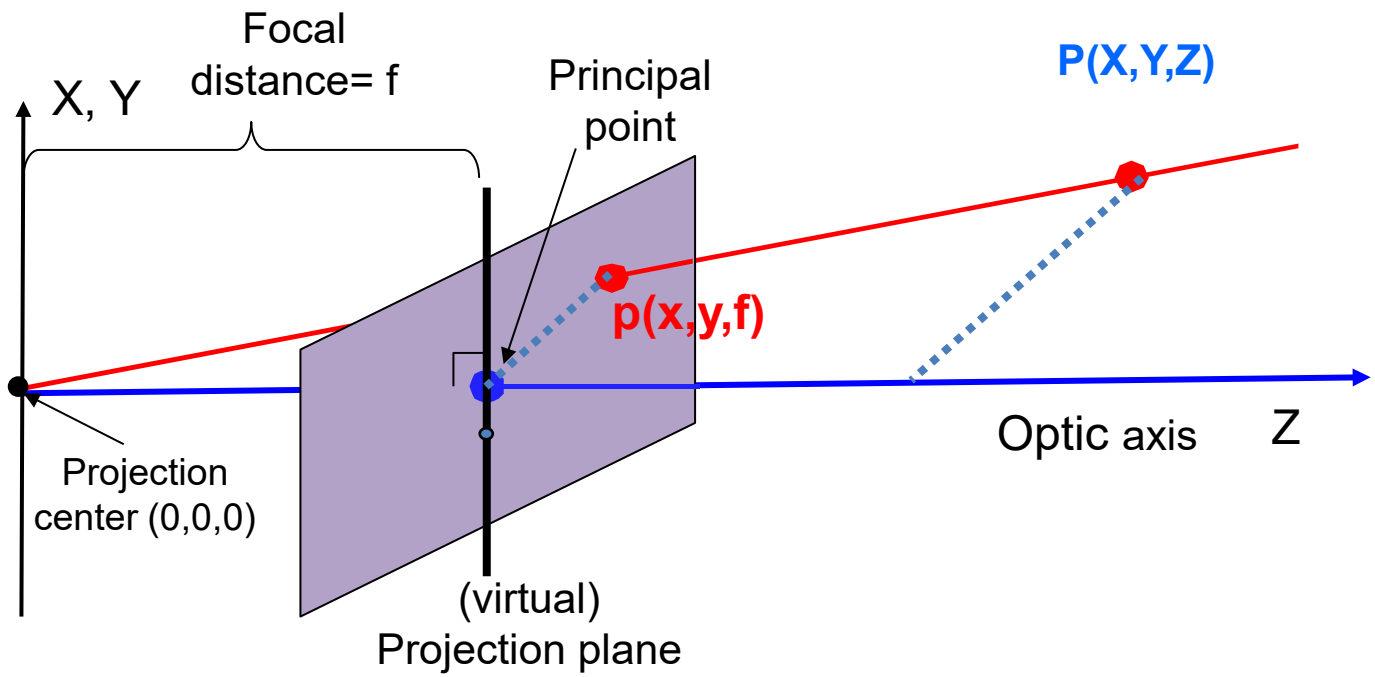
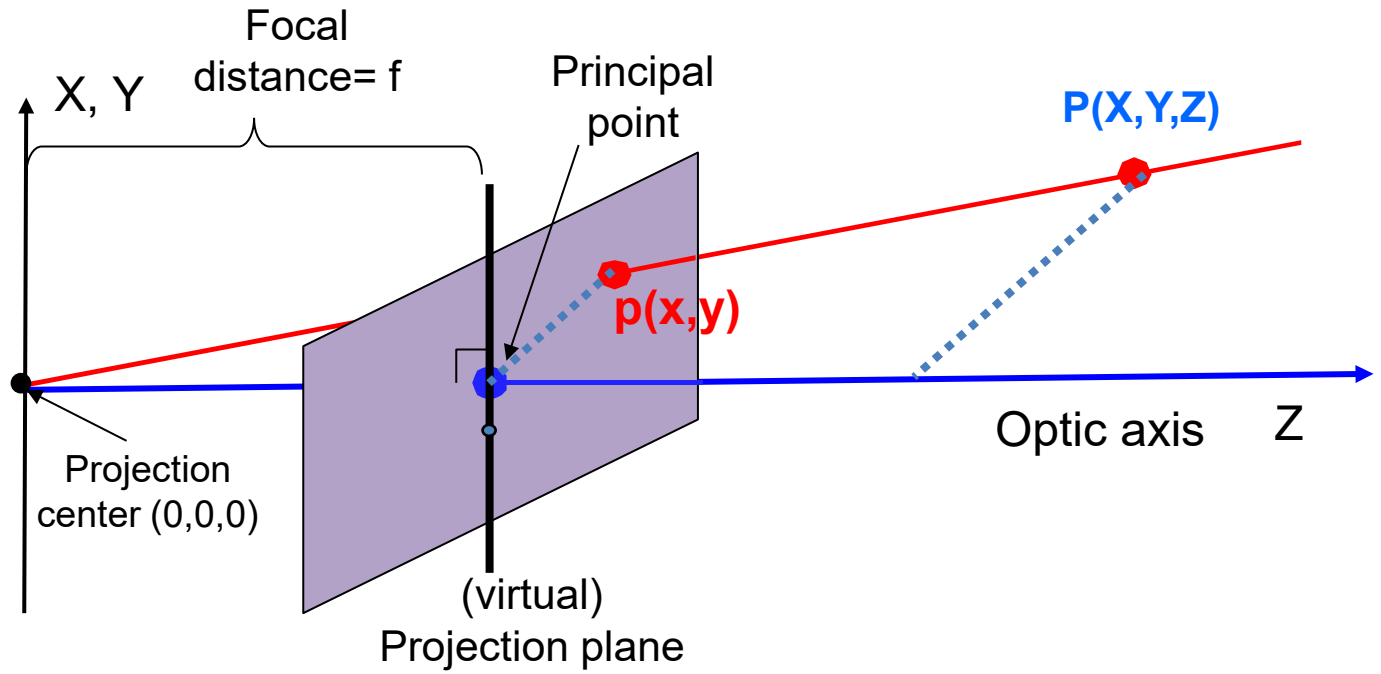


Image Formation



- Using similar triangles:

  $x/f=X/Z \rightarrow x=fX/Z$ $\quad\quad$ $y/f=Y/Z \rightarrow y=fY/Z$

- Projection equation: $\quad\quad$ **$P=(X, Y, Z) \rightarrow p= (fX/Z, fY/Z)$**

Non linear transformation!

Homogeneous coordinates

General camera model in homogeneous coordinates:

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

From image/scene coordinates to homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \rightarrow \left(\frac{x}{w}, \frac{y}{w}\right) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$$

Converting from homogeneous coordinates

Using **homogeneous coordinates** we can express relationship between 3D point and its 2D projection as a vector-matrix product:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

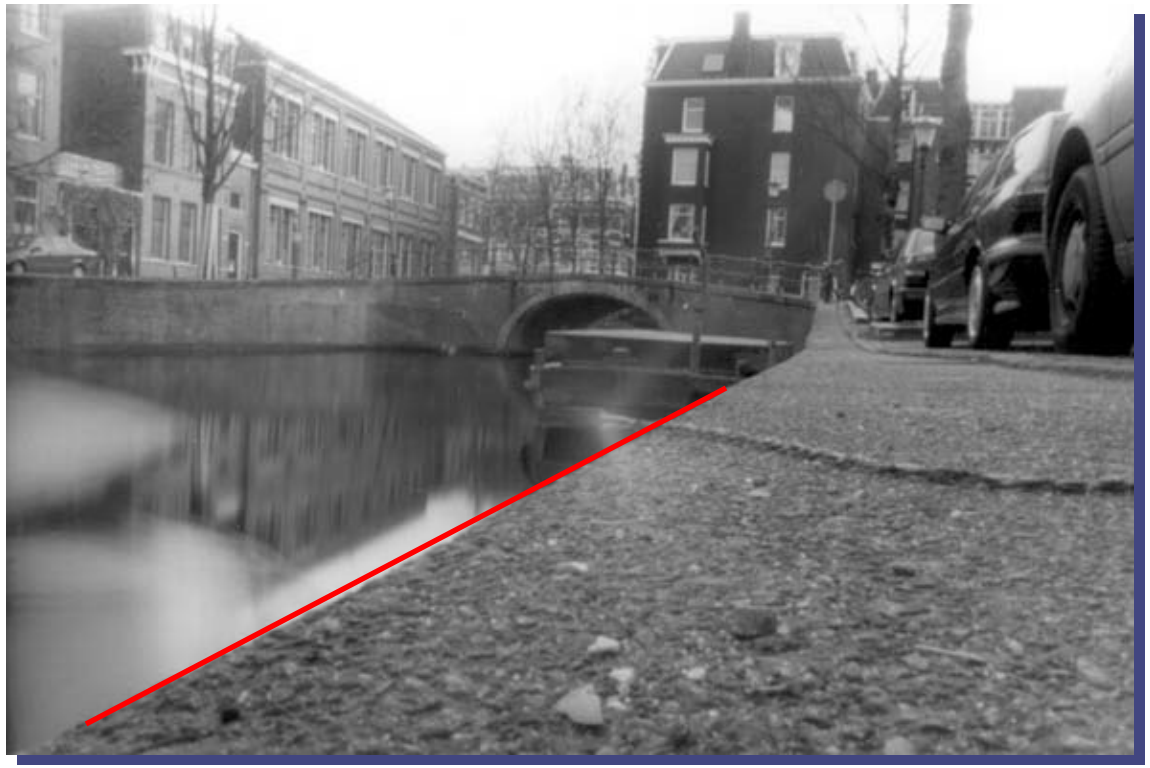
What does the pinhole camera projection matrix look like?

$$\mathrm{K} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Check it!

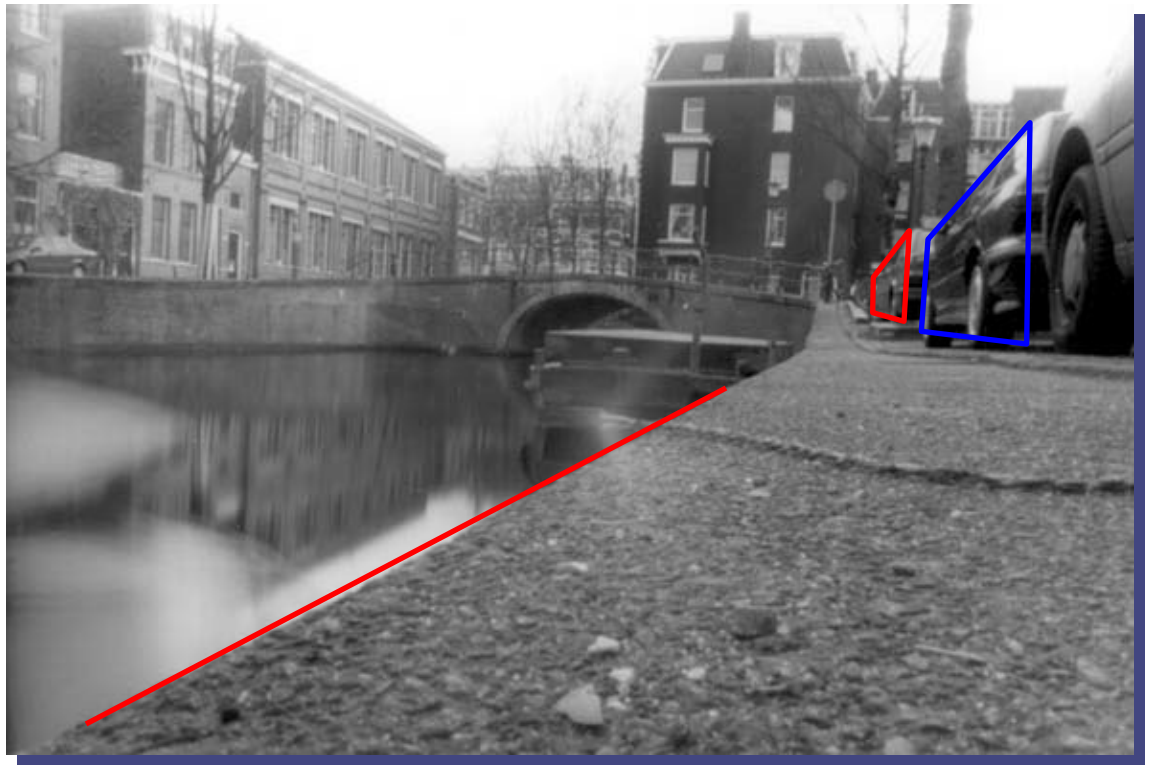
Perspective Projection

- ✓ straight line
- size
- parallelism/angle
- shape
- shape of planes



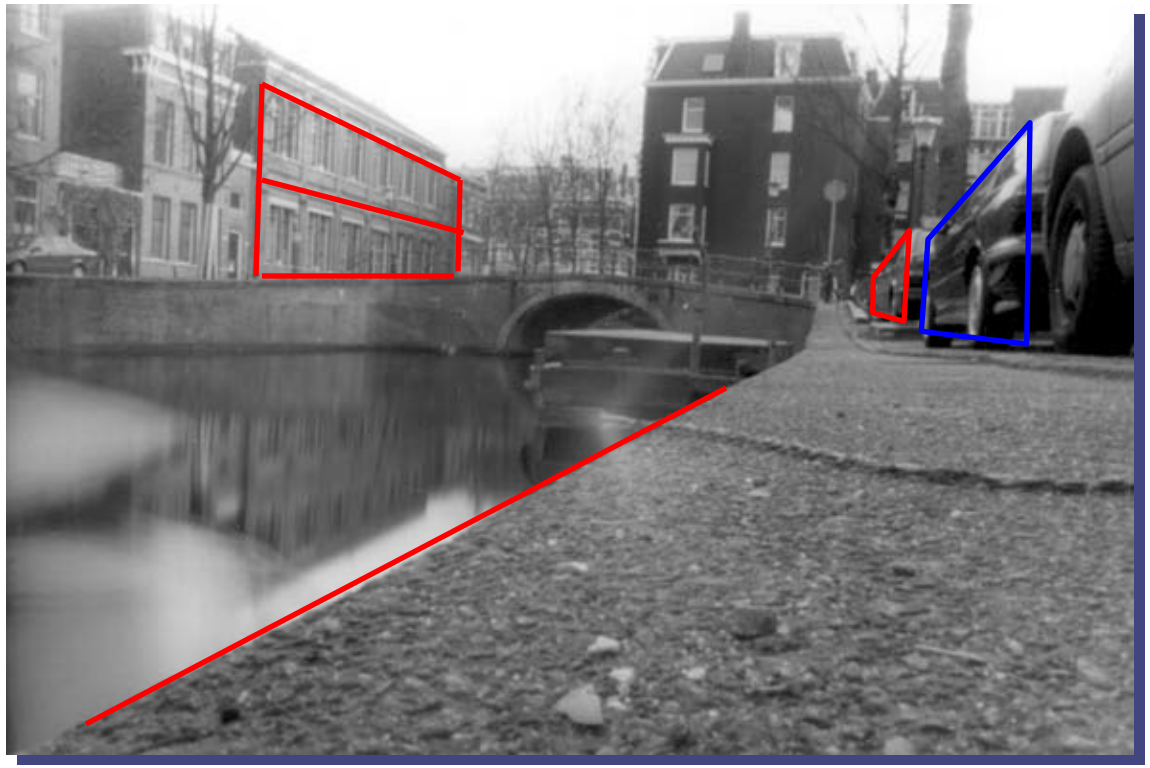
Perspective Projection

- ✓ straight line
- x size
- parallelism/angle
- shape
- shape of planes



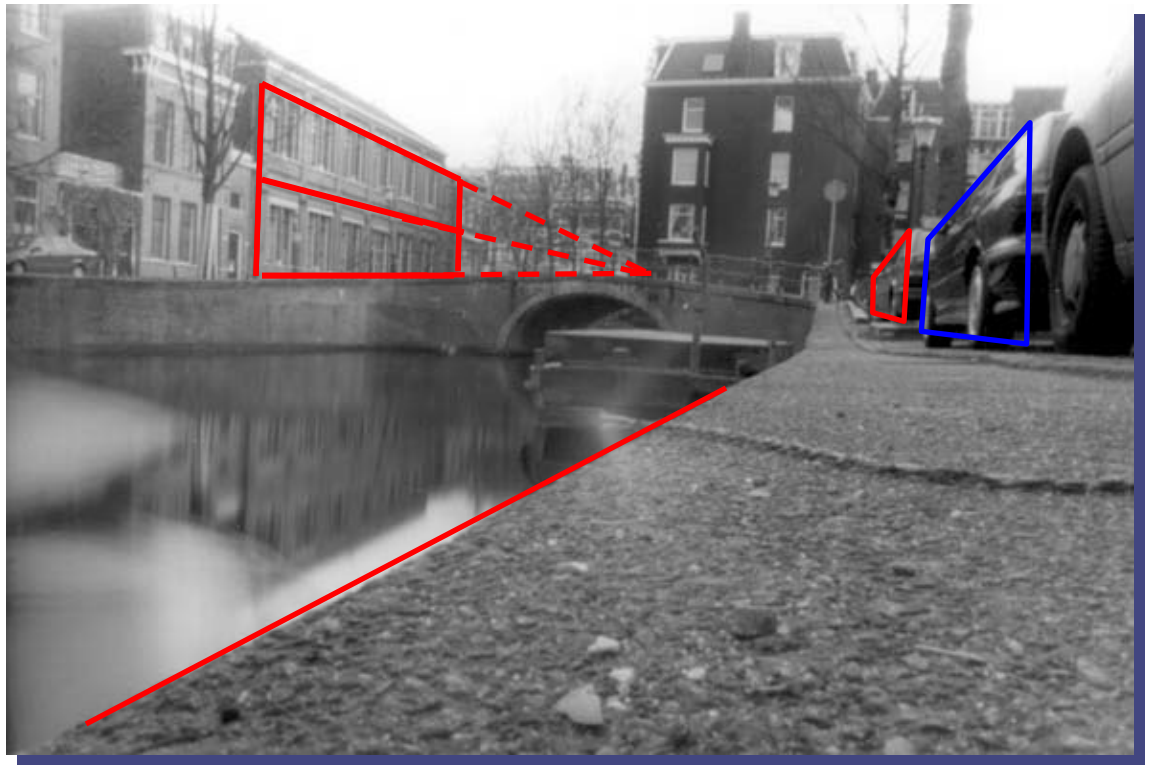
Perspective Projection

- ✓ straight line
- x size
- x parallelism/angle
- shape
- shape of planes



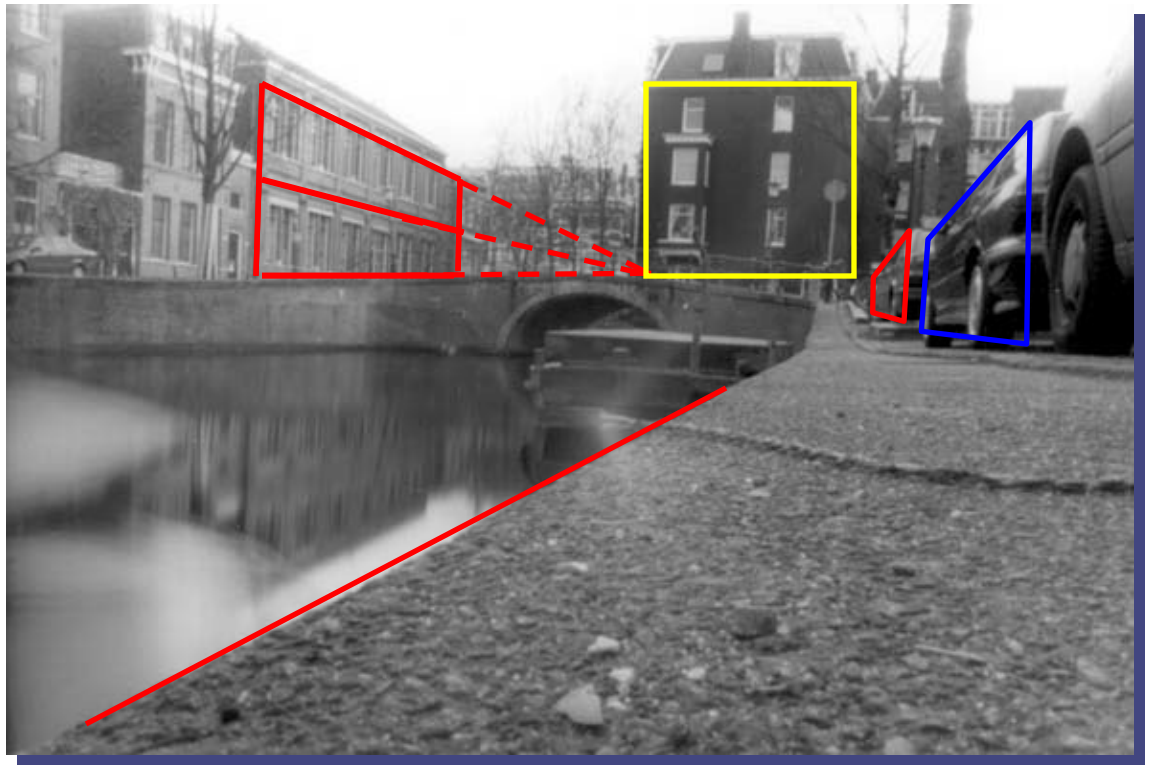
Perspective Projection

- ✓ straight line
- x size
- x parallelism/angle
- x shape
- • shape of planes



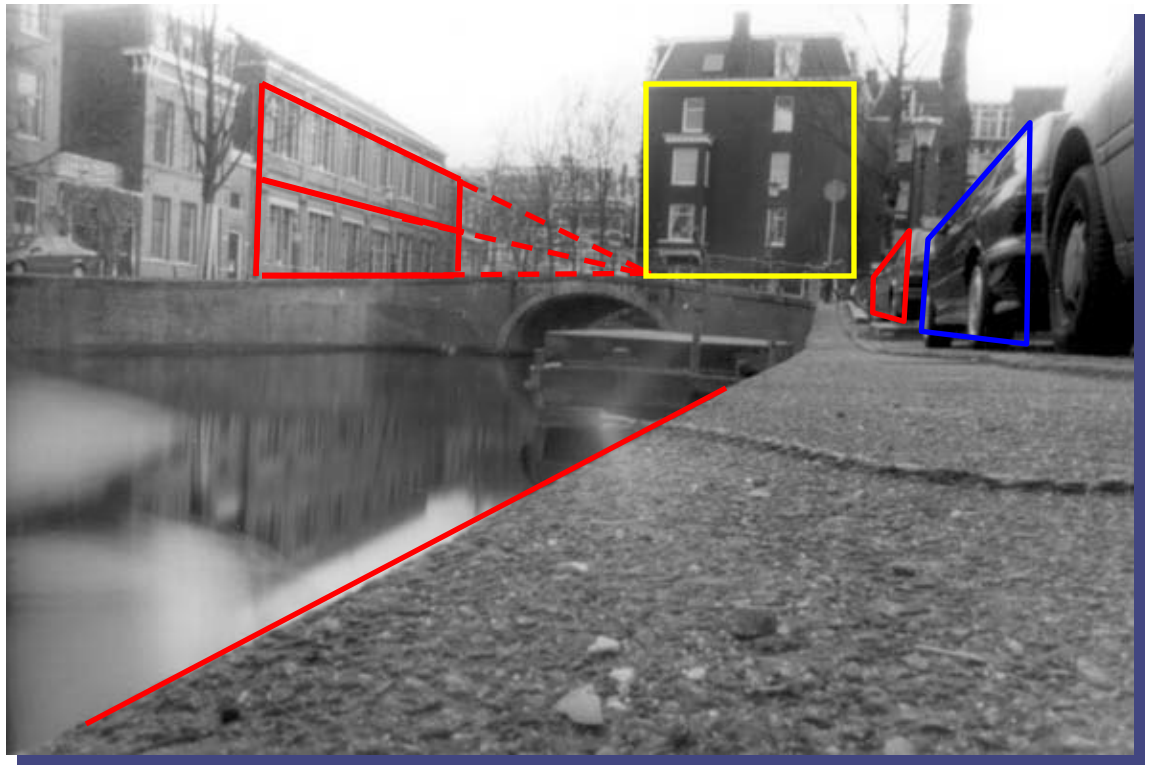
Perspective Projection

- ✓ straight line
- x size
- x parallelism/angle
- x shape
- ✓ shape of planes parallels to image plane



Perspective Projection

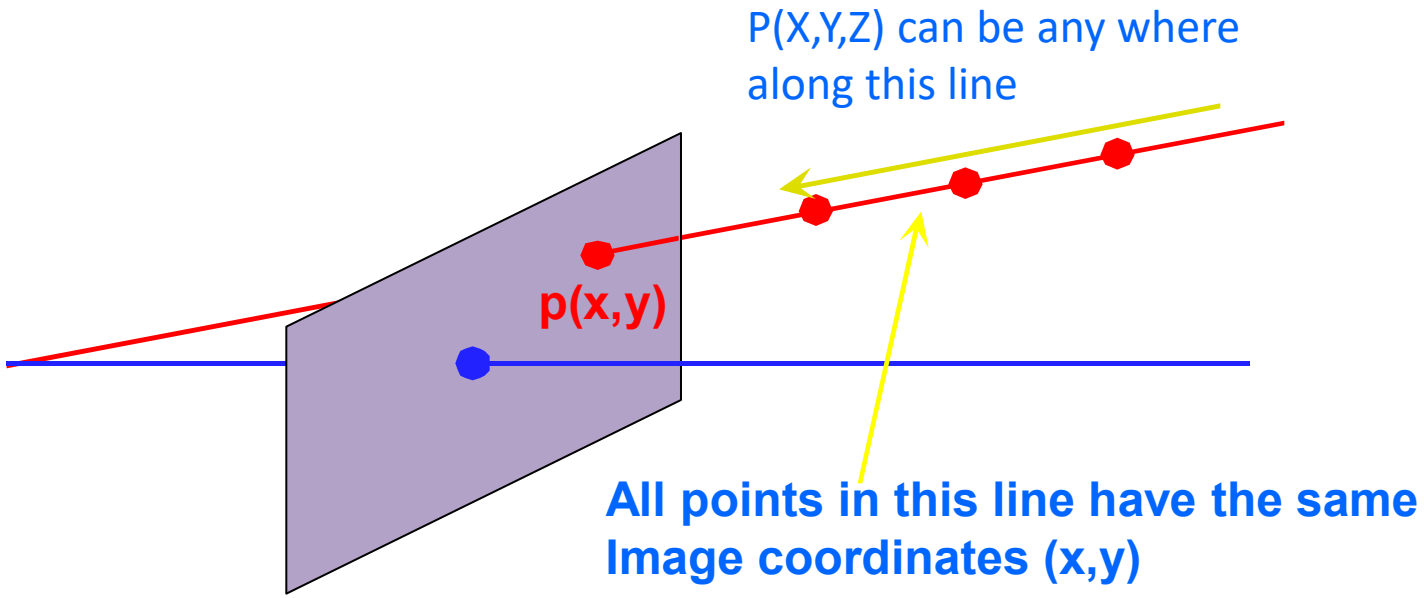
- ✓ straight line
- x size
- x parallelism/angle
- x shape
- ✓ shape of planes parallels to image
- ❑ Depth?



Monocular Vision

Reverse projection

Given a center of projection and image coordinates of a point, it is not possible to recover the 3D depth of the point from a single image.



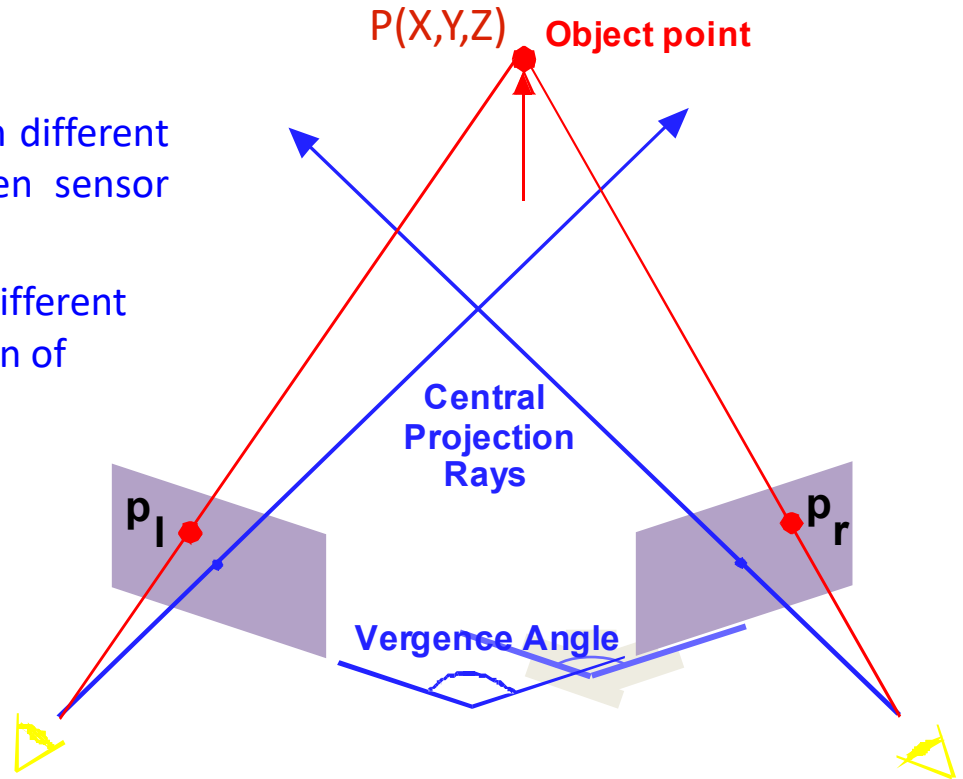
In general, at least two images of the same point taken from two different locations are required to recover depth.

Stereo Vision

- Converging Axes – Usual setup of human eyes
- Depth obtained by triangulation
- Correspondence problem: p_l and p_r correspond to the left and right projections of P, respectively.

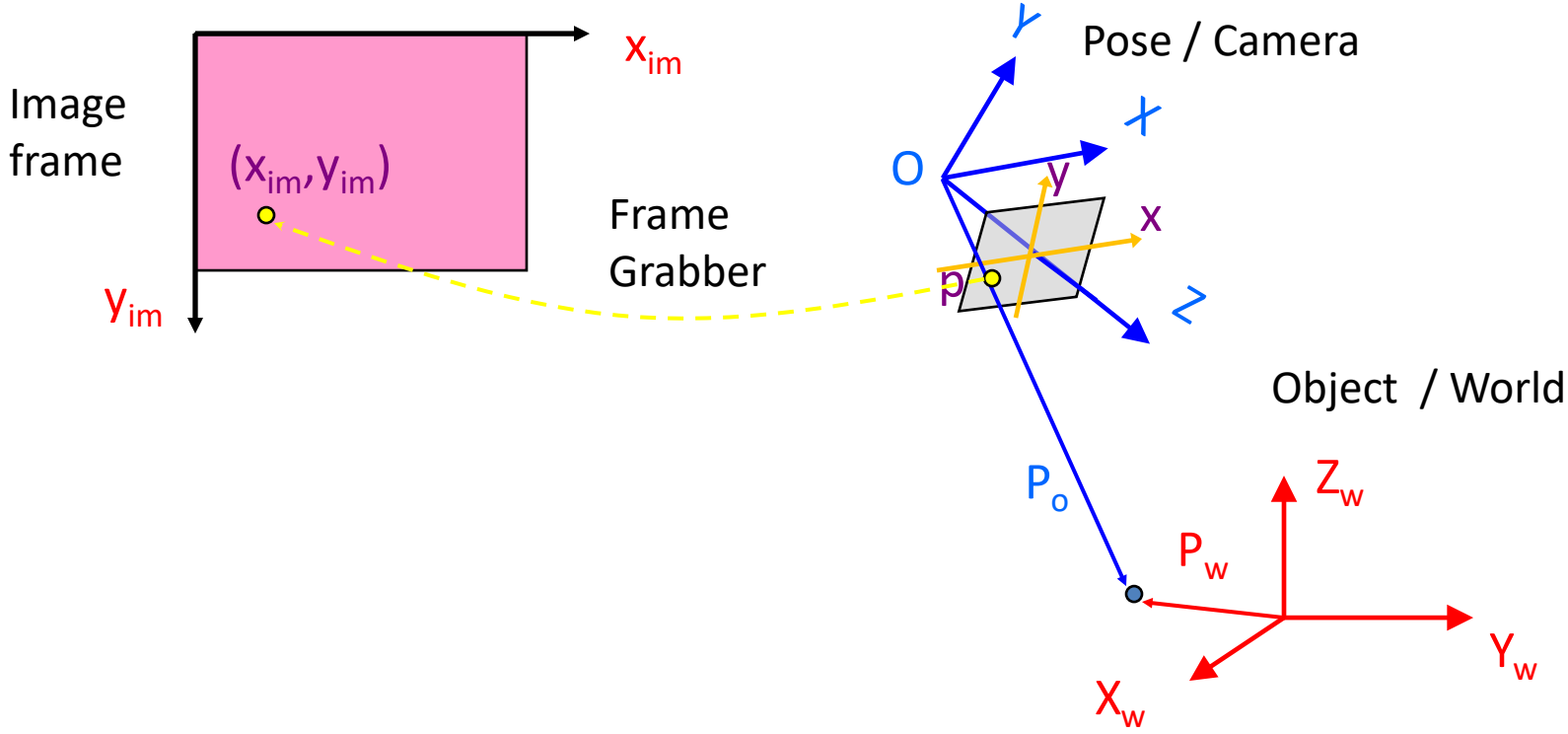
if we know the coordinates of P in two images taken from different world coordinate locations, and the relationship between sensor poses, then 3D information can be recovered.

In the human visual system, our two eyes produce slightly different images of the scene and from this we experience a sensation of depth.



General Camera Matrix

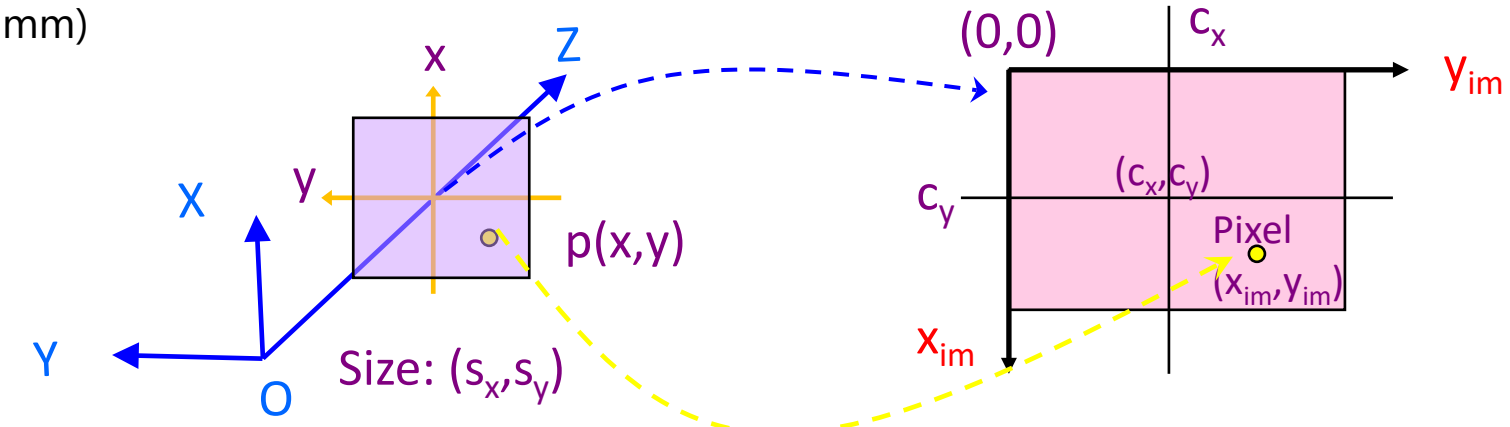
In general, camera, image and world have *different* coordinate systems.



General Camera Matrix

From Camera to Frame

- Intrinsic Parameters
  - (c_x, x_y): image plane center (in pixels)
  - (s_x, s_y): size of the pixel (in mm)
  - f: focal length



- From image plane to image frame:

$$x = -(x_{im} - c_x)s_x \qquad y = -(y_{im} - c_y)s_y$$

$$x_{im} = -\frac{x}{s_x} + c_x \qquad y_{im} = -\frac{y}{s_y} + c_y$$

- From 3D (camera reference) to 2D (image plane):

$$x = f\frac{X}{Z}, \qquad y = f\frac{Y}{Z}$$

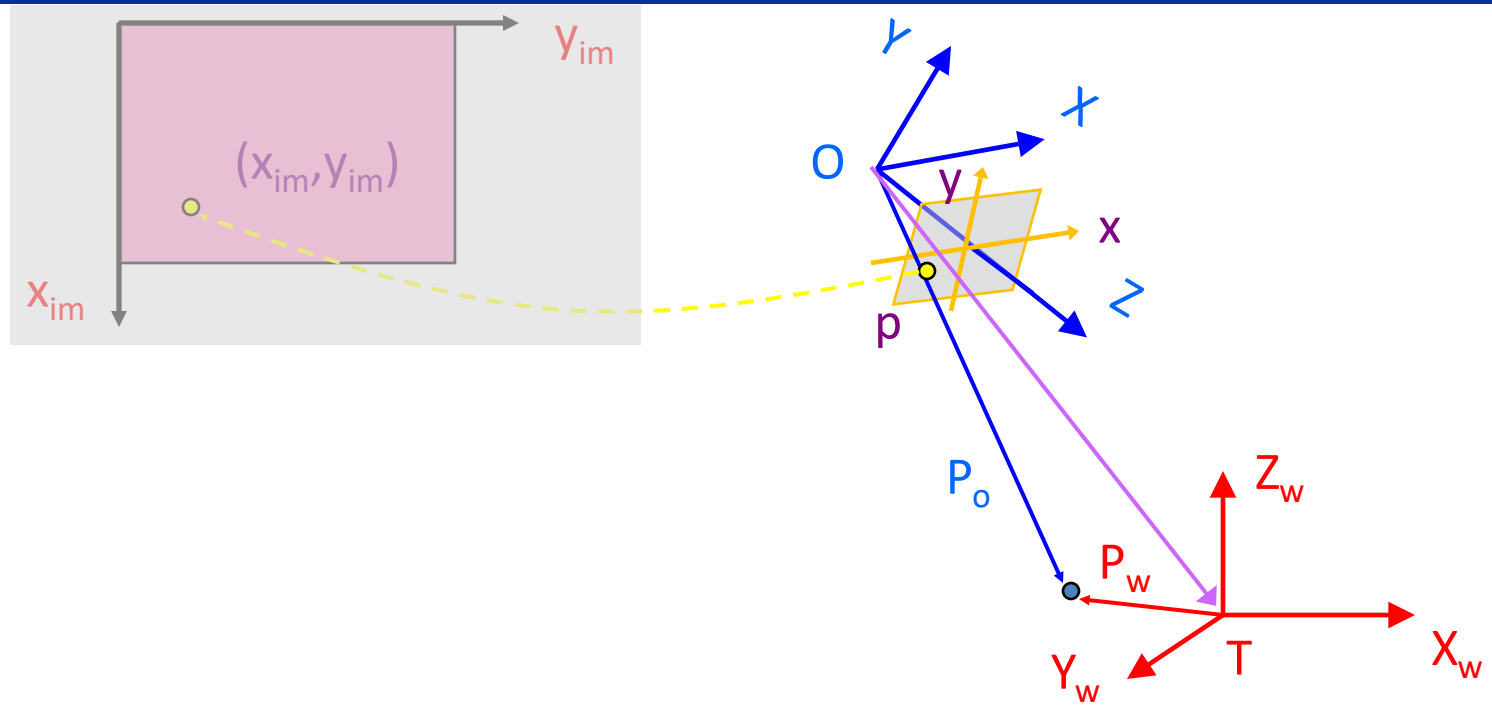
- **From 3D Camera Reference System to image Frame Reference System:**
  - (f_x, f_y) : focal length scaled (f/s)

$$\begin{bmatrix} x_{im} \\ x_{im} \\ w \end{bmatrix} = \mathbf{K} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad \mathbf{K} = \begin{bmatrix} -f_x & 0 & c_x & 0 \\ 0 & -f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Without considering lens distortions!!

General Camera Matrix

From World to Camera

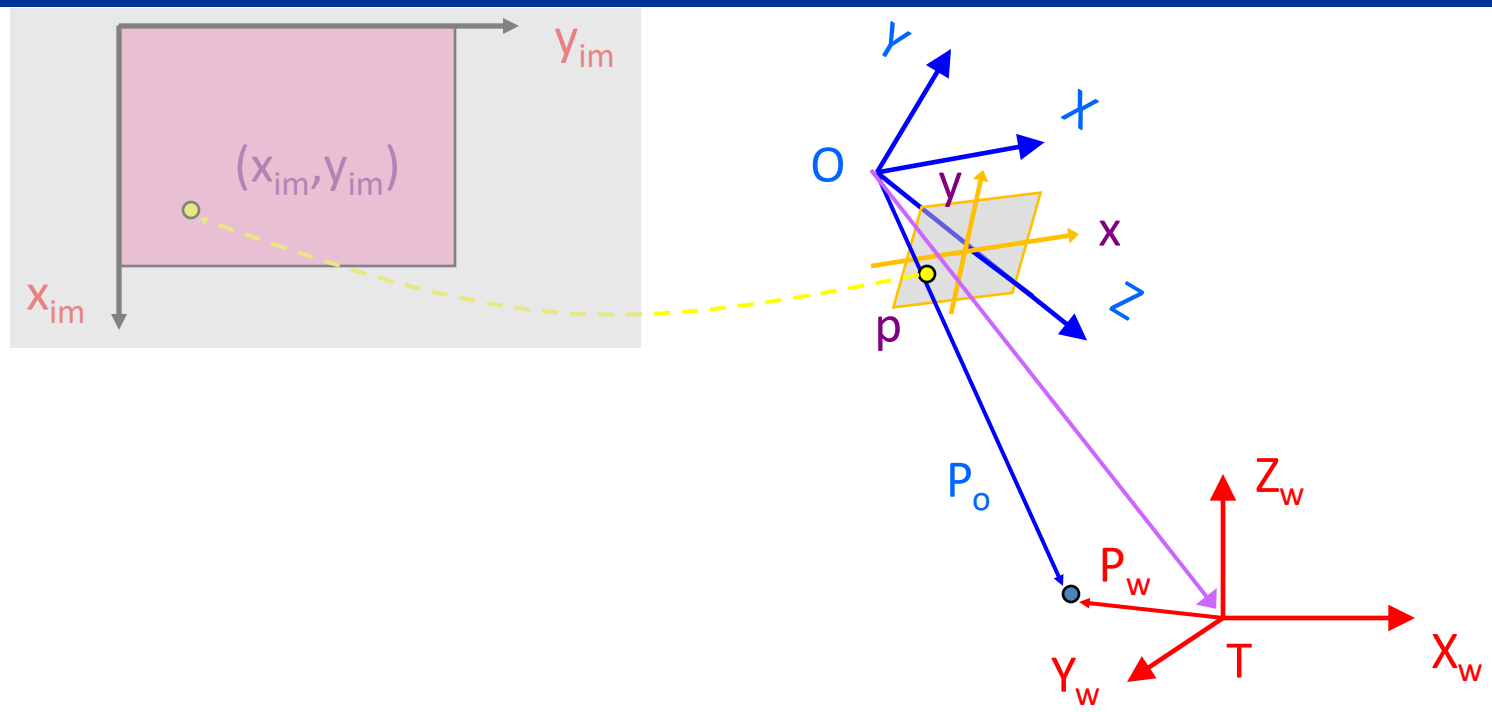


- Extrinsic Parameters
  - R: 3x3 rotation matrix that aligns the corresponding axes of the two systems
  - T: 3D translation vector (between origins of the coordinate systems)
- **From 3D World Reference System to 3D Camera Reference System:**

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} - T = \begin{bmatrix} R & -T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

General Camera Matrix

From World to Frame



- **Global transformation: World Reference Systems to pixel coordinates:**

$$\begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = K \begin{bmatrix} R & -T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

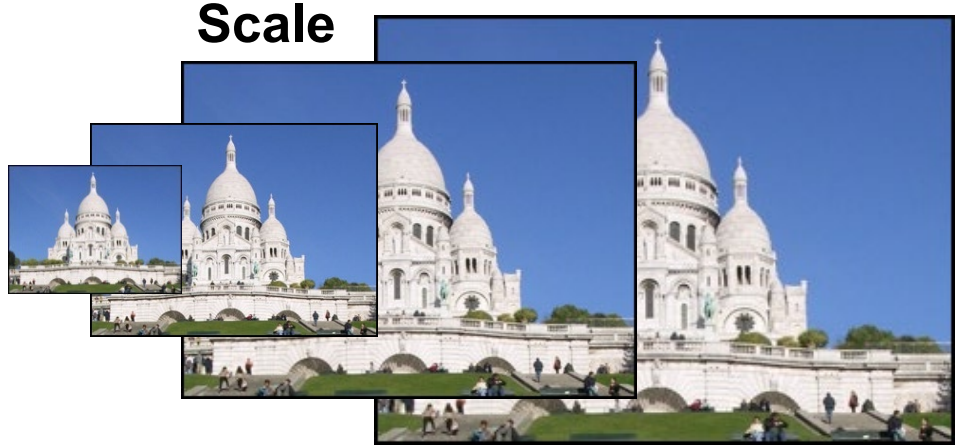
intrinsics extrinsics

Affine Transformations

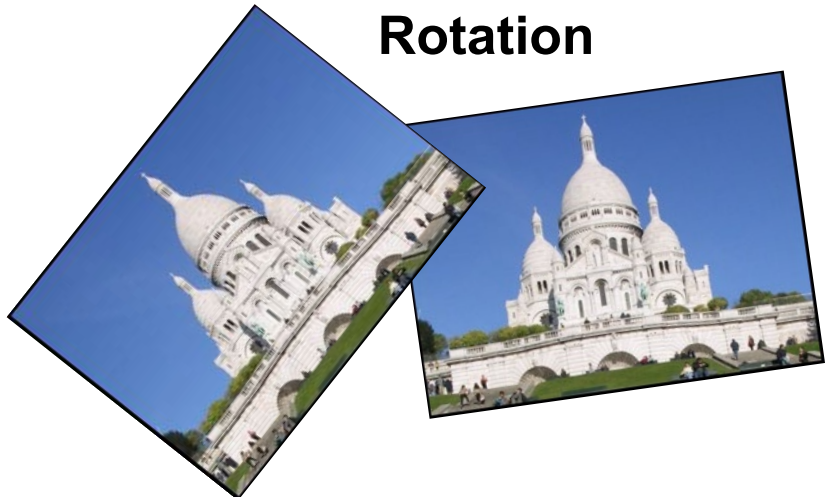
Affine transformation: preserve lines and parallelism, but not necessary angles and distances.

- Useful when distances to objects are much higher than their depth

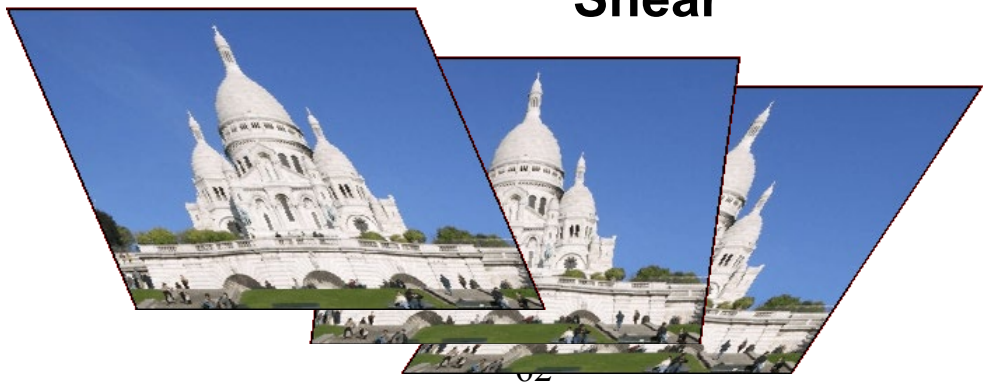
Translation



Scale



Rotation



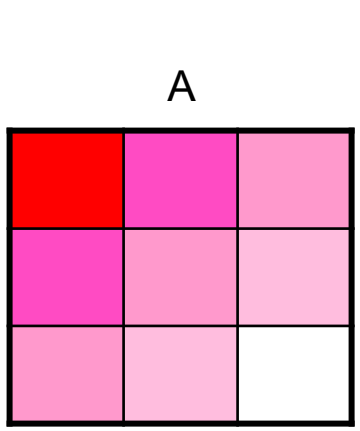
Shear

Affine Transformations

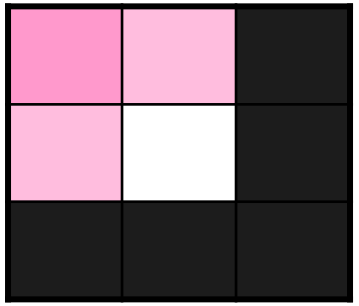
- We can define a generic affine transformation using homogeneous coordinates:

$$A(x,y) = R\left(\begin{bmatrix} c_{11z} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \right)$$

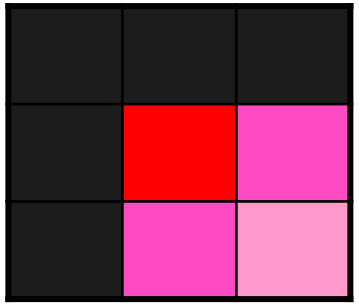
- Translation: $A(x,y) = R(x+t_x, y+t_y)$



Translation (-1, -1)
$R_1(x,y)= A(x+1,y+1)$



Translation (1, 1)
$R_2(x,y)= A(x-1,y-1)$



1	0	t_x
0	1	t_y
0	0	1

Affine Transformations

Scale transformation: $A(x,y) = R(x*s_x, y*s_y)$

- s_x, s_y greater than 1 → Image magnification → Apply bilinear or bicubic interpolation.
- s_x, s_y less than 1 → Image reduction → Apply supersampling.

A

R_1
$s_x = s_y = 0,8$

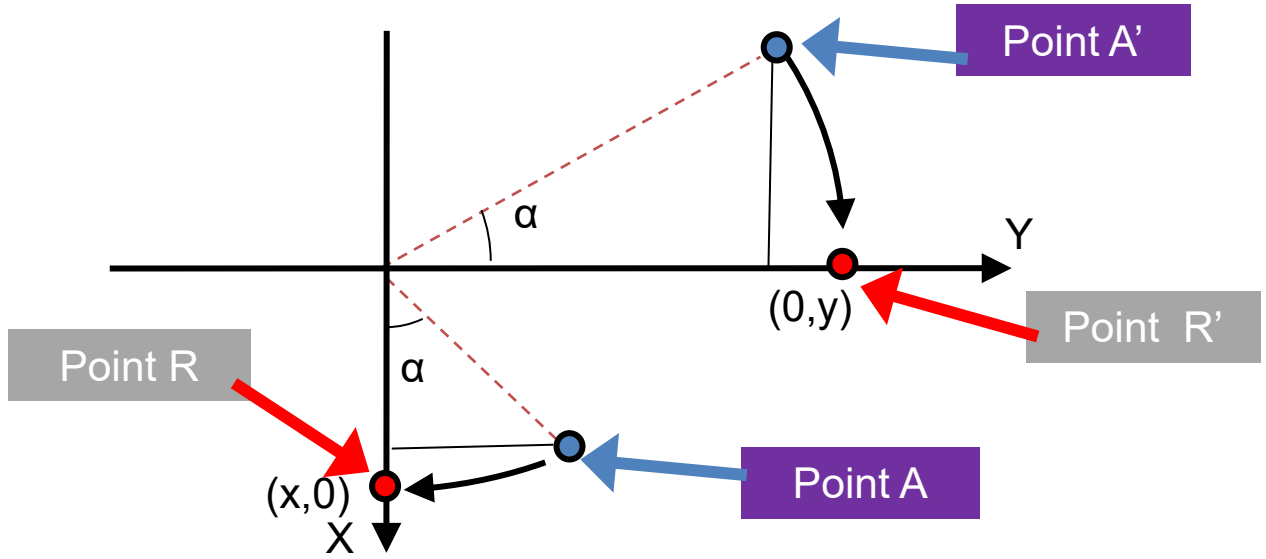
s_x	0	0
0	s_y	0
0	0	1

R_2
$s_x = 2 \quad s_y = 0,5$

Affine Transformations

- Rotation by angle α: A(x,y) → R(x·cos α - y·sin α, x·sin α + y·cos α)

cos α	-sin α	0
sin α	cos α	0
0	0	1



✖

Affine Transformation

- Derivation of rotation matrix

Expressing coordinates in the polar form:

$x = r \cos v$ --(1)

$y = r \sin v$ --(2)

$x' = r \cos (v + \theta)$

$y' = r \sin (v + \theta)$

Using trigonometric identities:

$x' = r \cos v.\cos \theta - r \sin v.\sin \theta$

$y' = r \sin v.\cos \theta + r \cos v.\sin \theta$

From(1) and (2) and for clockwise direction:

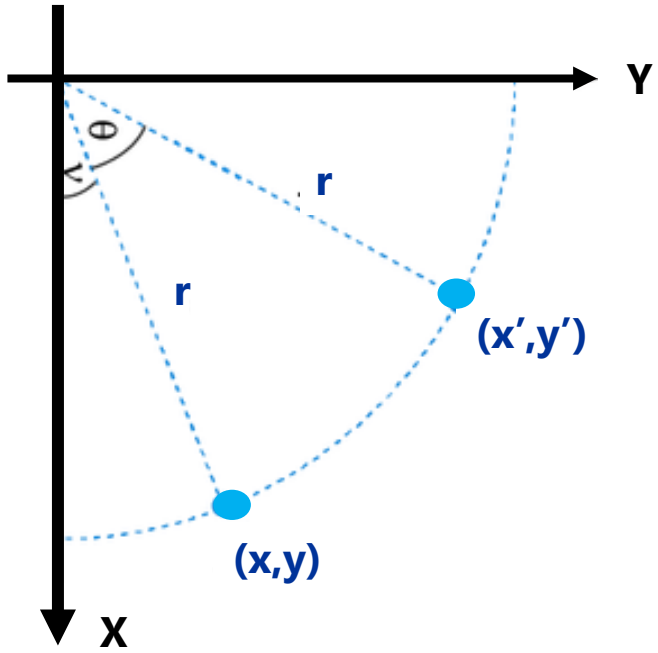
$x' = x \cos \theta - y \sin \theta$ --(3)

$y' = x \sin \theta + y \cos \theta$ --(4)

For counter clockwise direction:

$x = x' \cos \theta + y' \sin \theta$ --(5)

$y = -x' \sin \theta + y' \cos \theta$ --(6)

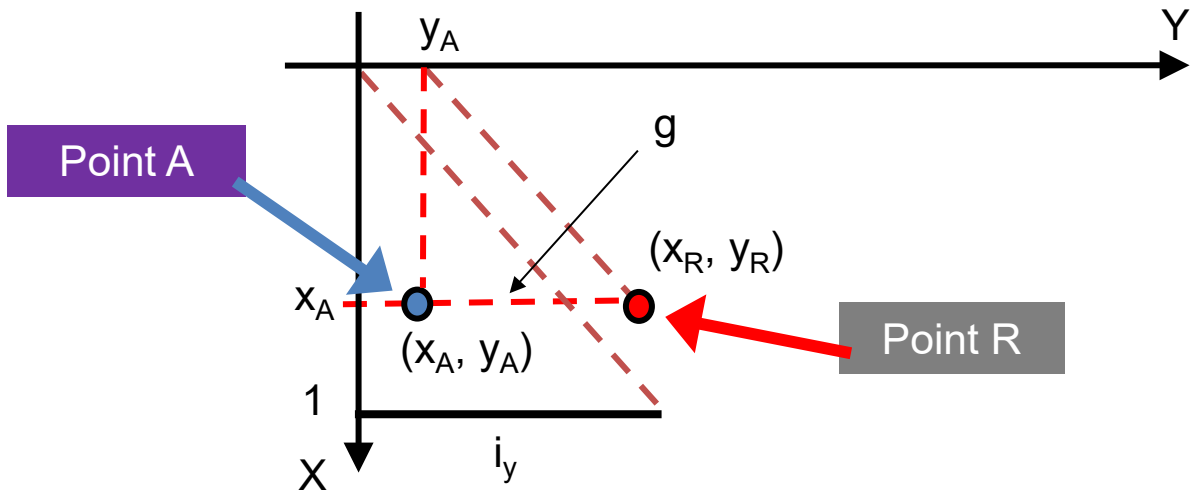


cos θ	-sin θ	0
sin θ	cos θ	0
0	0	1

cos θ	sin θ	0
-sin θ	cos θ	0
0	0	1

Affine Transformations

- Shear: slants the shape of an object
  - Example: shear in Y $i_{y:}$: $A(x, y) \rightarrow R(x, y + i_y \cdot x)$



  - $(x_A, y_A) = (x_R, y_R\text{-}g)$
  - $(1,0) = (1, i_y)$
  - By analogy of triangles: $g/x_R = i_y/1 \rightarrow g = i_y \cdot x_R$

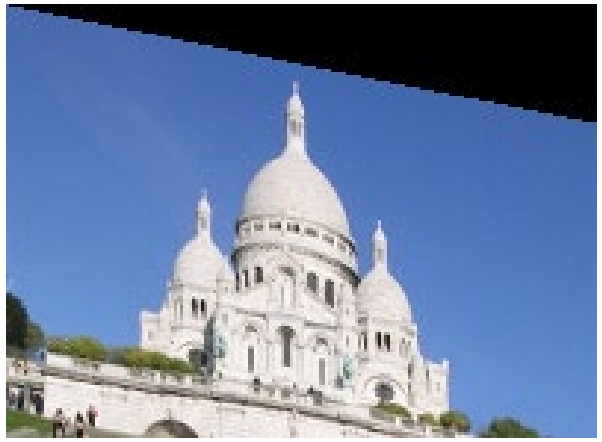
Affine Transformations

- Shear: slants the shape of an object
  - General transformation: $A(x, y)= R(x + i_x \cdot y, y + i_y \cdot x)$

1	i_x	0
i_y	1	0
0	0	1

$i_x=0$; $i_y=-0.4$

$i_x=0.2$; $i_y=0$



General Affine Transformation

- How to calculate the total affine transformation equivalent to two consecutive affine transformations?
  - Use matrix product by adding a row with (0, 0, 1).

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

General Affine Transformation

- Example: Equivalent transformation to apply
  - translation (a, b)
  - scale (c, d)
  - Rotation (θ)
  - translation (f, g).

1	0	f
0	1	g
0	0	1

x

cosθ	-sinθ	0
sinθ	cosθ	0
0	0	1

x

c	0	0
0	d	0
0	0	1

x

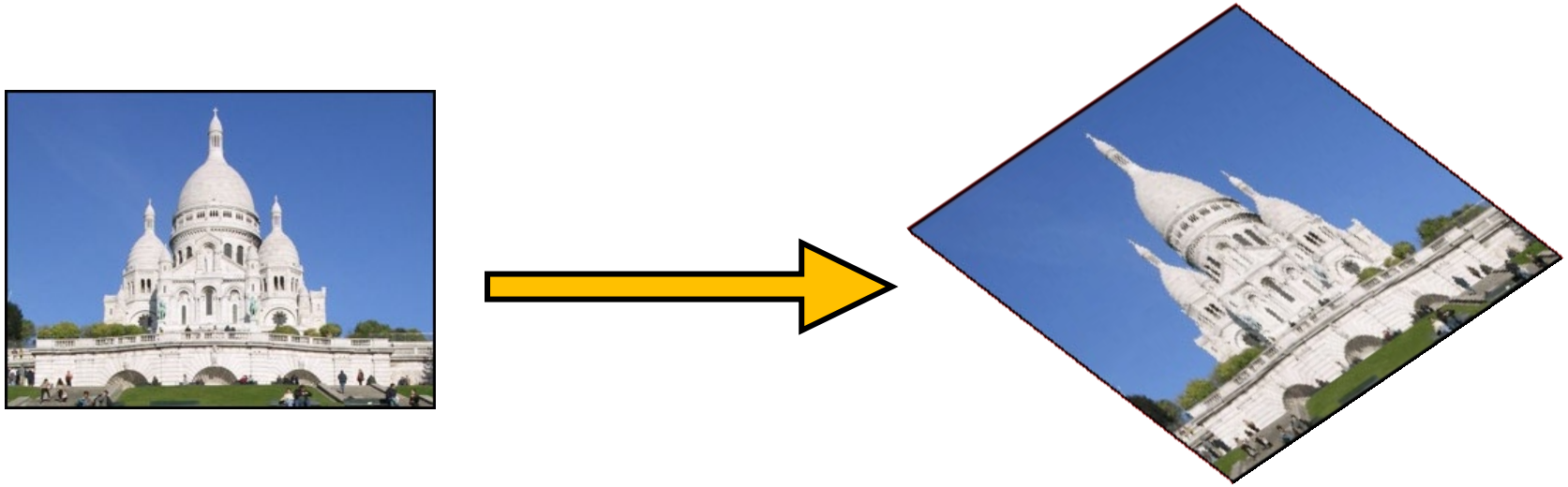
1	0	a
0	1	b
0	0	1

=

c·cosθ	-d·sin θ	a·c·cosθ +f -b·d·sinθ
c·sinθ	d·cos θ	b·d·cosθ +g+a·c·sinθ
0	0	1

General Affine Transformation

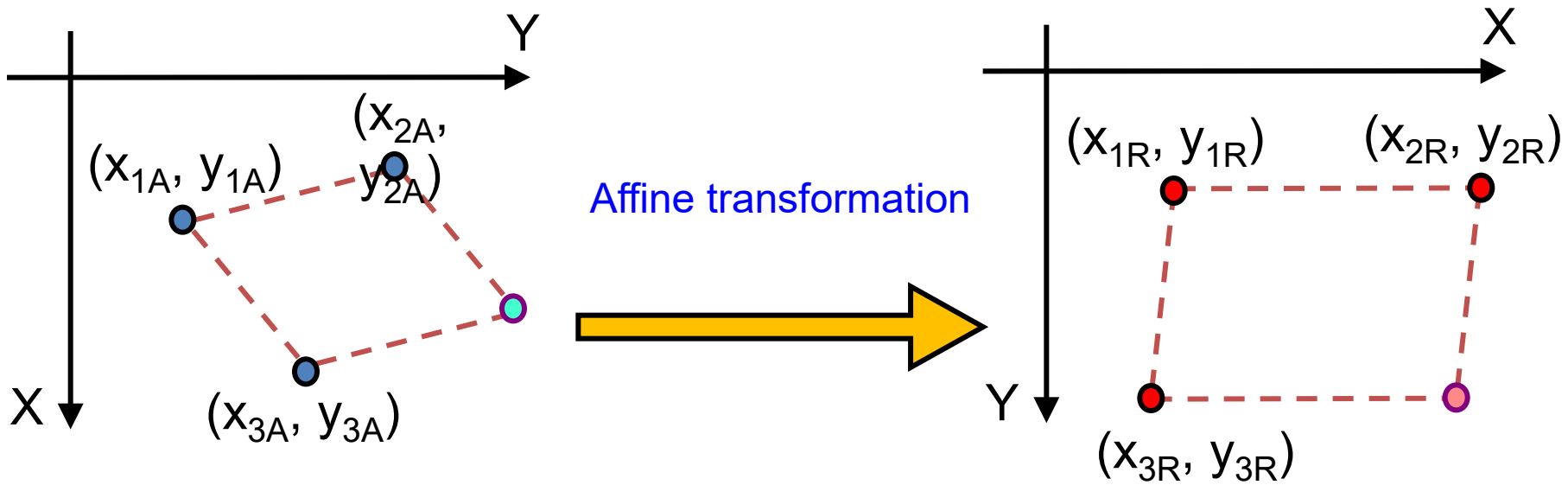
- An affine transformation allows you to "map" any rectangular region in any rhombus. Or, in general, any rhombus in another rhombus.



- How to compute the parameters of the transformation?

General Affine Transformation

- A rhombus is completely defined by 3 points.



- 6 unknowns (c_{ij}) → We need 6 equations to solve them.
- Each point produces two equations:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{iA} \\ y_{iA} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{iR} \\ y_{iR} \\ 1 \end{bmatrix}$$

General Affine Transformation

- Equations to solve:

$$c_{11}x_{1A} + c_{12}y_{1A} + c_{13} = x_{1R} \qquad c_{21}x_{1A} + c_{22}y_{1A} + c_{23} = y_{1R}$$
$$c_{11}x_{2A} + c_{12}y_{2A} + c_{13} = x_{2R} \qquad c_{21}x_{2A} + c_{22}y_{2A} + c_{23} = y_{2R}$$
$$c_{11}x_{3A} + c_{12}y_{3A} + c_{13} = x_{2R} \qquad c_{21}x_{3A} + c_{22}y_{3A} + c_{23} = y_{3R}$$

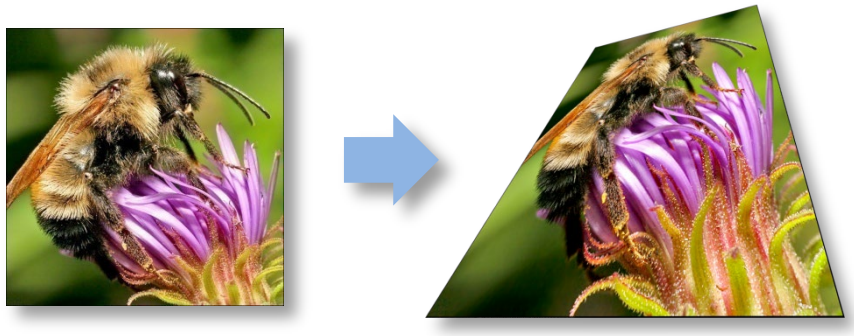
  - 6 equations and 6 unknowns.
  - There will be solution if the 3 points of A are not on the same line, and the 3 of R are not either.

Homography

- What happens when we change the last row?

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \longrightarrow \mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

- Projective Transformation, Homography or Planar Perspective Map
  - It is a transformation between two plane surfaces in two views of an scene
  - The planar surface is not assumed to be perpendicular to the optic axis
  - Possible to write the action of a perspective camera as a matrix



Homography

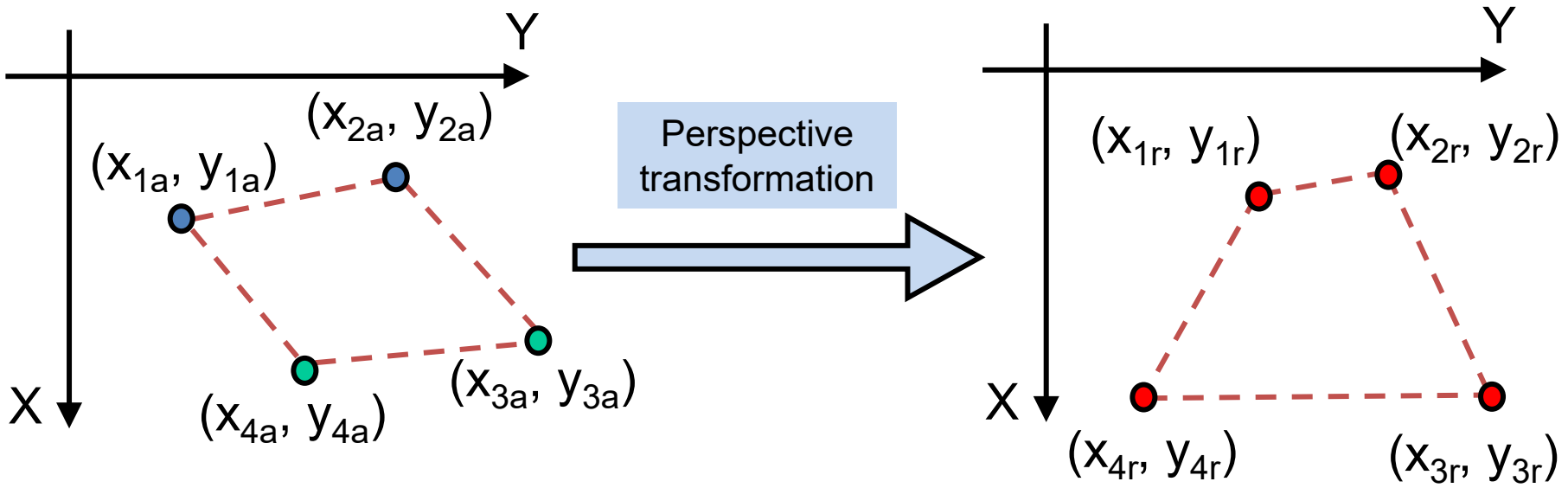
- Homography:

$$A(x,y) = R(x'/z', y'/z') \quad \text{with} \quad \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Problem: Given 4 coplanar points in the view from a camera and another 4 in the view from a second camera, calculate the perspective transformation that produces the map.
  - Solution: Set the corresponding system of equations and solve the unknowns.

Homography

- Homography



Homography

- Perspective transformation: 9 unknowns, 4 points… ?
  - Each pair of points (x_{ia}, y_{ia}), (x_{ir}, y_{ir}) gives two equations:

    $(c_{11}x_{ia}+c_{12}y_{ia}+c_{13})/(c_{31}x_{ia}+c_{32}y_{ia}+c_{33}) = x_{ir}$

    $(c_{21}x_{ia}+c_{22}y_{ia}+c_{23})/(c_{31}x_{ia}+c_{32}y_{ia}+c_{33}) = y_{ir}$

  - Homogeneous and indeterminate system (8 eqs., 9 unks.).
    - Observe that a scale factor appears. If we multiply all the coefficients by k the system does not change.
    - It can be solved by "fixing" the unknown c33 = 1. (with 4 points in a plane we can not recover full 3D information: different combinations of camera parameters could give the same view)

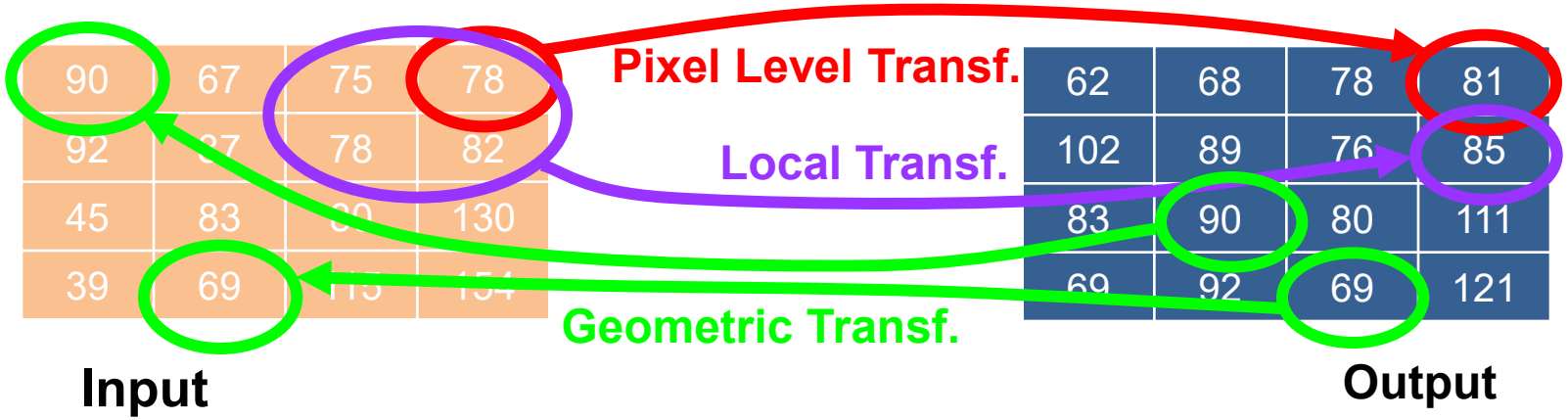
- Inverse transformations:
  - A perspective transformation defined by H a (3x3) matrix; the inverse transformation is given by H^{-1}.

Interpolation

- Geometric transformation:

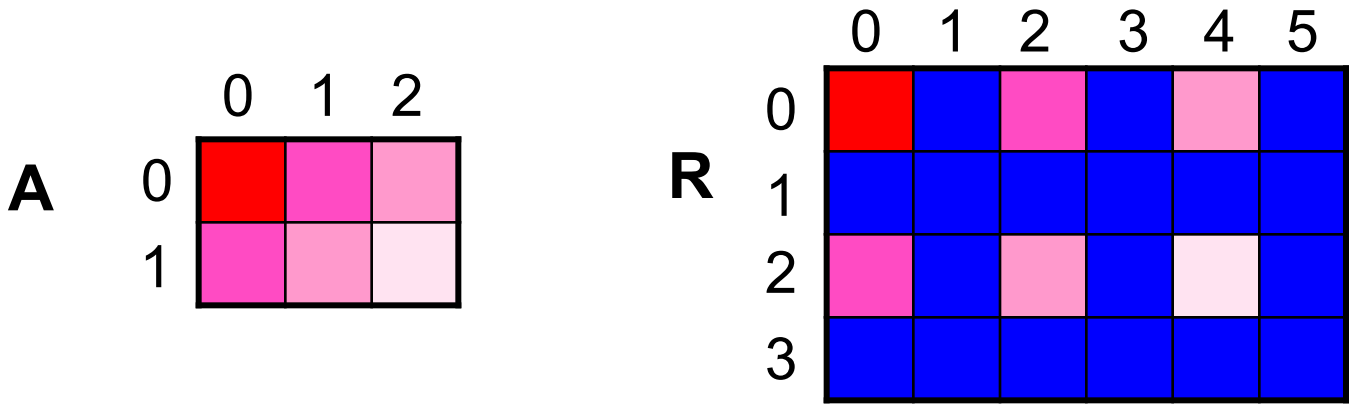
  $R(x, y) = A(f_1(x, y), f_2(x, y))$

  - The value of each pixel in the output depends on anothers in the input whose position are calculated through a pair of functions f_1 and f_2.
  - The output image size may differ from the input image size.



Interpolation

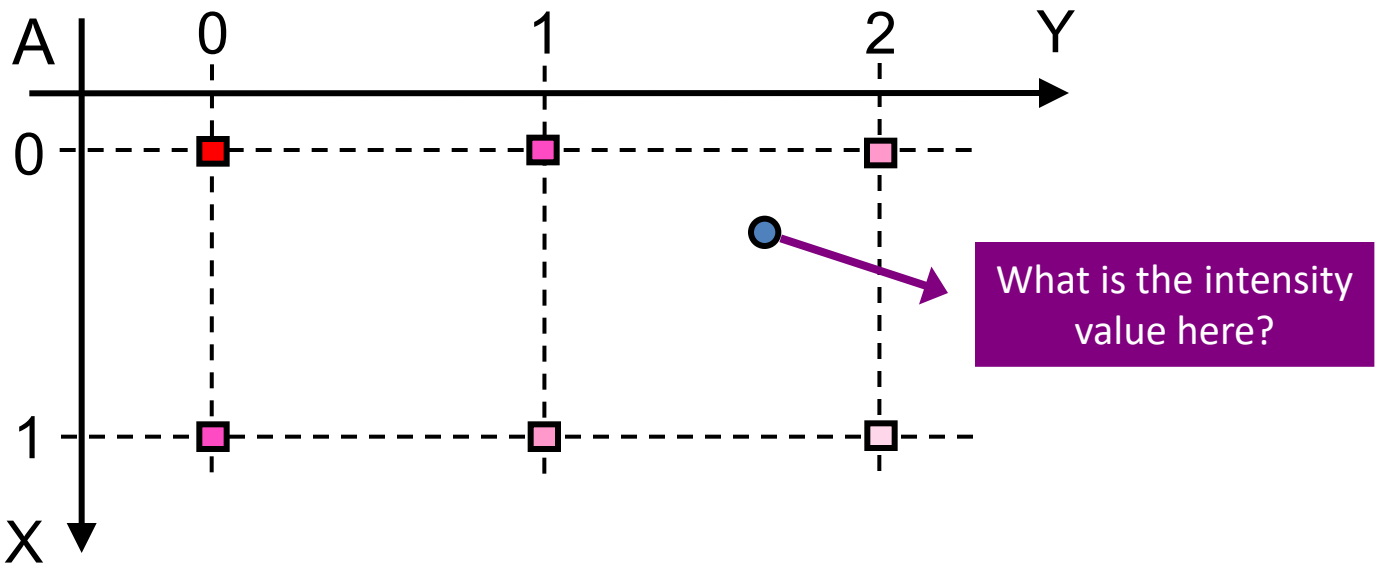
- General transformation: $R(x, y) = A(f_1(x,y), f_2(x,y))$
  - $f_1, f_2: N \times N \rightarrow R$
    - f_1: x-coordinate in the input image of the output pixel (x,y)
    - f_2: y-coordinate in the input image of the ouput pixel (x,y)
  - What if the result is not an integer?
    - Example: 2x magnification with a transformation R (x, y) = A (x / 2, y / 2)



- R(0, 0)= A(0/2, 0/2) = A(0, 0) **OK**
- R(1, 0)= A(1/2, 0/2) = A(0.5, 0) ??
- R(1, 1)= A(1/2, 1/2) = A(0.5, 0.5) ??

Interpolation

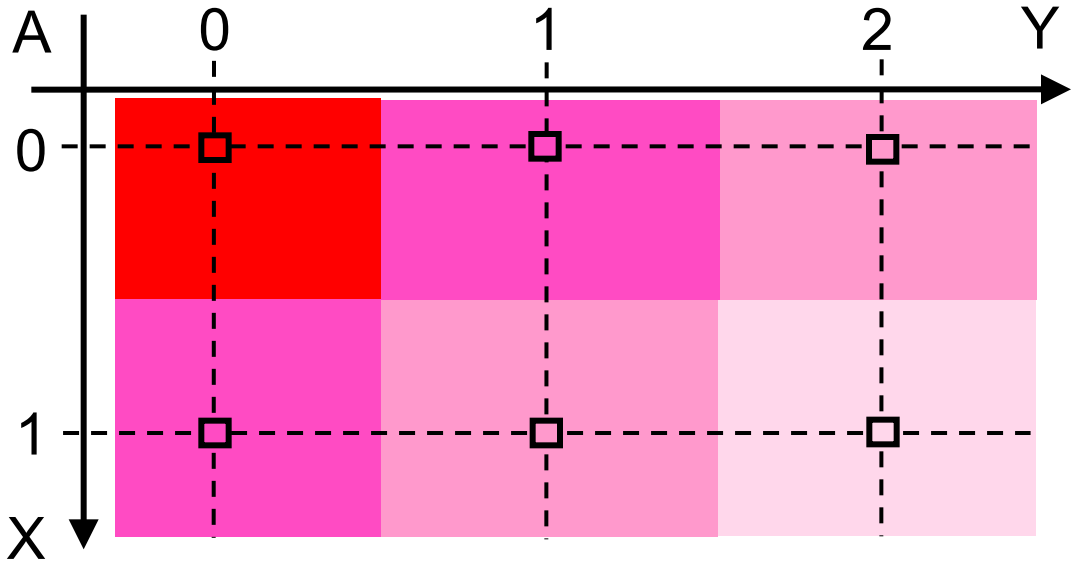
- Problem: Images are discrete signals, but geometric transformation treats them as continuous.



- Solution: Apply an interpolation.
- Types of interpolation: nearest neighbor, bilinear, bicubic, ...

Interpolation

- Nearest Neighbor Interpolation:
  - Each point in space takes the value of the pixel closest to it.



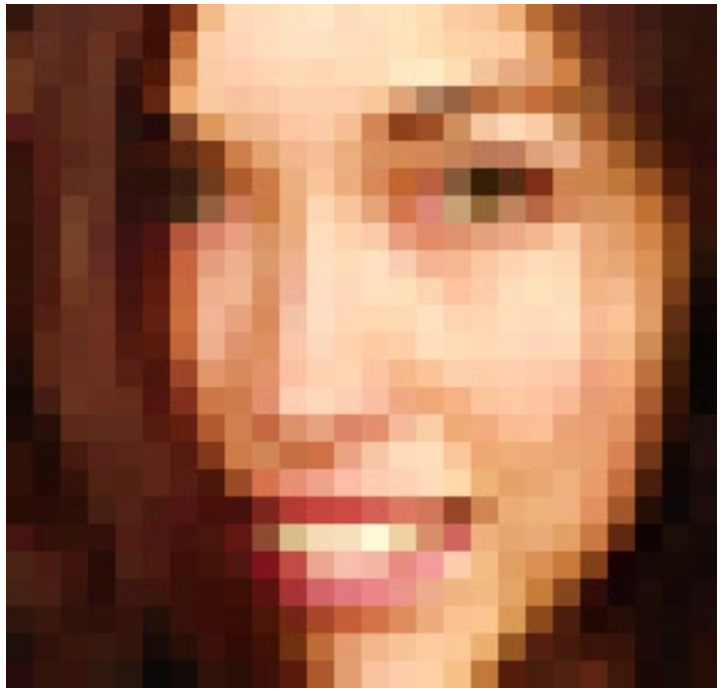
$$\left.\begin{array}{l} f_1(x,y) \rightarrow \lfloor f_1(x,y) + 0{,}5 \rfloor \\ f_2(x,y) \rightarrow \lfloor f_2(x,y) + 0{,}5 \rfloor \end{array}\right\} R(x,y) = A(\lfloor f_1(x,y) + 0{,}5 \rfloor, \lfloor f_2(x,y) + 0{,}5 \rfloor)$$

Interpolation

Nearest Neighbor

- Example: 10x zoom with nearest neighbor:
  - $R(x,y) = A(\lfloor x/10 + 0.5 \rfloor, \lfloor y/10 + 0.5 \rfloor)$

Input 26x25



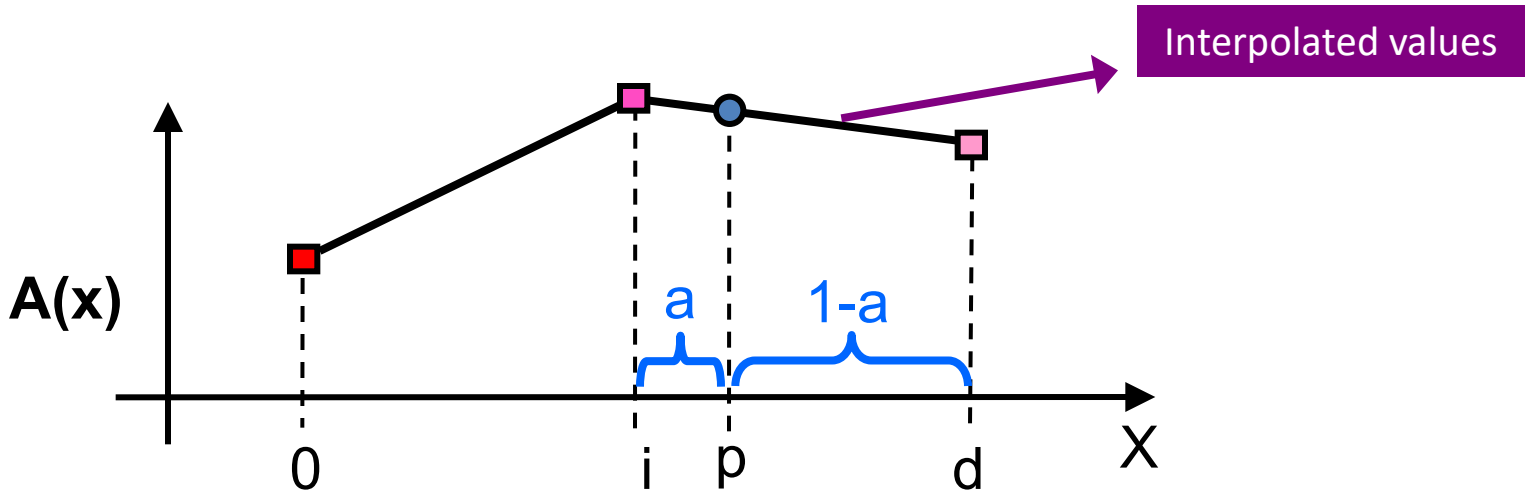
Output
260x250

  - Advantages: simple and fast.
  - Disadvantages: grid effect, low quality images.

Interpolation

Linear

- Linear Interpolation (1D)
  - Draw a straight line between each pair of consecutive points.



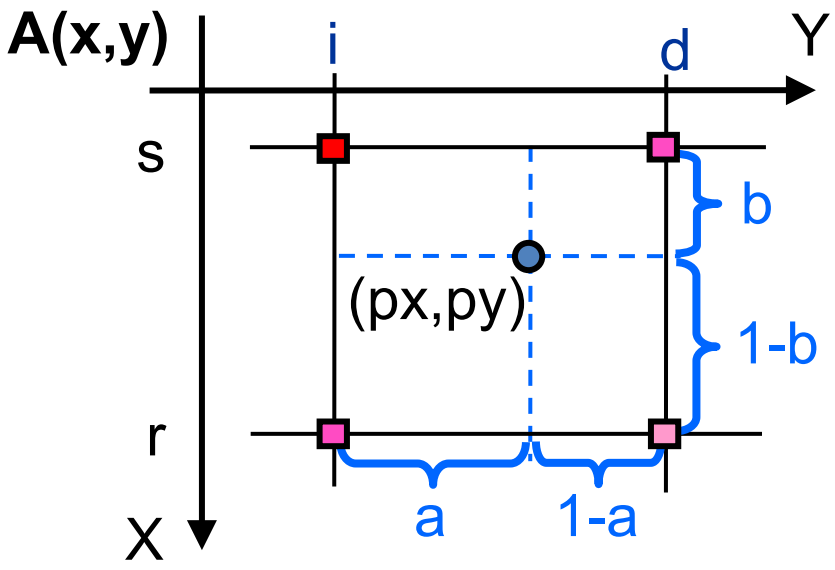
  - Let p be the point to interpolate, which lies between i and d:
    i= ⌊p⌋, **d**= i+1

  - The interpolated value at p will be:
    A (p) = (1-a) A (i) + aA (d); a = p-i

Interpolation

Bilinear

- Bilinear interpolation (2D): consists of applying two linear interpolations
  - Consider the point (px, py), where
    - **i**= ⌊px⌋, **d**= i+1, **s**= ⌊py⌋, **r**= s+1
    - **a**= px-i, **b**= py-s



  - Interpolated values:
    - A(s, py) = (1-a) A(s, i) + a A(s, d)
    - A(r, py) = (1-a) A(r, i) + a A(r, d)
    - A(px, py) = (1-b) A(s, py) + b A(r, py)

    - Weighted average of the 4 surrounding pixels (nearest integer coordinates):

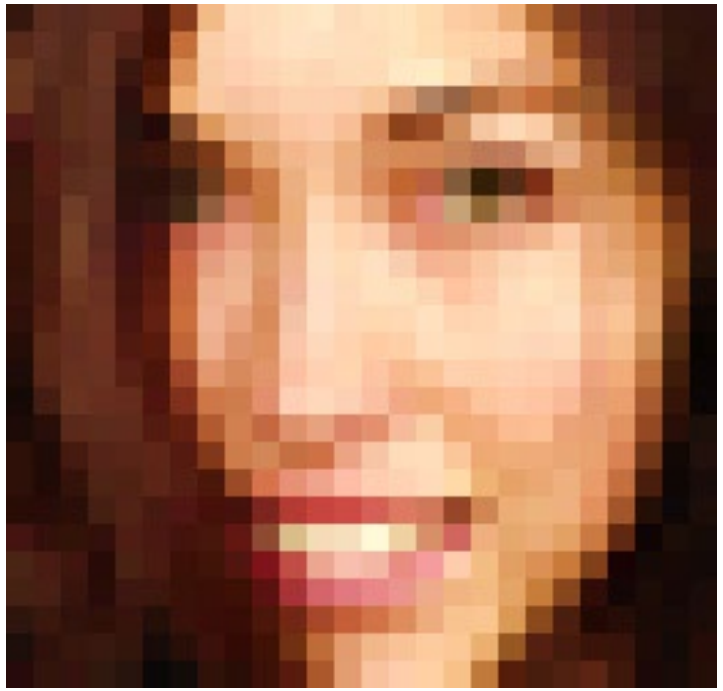
A(px, py) = (1-a)(1-b)A(s, i) + a(1-b)A(s, d) + (1-a)bA(r, i) + abA(r, d)

Interpolation

Nearest Neighbor vs. Bilinear

- Example: 10x zoom

Input



Nearest Neighbor

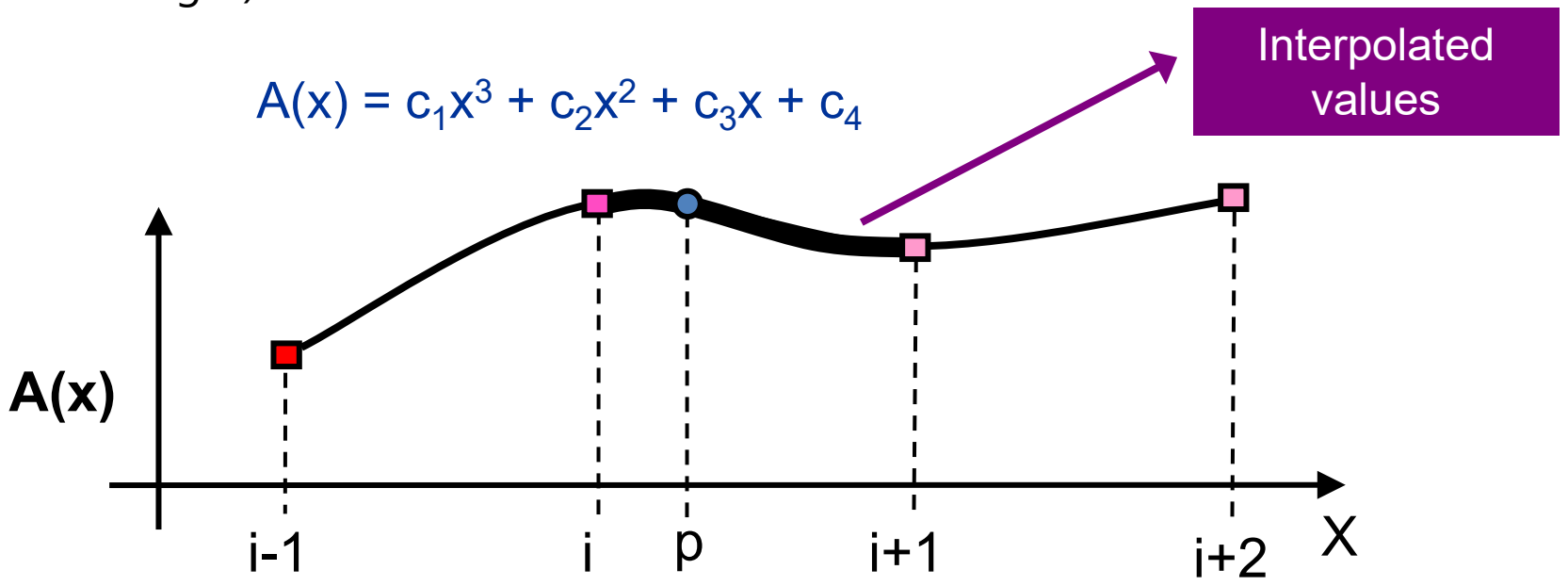
Bilinear interpolation

- Bilinear interpolation is better but produces a “rectangular zone” effect.

Interpolation

Cubic

- Cubic interpolation: In 1D the cubic interpolation consists of tracing a cubic curve between the 4 nearest points (2 to the left and 2 to the right).



  - Let p be the point to interpolate and **i**= ⌊p⌋
  - Given the values A (i-1), A (i), A (i+1), A (i+2)
    - Obtain the 4 equations, with 4 unknowns
    - Solve and obtain c1, c2, c3, c4
    - Apply the constants, and get A (p)

Interpolation

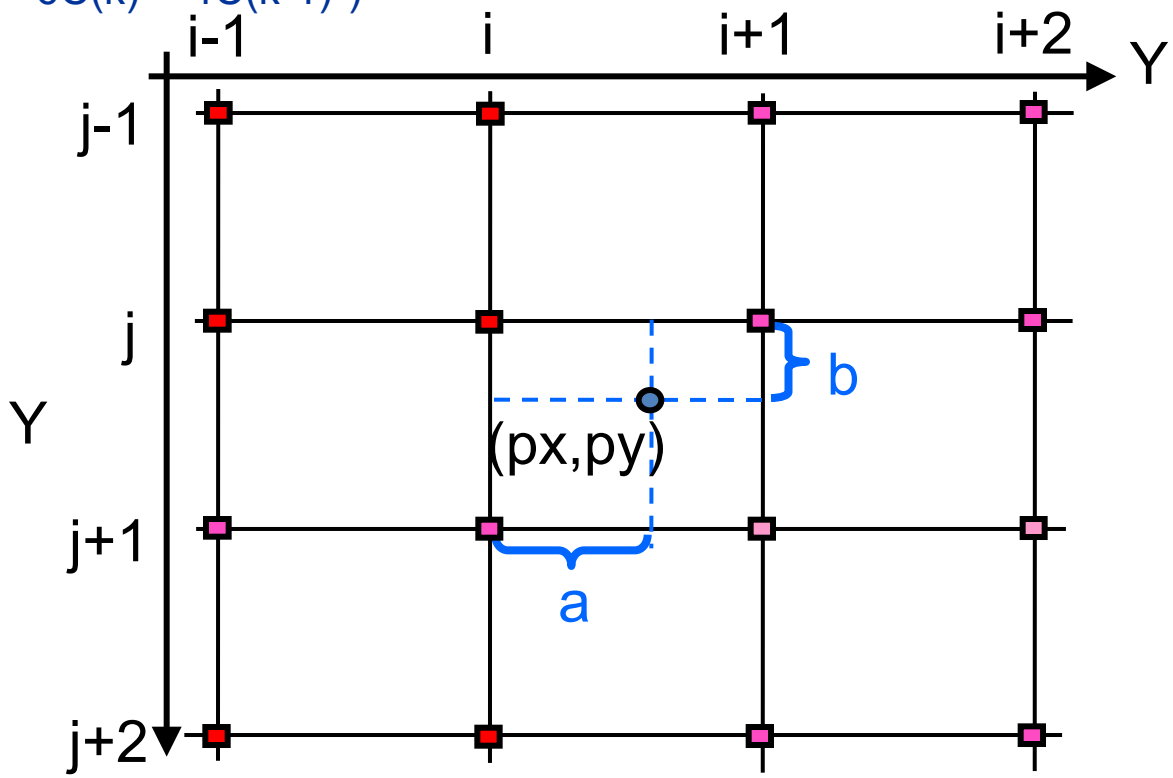
Bicubic

- Bicubic interpolation:
  - Horizontal cubic interpolation, in the original rows.
  - Vertical cubic interpolation using the above interpolation.
  - In the bicubic interpolation the 16 surrounding points are involved

$$A(px,py) = \Sigma_{n=-1..2}\Sigma_{m=-1..2} A(i+n,j+m)\cdot P(n-a)\cdot P(b-m)$$

$$P(k) = 1/6(C(k+2)^3 - 4C(k+1)^3+6C(k)^3 - 4C(k-1)^3)$$

$$C(k) = \max(0, k)$$



Interpolation

Bilinear vs. Bicubic

- Example: 10x zoom

Input

Bilinear interpolation

Bicubic interpolation

Zoom out

- What happens with the zoom out (down-scale)?
  - Example: zoom out 5x.

input

ouput: nearest neighbor

Zoom out

- The problem is not solved with bilinear or bicubic interpolation:
  - Details are smaller than the output resolution.

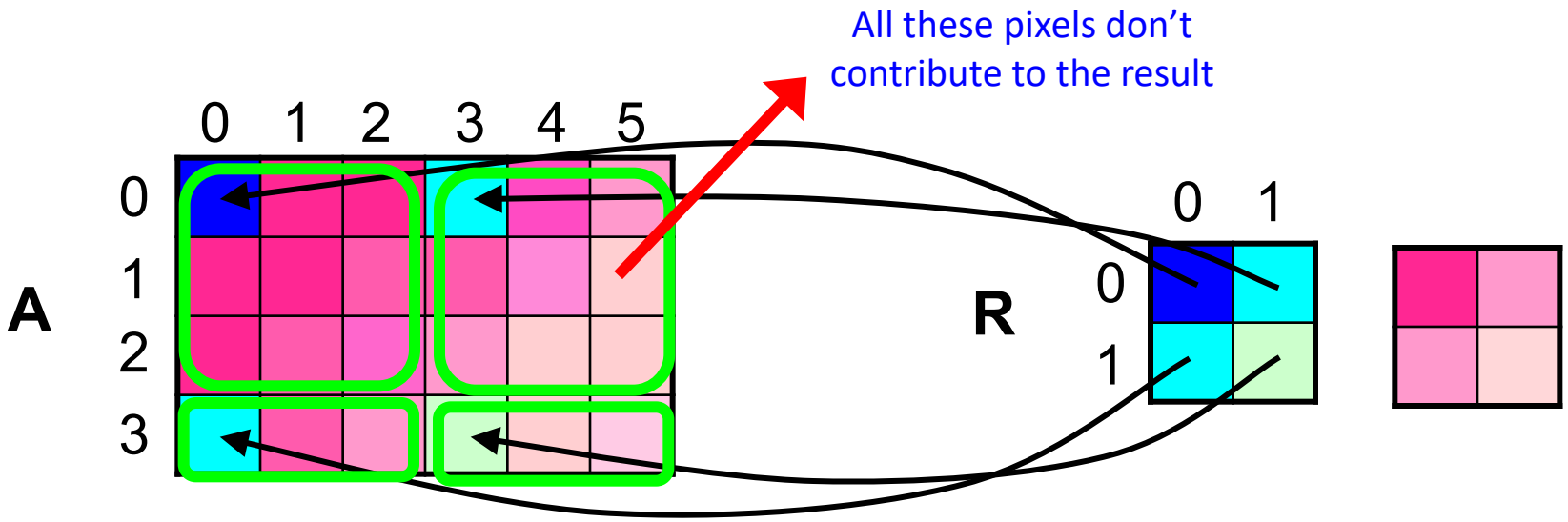
input

ouput: bilinear interpolation

ouput: bicubic interpolation

Zoom out

- Example: zoom out 3x. R(x,y) = A(3x, 3y)

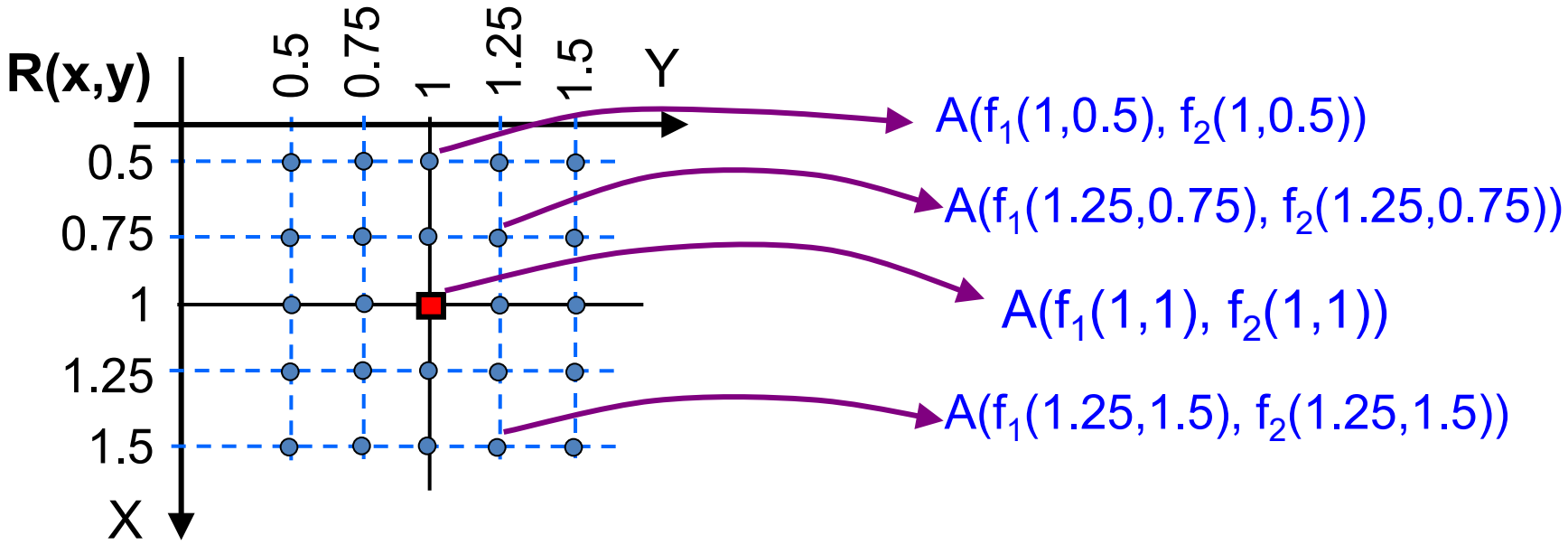


- Solution: each output pixel should be the average of the corresponding 3x3 input pixels.
  - Super sampling Interpolation: Consider the pixel as a region with a certain area.

Zoom out

Supersampling

- Uniform supersampling:



$R(x,y)= \text{mean } \{A(f_1(x-0.5,y-0.5), f_2(x-0.5,y-0.5)), \ldots$

$A(f_1(x-0.5,y+0.5), f_2(x-0.5,y+0.5)), \ldots, A(f_1(x,y-0.5), f_2(x,y-0.5)), \ldots,$
$A(f_1(x,y+0.5), f_2(x,y+0.5)), \ldots, A(f_1(x+0.5,y-0.5), f_2(x+0.5,y-0.5)), \ldots,$
$A(f_1(x+0.5,y+0.5), f_2(x+0.5,y+0.5))\}$

Zoom out

- Example: zoom out 5x

input

ouput: bilinear interpolation

ouput: uniform supersampling

- Supersampling achieves a much higher quality result.
- However, supersampling is much more expensive.

Zoom out

- Conclusions
  - Geometric Transformation: Each output pixel matches an input pixel whose position is calculated according to a pair of functions.
  - Positions can be non-integers: nearest neighbor interpolations, bilinear, bicubic, and so on.
  - Bicubic interpolation works better in zoom and the supersampling in zoom out. But they are more expensive than others.