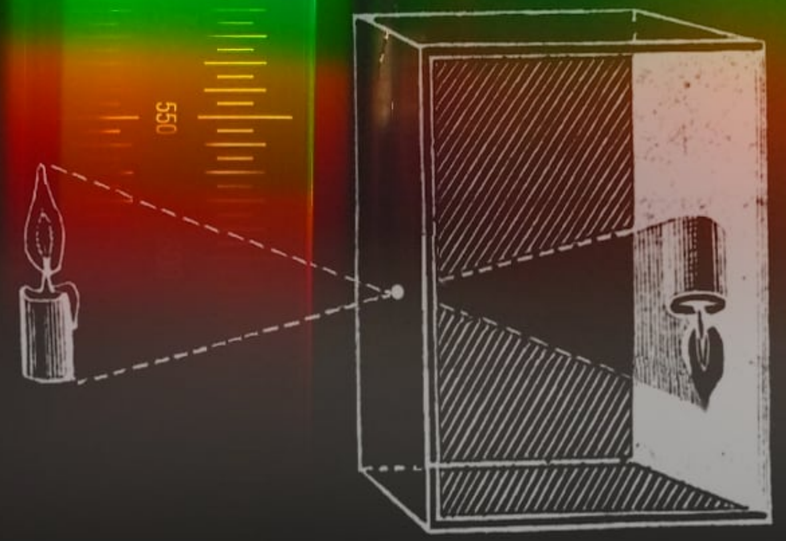
ADVANCES IN COMPUTER VISION

MODULE 1: Geometry

IMAGE FORMATION



Prof. Vincent Sitzmann

Prof. Fredo Durand

Some slides adapted from

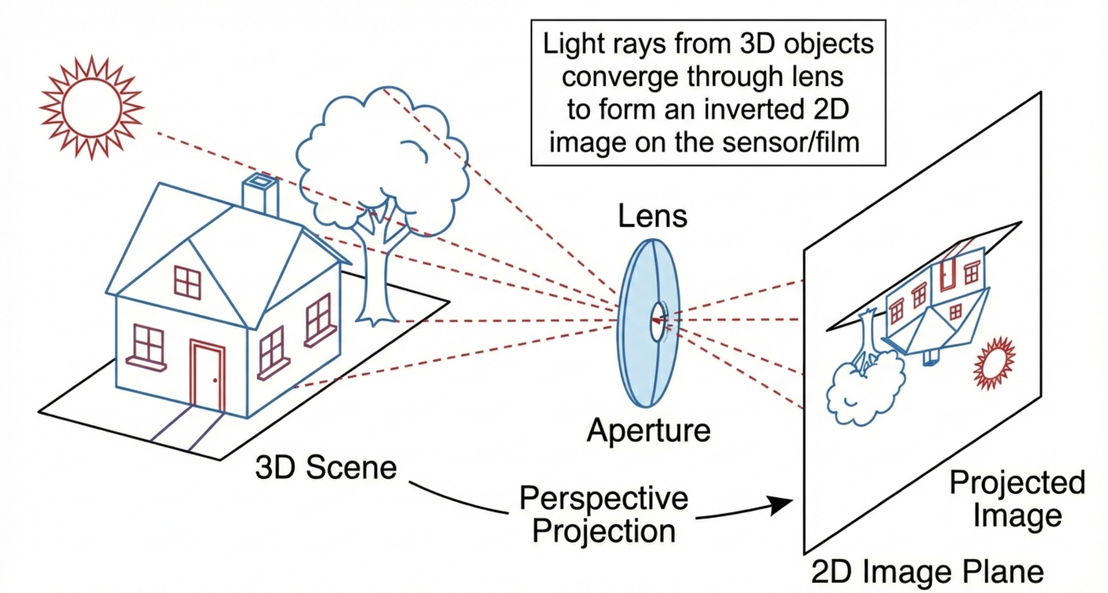
CMU 16-889: Learning for 3D Vision
Prof. Shubham Tulsiani

CMU 16-385: Computer Vision
Prof. Kris Kitani

MIT 6.819/6.869: Advances in Computer Vision,
Profs. Bill Freeman, Phillip Isola, Antonio Torralba

Philosophical discussion

- Ingredients of Vision include 2D and 3D image
- The connection between the two (image formation, perspective) is important
- So far still mostly hand-coded
  - Inductive bias
- Who knows if it will eventually be learned
  - e.g. transformers to learn mapping from 3D triangles to image https://microsoft.github.io/renderformer/



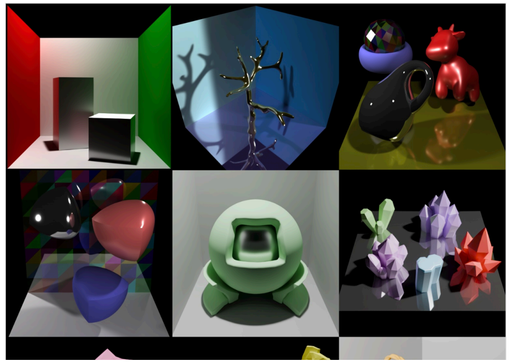
RenderFormer

Transformer-based Neural Rendering of Triangle Meshes with Global Illumination

SIGGRAPH 2025

Chong Zeng[1,2] Yue Dong[2] Pieter Peers[3] Hongzhi Wu[1] Xin Tong[2]

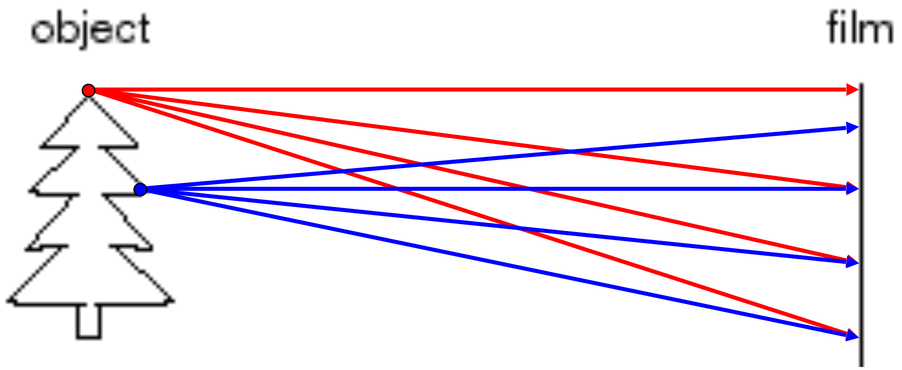
[1]State Key Lab of CAD and CG, Zhejiang University
[2]Microsoft Research Asia
[3]College of William & Mary



Let’s make a Camera!

Slide credit: Steve Seitz

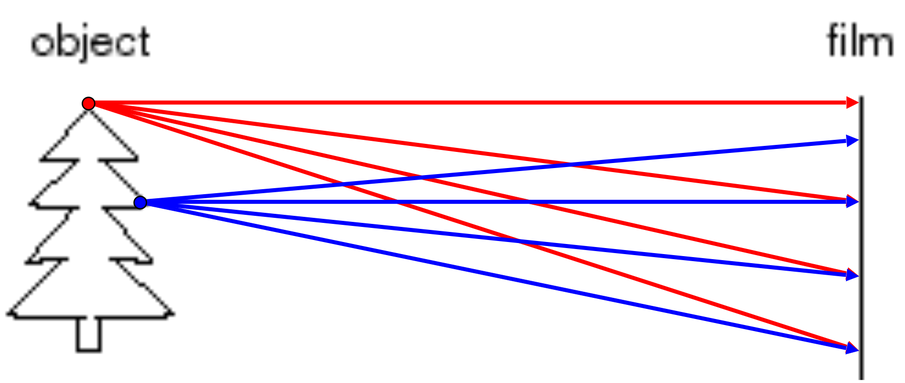
Let’s make a Camera!



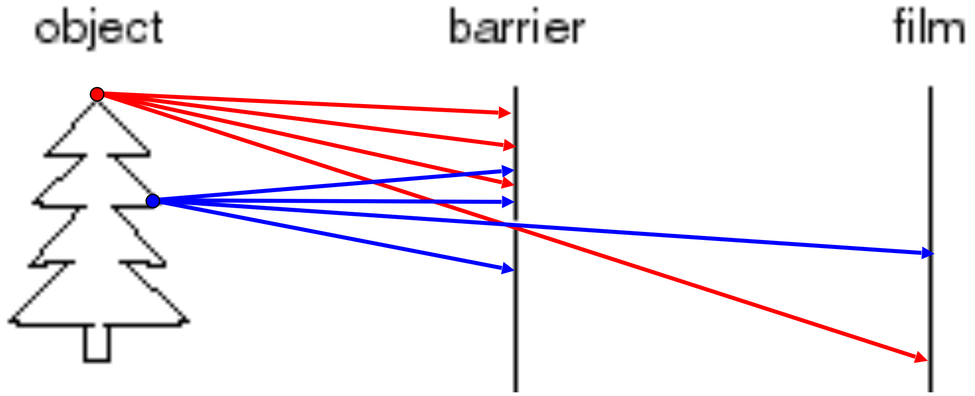
Idea 1: Put piece of film in front of an object

Slide credit: Steve Seitz

Let’s make a Camera!



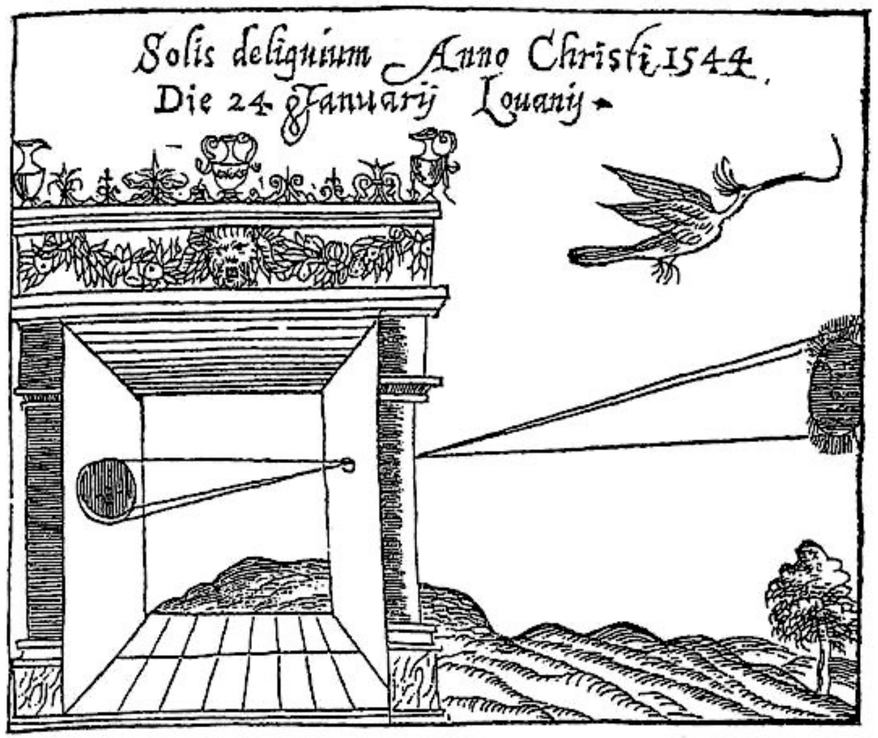
Idea 1: Put piece of film in front of an object



Add a barrier to block most rays.

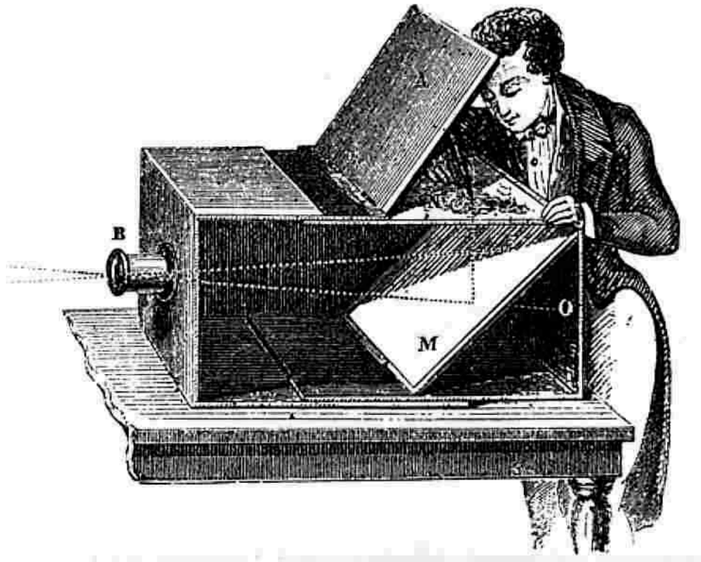
Slide credit: Steve Seitz

Camera Obscura



Gemma Frisius, 1558

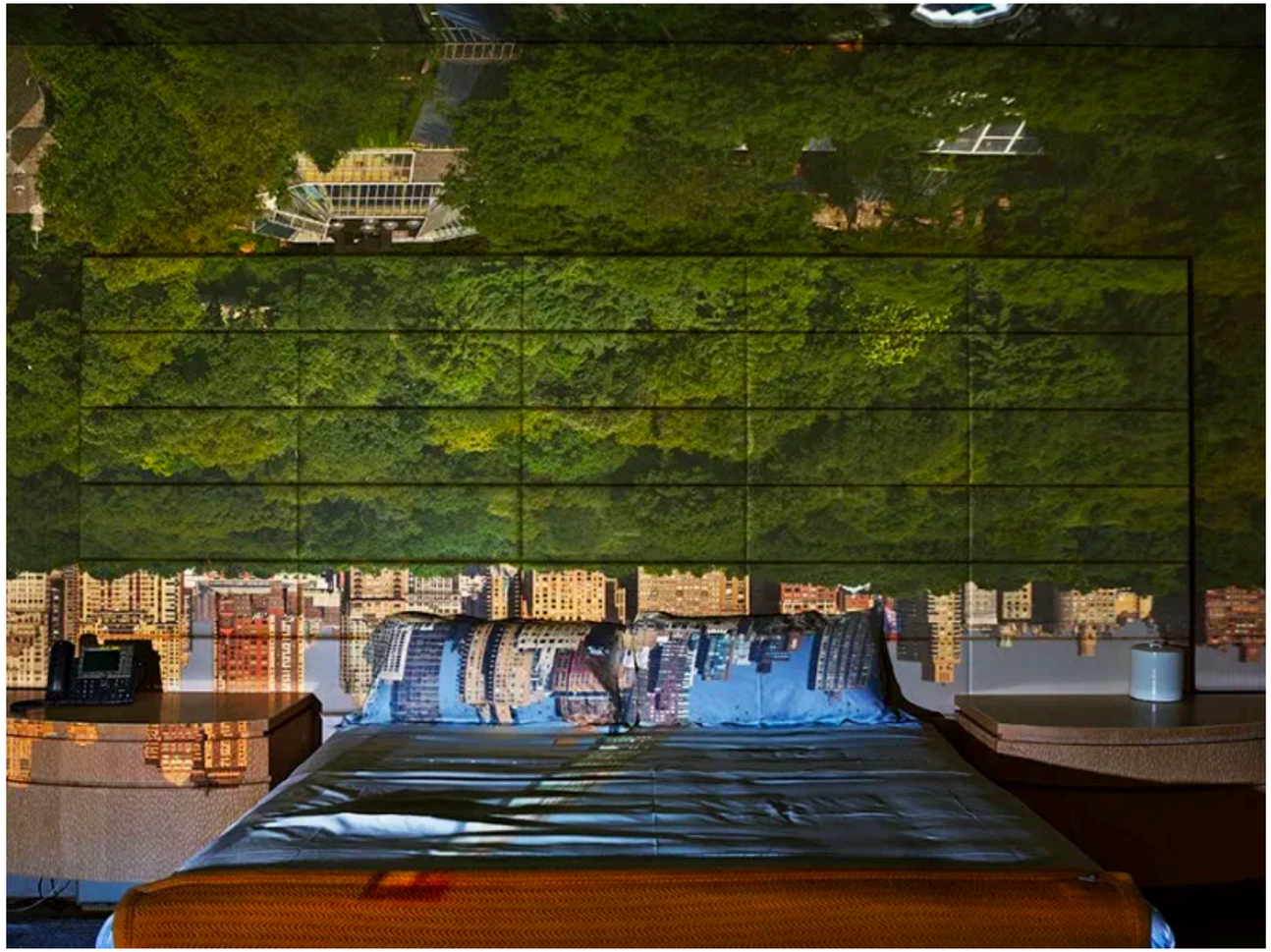
- Basic principle known to Mozi (470-390 BCE), Aristotle (384-322 BCE)
- Drawing aid for artists: described by Leonardo da Vinci (1452-1519)



Slide credit: Lana Lazebnik

Image source

Camera Obscura (*Dark Room*)



Camera Obscura: View of Central Park Looking West in Bedroom. Summer, 2018

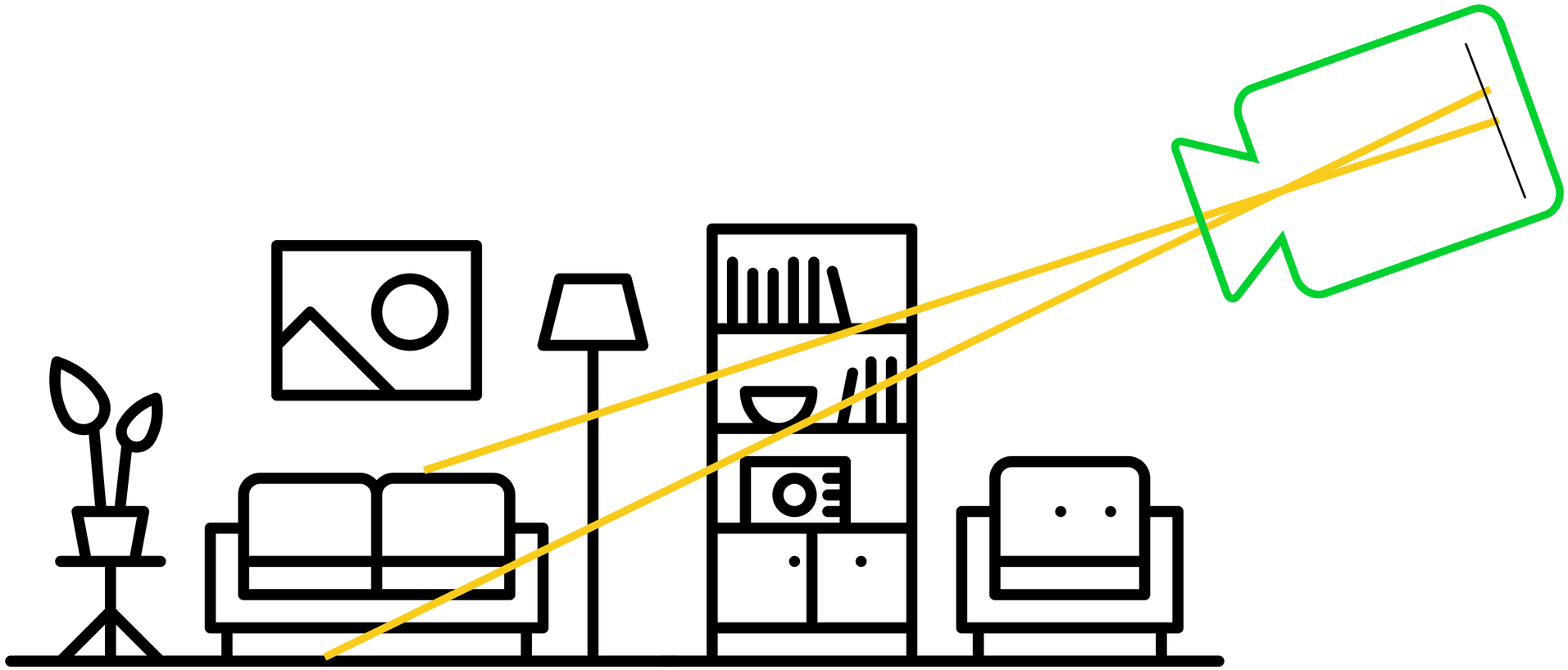
After scouting rooms and reserving one for at least a day, Morell masks the windows except for the aperture. He controls three elements: the size of the hole, with a smaller one yielding a sharper but dimmer image; the length of the exposure, usually eight hours; and the distance from the hole to the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

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

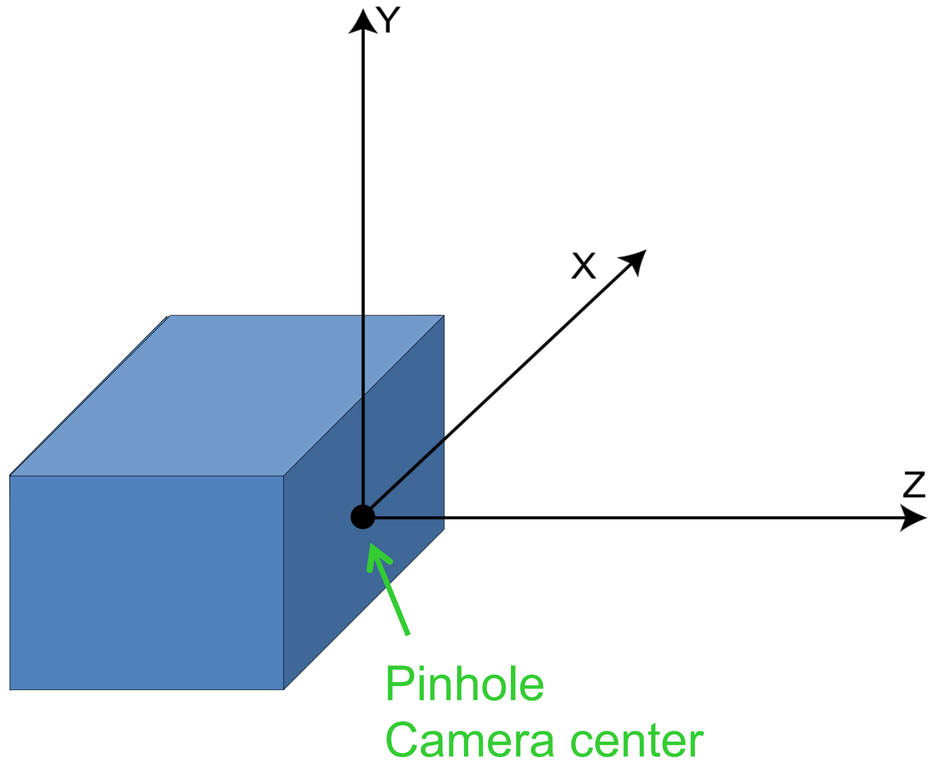
http://www.abelardomorell.net/project/camera-obscura/

Slide credit: Lana Lazebnik

Today: Model the camera, describe the rays.

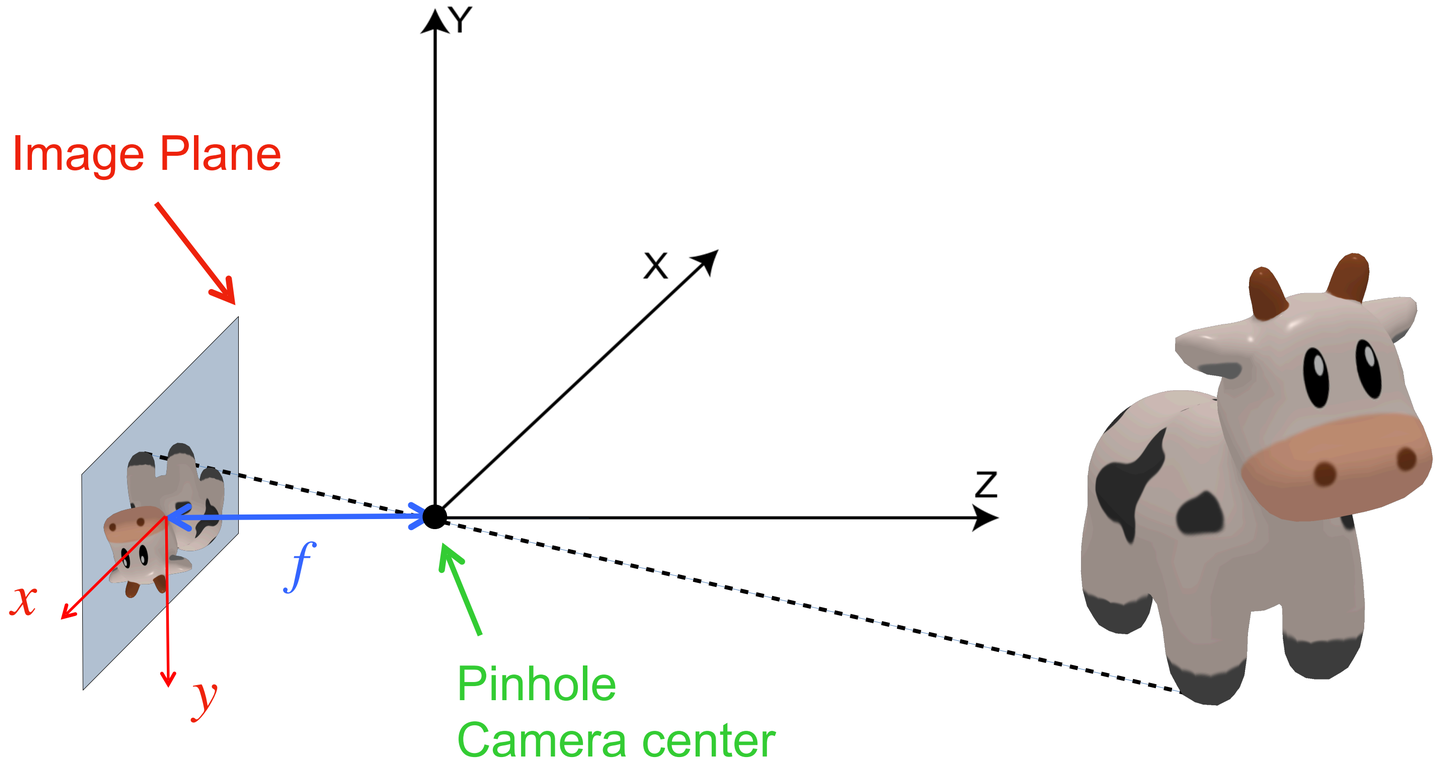


Perspective projection

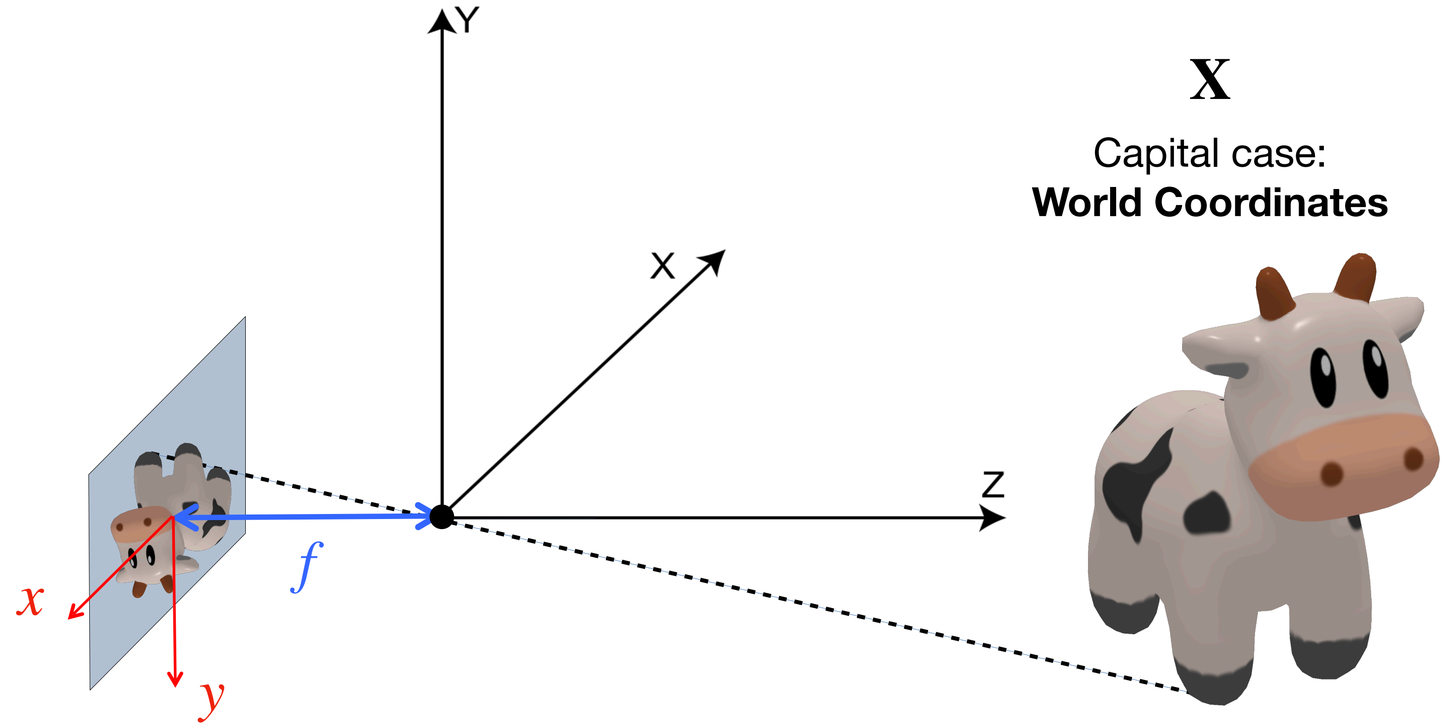




Perspective projection



Perspective projection



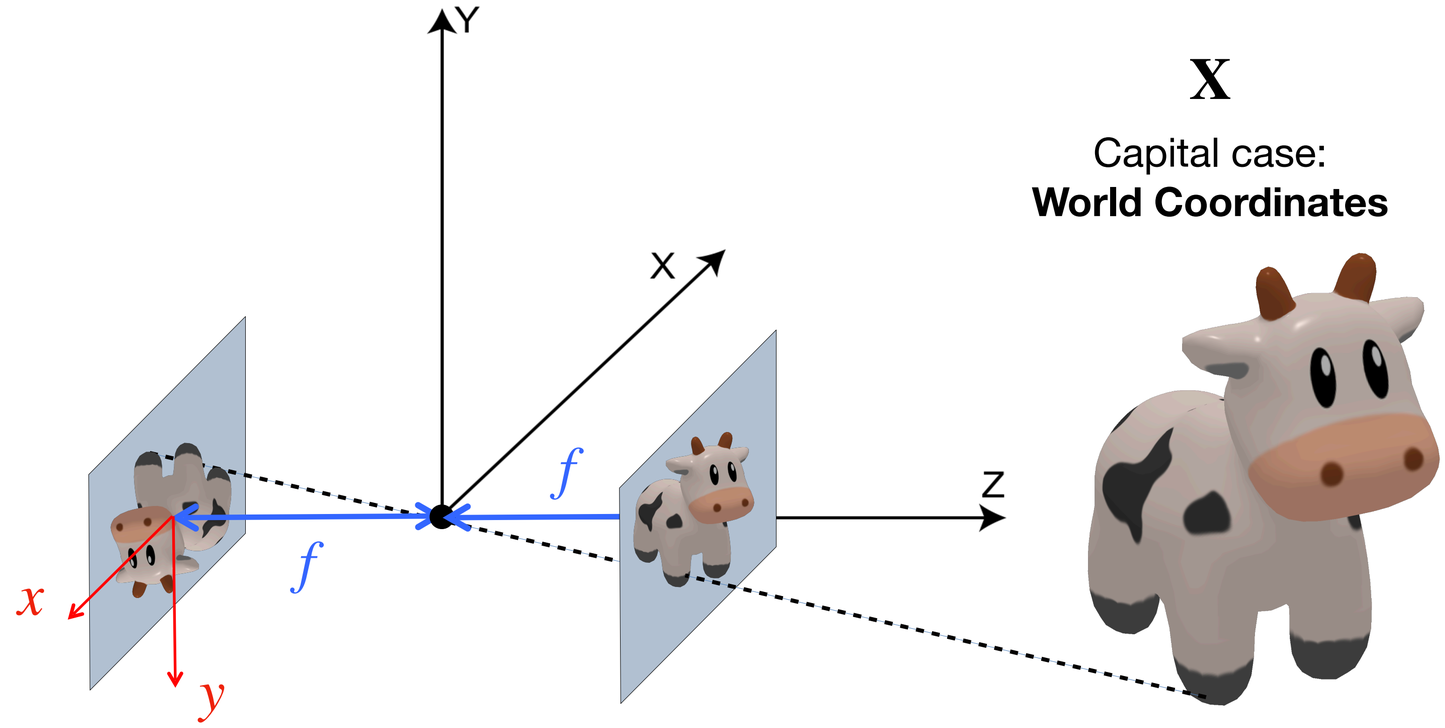
X

Capital case:
World Coordinates

x

Lowercase:
Pixel / Image Coordinates

Perspective projection



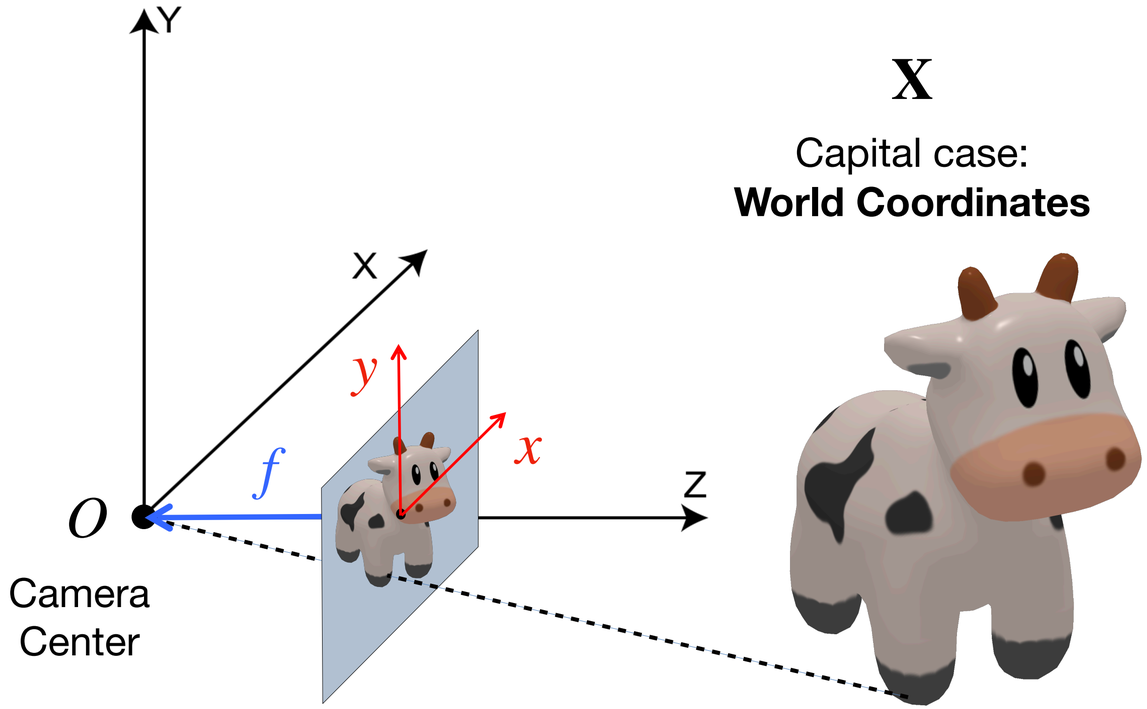
X

Capital case:
World Coordinates

x

Lowercase:
Pixel / Image Coordinates

Perspective projection



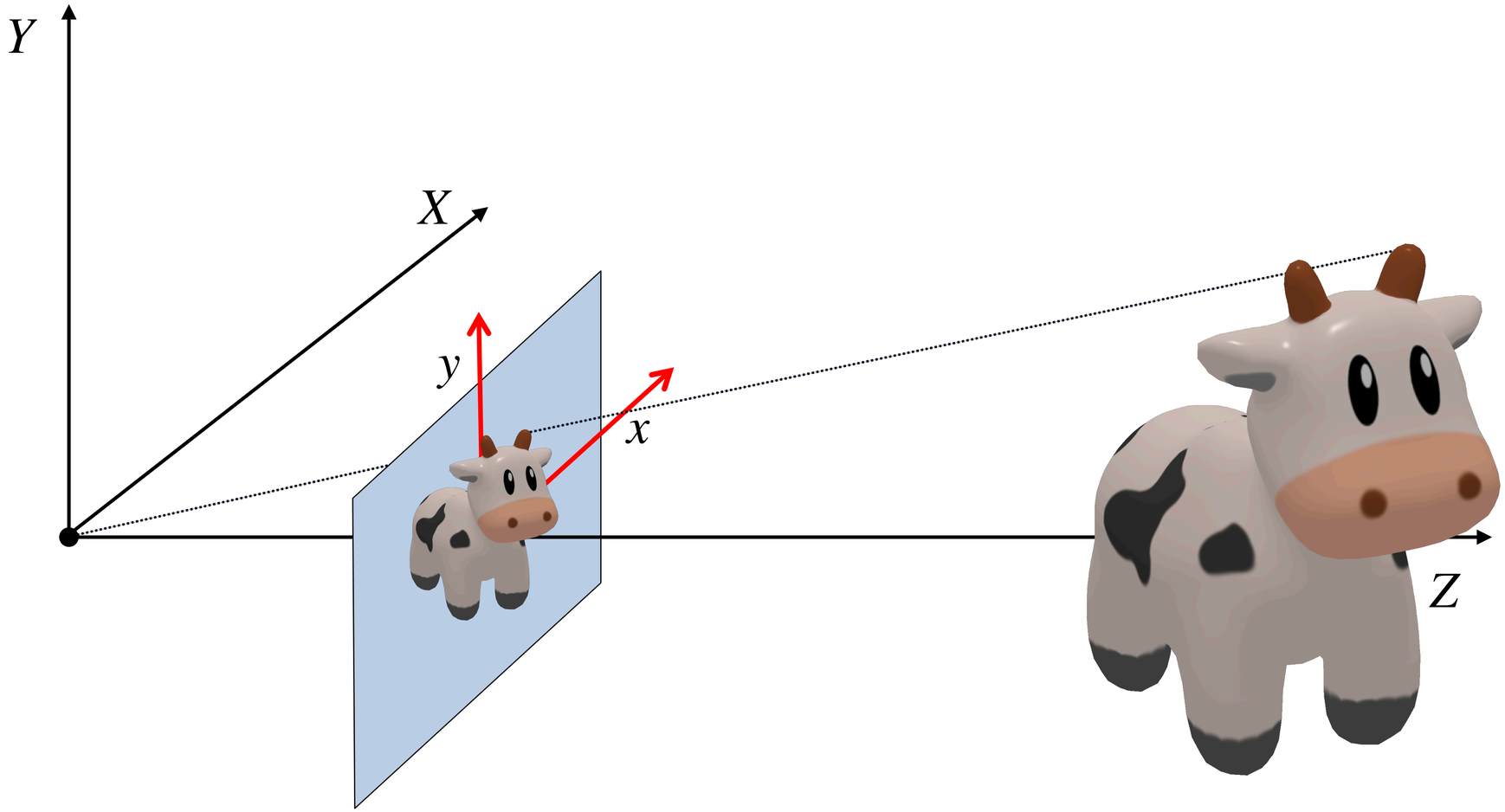
X

Capital case:
World Coordinates

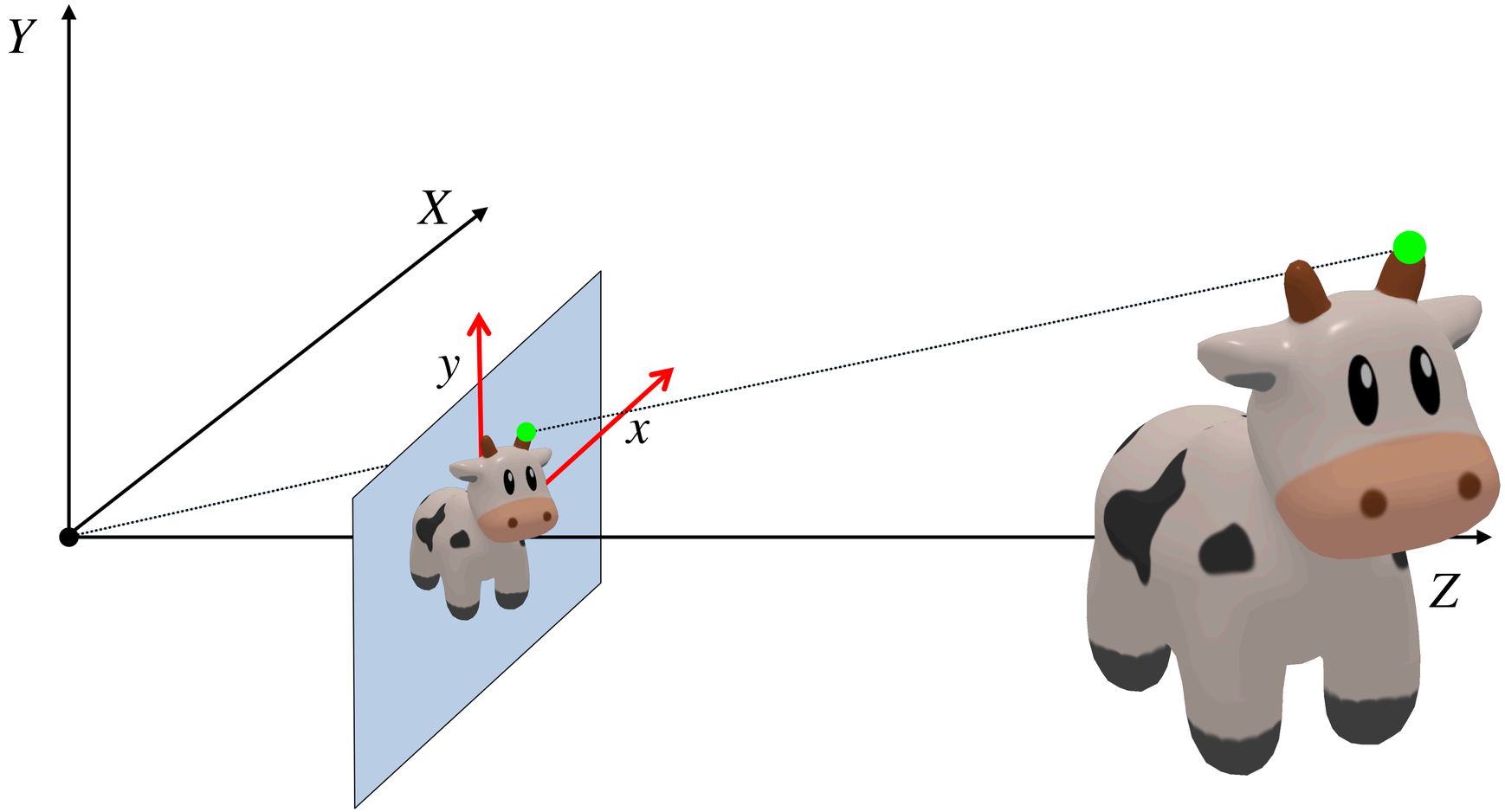
x

Lowercase:
Pixel / Image Coordinates

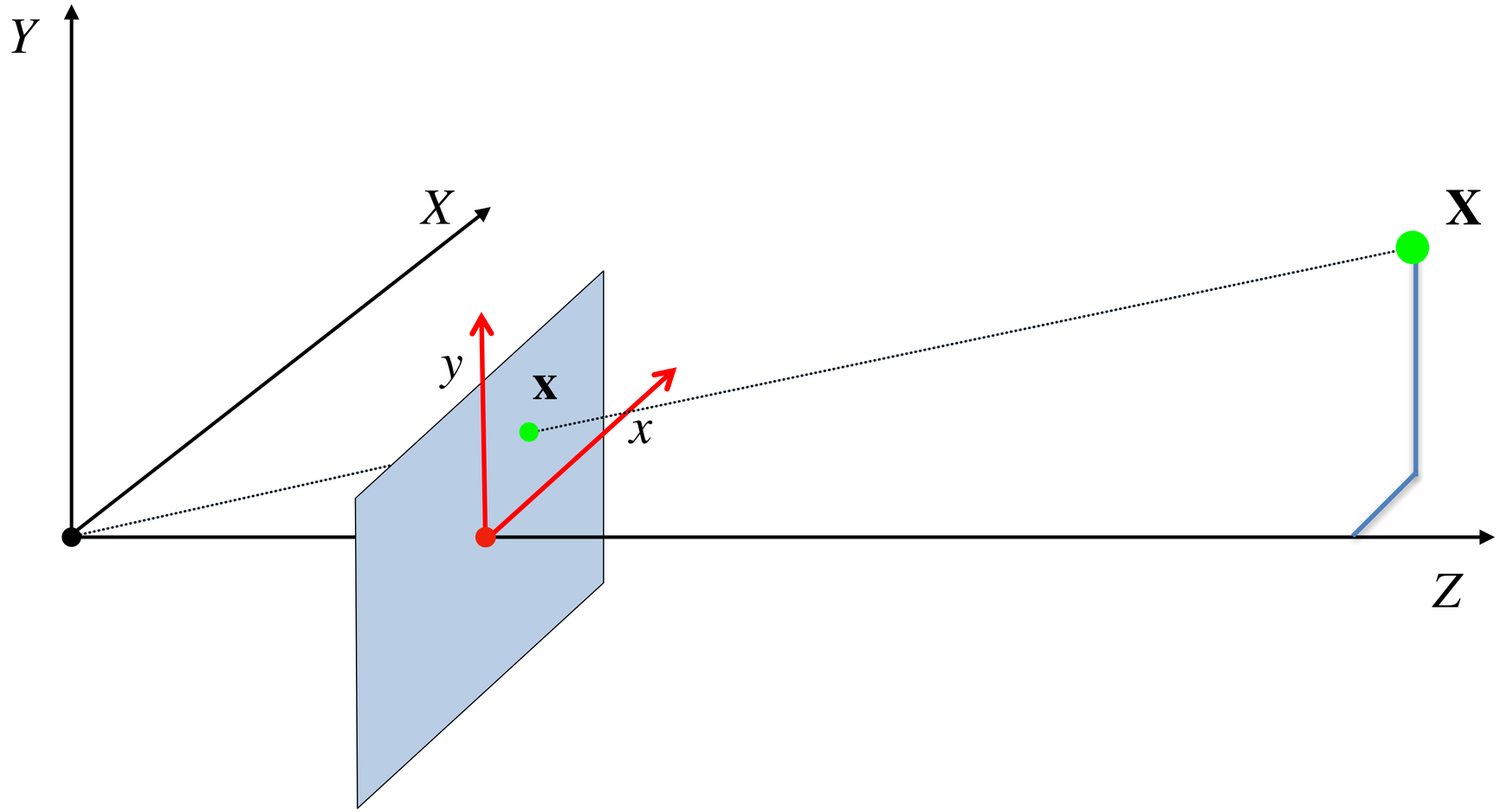
Perspective projection



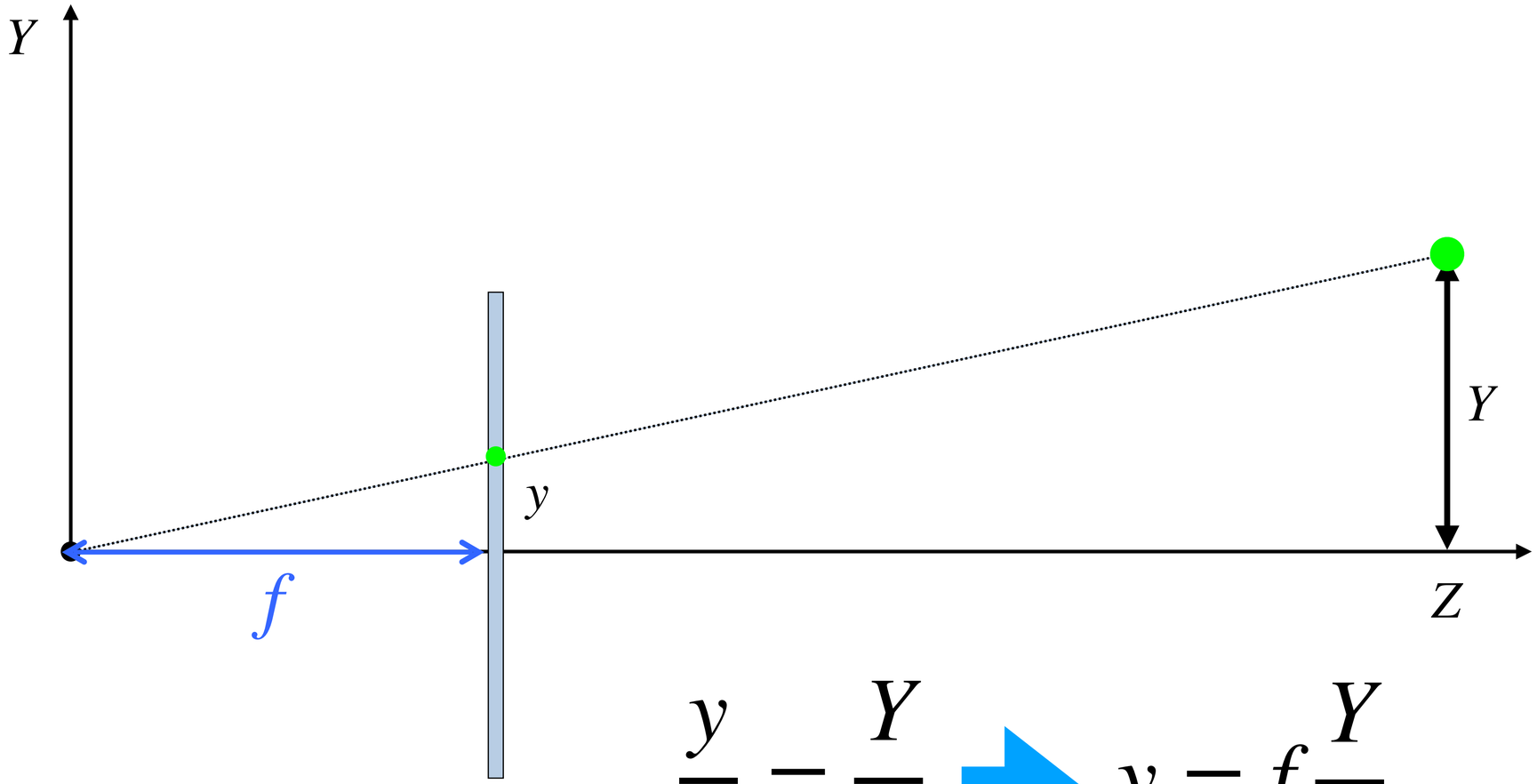
Perspective projection



Perspective projection

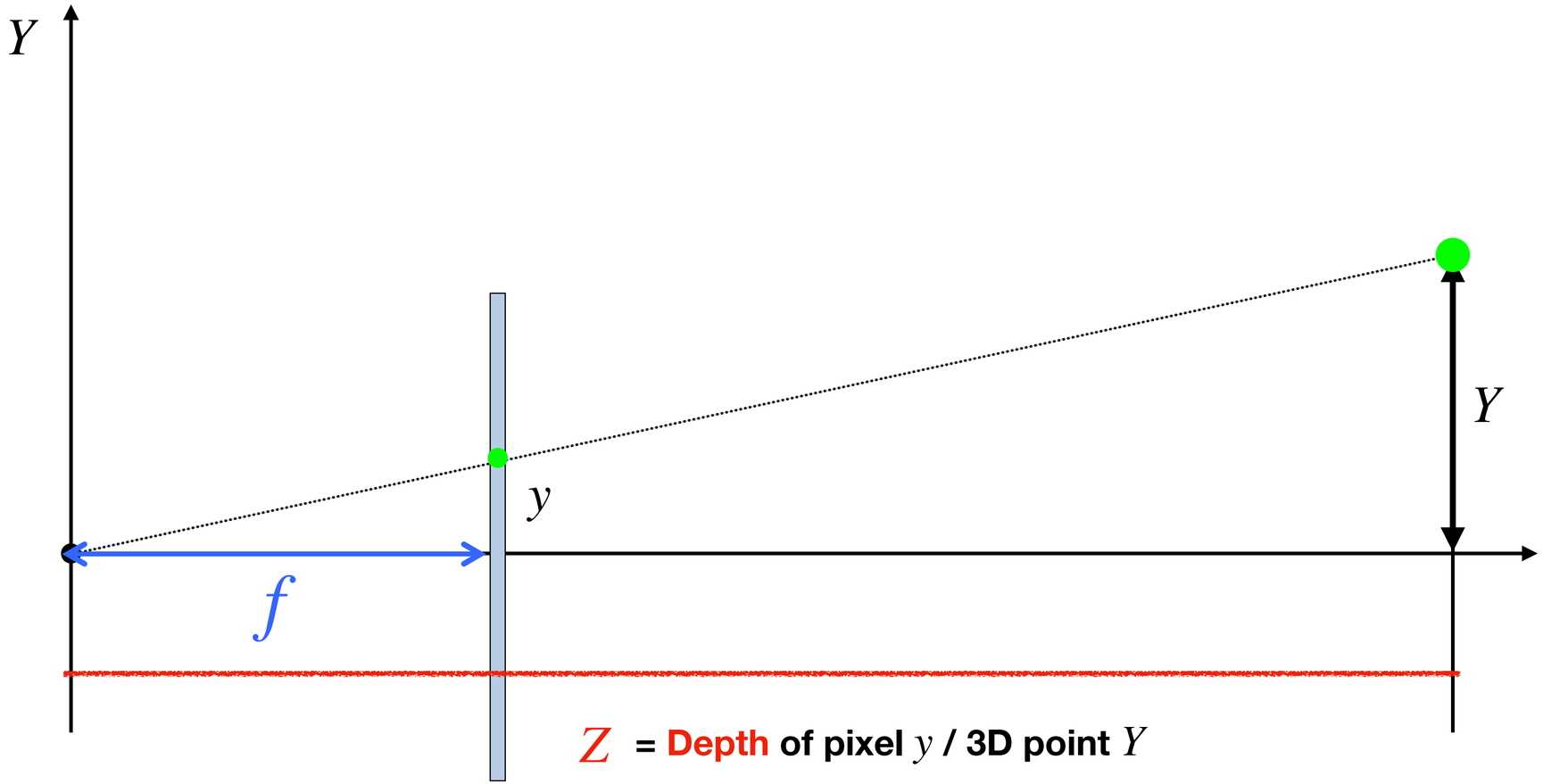


Perspective projection



$$\frac{y}{f} = \frac{Y}{Z} \Rightarrow y = f\frac{Y}{Z}$$

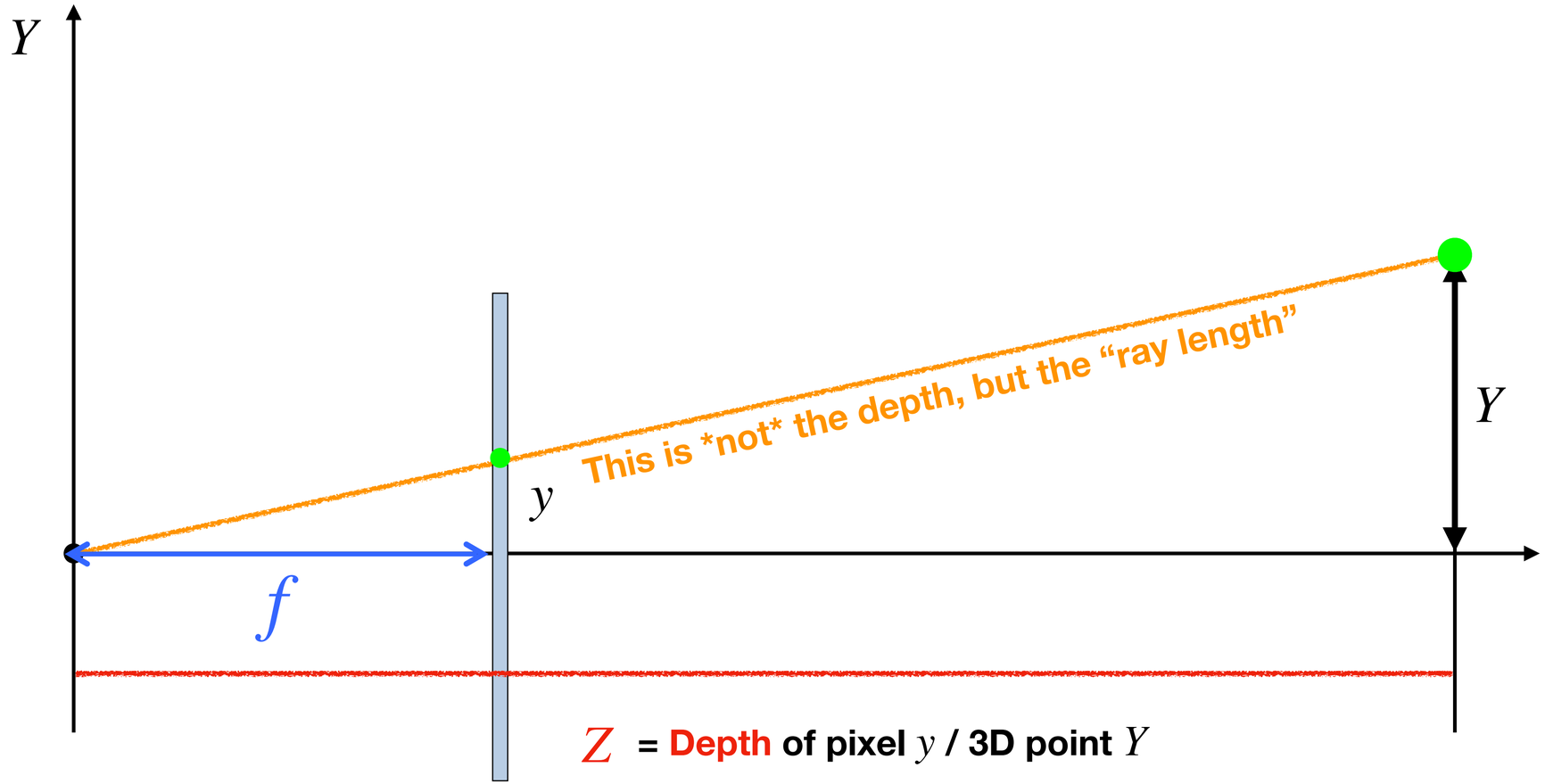
Depth of a pixel / 3D point



Z = **Depth of pixel y / 3D point Y**

$$y = f\frac{Y}{Z}$$

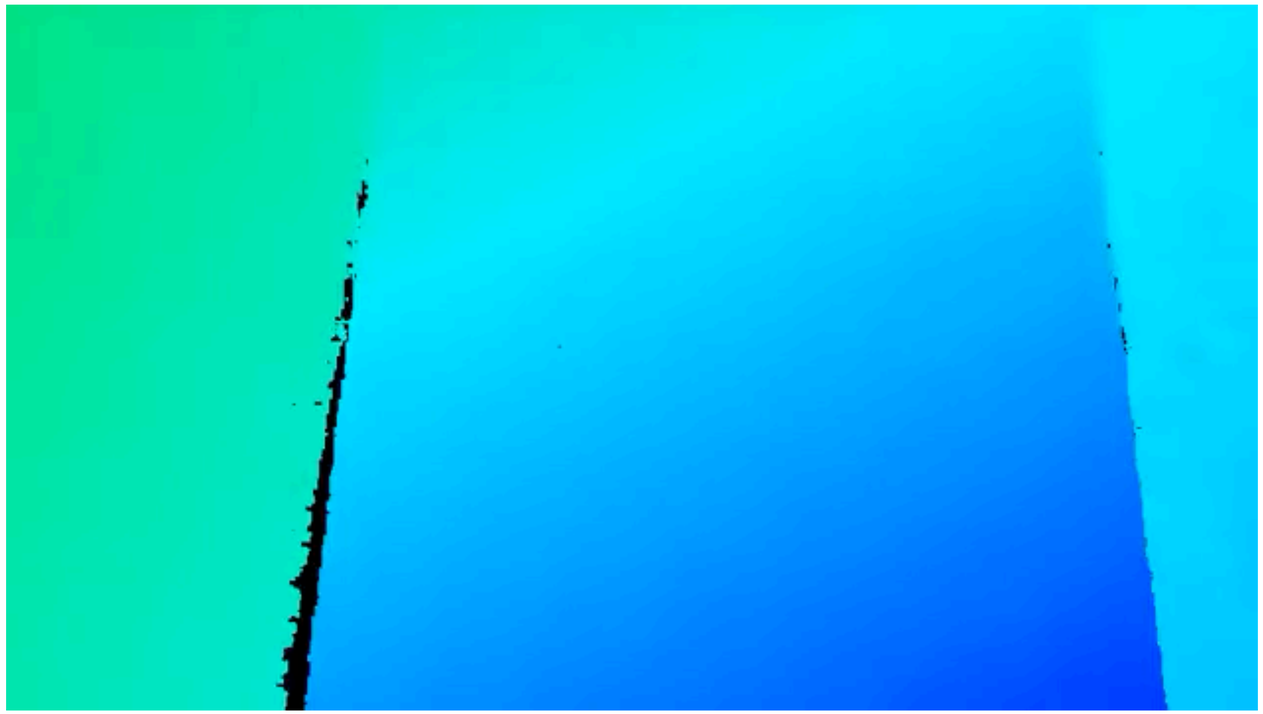
Depth vs. Ray Length



$$y = f\frac{Y}{Z}$$

Side note : there exists Depth Sensors



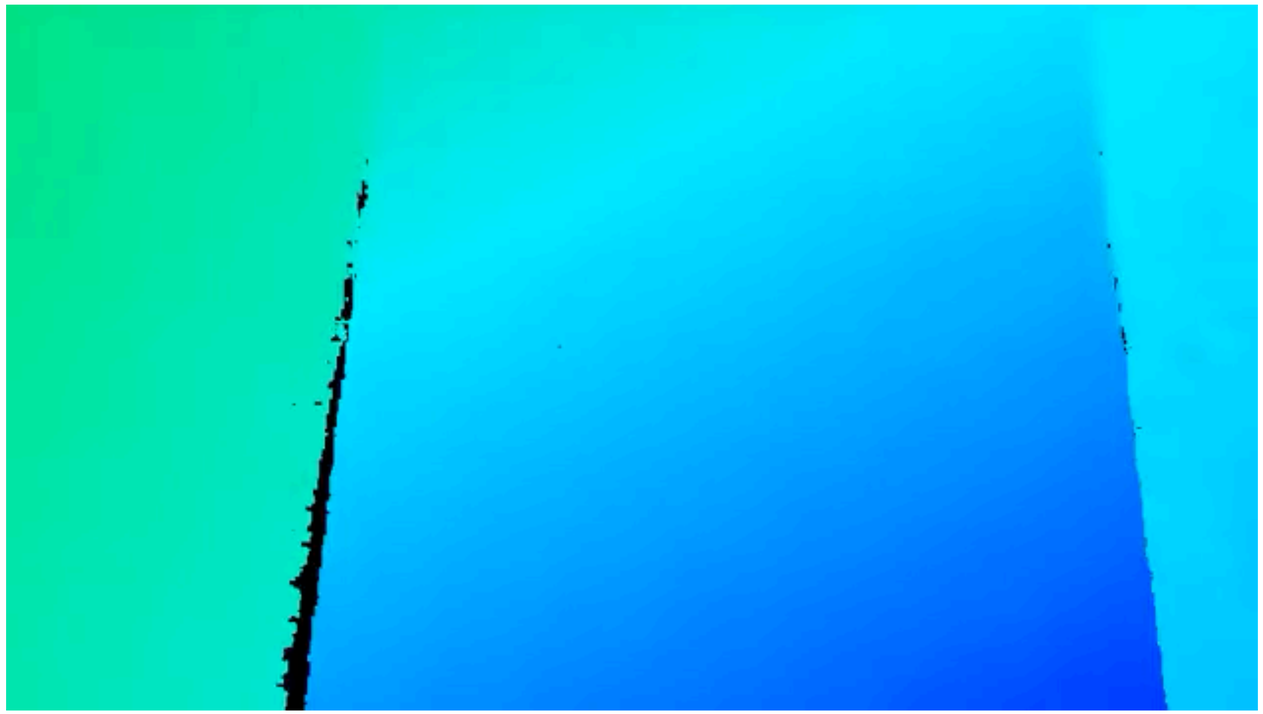


Output of Intel RealSense Camera

https://www.youtube.com/watch?v=t-X7tnT8CVY

Side note : there exists Depth Sensors

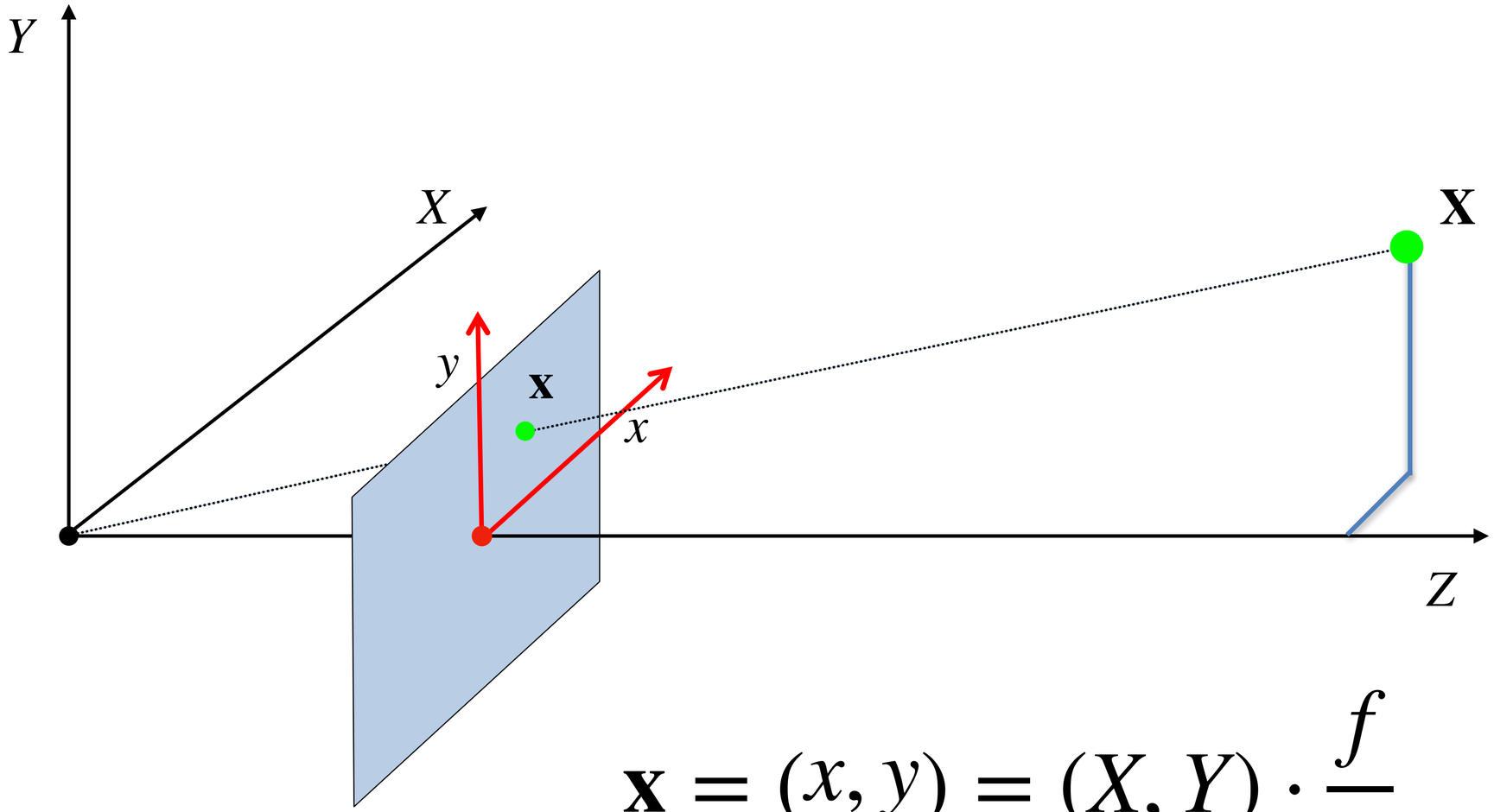




Output of Intel RealSense Camera

https://www.youtube.com/watch?v=t-X7tnT8CVY

Perspective projection

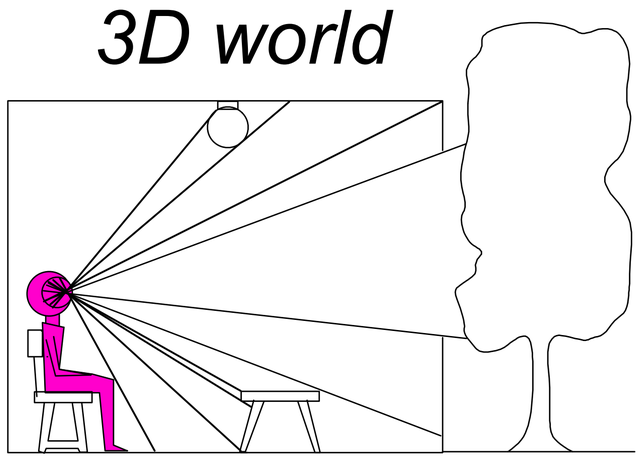


$$\mathbf{x} = (x, y) = (X, Y) \cdot \frac{f}{Z}$$

Image / Pixel Coordinates

World Coordinates

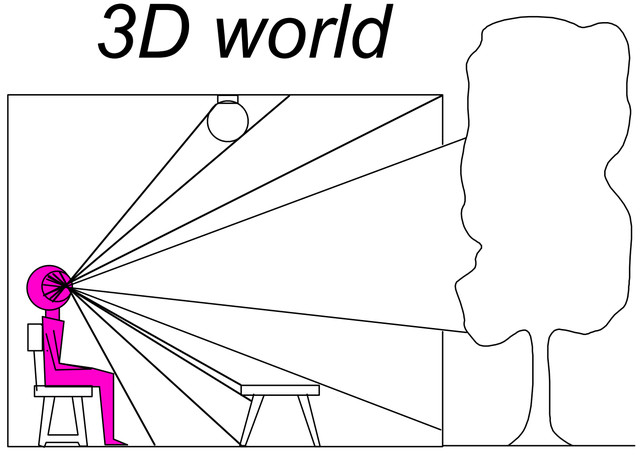
Perspective Projection



Figures © Stephen E. Palmer, 2002

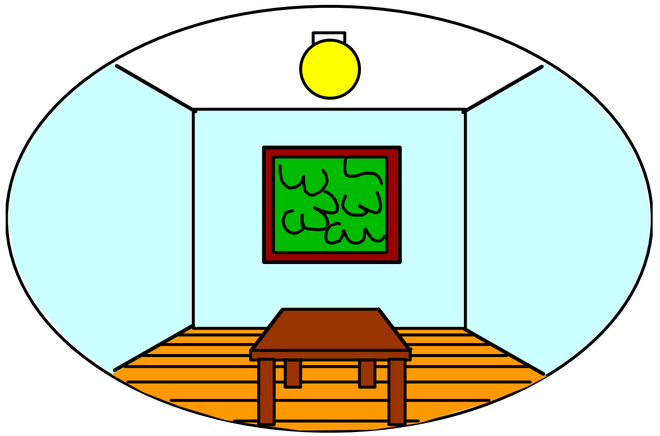
Perspective Projection

3D world



Point of observation

2D image

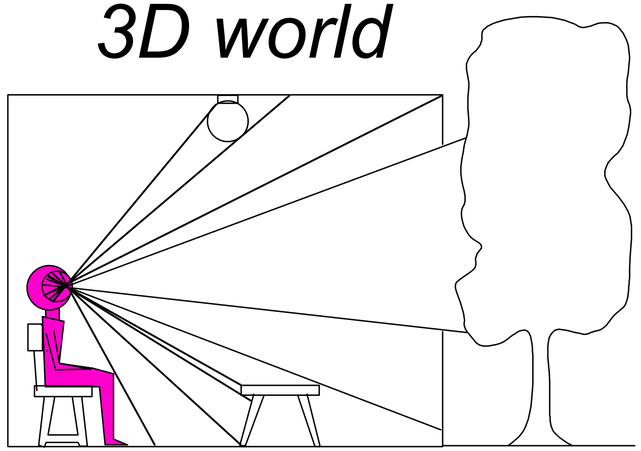


Slide by A. Efros

Figures © Stephen E. Palmer, 2002

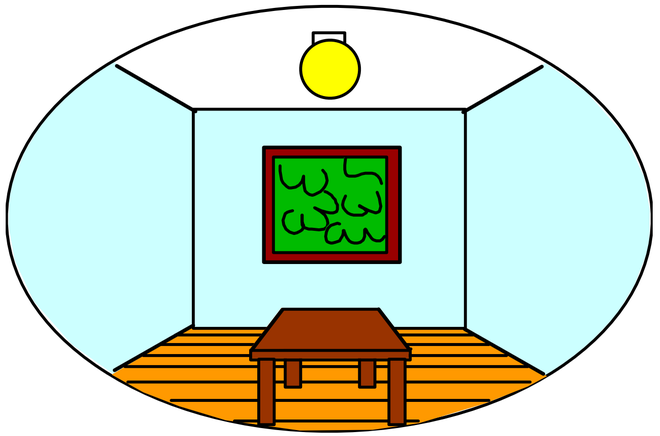
Perspective Projection

3D world



Point of observation

2D image



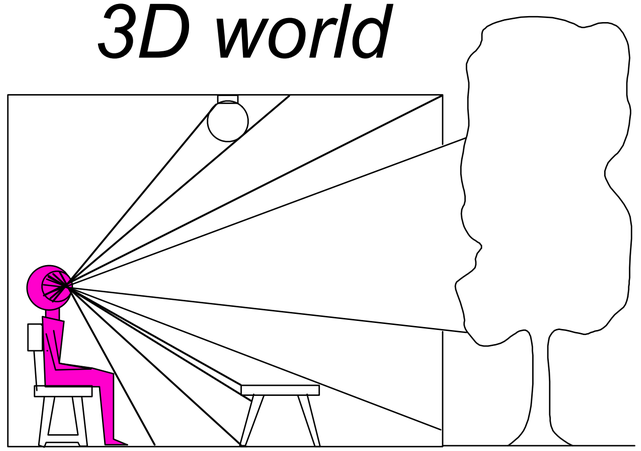
What properties of the world are preserved?

Slide by A. Efros

Figures © Stephen E. Palmer, 2002

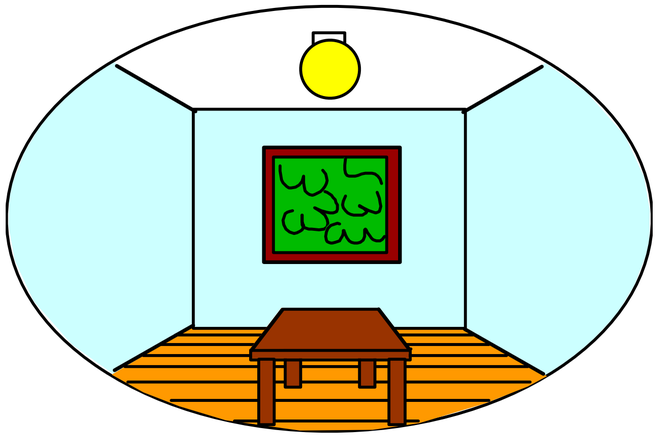
Perspective Projection

3D world



Point of observation

2D image



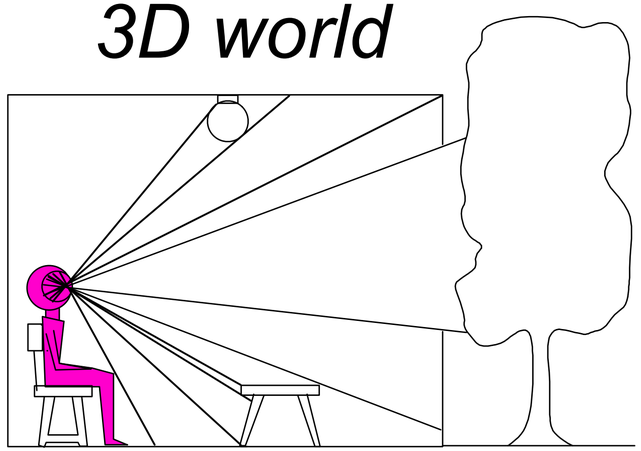
What properties of the world are preserved?

- Straight lines, incidence

Figures © Stephen E. Palmer, 2002

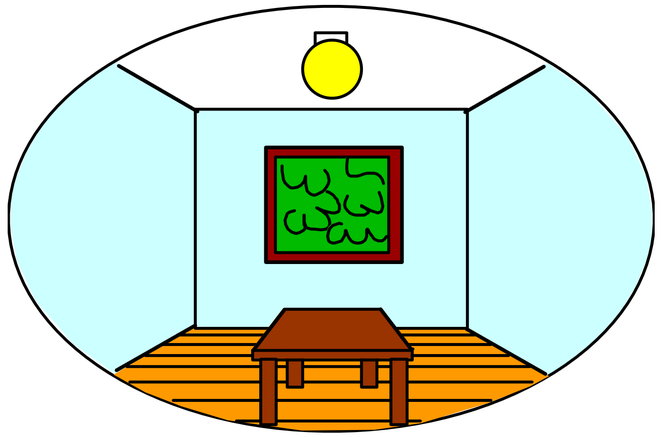
Perspective Projection

3D world



Point of observation

2D image



What properties of the world are preserved?

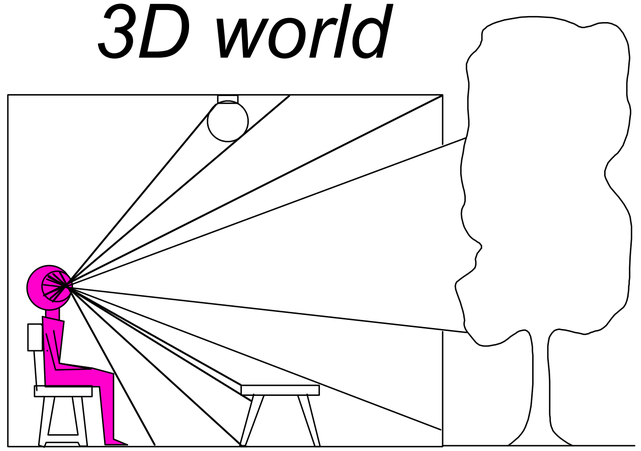
- Straight lines, incidence

What properties are not preserved?

Figures © Stephen E. Palmer, 2002

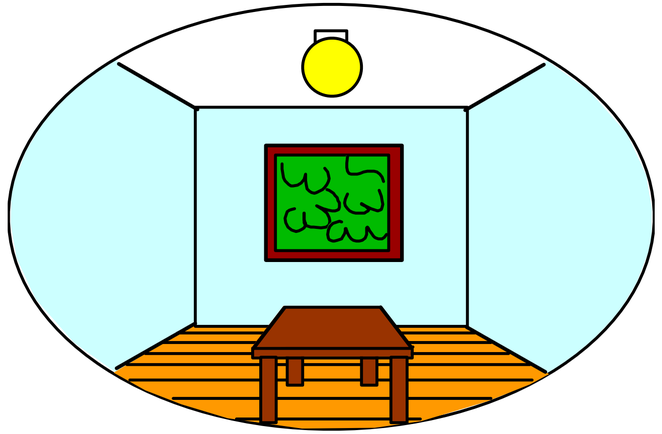
Perspective Projection

3D world



Point of observation

2D image



What properties of the world are preserved?

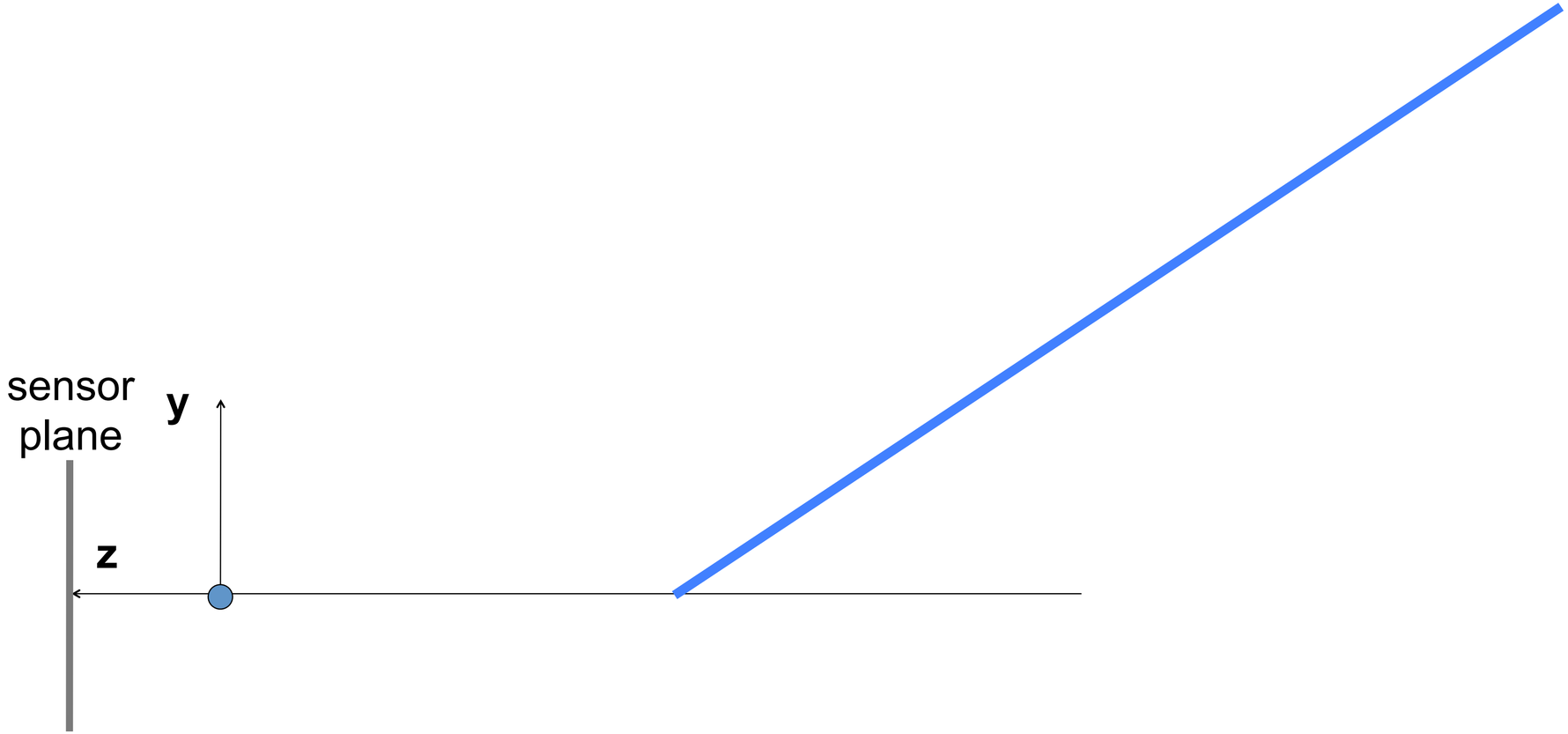
- Straight lines, incidence

What properties are not preserved?

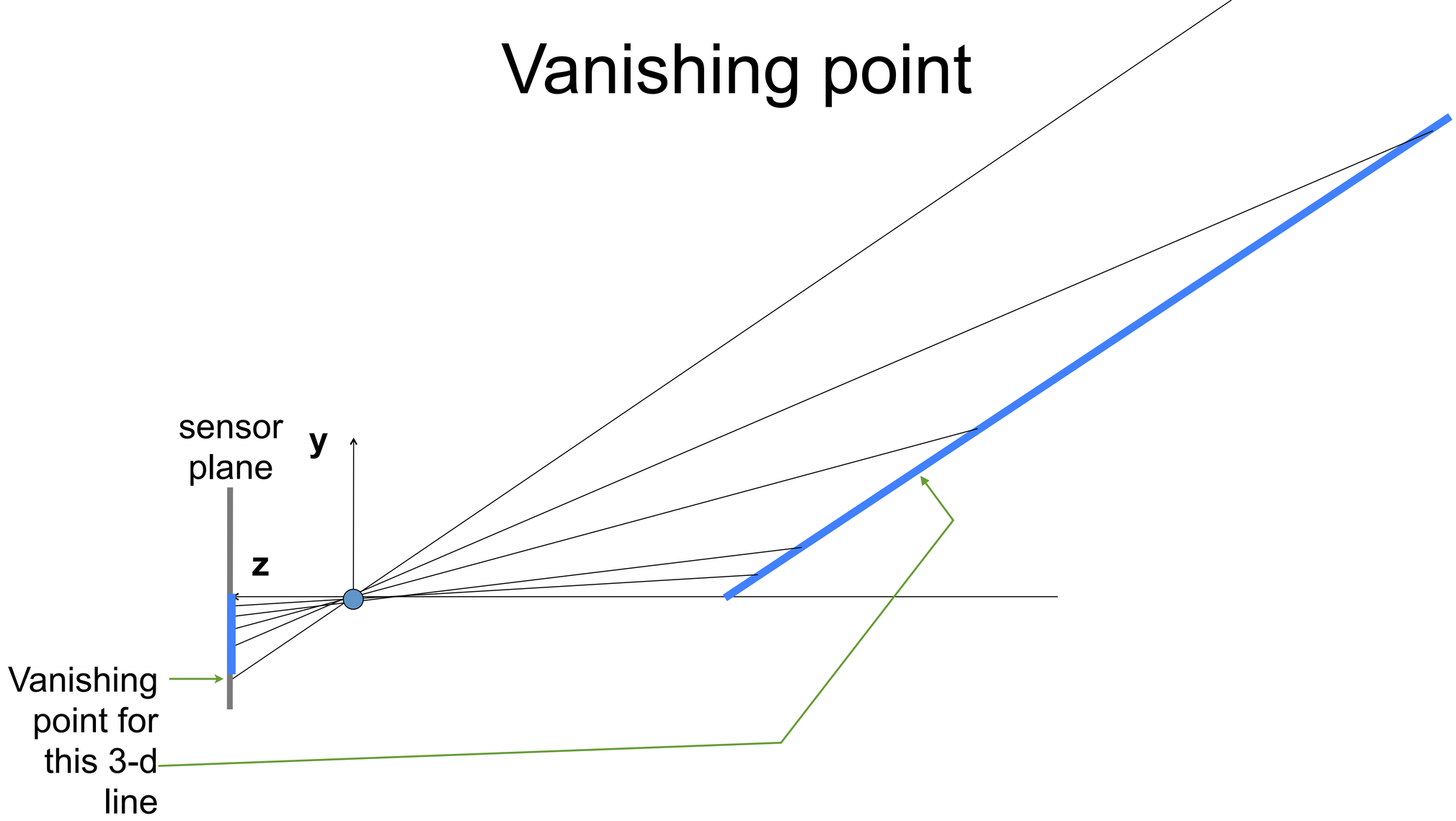
- Angles, lengths

Figures © Stephen E. Palmer, 2002

Vanishing point



Vanishing point



Line in 3-space

$$X(t) = X_0 + at$$

$$Y(t) = Y_0 + bt$$

$$Z(t) = Z_0 + ct$$

Perspective projection of that line

$$x(t) =$$

$$y(t) =$$

Line in 3-space

$$X(t) = X_0 + at$$

$$Y(t) = Y_0 + bt$$

$$Z(t) = Z_0 + ct$$

Perspective projection of that line

$$x(t) = \frac{fX}{Z} = \frac{fX_0 + fat}{Z_0 + ct}$$

$$y(t) = \frac{fY}{Z} = \frac{fY_0 + fbt}{Z_0 + ct}$$

In the limit as $t \rightarrow \pm\infty$
we have (for $c \neq 0$ **):**

Line in 3-space

$$X(t) = X_0 + at$$

$$Y(t) = Y_0 + bt$$

$$Z(t) = Z_0 + ct$$

Perspective projection of that line

$$x(t) = \frac{fX}{Z} = \frac{fX_0 + fat}{Z_0 + ct}$$

$$y(t) = \frac{fY}{Z} = \frac{fY_0 + fbt}{Z_0 + ct}$$

In the limit as $t \rightarrow \pm\infty$
we have (for $c \neq 0$ **):**

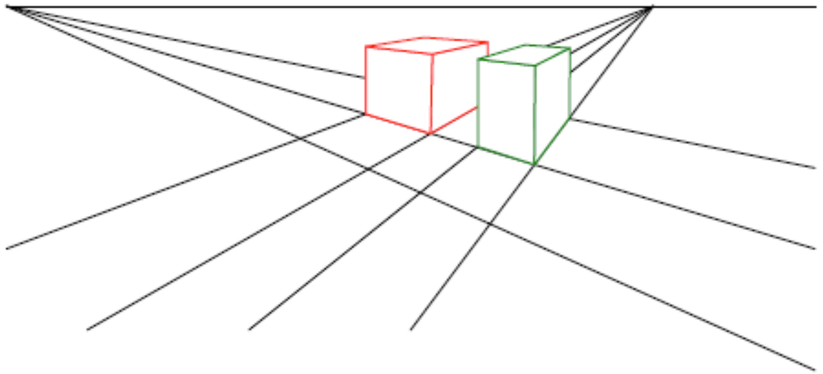
$$x(t \rightarrow \infty) \rightarrow \frac{fa}{c}$$

$$y(t \rightarrow \infty) \rightarrow \frac{fb}{c}$$

Any set of parallel lines (same a, b, c) converge to the same point (called the vanishing point).

Vanishing points

- Each set of parallel lines (=direction) meets at a different point
  - The *vanishing point* for this direction
- Sets of parallel lines on the same plane lead to *colinear* vanishing points.
  - The line is called the *horizon* for that plane



Perspective Projection



Video source: Camera Obscura & World of Illusions

Slide credit: CMU 16-889: Learning for 3D Vision, Prof. Shubham Tulsiani

Perspective Projection



Video source: Camera Obscura & World of Illusions

Slide credit: CMU 16-889: Learning for 3D Vision, Prof. Shubham Tulsiani

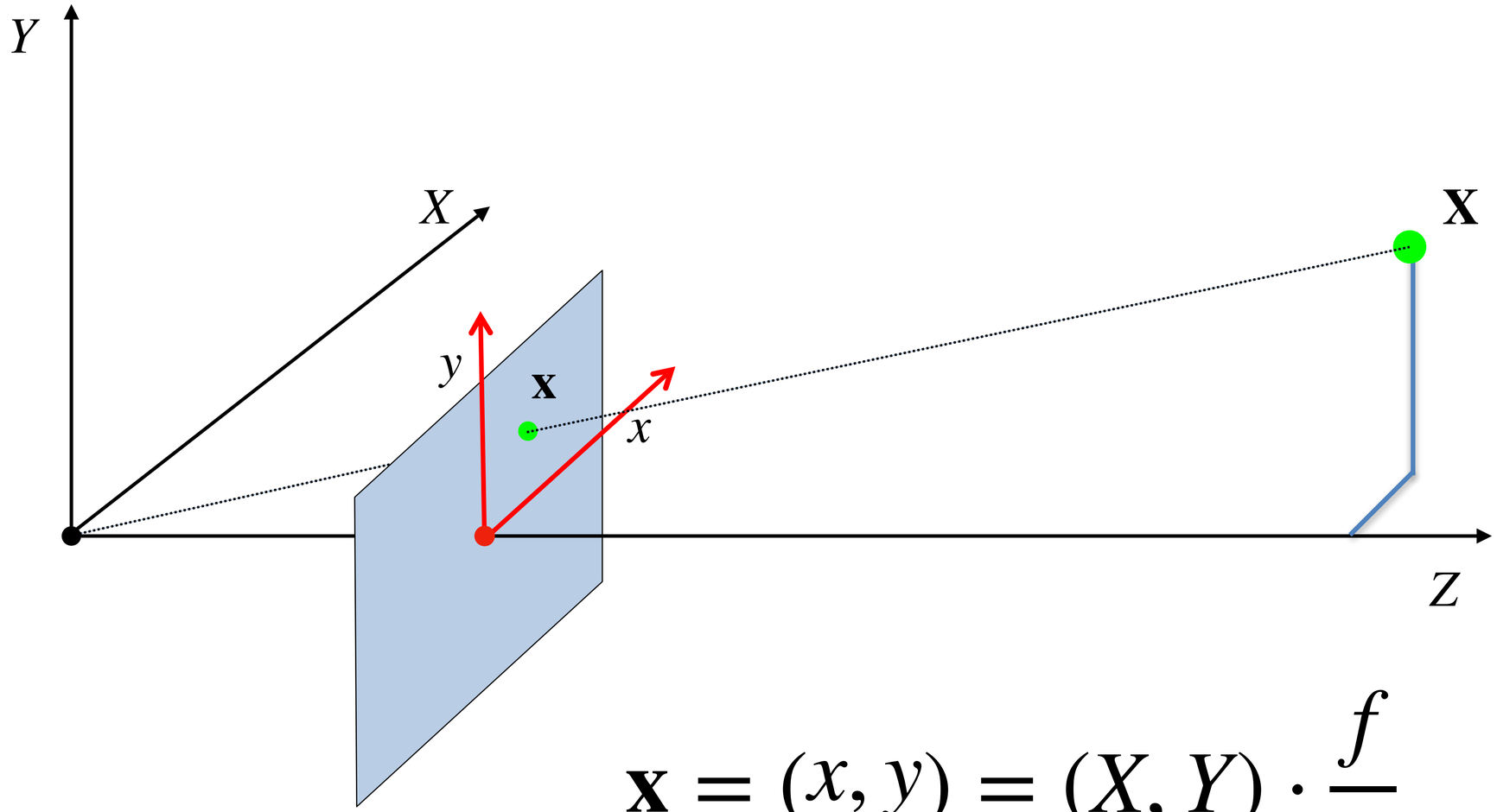
See also Lord of The Ring’s version with a moving camera

- https://www.youtube.com/watch?v=QWMFpxkGO_s
- https://nofilmschool.com/lord-of-the-rings-forced-perspective



Questions?

Perspective projection

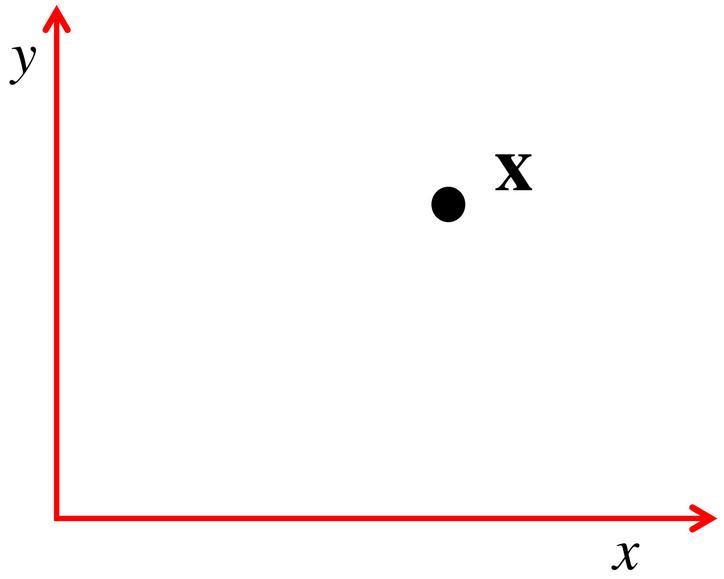


$$\mathbf{x} = (x, y) = (X, Y) \cdot \frac{f}{Z}$$

This is awkward… Not a linear operator.

Homogeneous coordinates

2D



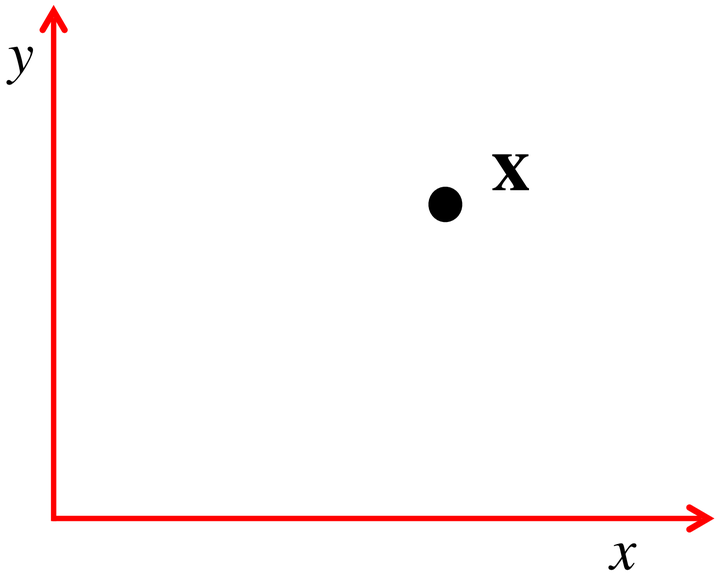
$$\mathbf{x} = (x, y) \quad \Rightarrow \quad \tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

heterogeneous coordinates

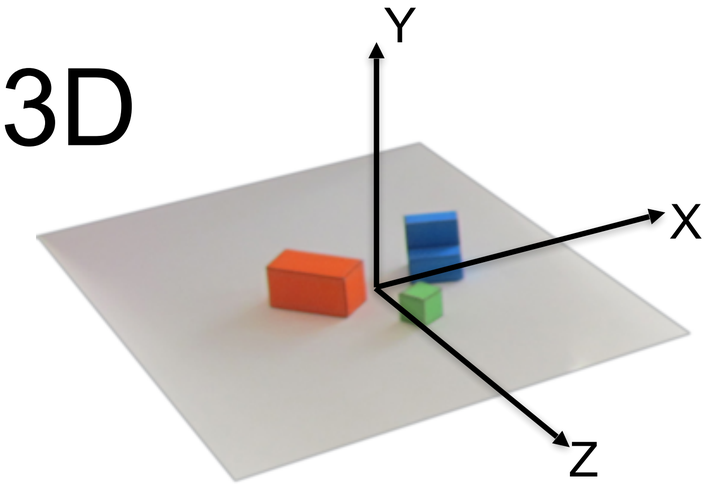
homogeneous coordinates

Homogeneous coordinates

2D



3D



$$\mathbf{x} = (x, y) \quad \Rightarrow \quad \tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

heterogeneous coordinates

homogeneous coordinates

$$\mathbf{X} = (X, Y, Z) \quad \Rightarrow \quad \tilde{\mathbf{X}} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

From heterogeneous to homogeneous:

$$\mathbf{x} = (x, y) \quad \Rightarrow \quad \tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} w \cdot x \\ w \cdot y \\ w \cdot 1 \end{bmatrix}$$

scale invariant (all w $\tilde{\mathbf{x}}$ represent the same point)

For w≠0

From heterogeneous to homogeneous:

$$\mathbf{x} = (x, y) \quad \Rightarrow \quad \tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} w \cdot x \\ w \cdot y \\ w \cdot 1 \end{bmatrix}$$

scale invariant (all w$\tilde{\mathbf{x}}$ represent the same point)

For w≠0

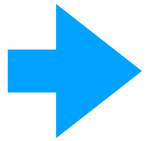
From homogeneous to heterogeneous:

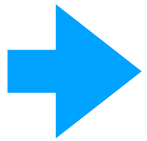
$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad \Rightarrow \quad \mathbf{x} = \left(x/w, y/w\right)$$

Homogeneous coordinates

2D

$\mathbb{R}^2$ $\mathbb{P}^2$

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

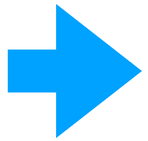
$\begin{bmatrix} x \\ y \\ w \end{bmatrix}$  $(x/w, y/w)$

Homogeneous coordinates

2D

$\mathbb{R}^2$

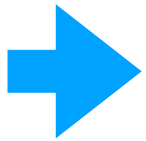
(x, y)



$\mathbb{P}^2$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix}$$



$$(x/w, y/w)$$

3D

$\mathbb{R}^3$

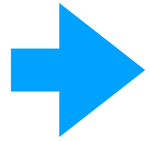
(X, Y, Z)



$\mathbb{P}^3$

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

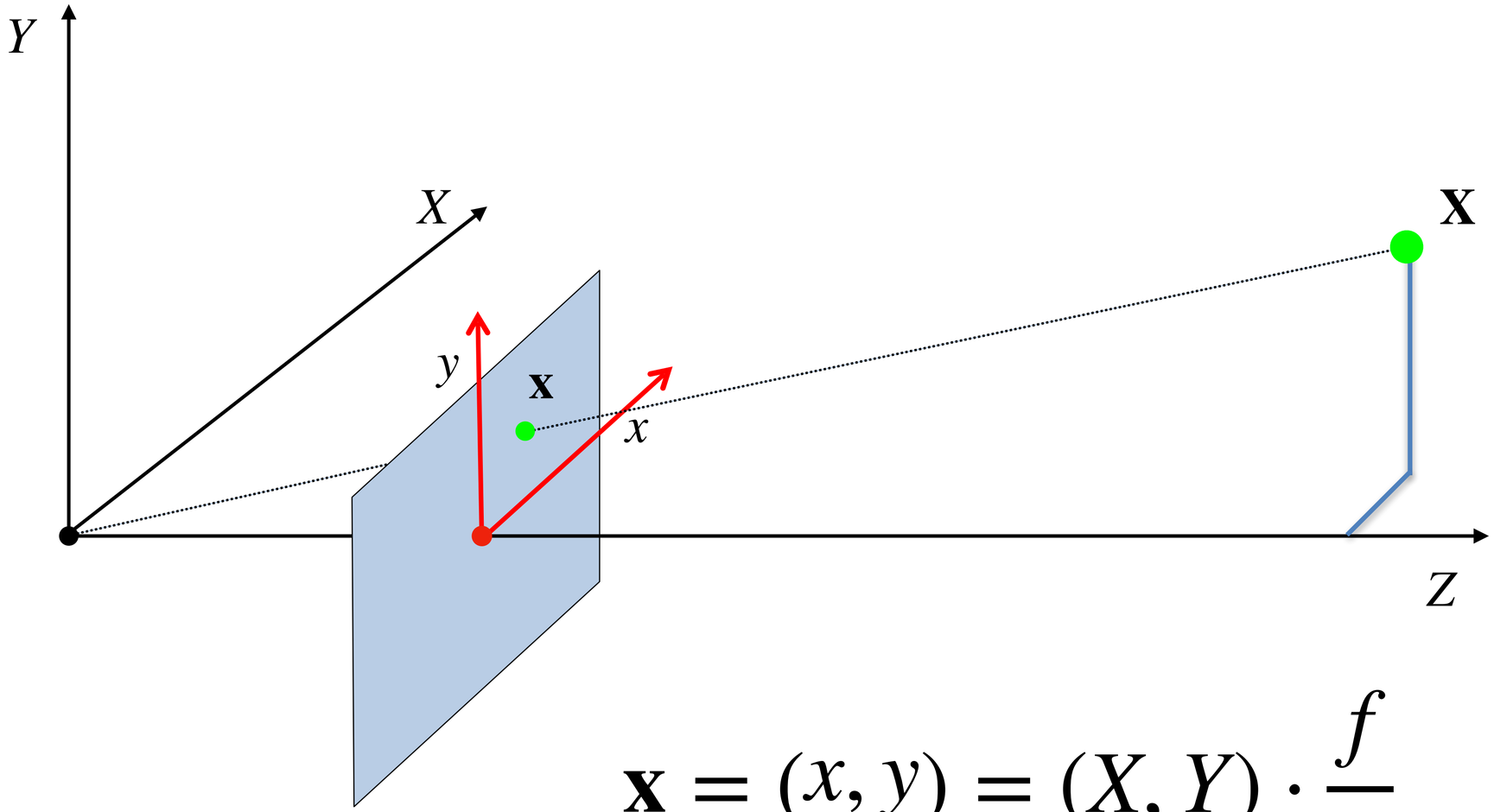
$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$



$$(X/W, Y/W, Z/W)$$

Questions?

Perspective projection



$$\mathbf{x} = (x, y) = (X, Y) \cdot \frac{f}{Z}$$

This is awkward… Not a linear operator.

Perspective projection

Heterogeneous coordinates

$$\mathbf{x} = (x, y) = (X, Y) \cdot \frac{f}{Z}$$

Image / Pixel Coordinates

World Coordinates

Homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = ? \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Image / Pixel Coordinates

World coords.

Perspective projection

Heterogeneous coordinates

$$\mathbf{x} = (x, y) = (X, Y) \cdot \frac{f}{Z}$$

Image / Pixel Coordinates

World Coordinates

Homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Image / Pixel Coordinates

Projection Matrix

World coords.

Perspective projection

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z/f \end{bmatrix}$$

Image / Pixel Coordinates

Projection Matrix

World coords.

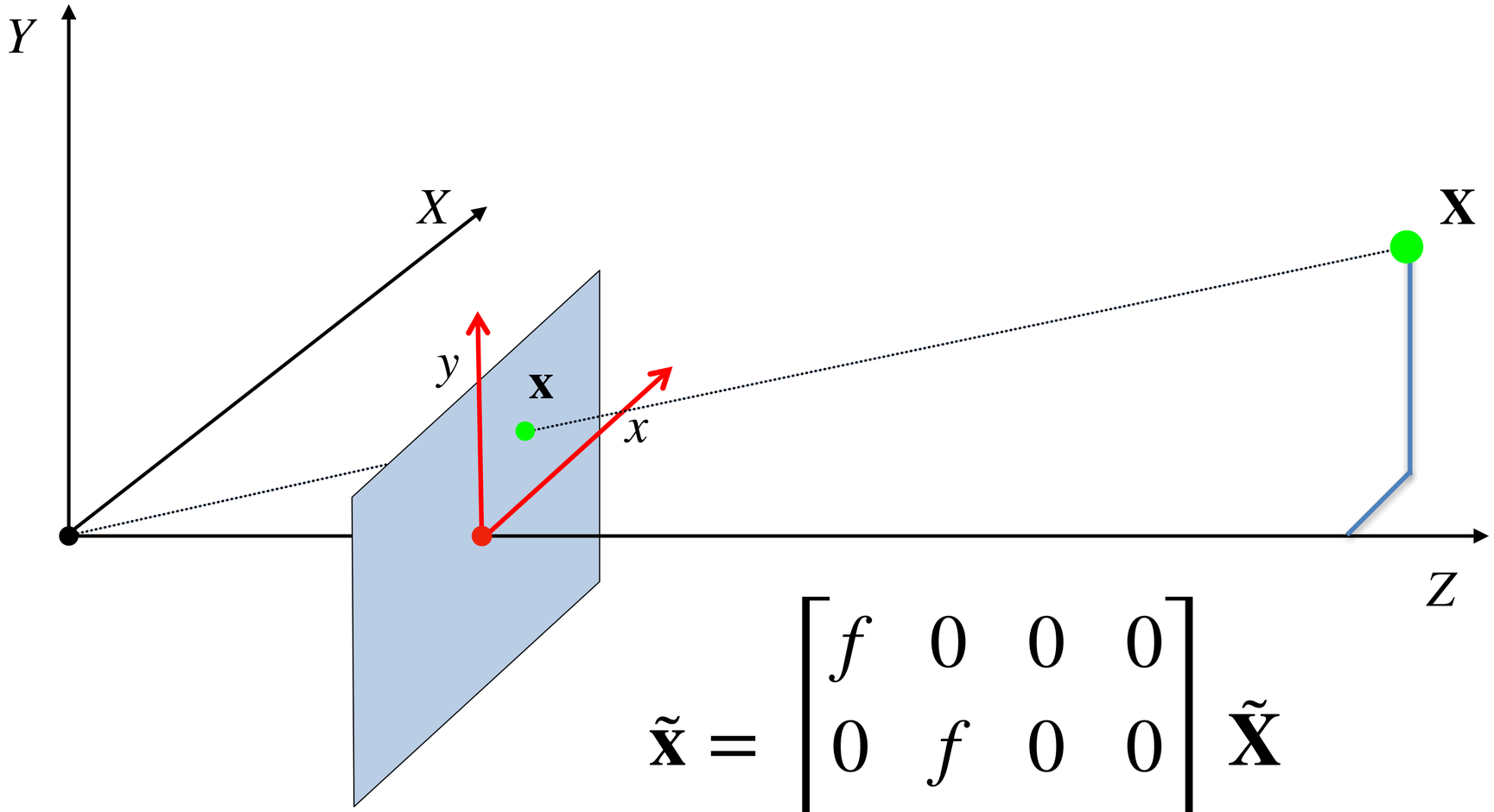
$$\tilde{\mathbf{x}} = \begin{bmatrix} X \\ Y \\ Z/f \end{bmatrix} \quad \mathbf{x} = \left(f\frac{X}{Z}, \quad f\frac{Y}{Z} \right)$$

Perspective projection, equivalently

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \tilde{\mathbf{X}} = \begin{bmatrix} X \\ Y \\ Z/f \end{bmatrix} \quad \Rightarrow \quad \mathbf{x} = \left(f\frac{X}{Z}, \quad f\frac{Y}{Z} \right)$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{X}} = \begin{bmatrix} f \cdot X \\ f \cdot Y \\ Z \end{bmatrix} \quad \Rightarrow \quad \mathbf{x} = \left(f\frac{X}{Z}, \quad f\frac{Y}{Z} \right)$$

Perspective projection recap & questions?



$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{X}}$$

Image / Pixel coords.

World coords.

Projection Matrix

Bonus: Translations are now matrices!

e.g. in 2D

$$\begin{pmatrix} 1 & 0 & t_y \\ 0 & 1 & t_x \\ 0 & 0 & 1 \end{pmatrix}$$

Also cool duality lines-points in 2D

Line $ax+by+c = 0$ represented by (a, b, c), up to a scale factor

Point (x, y) represented by (wx, wy, w) up to a scale factor w

Also cool duality lines-points in 2D

Line $ax+by+c = 0$ represented by (a, b, c), up to a scale factor

Point (x, y) represented by (wx, wy, w) up to a scale factor w

A point is on a line if the dot product $(a, b, c).(x, y,1)$ is zero

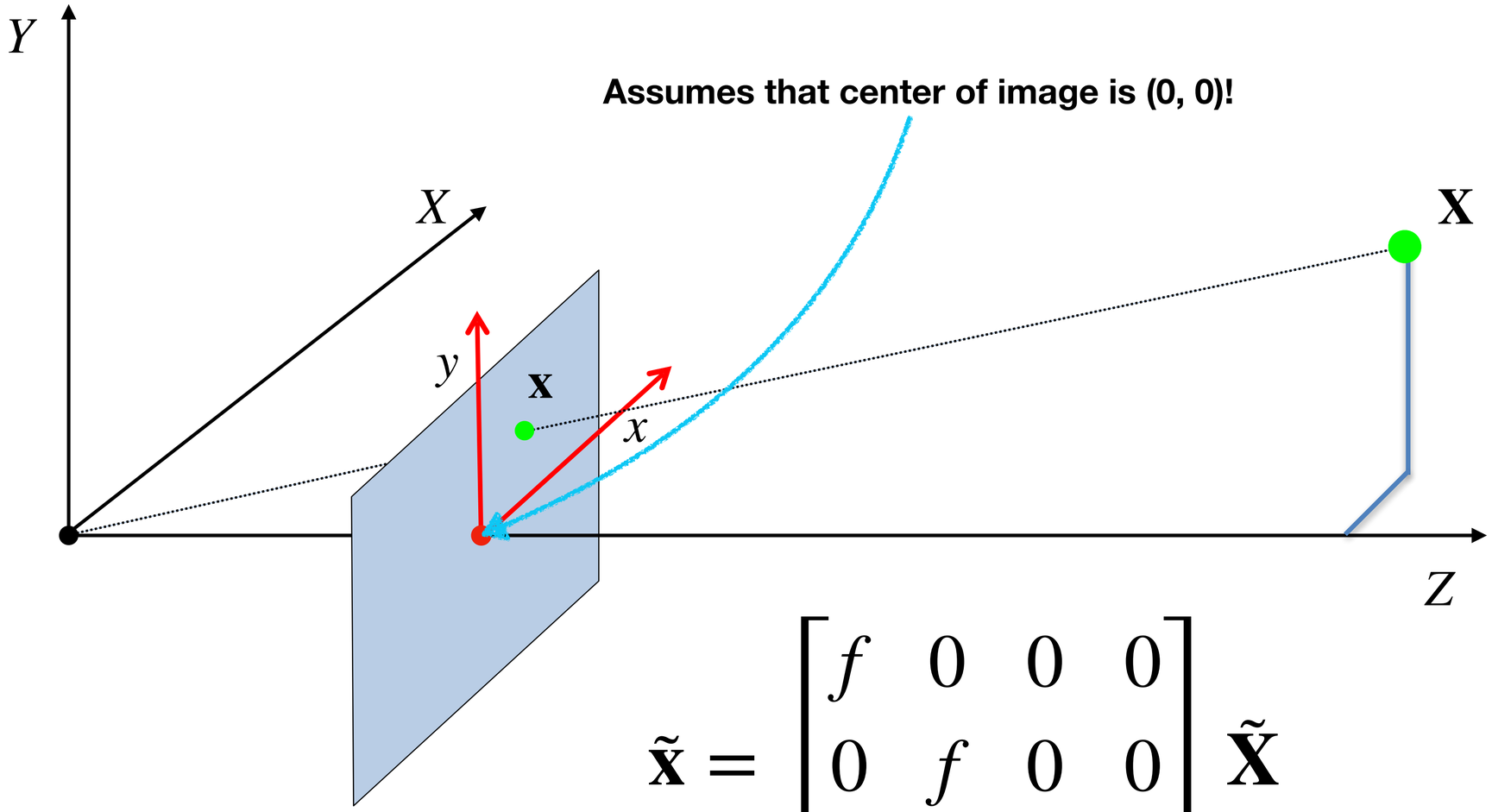
Points and lines are perfectly symmetrical in this equation

Anything you say about point & lines is also true of lines & points

In general true of points-hyperplanes

Questions?

Perspective projection



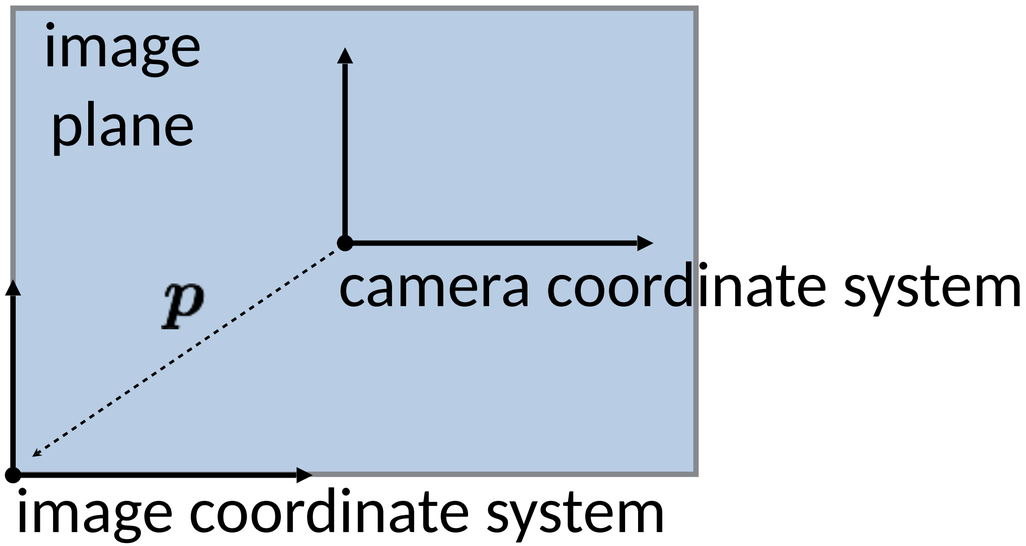
$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{X}}$$

Image / Pixel coords.

World coords.

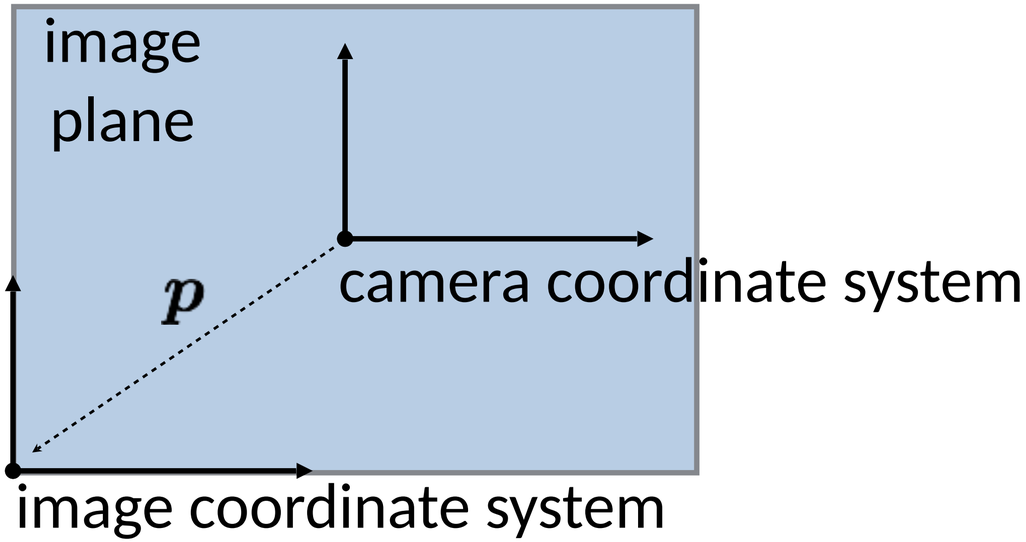
Projection Matrix

More General Case: Arbitrary Image Centre



Slide credit: CMU 16-385 (Yannis, Kris)

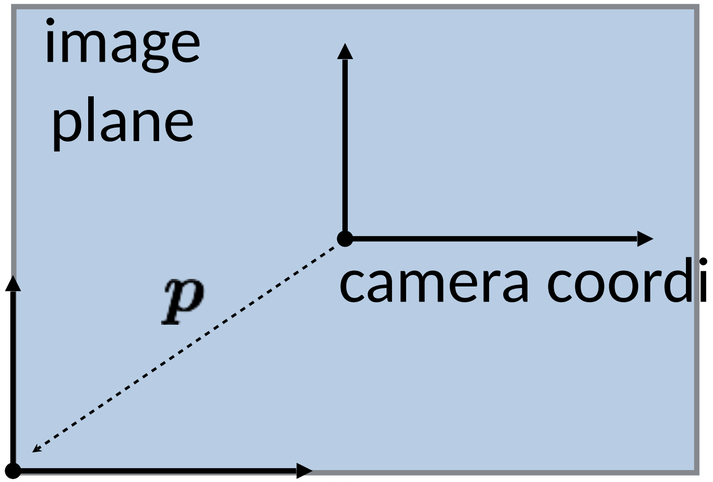
More General Case: Arbitrary Image Centre



$$\begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Slide credit: CMU 16-385 (Yannis, Kris)

More General Case: Arbitrary Image Centre

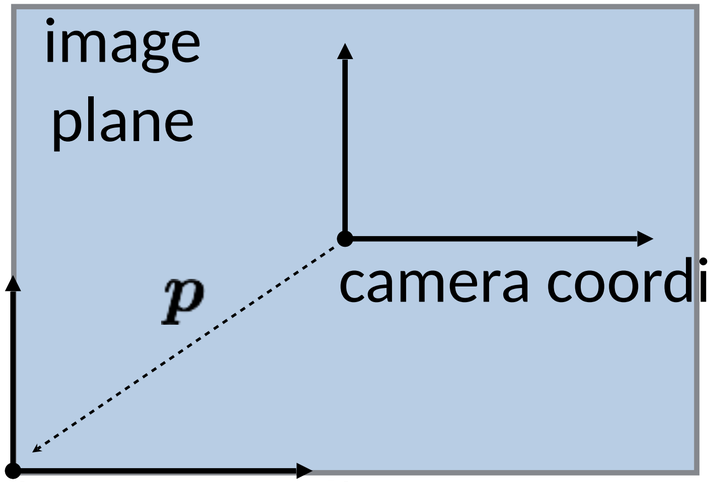


How does the projection matrix change?

$$\begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Slide credit: CMU 16-385 (Yannis, Kris)

More General Case: Arbitrary Image Centre



How does the projection matrix change?

$$\begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

shift vector transforming camera origin to image origin

Slide credit: CMU 16-385 (Yannis, Kris)

Decomposing the Projection Matrix

We can decompose the projection matrix like this:

$$\begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

Slide credit: CMU 16-385 (Yannis, Kris)

Decomposing the Projection Matrix

We can decompose the projection matrix like this:

$$\begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

What does each part of the matrix represent?

Slide credit: CMU 16-385 (Yannis, Kris)

Decomposing the Projection Matrix

We can decompose the projection matrix like this:

$$\begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

(homogeneous) transformation from 2D to 2D, accounting for not unit focal length and origin shift

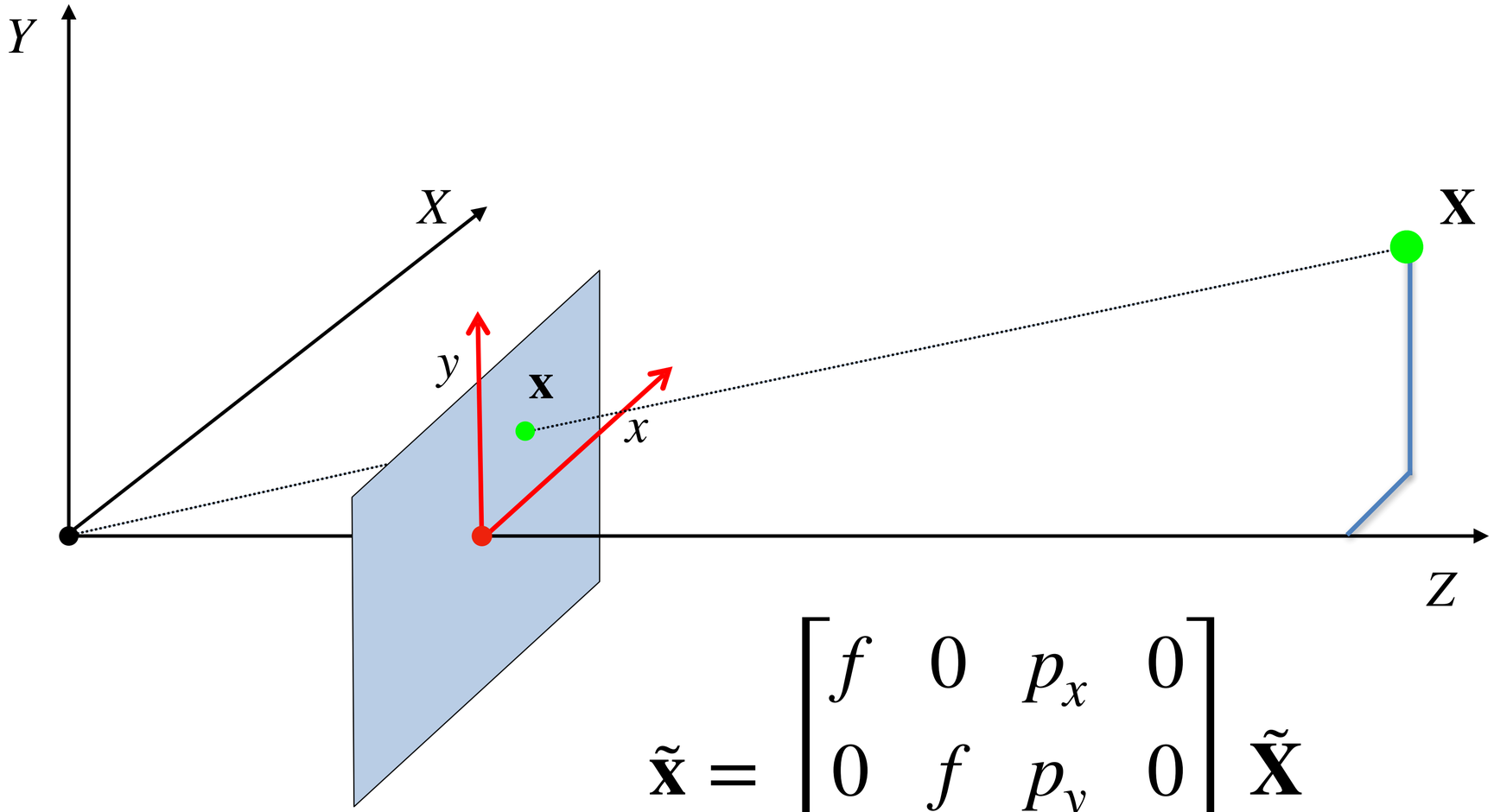
(homogeneous) perspective projection from 3D to 2D, assuming image plane at z = 1 and shared camera/image origin

Also written as: $\mathbf{K}\left[\mathbf{I}\,|\,\mathbf{0}\right]$

where $\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$

Slide credit: CMU 16-385 (Yannis, Kris)

Perspective projection



$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{X}}$$

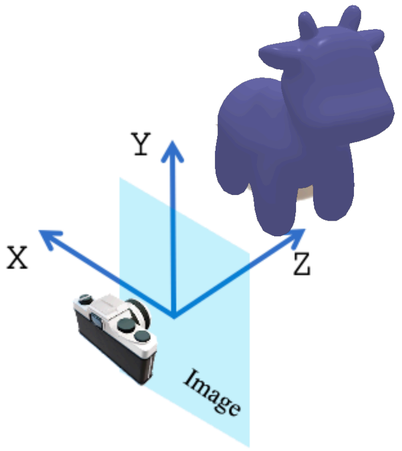
Image / Pixel coords.

World coords.

Projection Matrix

In practice: Decoupling Projection from Image Size

$$K \equiv K_{img \to pix} \; K_{cam \to img}$$



$\mathbb{P}^3$
3D in Camera Frame

$K_{cam \to img}$

(0, 0)



(1, 1)

Normalized Image Frame (here OpenCV convention)

$K_{img \to pix}$

(0,0)



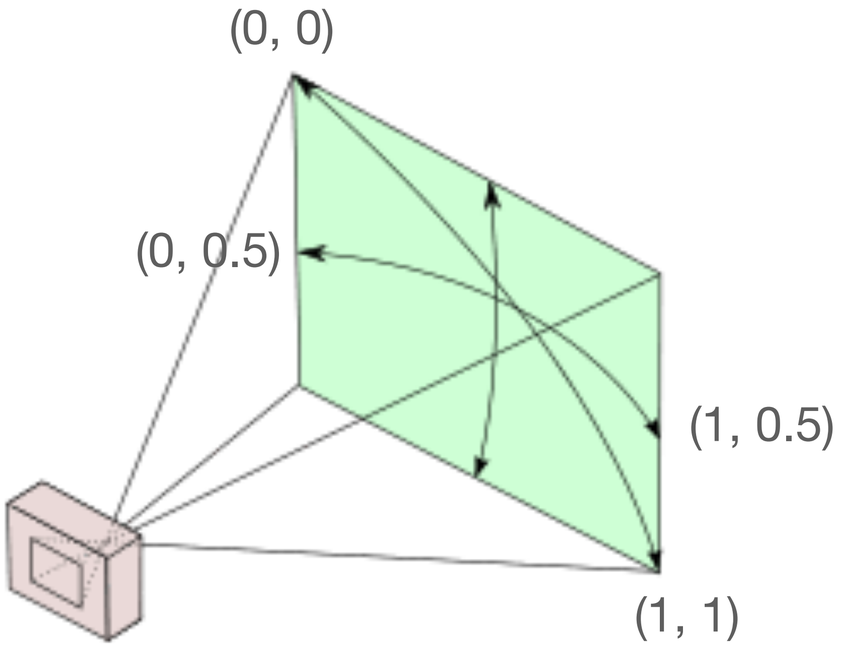
(255,255)

$\mathbb{P}^2$

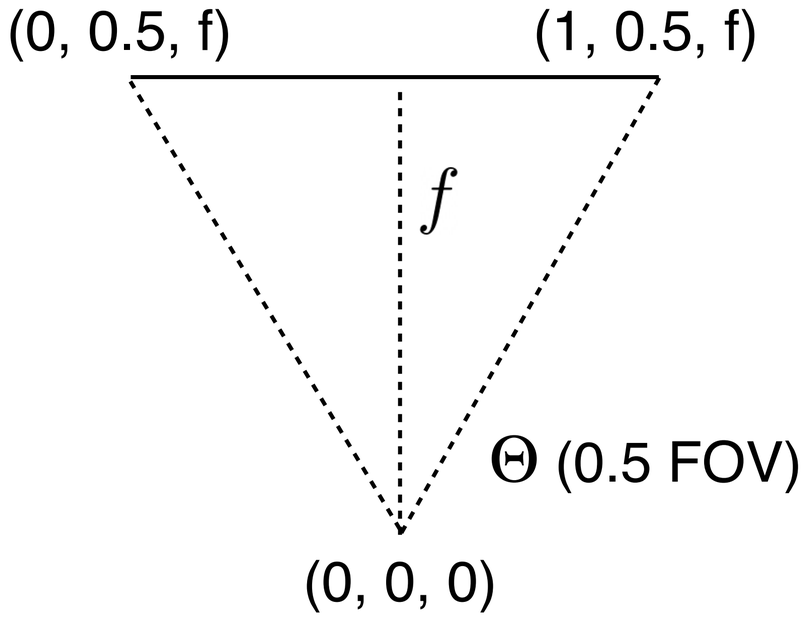
Slide credit: CMU 16-889: Learning for 3D Vision, Prof. Shubham Tulsiani

Exercise: Focal Length as Function of FOV

$$K_{cam \to img}$$

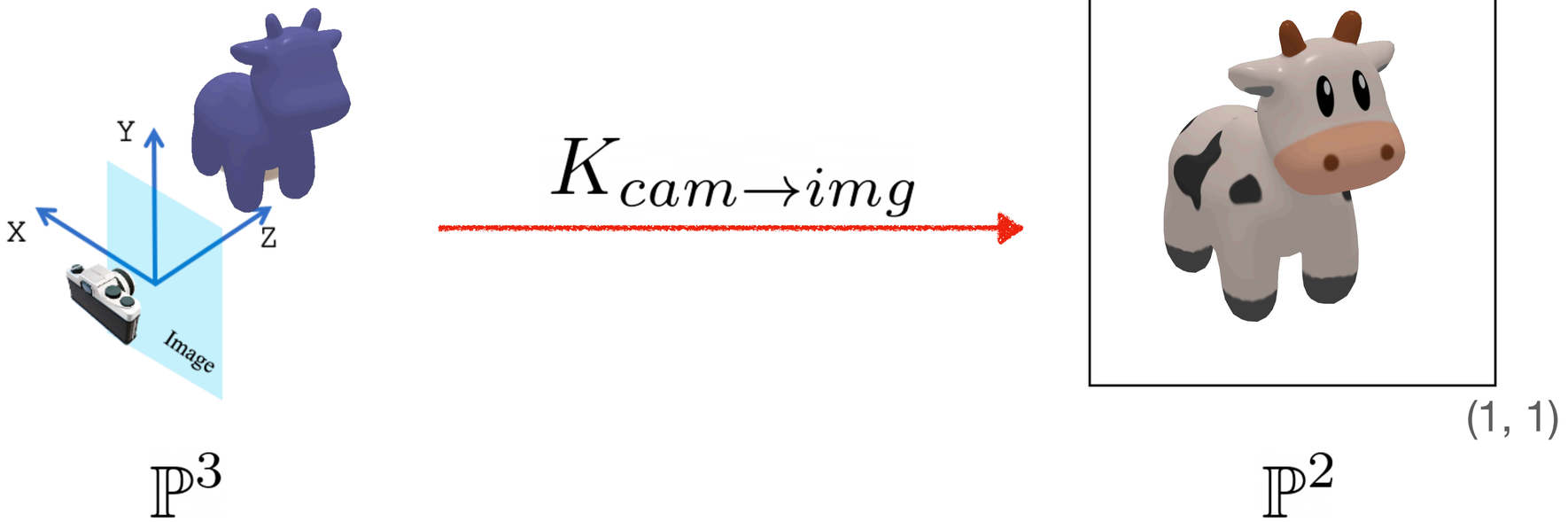


$$\begin{bmatrix} \frac{1}{2\tan(\Theta)} & 0 & 0.5 & 0 \\ 0 & \frac{1}{2\tan(\Theta)} & 0.5 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



Slide credit: CMU 16-889: Learning for 3D Vision, Prof. Shubham Tulsiani

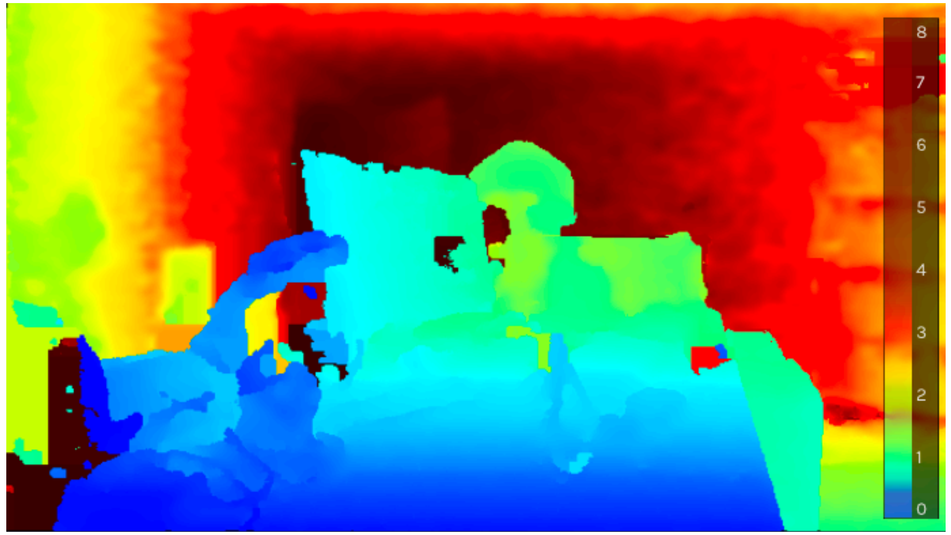
In practice: Decoupling Projection from Image Size

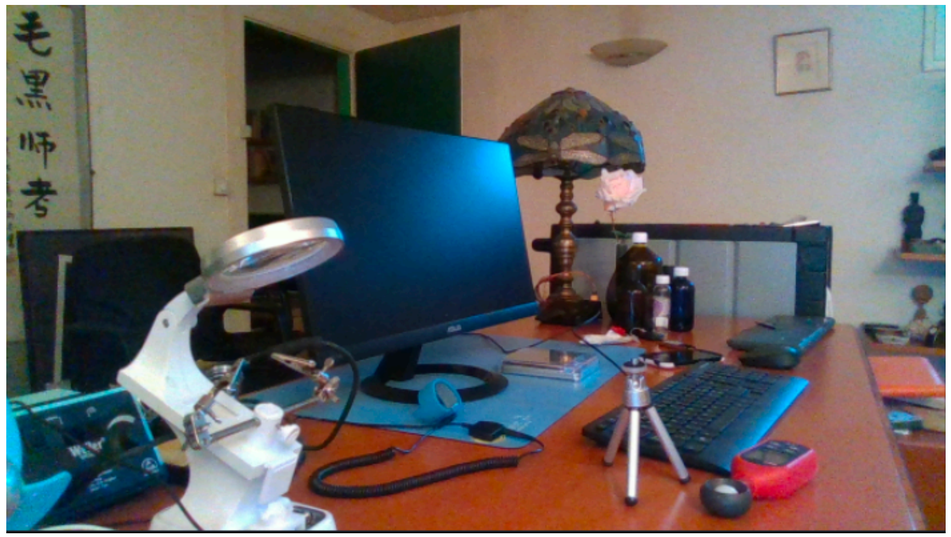


Helps (conceptually and in implementation) to reason in normalized image coordinates

Slide credit: CMU 16-889: Learning for 3D Vision, Prof. Shubham Tulsiani

“Unproject”: Go from pixel and depth to 3D point





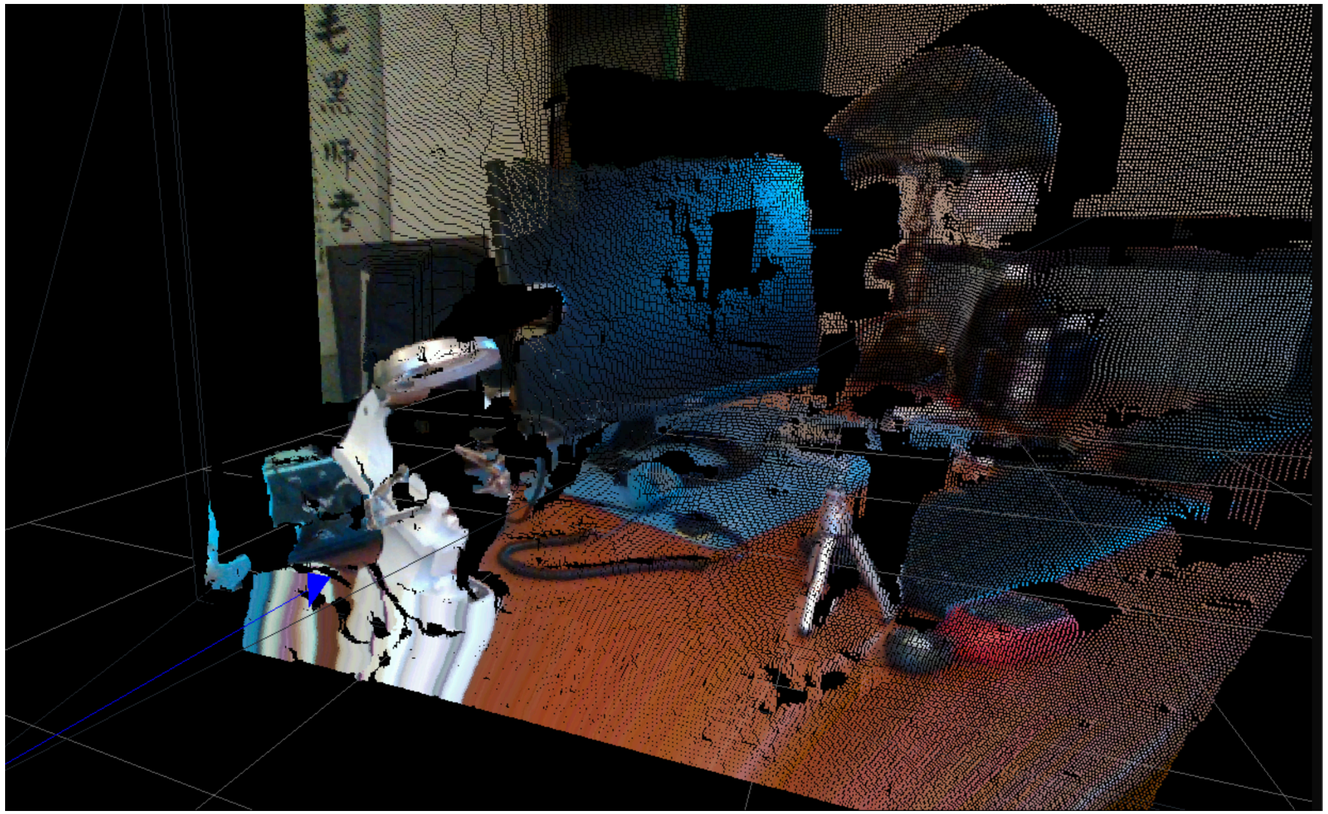


Image source: https://www.calvert.ch/maurice/2018/06/12/intel-realsense-d435-review/

“Unproject”: Go from pixel and depth to 3D point

$$\begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

“Unproject”: Go from pixel and depth to 3D point

$$\begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{K}^{-1} \begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix}$$

“Unproject”: Go from pixel and depth to 3D point

$$\begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{K}^{-1} \begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} \frac{f \cdot X}{Z} + p_x \\ \frac{f \cdot Y}{Z} + p_y \\ 1 \end{pmatrix}$$

“Unproject”: Go from pixel and depth to 3D point

$$\begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{K}^{-1} \begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} \frac{f \cdot X}{Z} + p_x \\ \frac{f \cdot Y}{Z} + p_y \\ 1 \end{pmatrix}$$

$$\mathbf{X} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix}$$

“Unproject”: Go from pixel and depth to 3D point

$$\mathbf{X} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix}$$

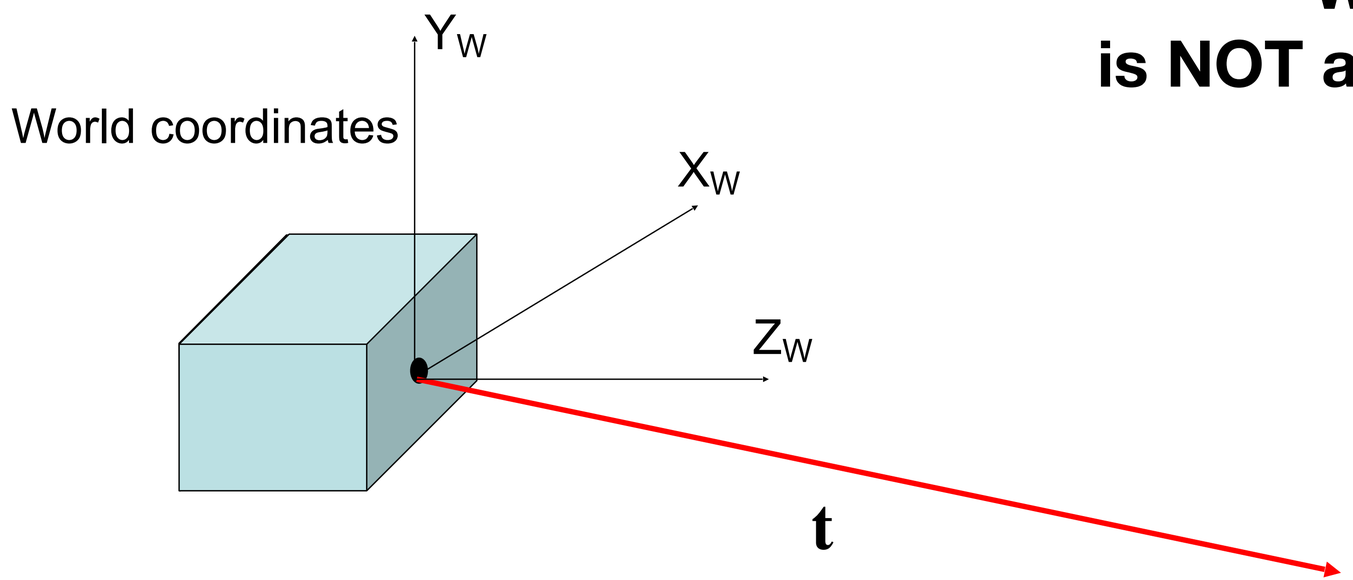
“Unproject”: Go from pixel and depth to 3D point

$$\mathbf{X} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix}$$

$$\texttt{unproject}(\mathbf{x}, Z) = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix}$$

Questions?

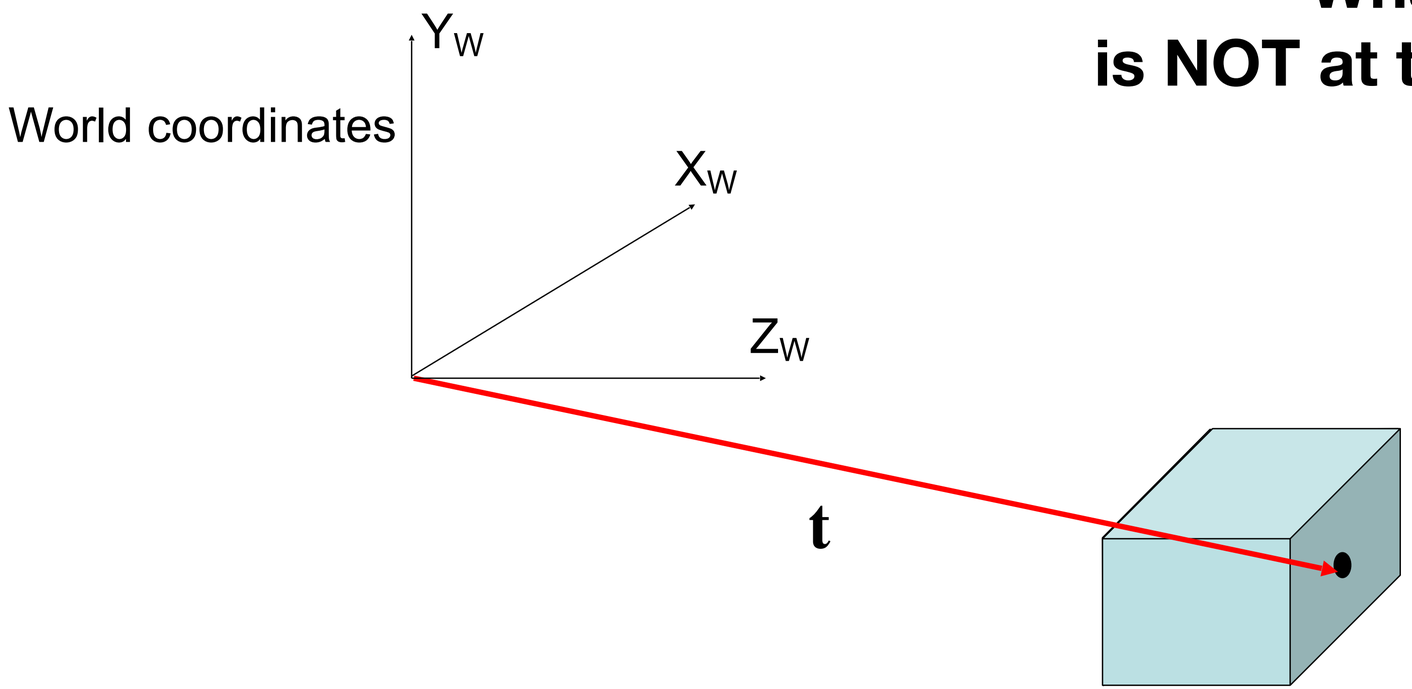
Camera parameters



What if the camera origin is NOT at the world coordinates origin?

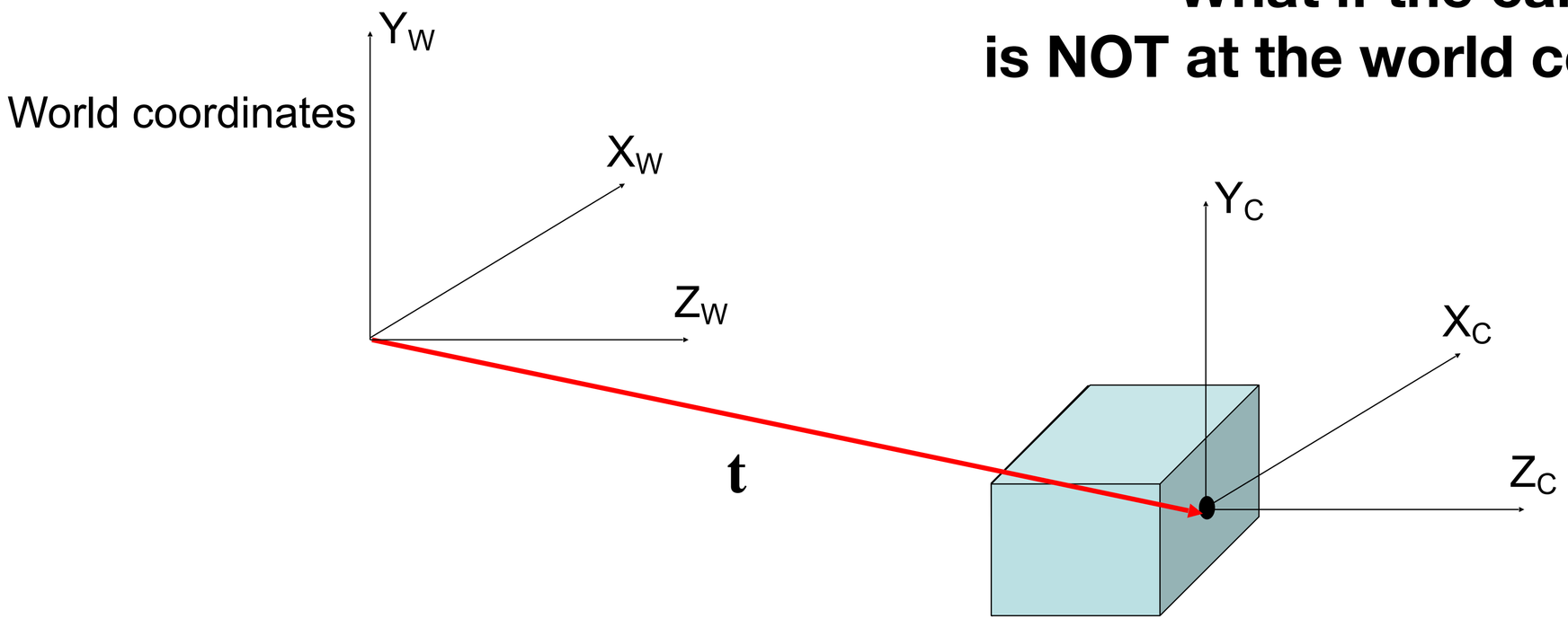
Camera parameters

What if the camera origin is NOT at the world coordinates origin?



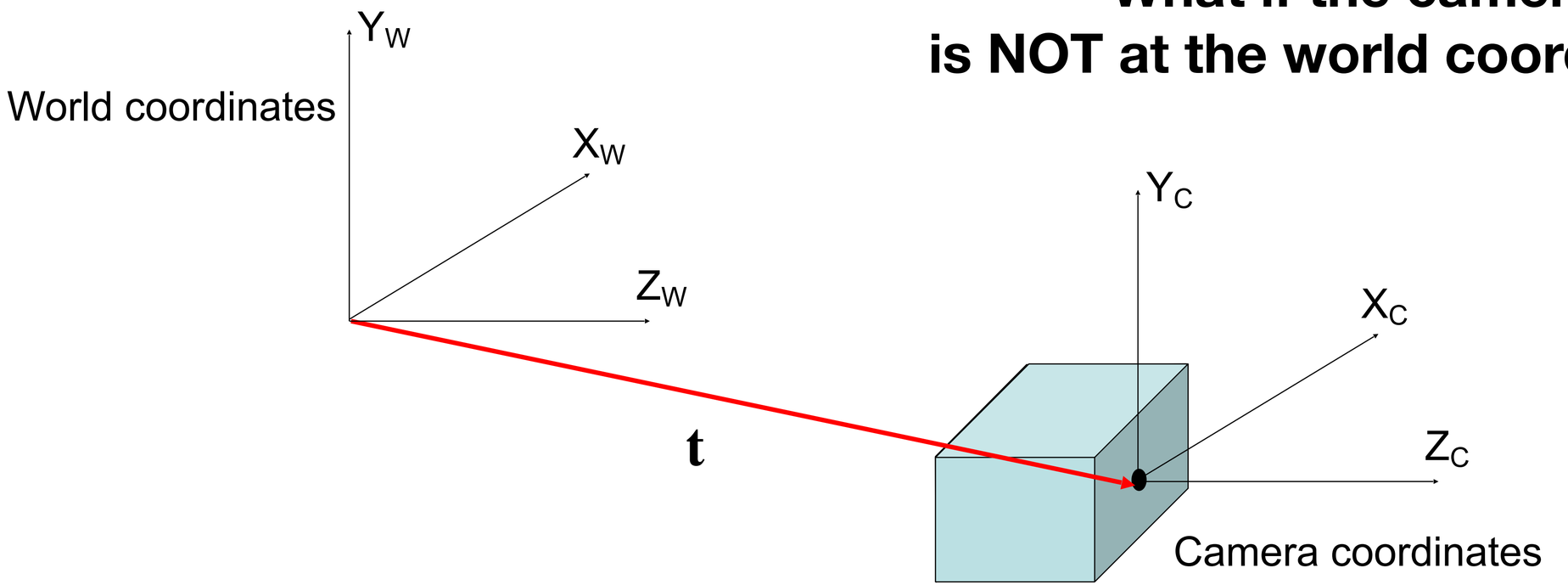
Camera parameters

What if the camera origin is NOT at the world coordinates origin?

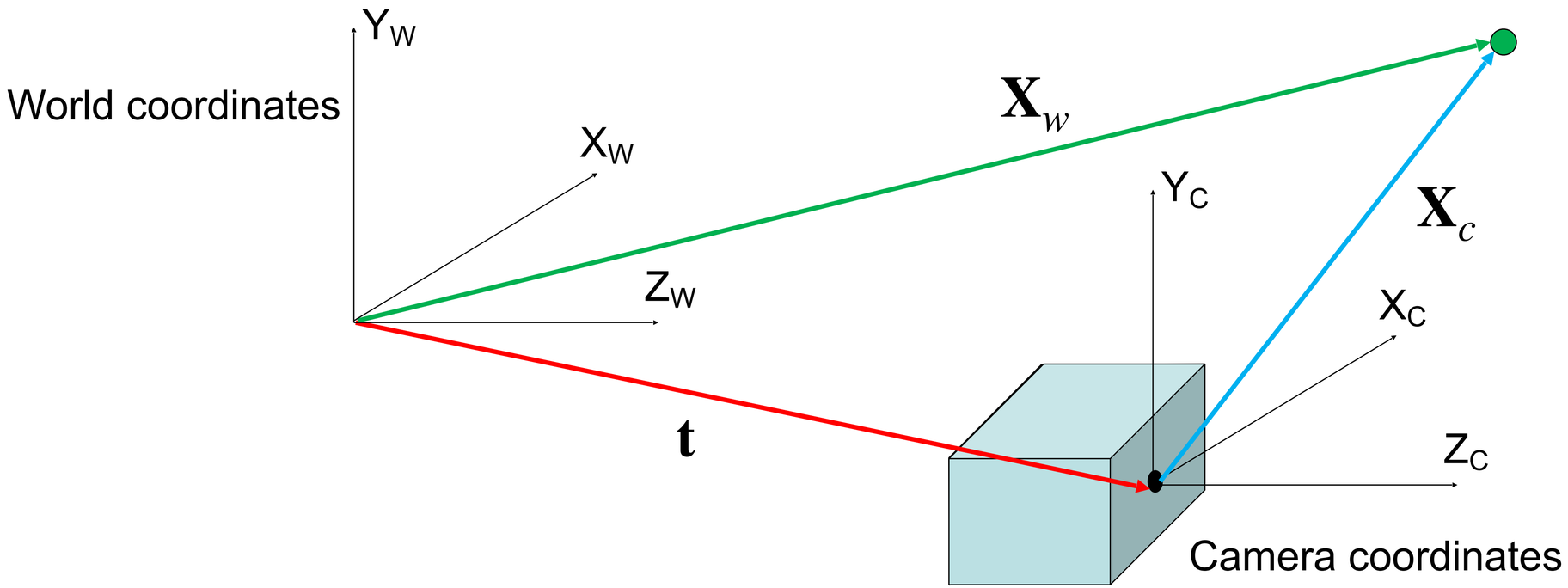


Camera parameters

What if the camera origin is NOT at the world coordinates origin?



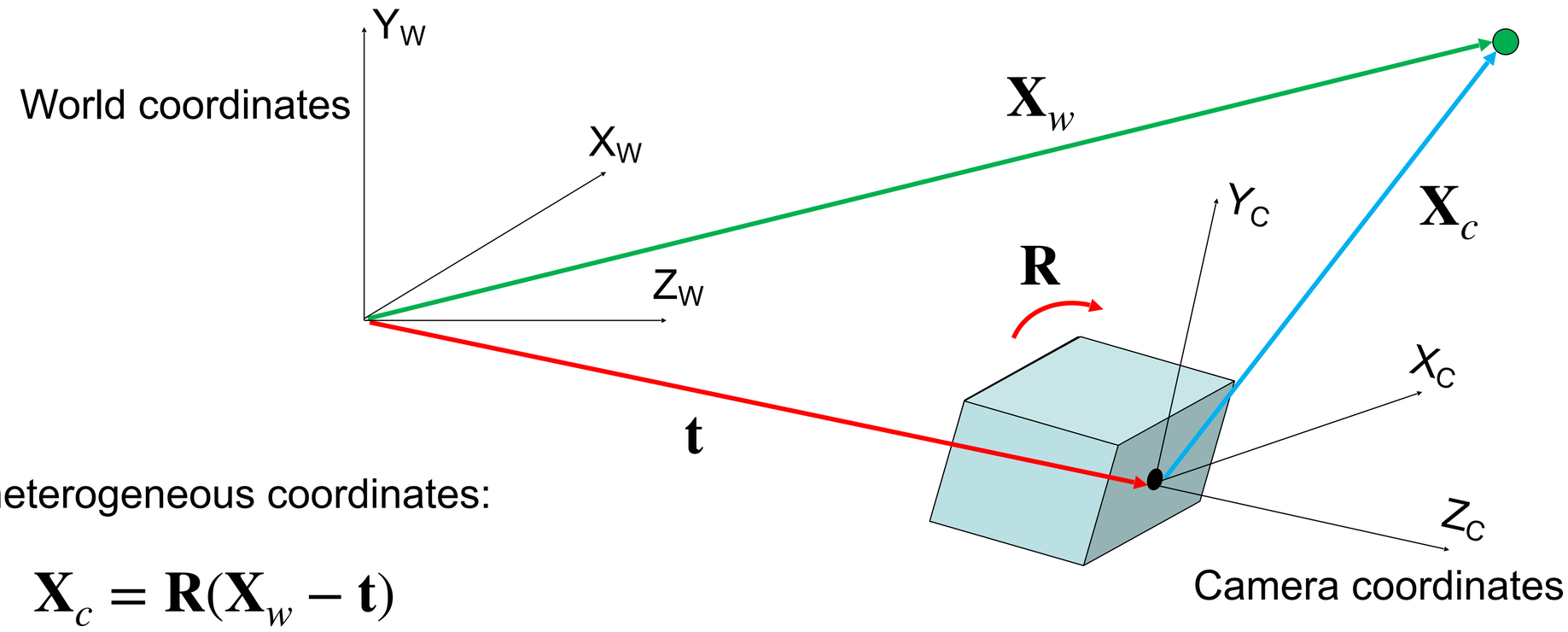
Camera parameters



In heterogeneous coordinates:

$$\mathbf{X}_c = \mathbf{X}_w - \mathbf{t}$$

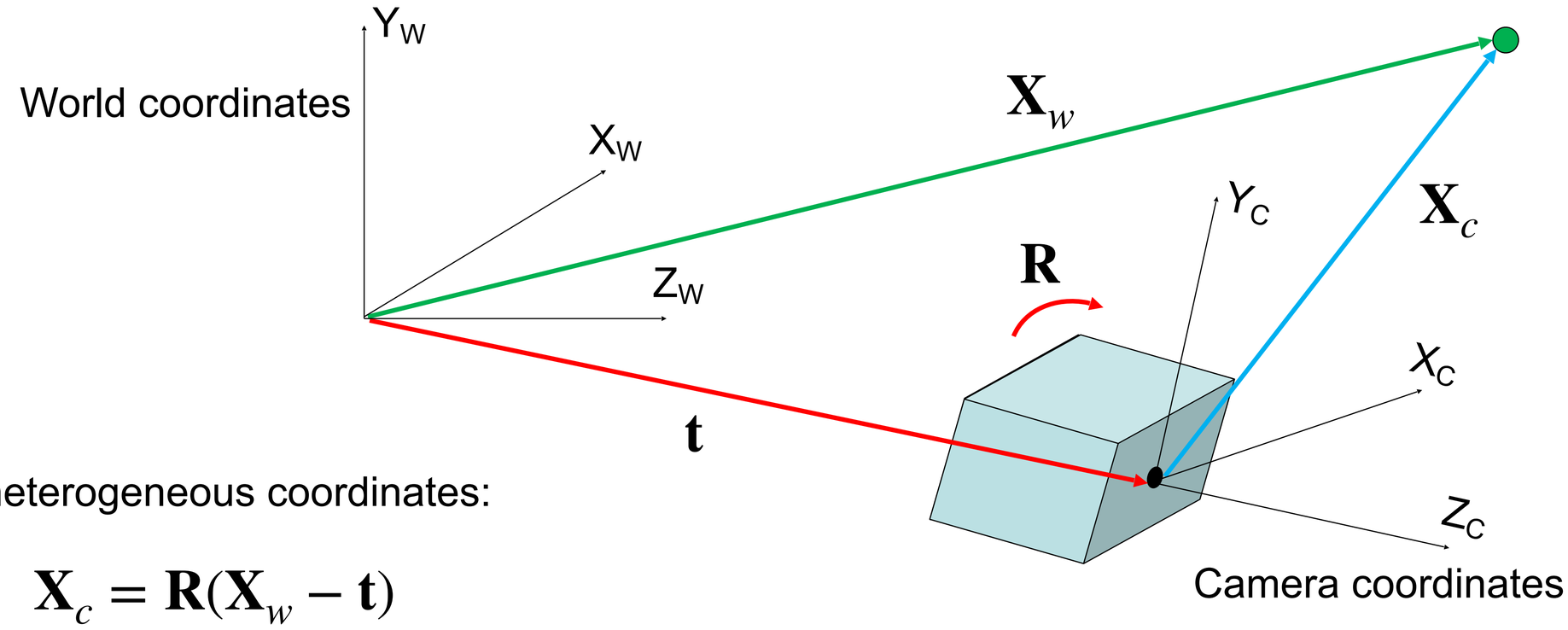
Camera parameters: rotation R



In heterogeneous coordinates:

$$\mathbf{X}_c = \mathbf{R}(\mathbf{X}_w - \mathbf{t})$$

Camera parameters



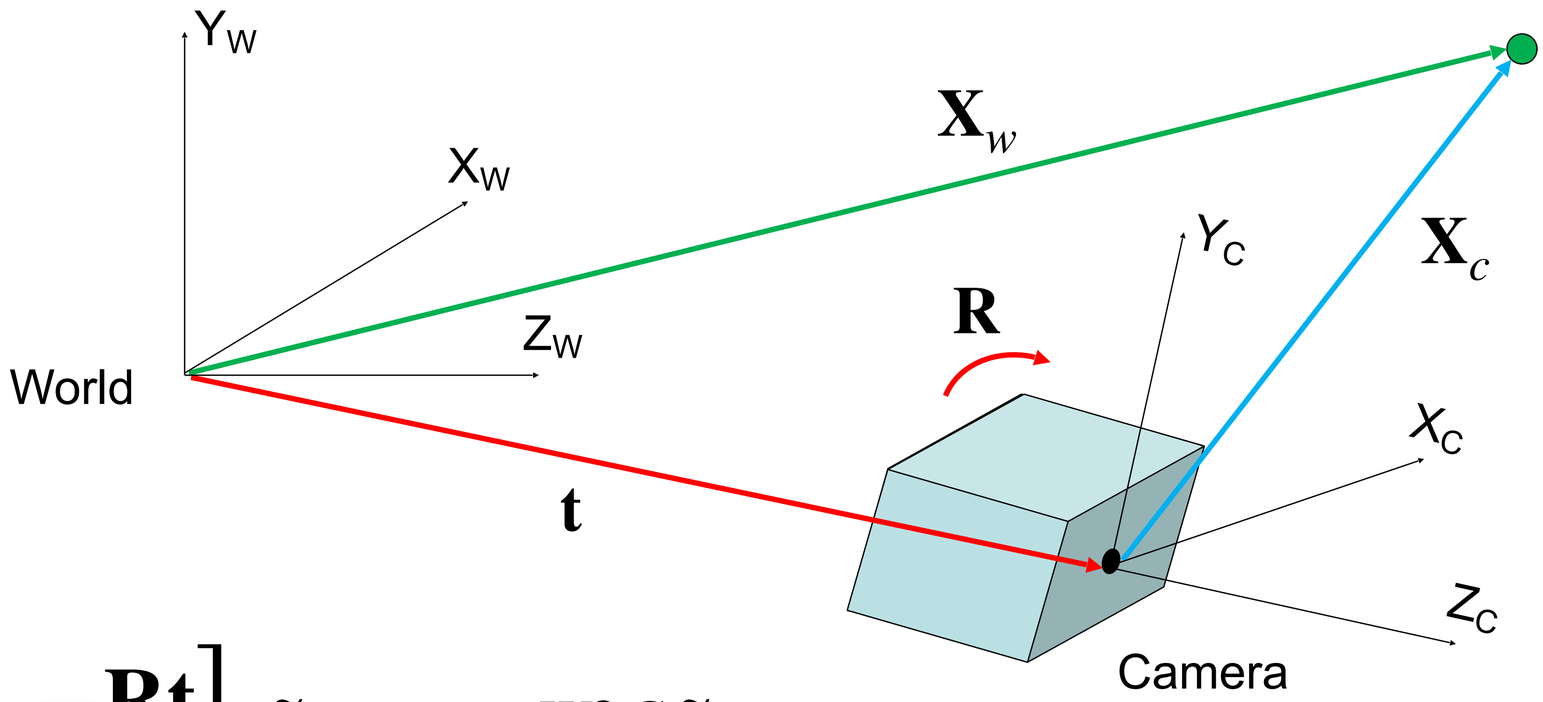
In heterogeneous coordinates:

$$\mathbf{X}_c = \mathbf{R}(\mathbf{X}_w - \mathbf{t})$$

In homogeneous coordinates:

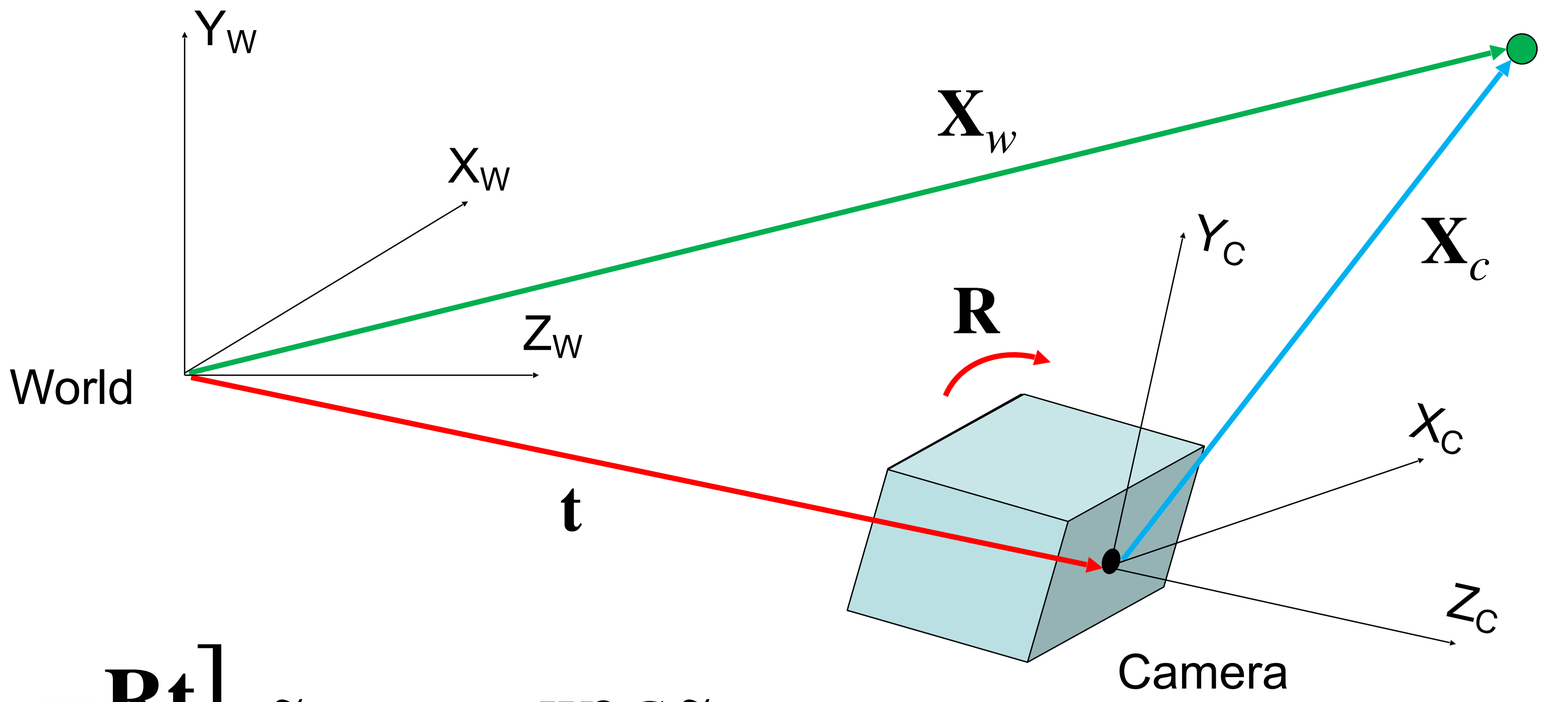
$$\tilde{\mathbf{X}}_c = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_w$$

Cam2World vs. World2Cam extrinsic parameters



$$\tilde{\mathbf{X}}_c = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_w = \mathbf{C}^{W2C}\tilde{\mathbf{X}}_w$$

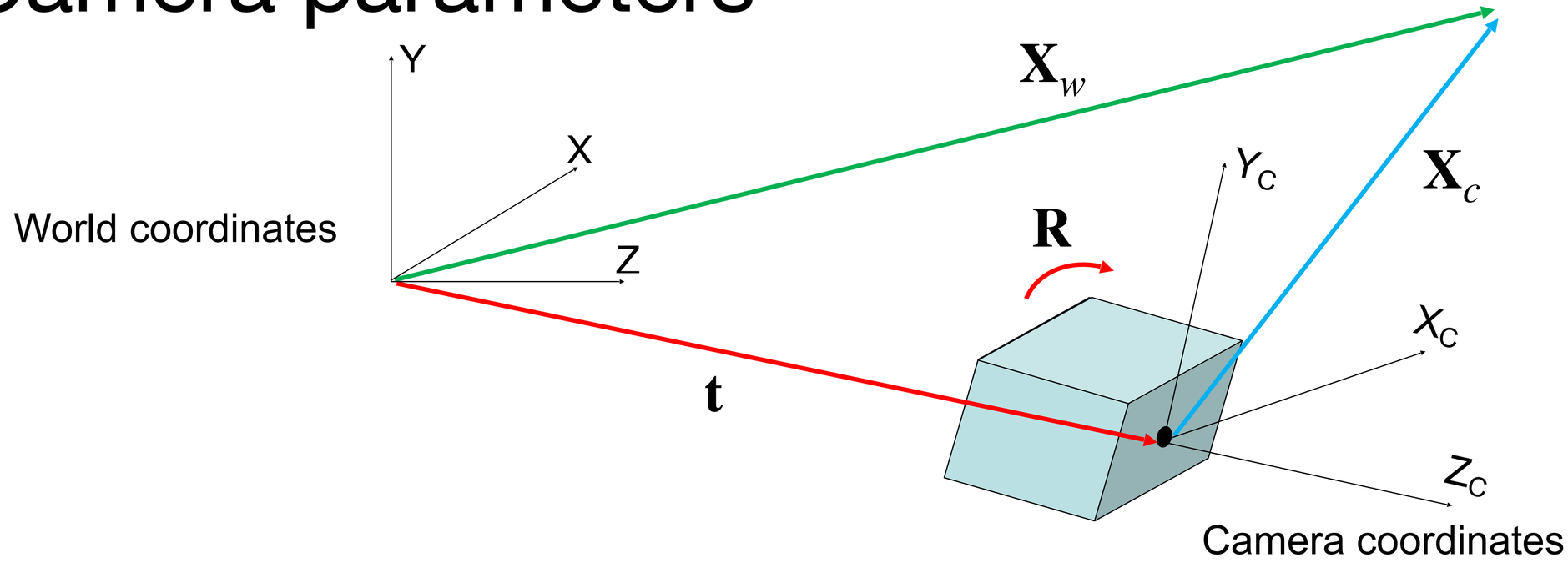
Cam2World vs. World2Cam extrinsic parameters



$$\tilde{\mathbf{X}}_c = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_w = \mathbf{C}^{W2C}\tilde{\mathbf{X}}_w$$

$$\tilde{\mathbf{X}}_w = \begin{bmatrix} \mathbf{R}^T & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_c = (\mathbf{C}^{W2C})^{-1}\tilde{\mathbf{X}}_c = \mathbf{C}^{C2W}\tilde{\mathbf{X}}_c$$

Camera parameters



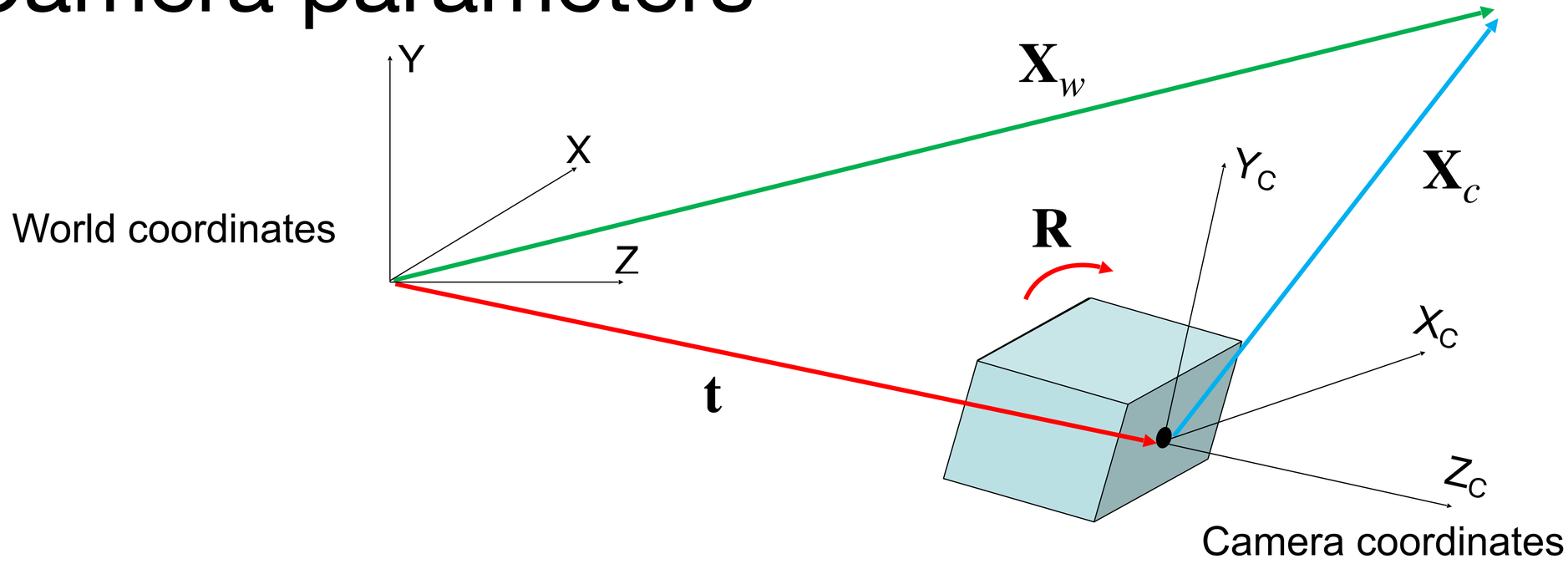
World coordinates to camera coordinates

$$\tilde{\mathbf{X}}_c = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_w$$

Camera coordinates to image coordinates

$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{X}}_c$$

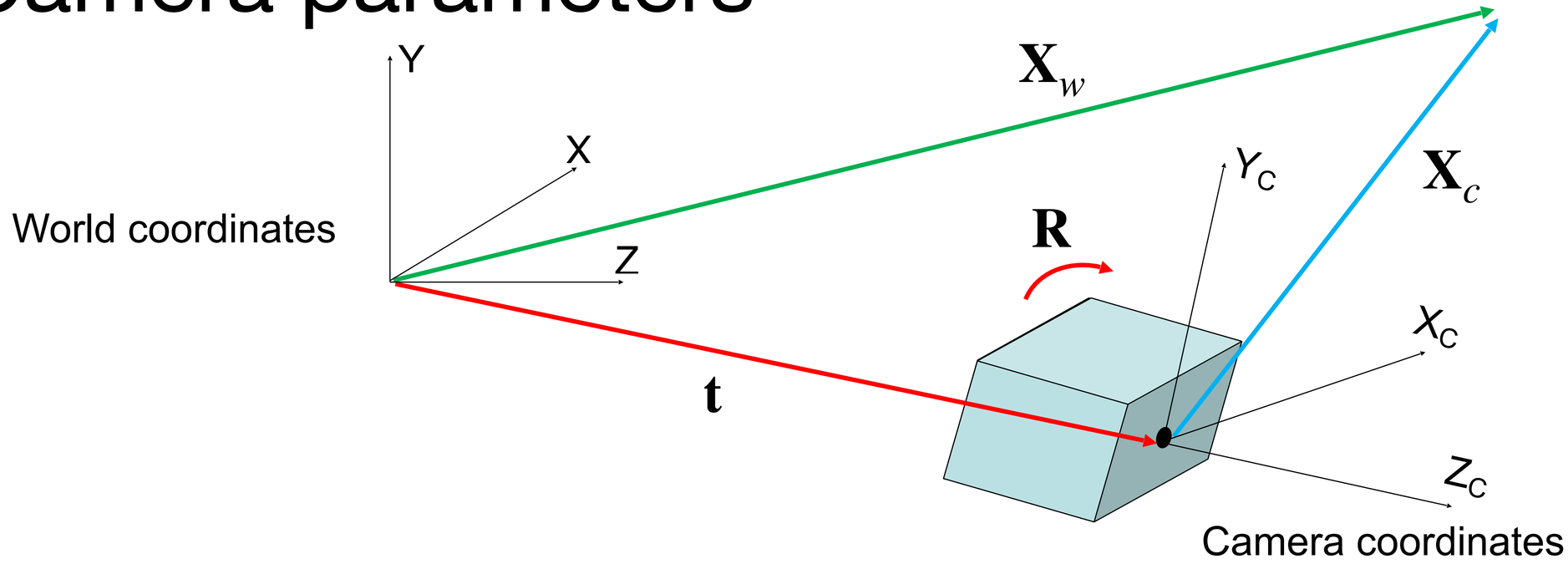
Camera parameters



World coordinates to image coordinates

$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & -\mathbf{Rt} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_w$$

Camera parameters



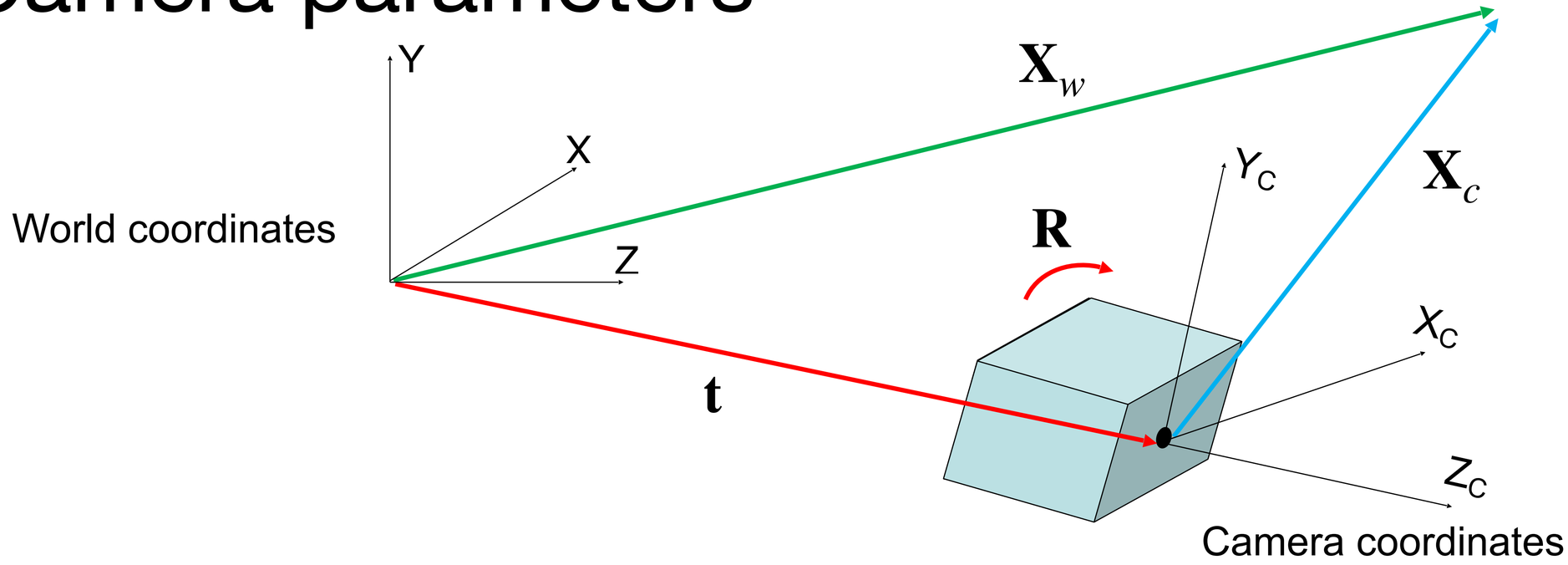
World coordinates to image coordinates

$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & -\mathbf{Rt} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_w = \mathbf{P}\tilde{\mathbf{X}}_w$$

Intrinsic parameters

Extrinsic parameters

Camera parameters



World coordinates to image coordinates

$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & -\mathbf{Rt} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_w = \mathbf{P}\tilde{\mathbf{X}}_w$$

Properties of the camera+lens that don't change (unless you zoom/crop)

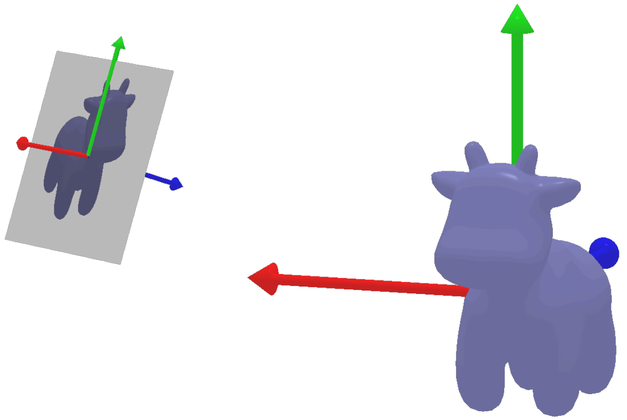
Intrinsic parameters

Extrinsic parameters

Parameters that vary when you move the camera around

World2Camera Transformations: Exercise

$$X_c = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} X_w; \quad \mathbf{t} = -R\tilde{C}$$



What are R, t for an upright (world and camera y-directions align) origin-facing camera 2m away from origin located at (0,0,-2)?

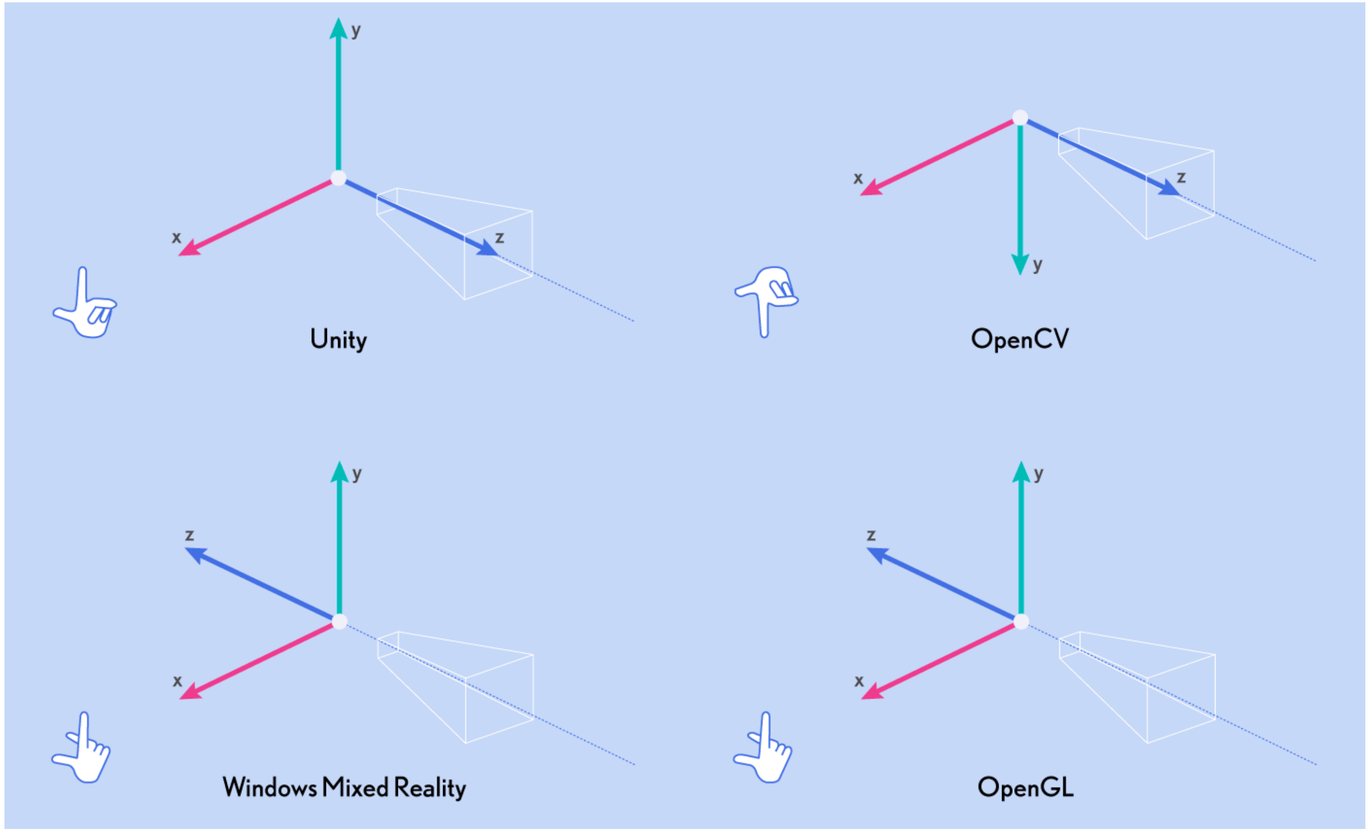
$$R = \mathbf{I}; \quad \mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

What are R, t for an upright (world and camera y-directions align) origin-facing camera 2m away from origin located at (-2,0,0)?

$$R = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}; \quad \mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

Slide credit: CMU 16-889: Learning for 3D Vision, Prof. Shubham Tulsiani

Beware of Conventions



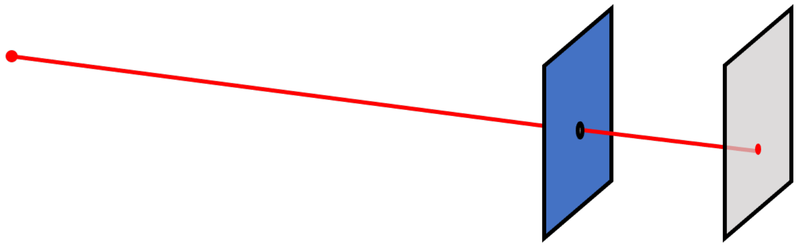
X=left or right? Y=up or down? etc..

If you externally obtain transformation matrices (e.g. using someone else's code), make sure of convention compatibility (one of the *most common* sources of bugs!)

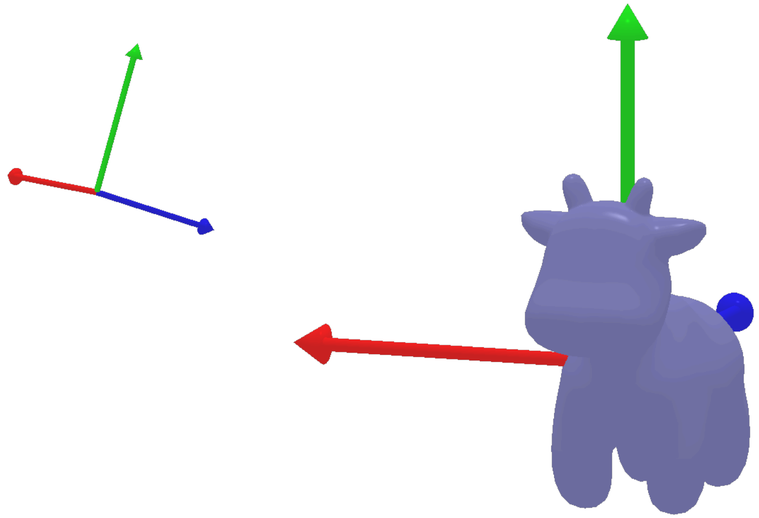
read https://pytorch3d.org/docs/cameras

Slide credit: Shubham Tulsiani. Image courtesy: Christoph Krautz

Summary



Projection — associating rays to points in a plane



Camera Transformation — from world to camera coordinates

Slide credit: CMU 16-889: Learning for 3D Vision, Prof. Shubham Tulsiani

Questions?

Vision systems

One camera



Two cameras



N cameras



Let’s consider two eyes

One camera

Two cameras

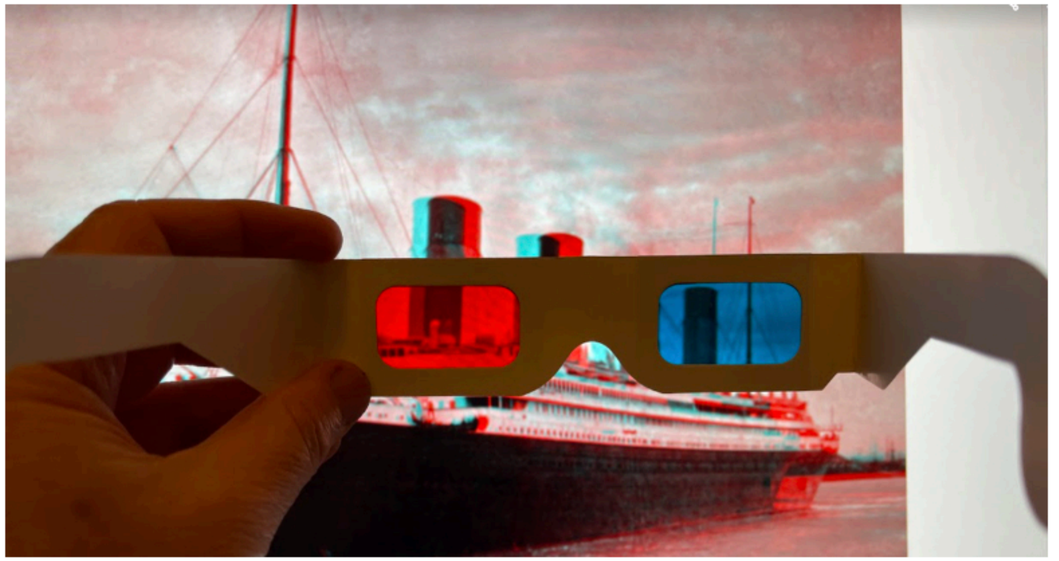


N cameras

Stereo images of the Titanic



(a)



(b)

Figure 1.1: (a) Stereo anaglyph of the ocean liner, the Titanic [McManus2022]. The red image shows the right eye's view, and cyan the left eye's view. When viewed through stereo red/cyan stereo glasses, as in (b), the cyan contrast appears in the left eye image and the red variations appear to the right eye, creating a the perception of 3d.

Stereoscope



View of Boston, c. 1860; an early stereoscopic card for viewing a scene from nature

More details

Soule, John P., 1827-1904 -- Photographer - This image is available from the New York Public Library's Digital Library under the digital ID G90F336_113F: digitalgallery.nypl.org → digitalcollections.nypl.org

Public Domain

File: Charles Street Mall, Boston Common, by Soule, John P., 1827-1904 3.jpg

Created: Coverage: 1860?-1890?. Source Imprint: 1860?-1890?. Digital item published 7-28-2005; updated 4-23-2009.



Brewster-type stereoscope, 1870

More details

Alessandro Nassiri - Museo della Scienza e della Tecnologia "Leonardo da Vinci"

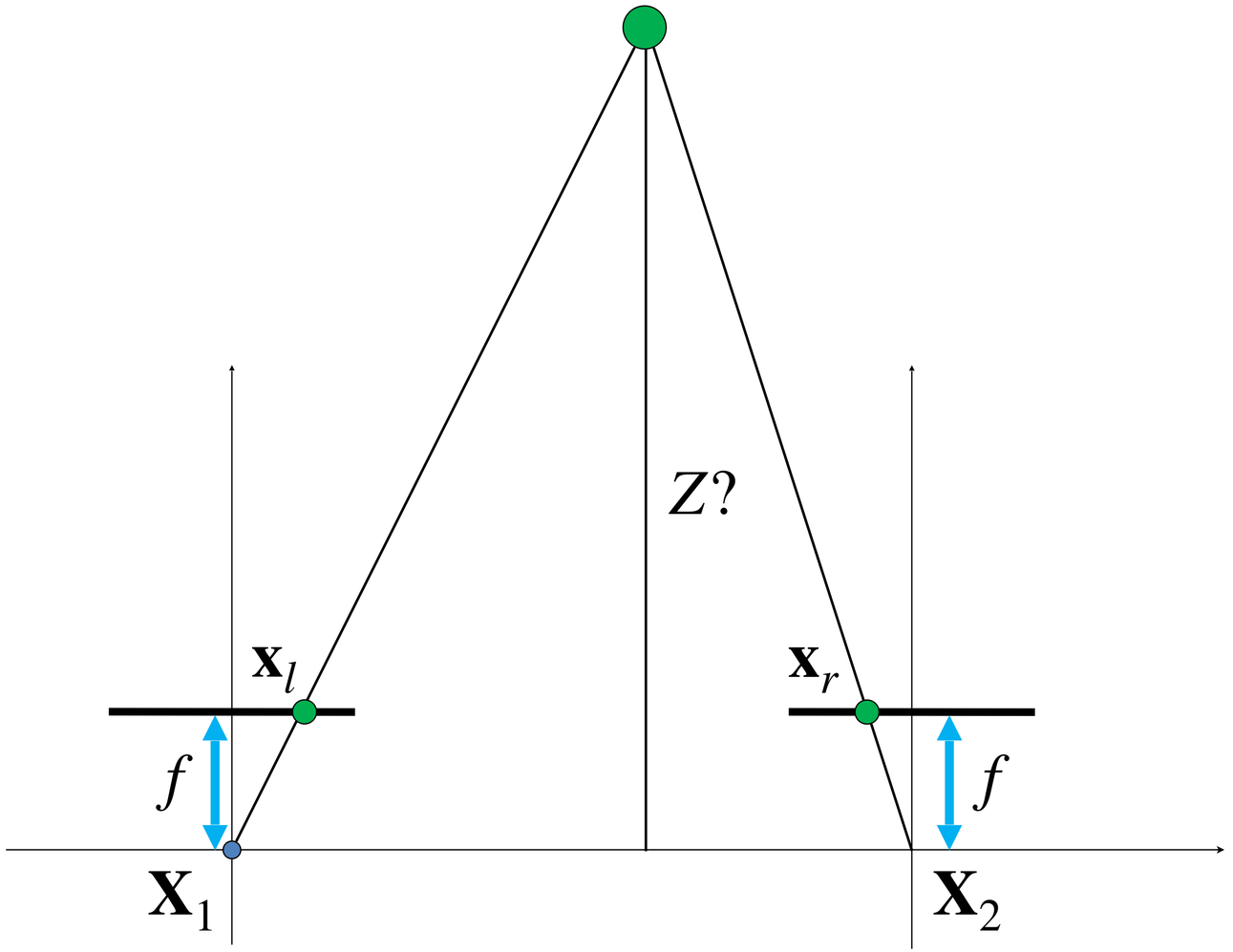
Visore stereoscopico portatile di tipo Brewster, J. Fleury - Hermagis, 1870, con messa a fuoco manuale. Per la visione di lastre e stampe stereoscopiche 8,5x17cm. Museo nazionale della scienza e della tecnologia Leonardo da Vinci, Milano.

CC BY-SA 4.0

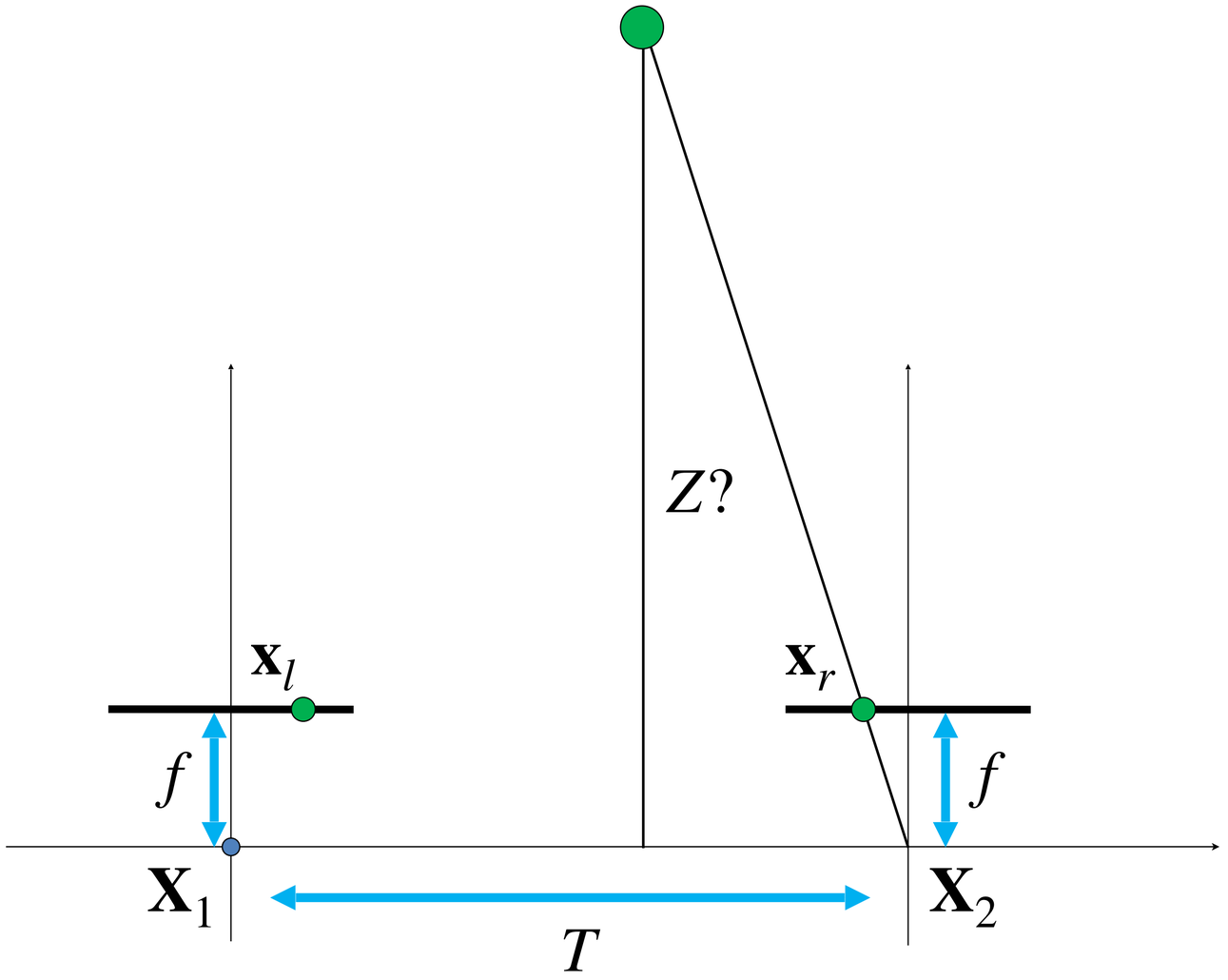
File: IGB 006055 Visore stereoscopico portatile Museo scienza e tecnologia Milano.jpg

Created: 1 July 2014

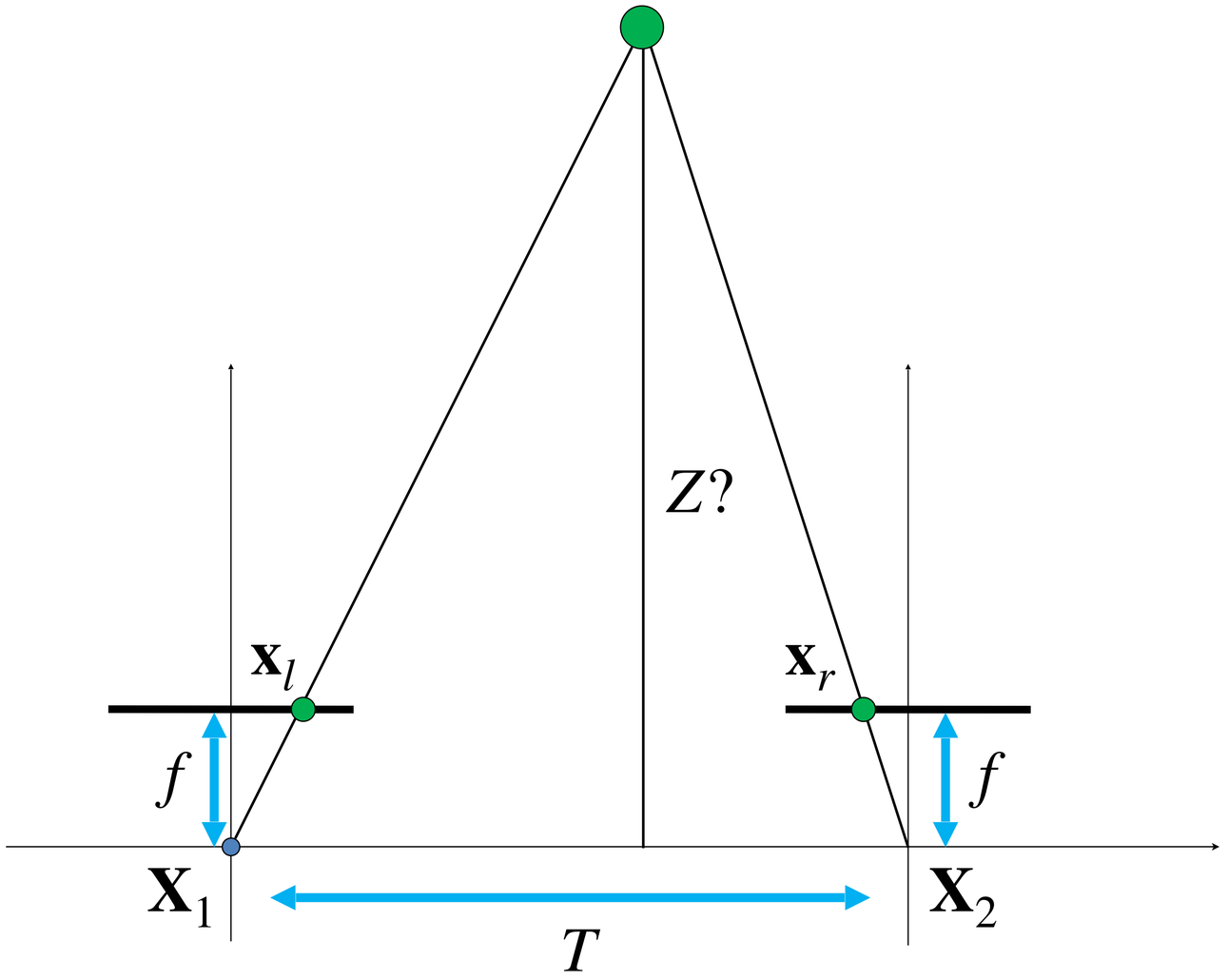
Geometry for a simple stereo system



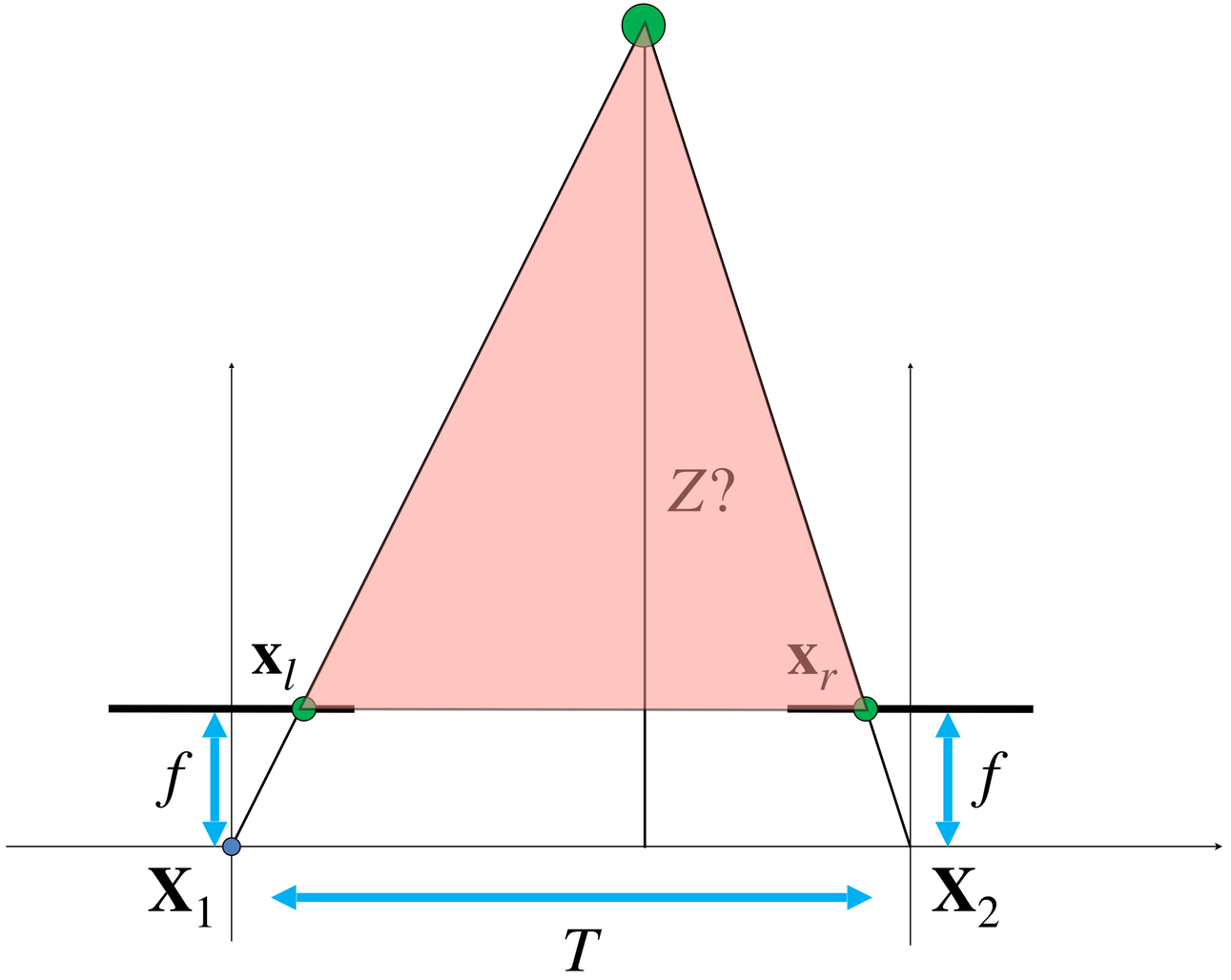
Geometry for a simple stereo system



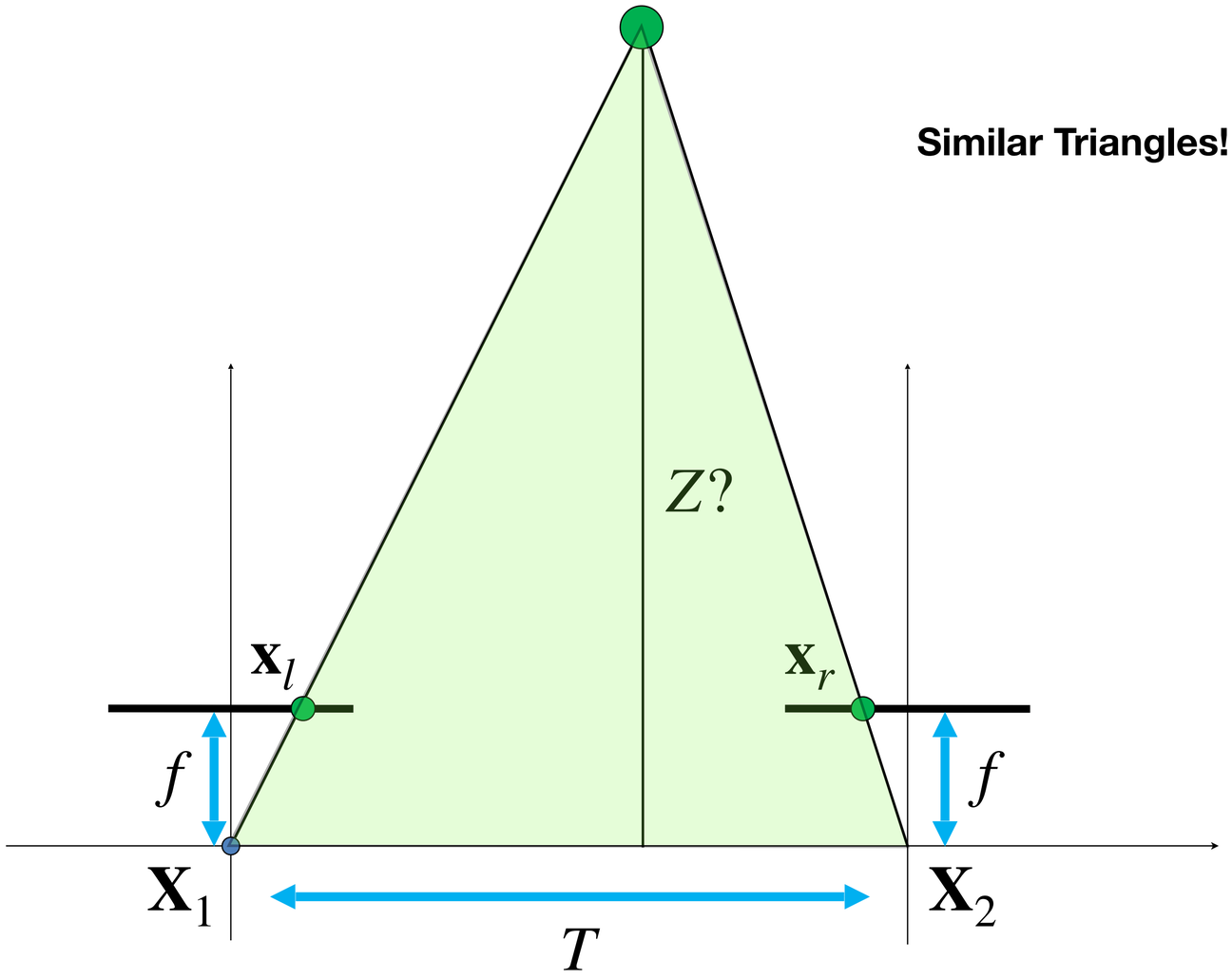
Geometry for a simple stereo system



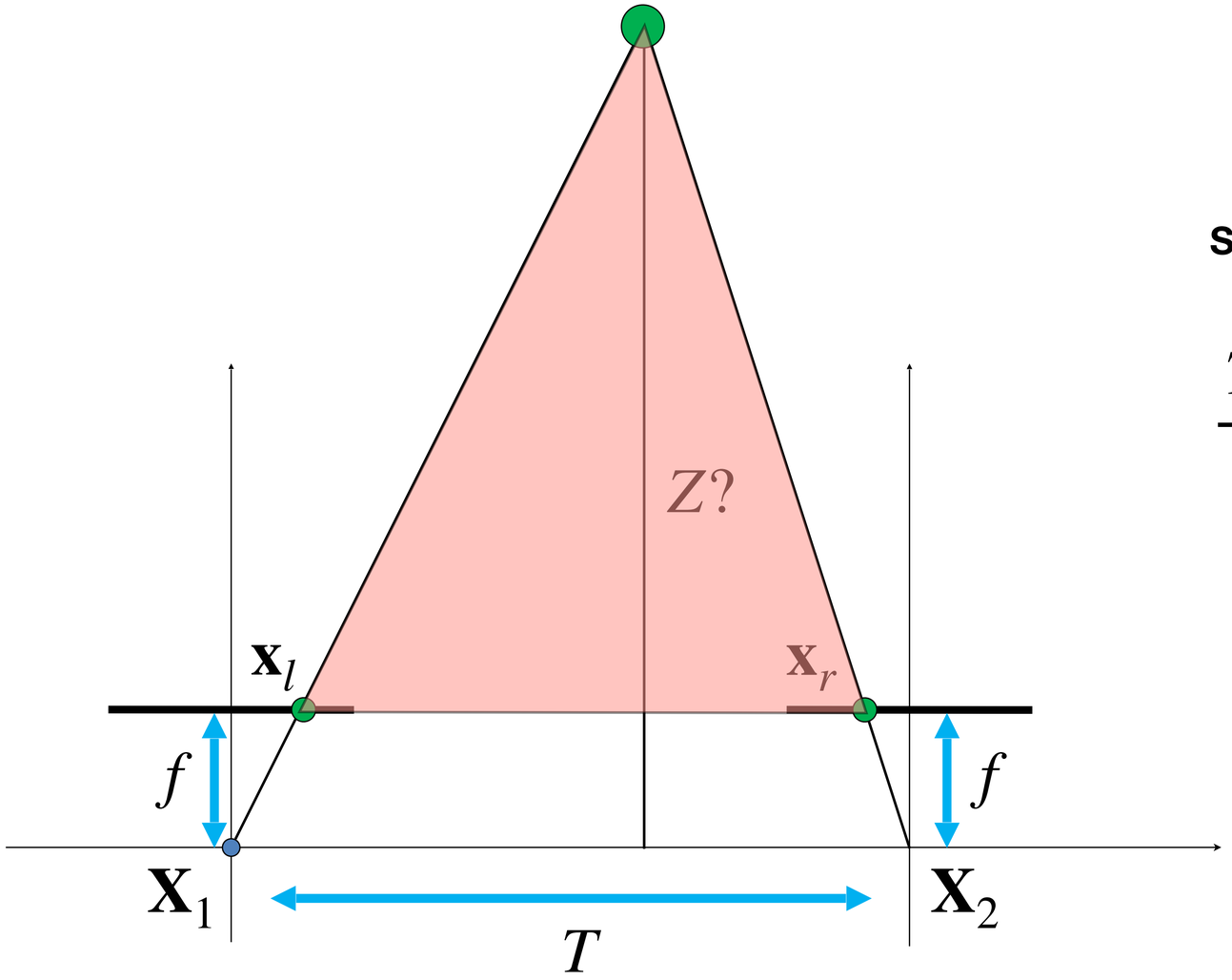
Geometry for a simple stereo system



Geometry for a simple stereo system



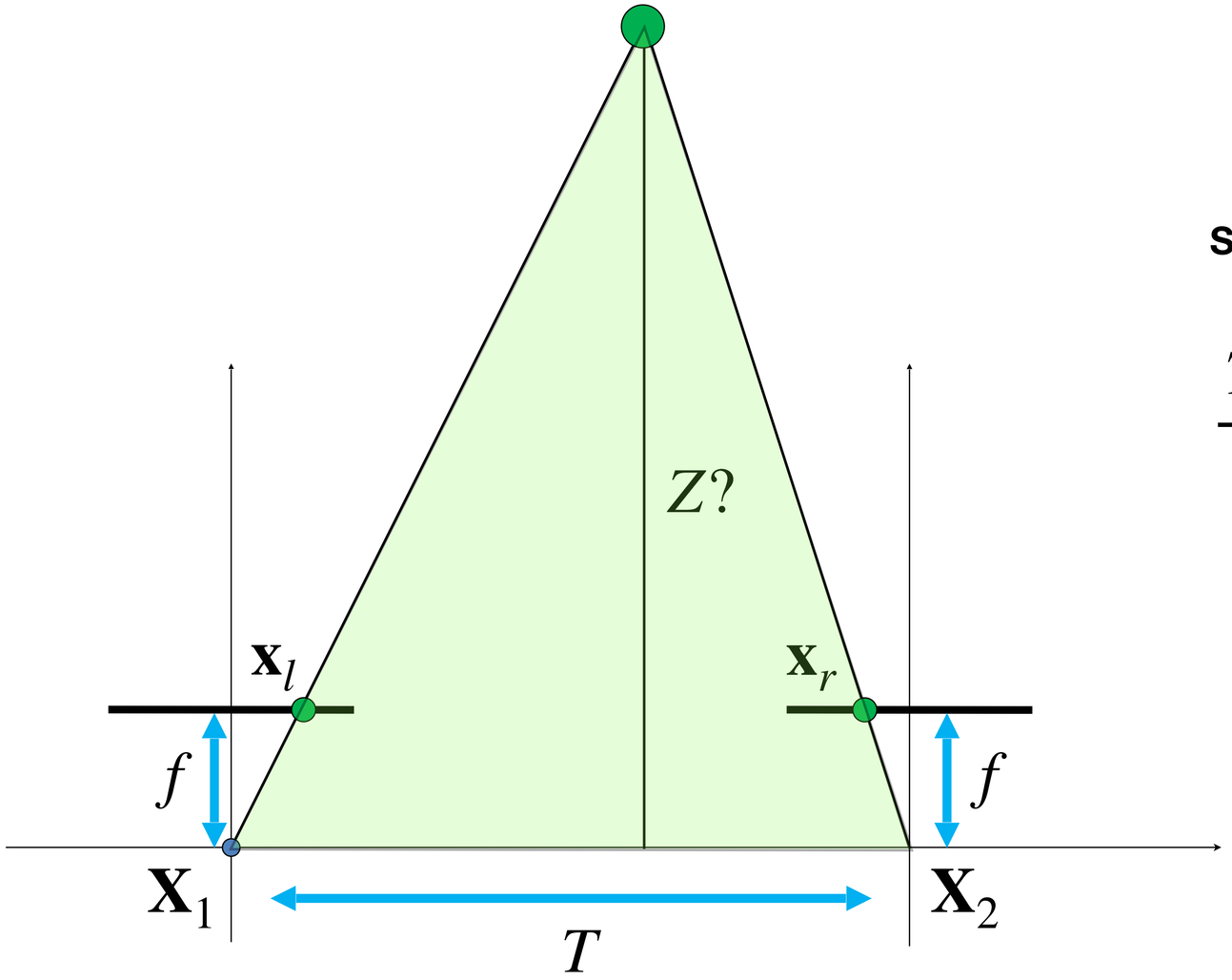
Geometry for a simple stereo system



Similar Triangles:

$$\frac{T + \mathbf{x}_r - \mathbf{x}_l}{Z - f} =$$

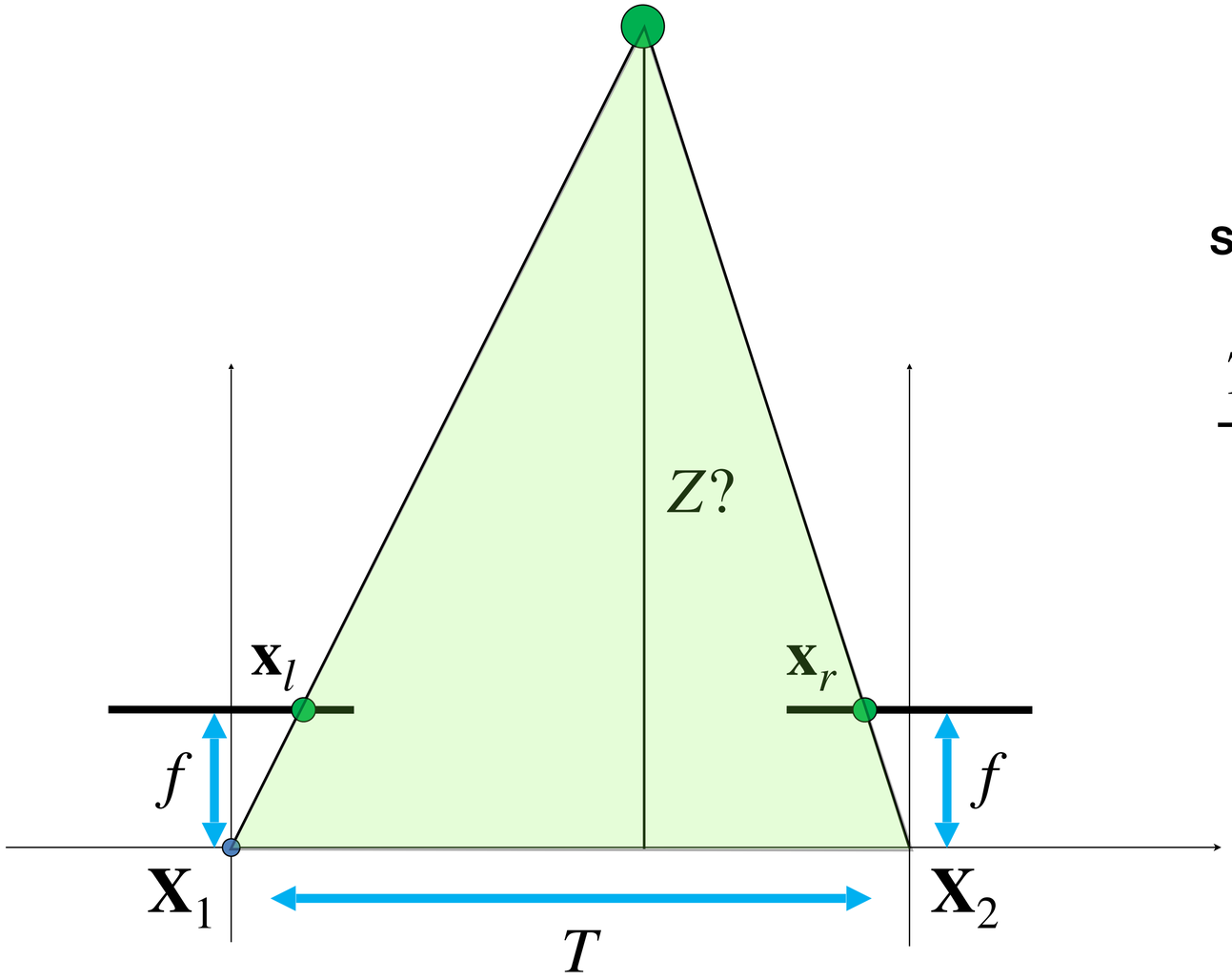
Geometry for a simple stereo system



Similar Triangles:

$$\frac{T + \mathbf{x}_r - \mathbf{x}_l}{Z - f} = \frac{T}{Z}$$

Geometry for a simple stereo system



Similar Triangles:

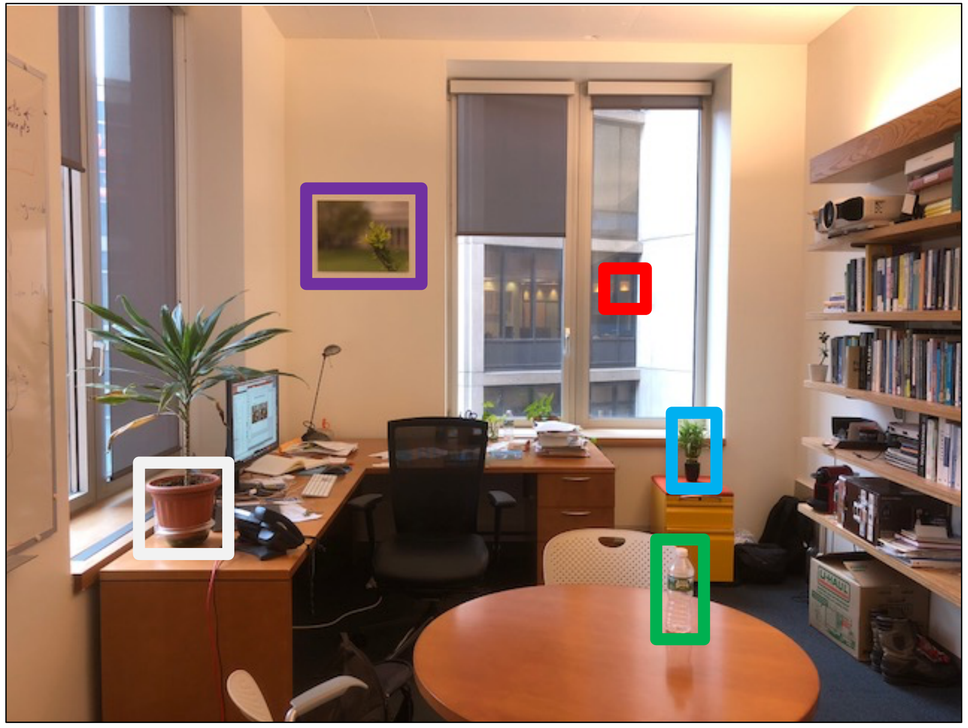
$$\frac{T + \mathbf{x}_r - \mathbf{x}_l}{Z - f} = \frac{T}{Z}$$

Solve for Z:

$$Z = f\frac{T}{\mathbf{x}_l - \mathbf{x}_r}$$

Disparity

Measuring disparity



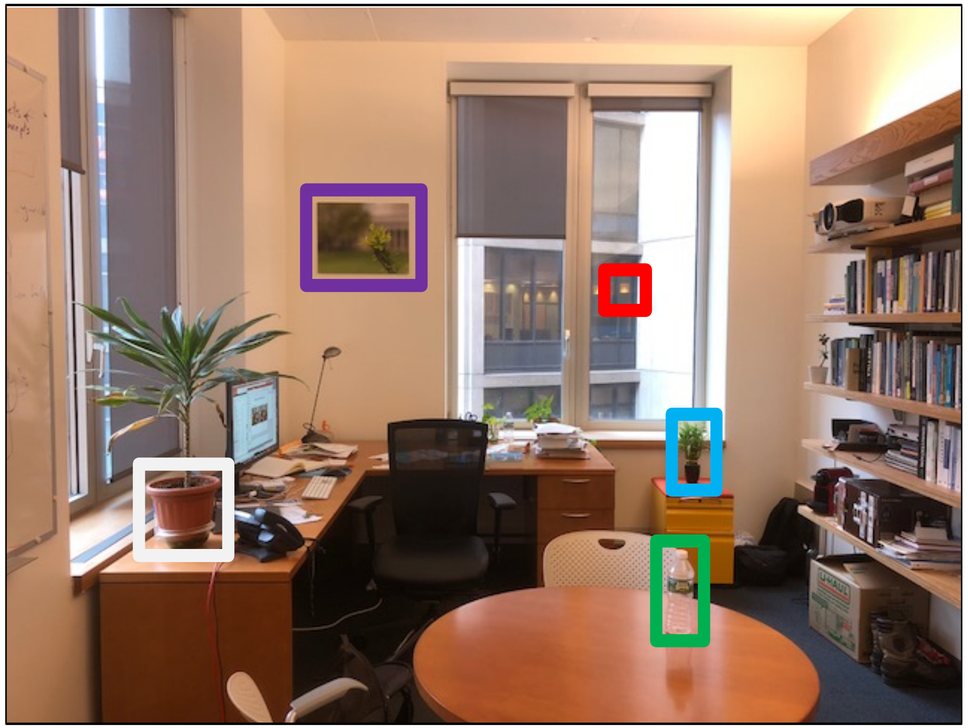
Left image



Right image

I took one picture, then I moved ~1m to the right and took a second picture.

Measuring disparity

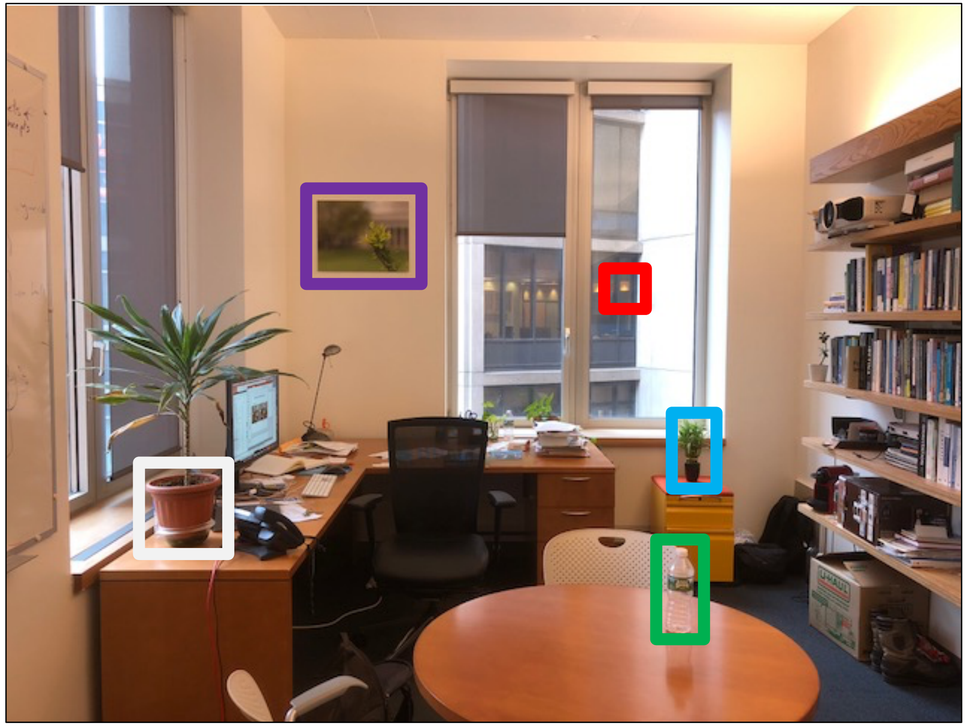


Left image

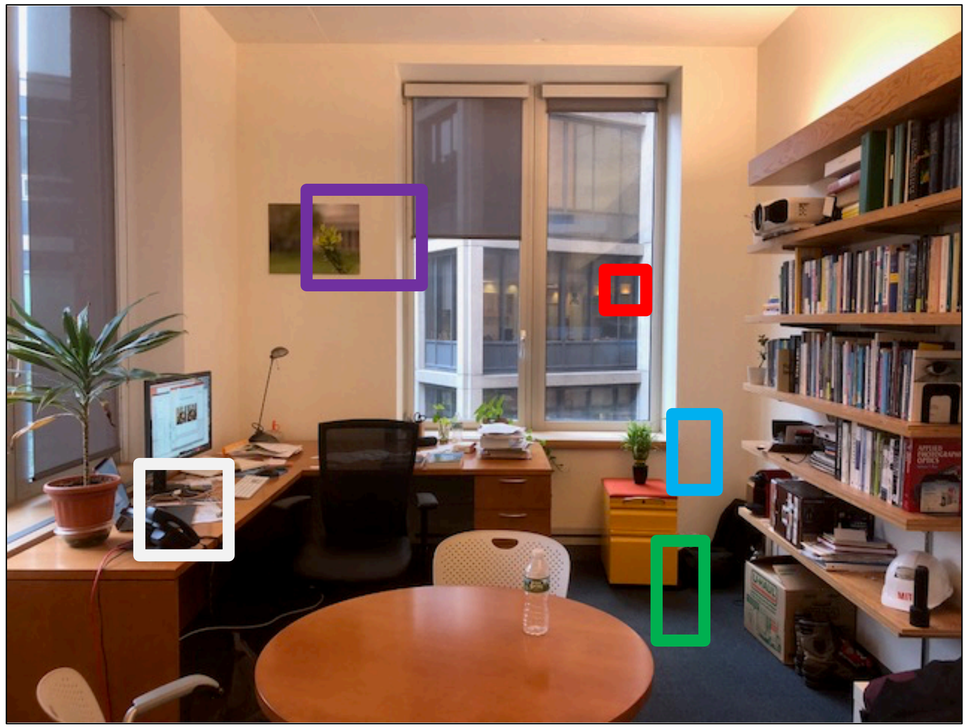


Right image

Measuring disparity

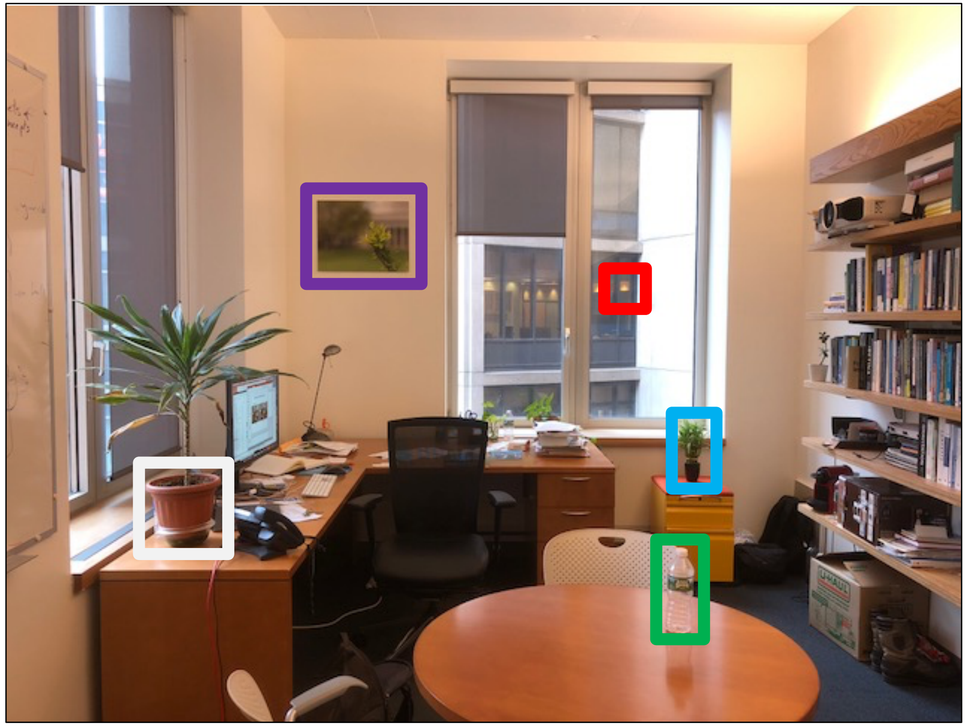


Left image

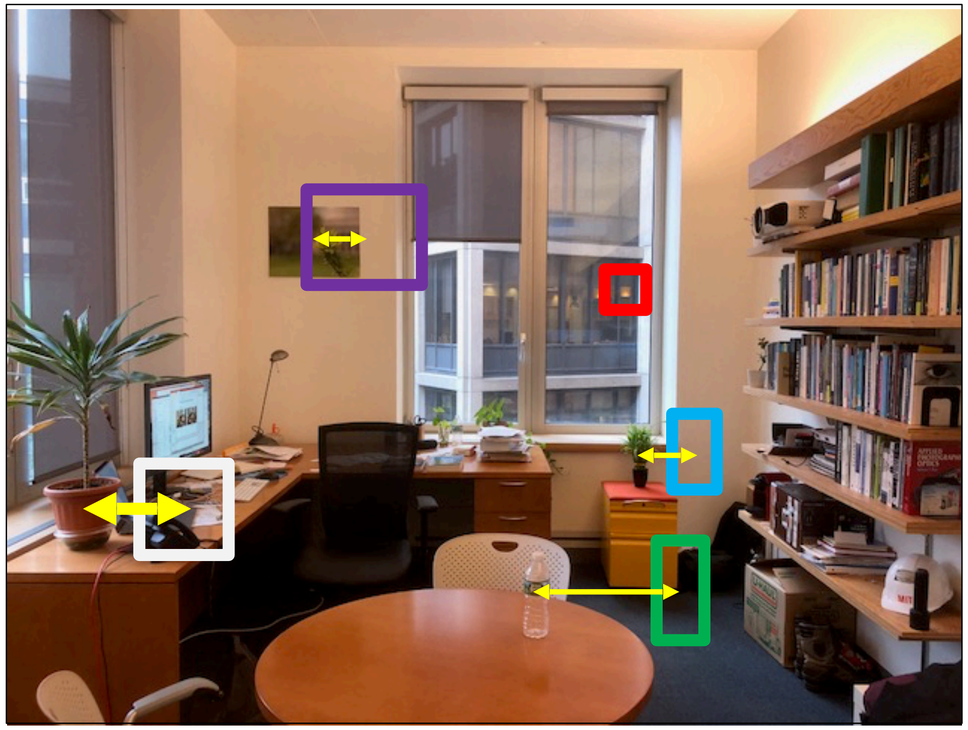


Right image

Measuring disparity



Left image



Right image

Disparity map

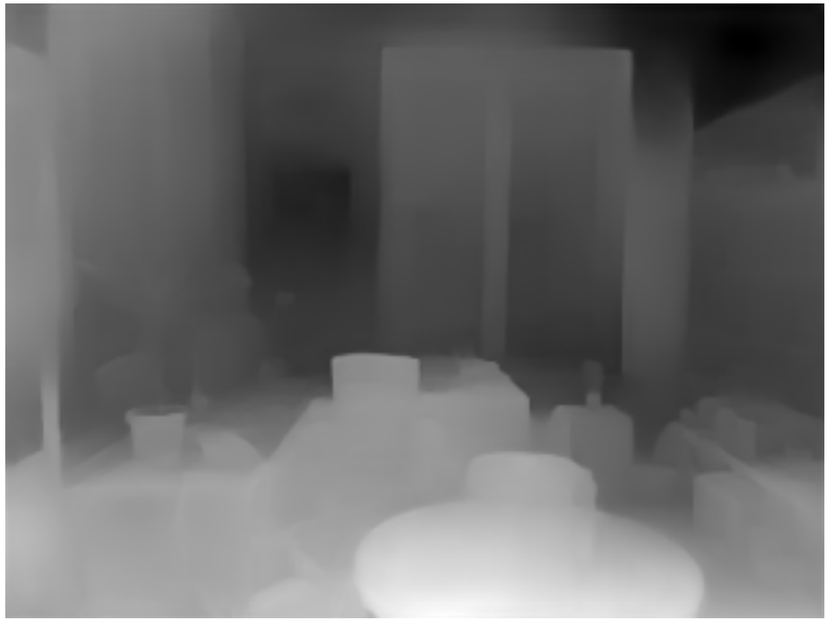
$I(x, y)$



$I'(x, y) = I(x + D(x, y), y)$



$D(x, y)$

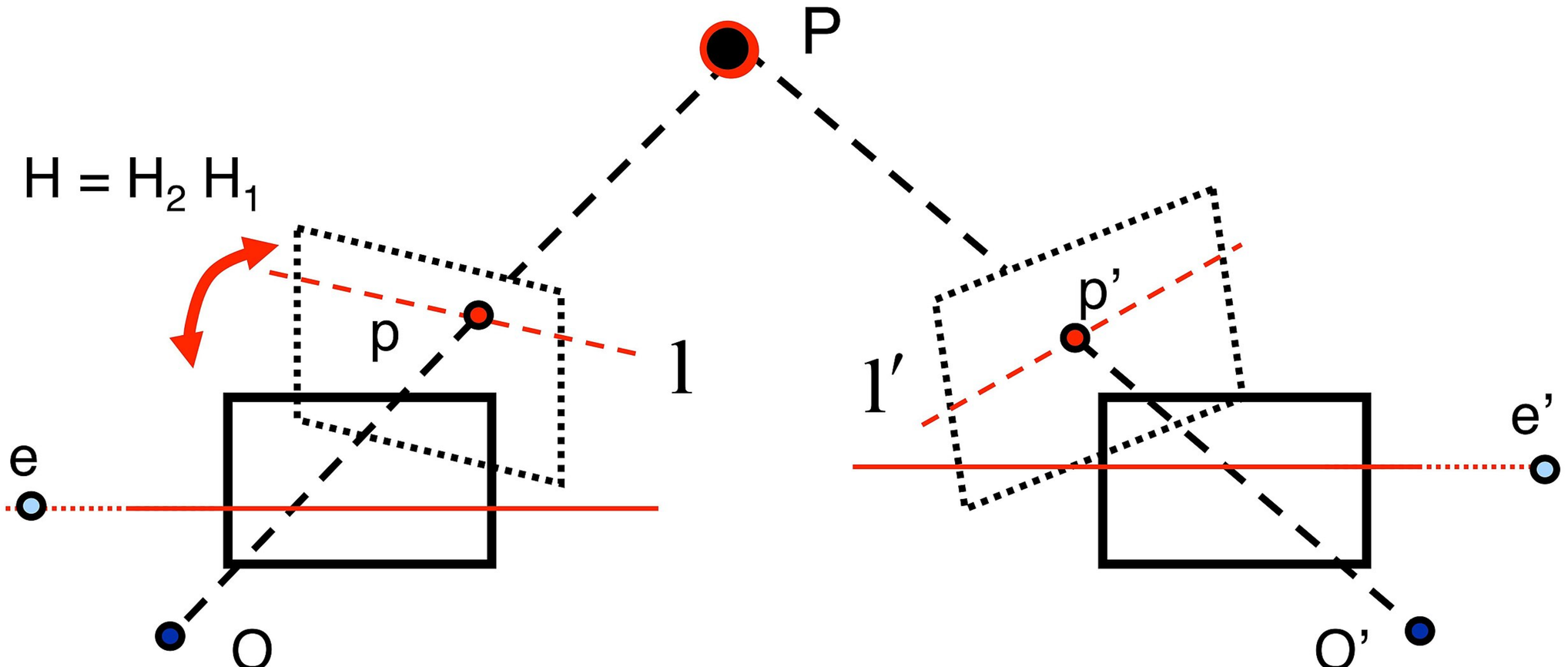


Depth map

$$Z = f\frac{T}{D(x, y)}$$

If the cameras are not parallel?

Reproject so that they are (rectification)
2D projective transform (homography)



Also useful for panorama stitching

Images from the same viewpoint but in different directions
2D projective transform (homography)

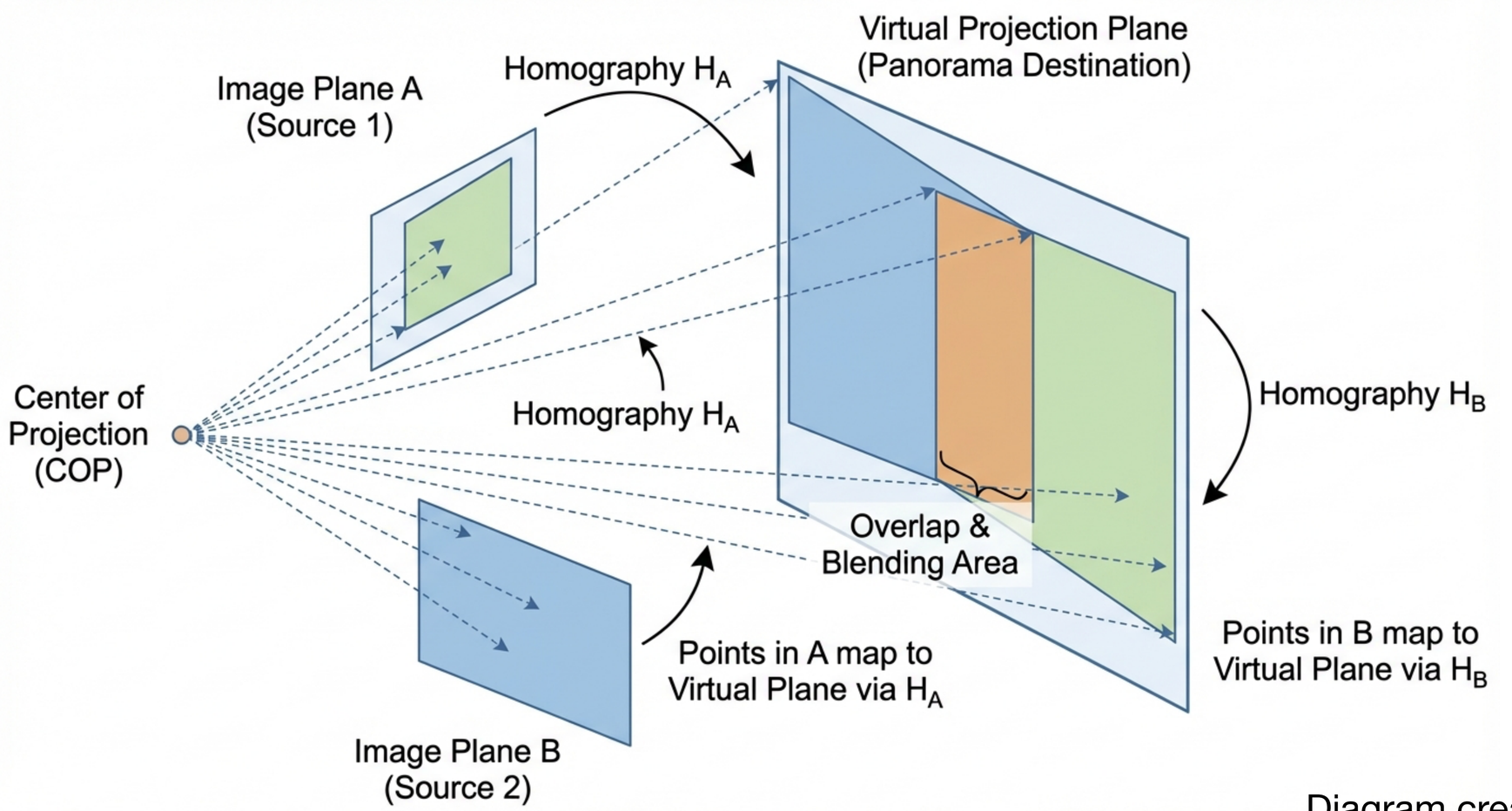


Diagram created with nano banana

Reprojection with homography

- Take my computational photography class (6.8370/1)



Contrarian take

- https://people.csail.mit.edu/bkph/articles/Harmful.pdf

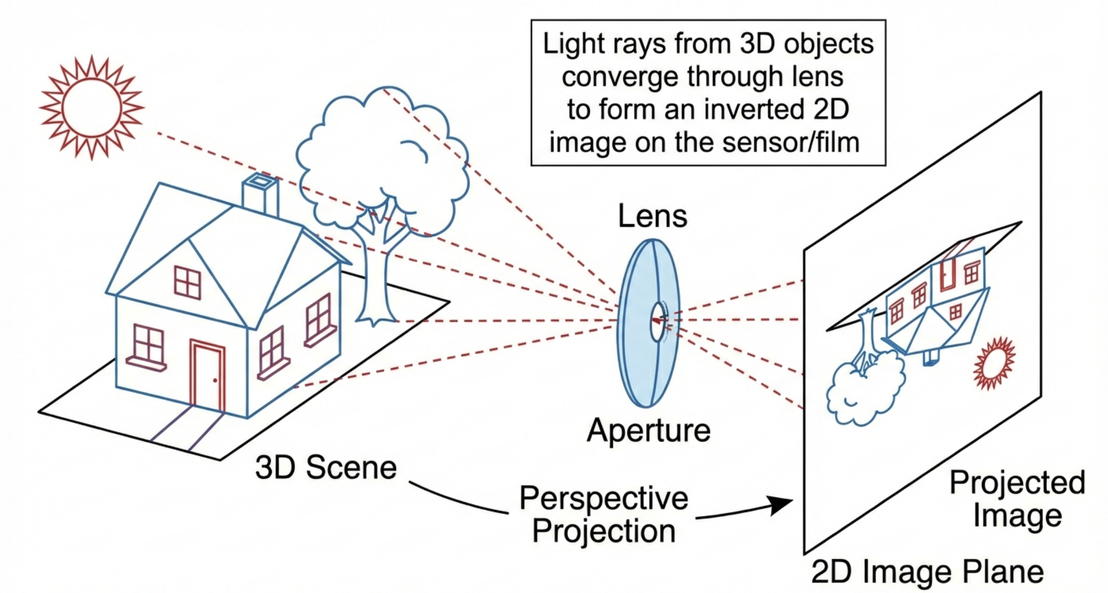
Projective Geometry Considered Harmful

Berthold K.P. Horn

Copright © 1999

Philosophical discussion

- Ingredients of Vision include 2D and 3D image
- The connection between the two (image formation, perspective) is important
- So far still mostly hand-coded
  - Inductive bias
- Who knows if it will eventually be learned
  - e.g. transformers to learn mapping from 3D triangles to image https://microsoft.github.io/renderformer/



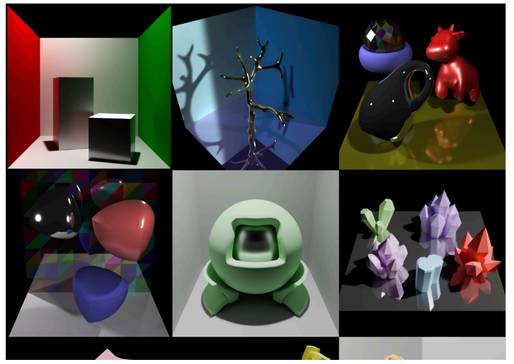
RenderFormer

Transformer-based Neural Rendering of Triangle Meshes with Global Illumination

SIGGRAPH 2025

Chong Zeng[1,2] Yue Dong[2] Pieter Peers[3] Hongzhi Wu[1] Xin Tong[2]

[1]State Key Lab of CAD and CG, Zhejiang University
[2]Microsoft Research Asia
[3]College of William & Mary



Degrees of classical vs neural

- Compute extrinsic / intrinsics using traditional vision
- Have a projective transform hard coded but let neural network figure out coefficients
- All through learning?