Robotic Mapping & Localization

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lec07: Pinhole Camera Model

Our Simple SLAM System

robotic-mapping / code / visual-odometry / vo_epipolar.cpp

KavehFathian first commit

Code Blame 97 lines (88 loc) · 3.52 KB

```cpp
#include <iostream>
#include <vector>
#include <string>
#include "opencv2/opencv.hpp"

int main()
{
    const char* video_file = "../data/KITTI07/image_0/%06d.png";
    double f = 707.0912;
    cv::Point2d c(601.8873, 183.1104);
    bool use_5pt = true;
    int min_inlier_num = 100;
    double min_inlier_ratio = 0.2;
    const char* traj_file = "vo_epipolar.xyz";
```

Our Simple SLAM System


```cpp
#include <iostream>
#include <vector>
#include <string>
#include "opencv2/opencv.hpp"

int main()
{
    const char* video_file = "../data/KITTI07/image_0/%06d.png";
    double f = 707.0912;
    cv::Point2d c(601.8873, 183.1104);
    bool use_5pt = true;
    int min_inlier_num = 100;
    double min_inlier_ratio = 0.2;
    const char* traj_file = "vo_epipolar.xyz";

    // Open a video and get an initial image
    cv::VideoCapture video;
    if (!video.open(video_file)) return -1;

    cv::Mat gray_prev;
    video >> gray_prev;
    if (gray_prev.empty())
    {
        video.release();
        return -1;
    }
    if (gray_prev.channels() > 1) cv::cvtColor(gray_prev, gray_prev,
    cv::COLOR_RGB2GRAY);

    // Run the monocular visual odometry
    cv::Mat K = (cv::Mat_<double>(3, 3) << f, 0, c.x, 0, f, c.y, 0, 0, 1);
    cv::Mat camera_pose = cv::Mat::eye(4, 4, CV_64F);
    FILE* camera_traj = fopen(traj_file, "wt");
    if (camera_traj == NULL) return -1;
```

OpenCV library

Camera parameters

SLAM system parameters

Convert RGB images to gray

Camera calibration matrix

Pose (position & orientation) estimates

Our Simple SLAM System

```cpp
while (true)
{
    // Grab an image from the video
    cv::Mat img, gray;
    video >> img;
    if (img.empty()) break;
    if (img.channels() > 1) cv::cvtColor(img, gray, cv::COLOR_RGB2GRAY);
    else                    gray = img.clone();

    // Extract optical flow
    std::vector<cv::Point2f> pts_prev, pts;
    cv::goodFeaturesToTrack(gray_prev, pts_prev, 2000, 0.01, 10);
    std::vector<uchar> status;
    cv::Mat err;
    cv::calcOpticalFlowPyrLK(gray_prev, gray, pts_prev, pts, status, err);
    gray_prev = gray;

    // Calculate relative pose
    cv::Mat E, inlier_mask;
    if (use_5pt)
    {
        E = cv::findEssentialMat(pts_prev, pts, f, c, cv::RANSAC, 0.999, 1,
        inlier_mask);
    }
    else
    {
        cv::Mat F = cv::findFundamentalMat(pts_prev, pts, cv::FM_RANSAC, 1, 0.
        99, inlier_mask);
        E = K.t() * F * K;
    }
    cv::Mat R, t;
    int inlier_num = cv::recoverPose(E, pts_prev, pts, R, t, f, c, inlier_mask);
    double inlier_ratio = static_cast<double>(inlier_num) / static_cast<double>
    (pts.size());

    // Accumulate relative pose if result is reliable
    cv::Vec3b info_color(0, 255, 0);
    if ((inlier_num > min_inlier_num) && (inlier_ratio > min_inlier_ratio))
    {
        cv::Mat T = cv::Mat::eye(4, 4, R.type());
        T(cv::Rect(0, 0, 3, 3)) = R * 1.0;
        T.col(3).rowRange(0, 3) = t * 1.0;
        camera_pose = camera_pose * T.inv();
        info_color = cv::Vec3b(0, 0, 255);
    }
```

Iterate over images

Feature detection & tracking using optical flow

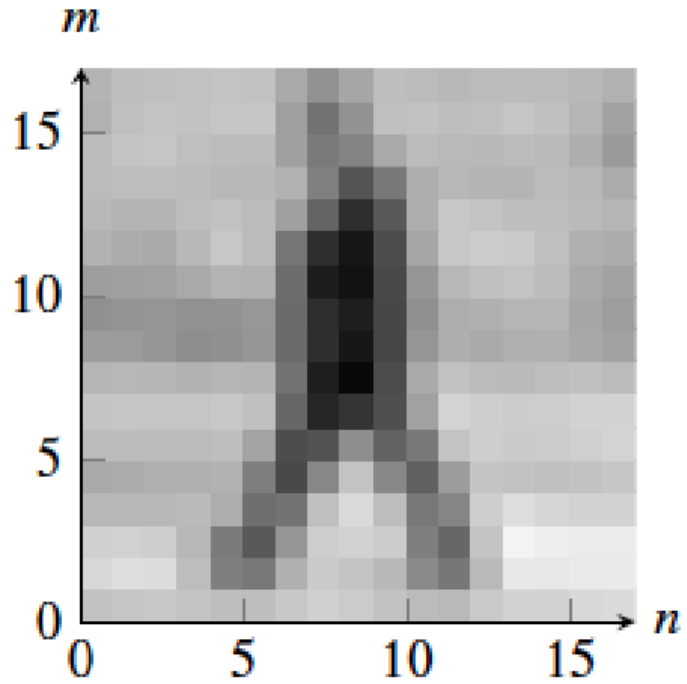
Estimate camera pose (in the form of essential matrix) from consecutive images

RANSAC outlier rejection

Extract camera pose from essential matrix

Camera pose (and trajectory) over time

Image as a 2D discrete signal

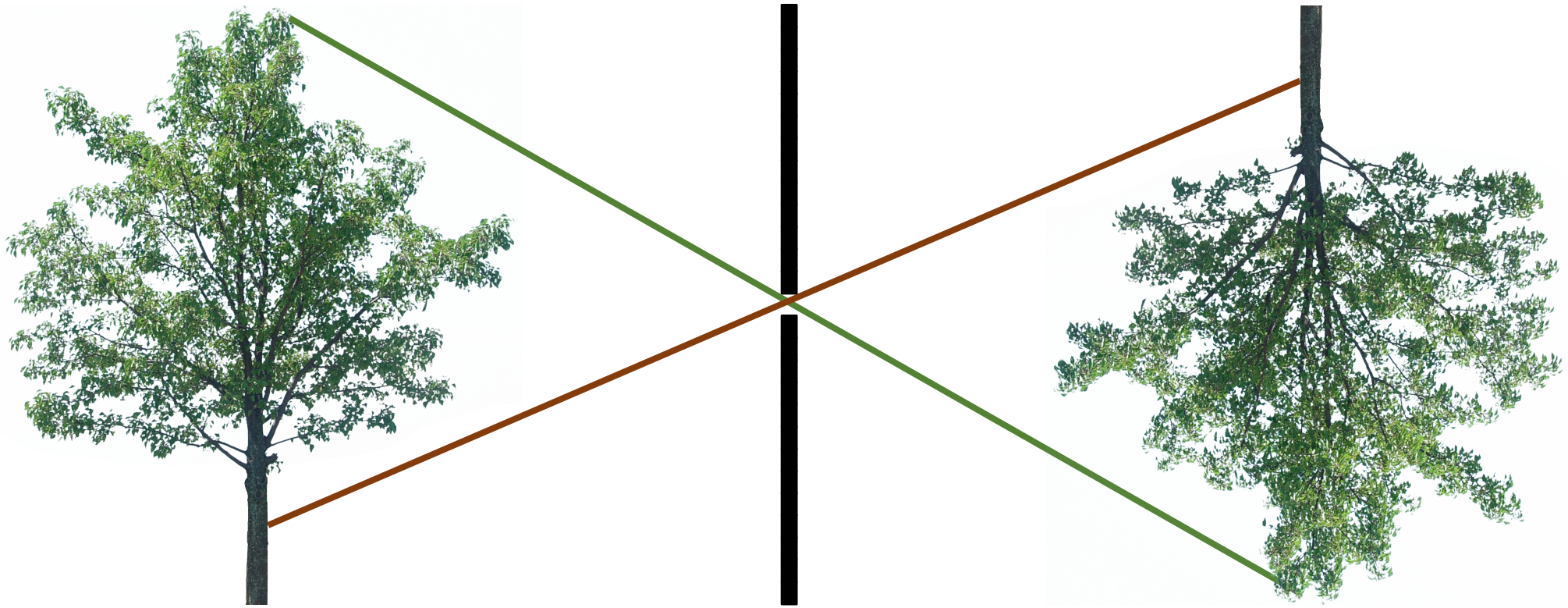


A tiny person of 18 x 18 pixels

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

Pinhole imaging

real-world object

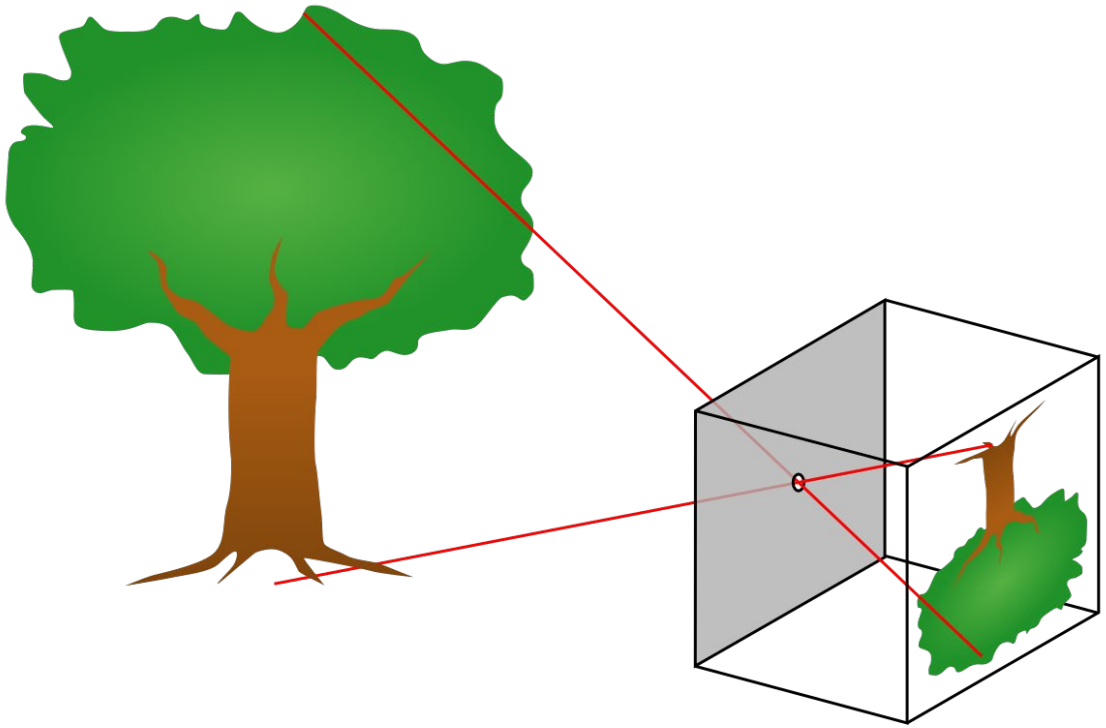


copy of real-world object (inverted and scaled)

Lecture Outline

Camera

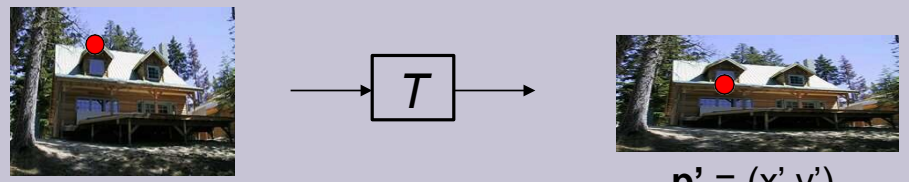
- Image transformations
- Pinhole camera model



Transformations

Definition

A transformation T is a coordinate-changing operation p’ = T(p)



For linear transformations, we can represent T as a matrix:

$$p' = \mathbf{T}p$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Common Transformations

Original

Transformed

Translation

Rotation

Scaling

Affine

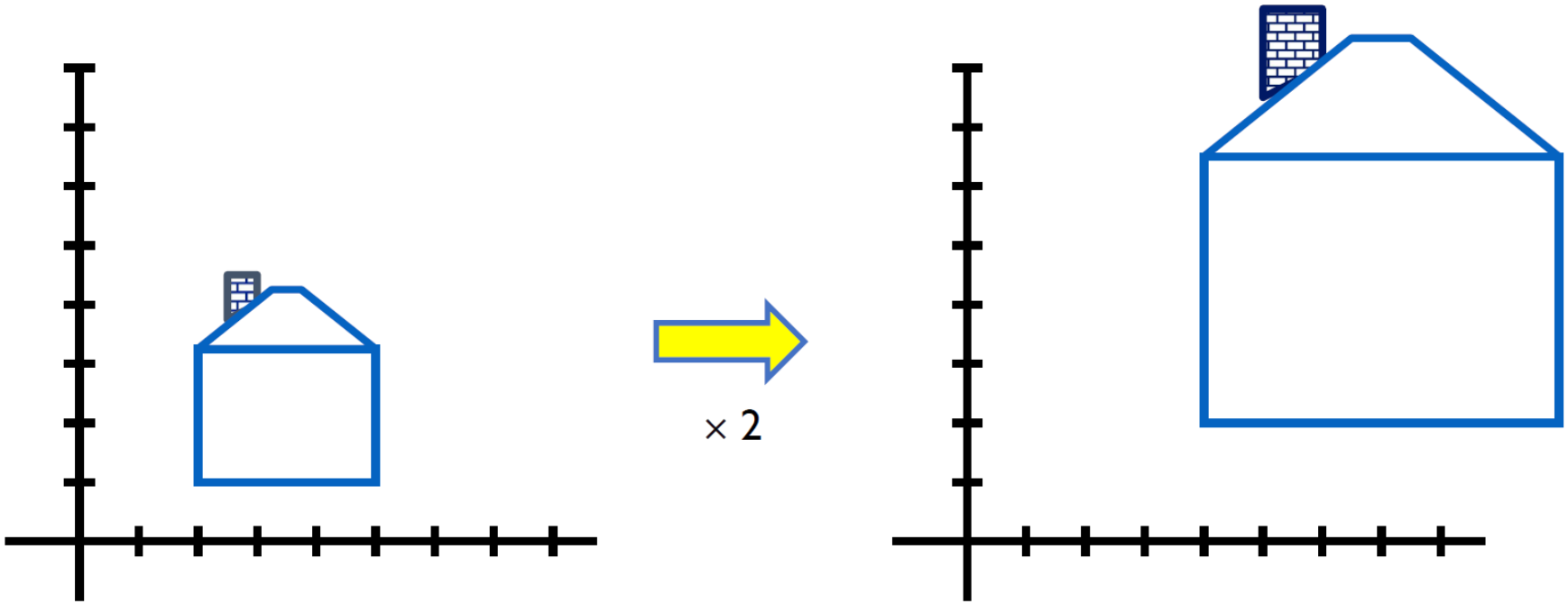
Perspective

next few slides):A. Efros and/or S. Seitz

Scaling

Definition

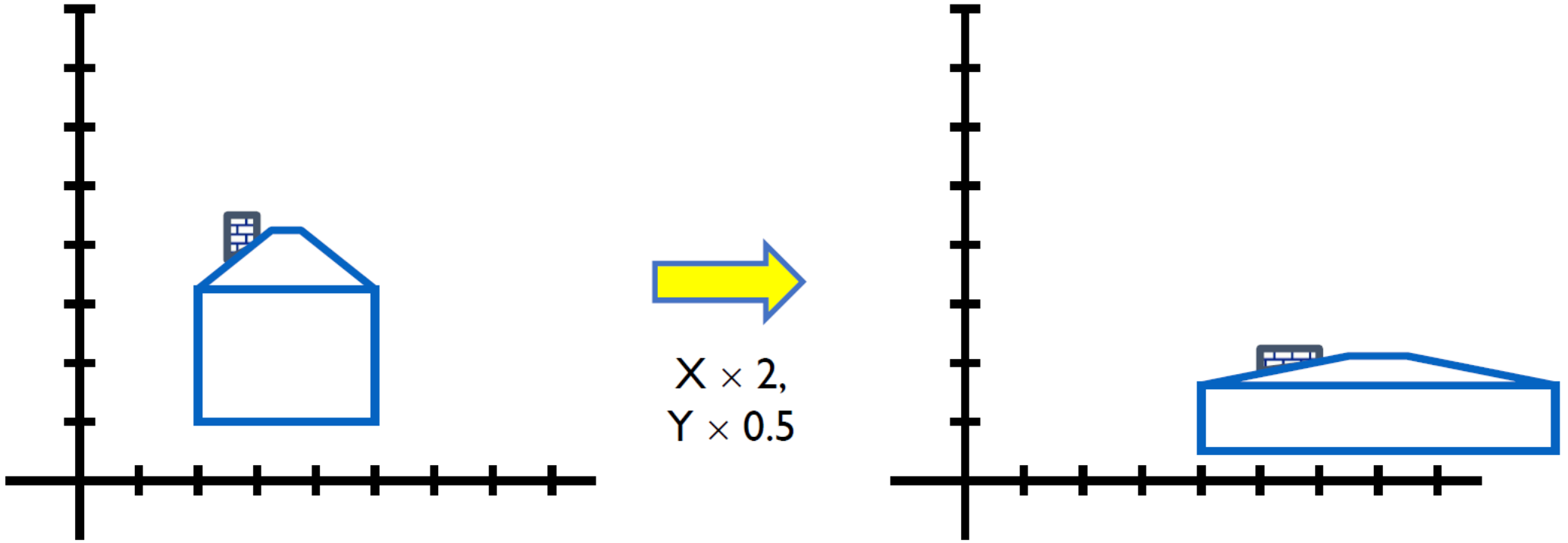
Scaling a coordinate means multiplying each of its components by a scalar.
Uniform scaling means this scalar is the same for all components.



Scaling

Definition

Non-uniform scaling: different scalars per component



Scaling

- Scaling operation:

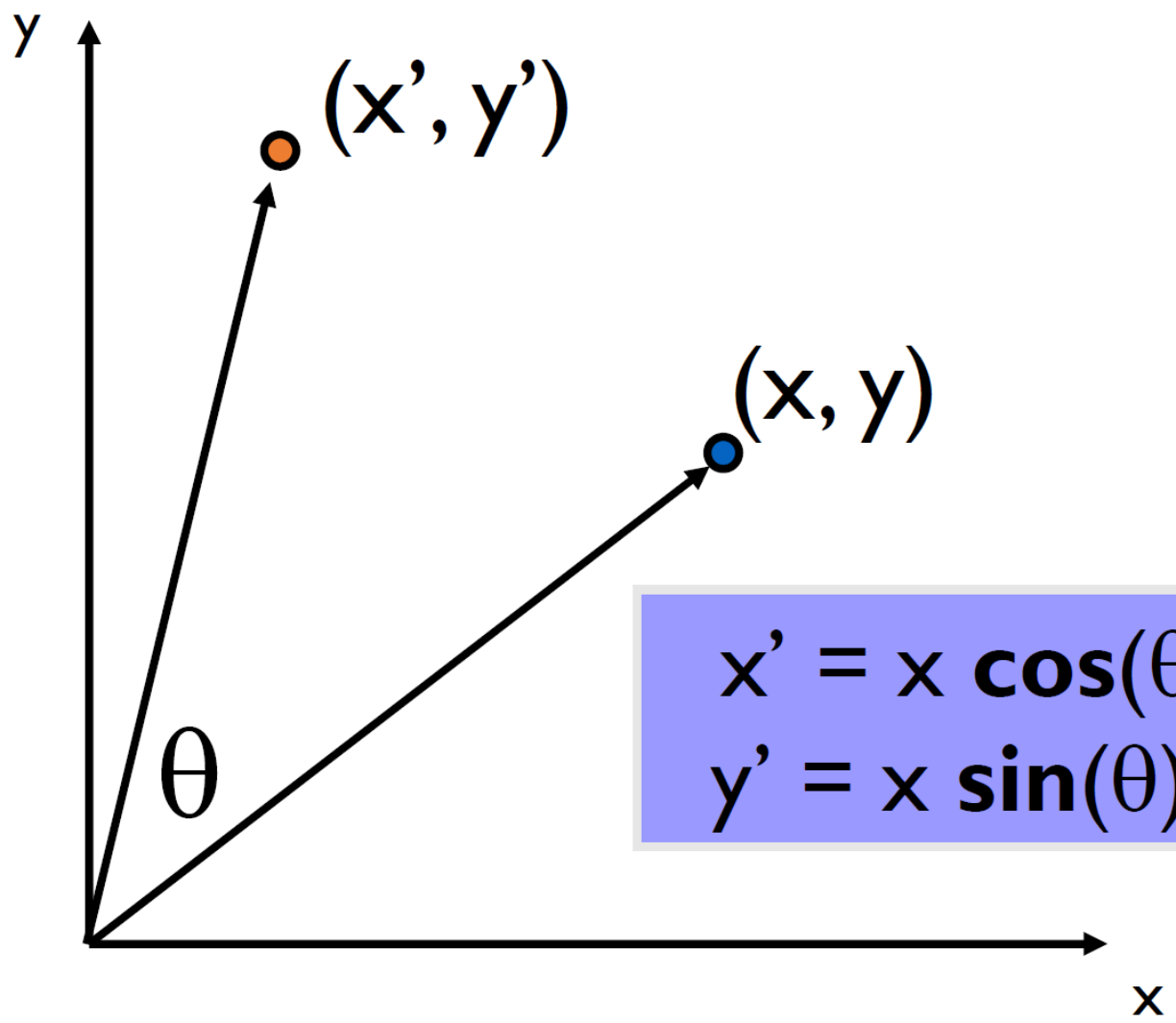
$$x' = ax$$

$$y' = by$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Rotation



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

Basic 2D Transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, shear

Affine Transformations

- Affine transformations are combinations of
  - Linear transformations, and
  - Translations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or

- Properties of affine transformations:
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

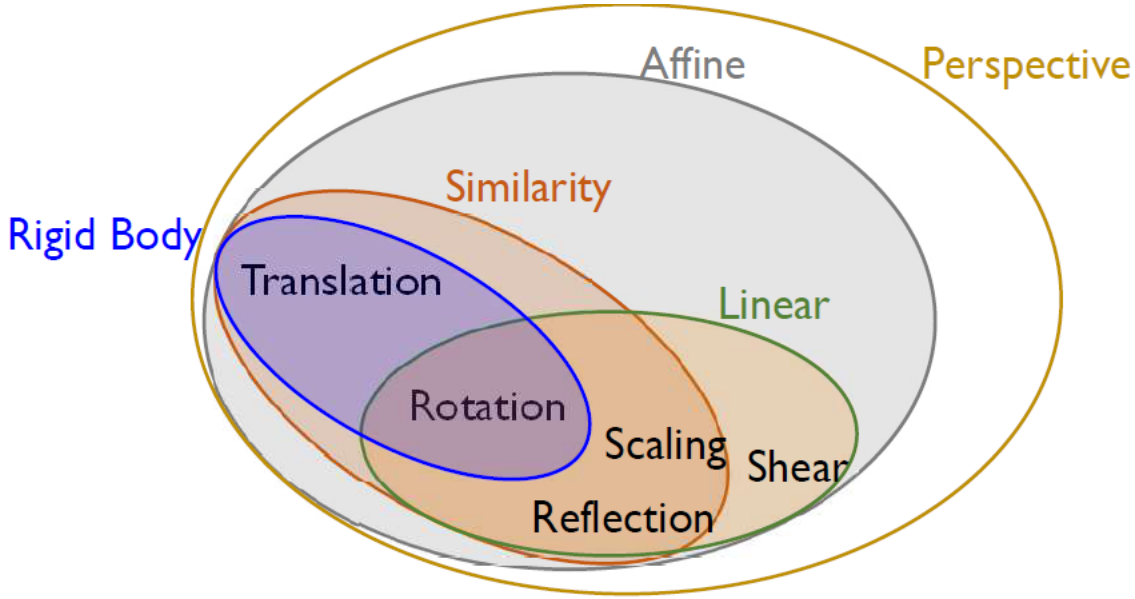
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

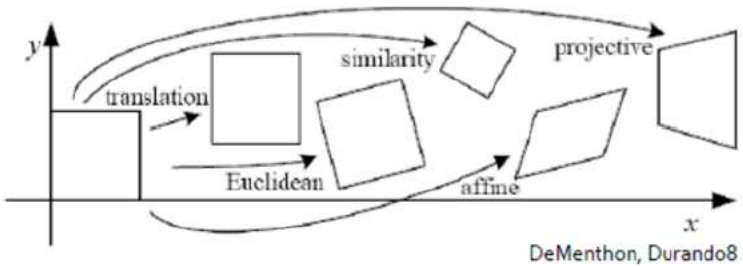
ɔugih Jamin

2D Transformations: Properties Preserved

Transformations can be classified based on the properties they preserve:

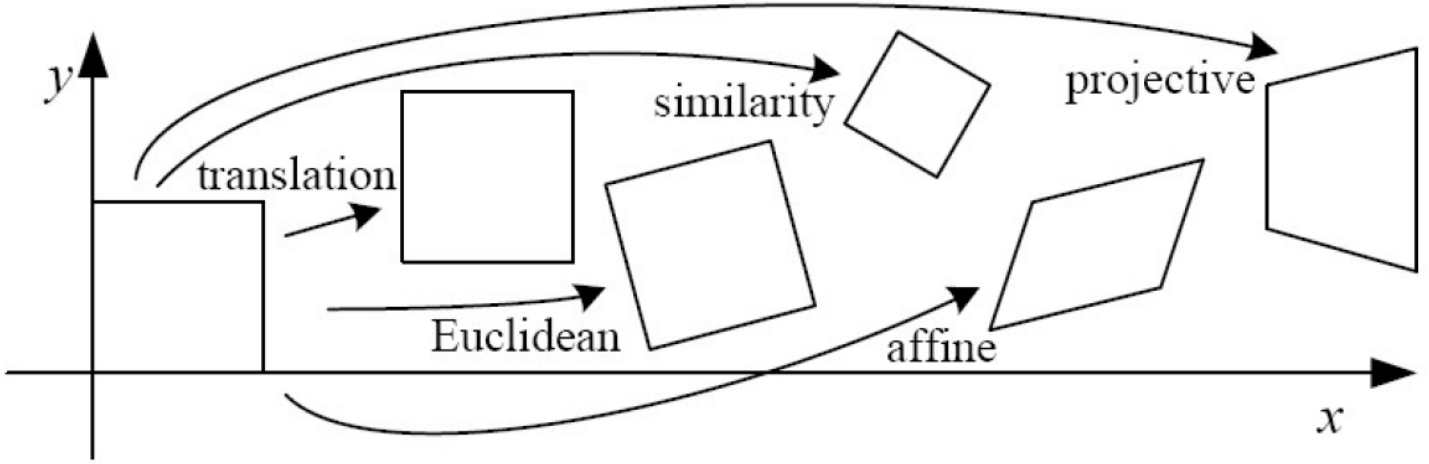
- Rigid Body
  - angles, lengths, areas
- Similarity
  - angles, length ratios
- Linear
  - linear combination
- Affine
  - parallel lines, length ratios, area ratios
- Perspective
  - collinearity, cross-ratio
- …





Sugih Jamin

2D Transformations



Szeliski 2.1

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

‘Homography’

Cameras are Projective Transformation Machines

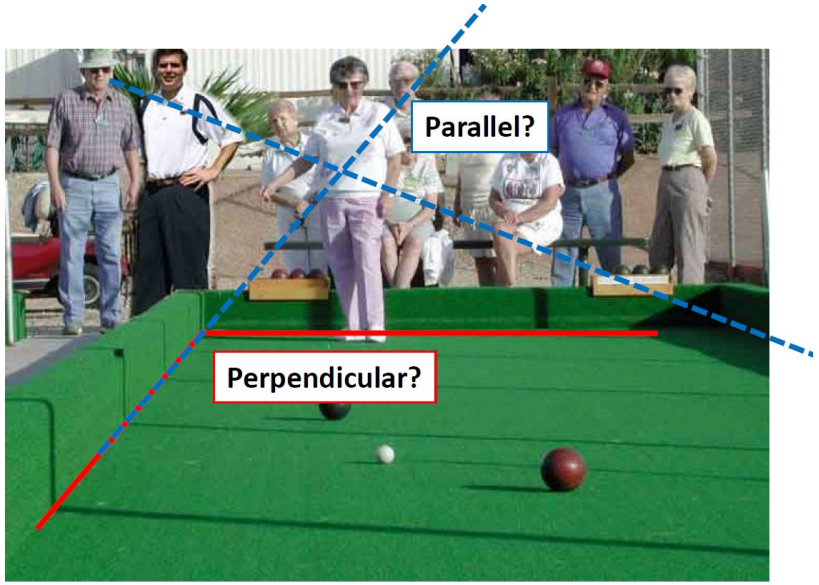
Projective Transformation

3D World

2D Image

Projective Transformations

- Projective transformations are combos of
  - Affine transformations, and
  - Projective warps



- Properties of projective transformations:
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition
  - Models change of basis
  - Projective matrix is defined up to a scale (8 degrees of freedom)

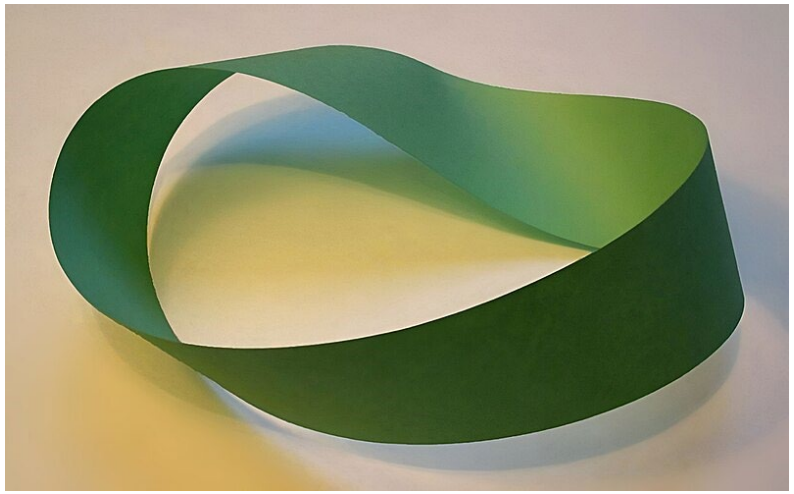
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Homogeneous Coordinates

- Introduced by August Ferdinand Möbius

- An alternative coordinate system for projective geometry
  - Compare to Cartesian coordinates

- Simplify formulas
  - Make translation a linear transformation
  - Facilitate matrix-vector formulas



Möbius strip - Wikipedia

Wikipedia

Homogeneous Coordinates

Converting to homogeneous coordinates

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

2D (image) coordinates

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

3D (scene) coordinates

Converting from homogeneous coordinates

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

2D (image) coordinates

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

3D (scene) coordinates

Homogeneous Coordinates

Scale invariance in projection space

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

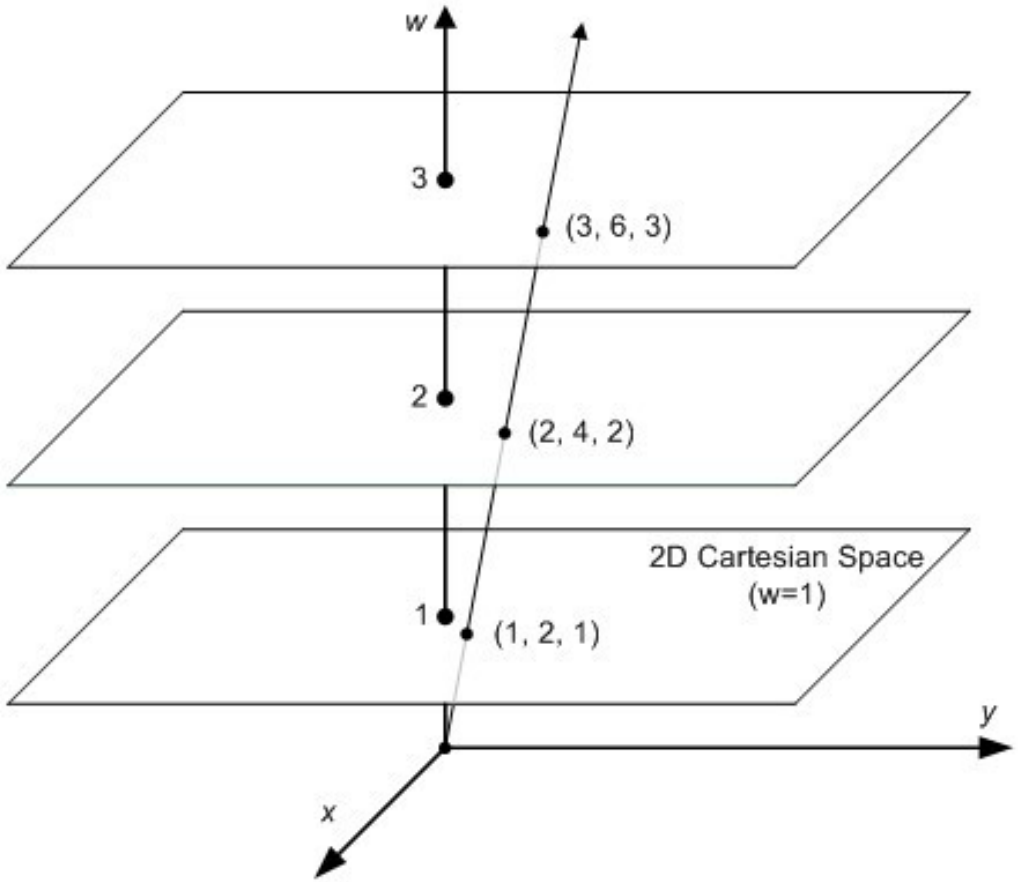
Homogeneous Coordinates

Cartesian Coordinates

e.g., we can uniformly scale the projective space, and it will still produce the same image ⇒ scale ambiguity

Homogeneous Coordinates

- Projective
- Point becomes a line



Song Ho Ahn

To homogeneous

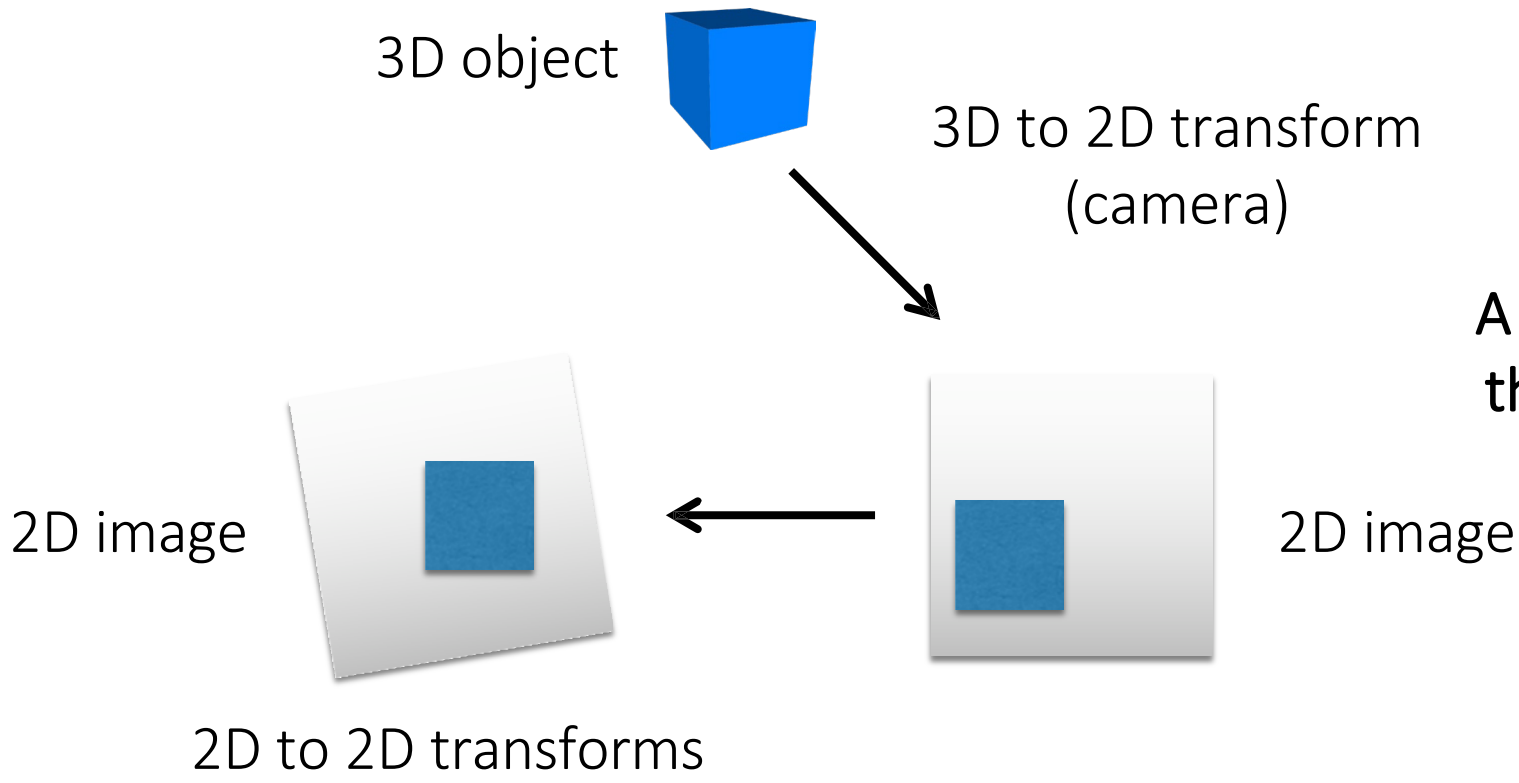
$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

From homogeneous

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

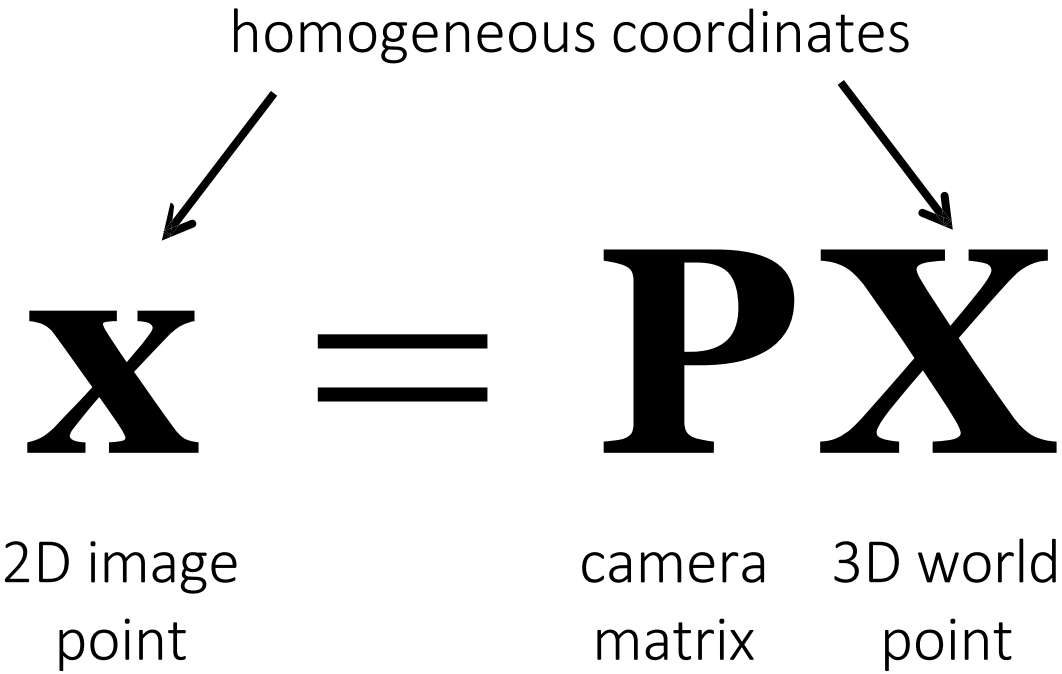
Camera Matrix

The camera as a coordinate transformation



A camera is a mapping from the 3D world to a 2D image

The camera as a coordinate transformation



A camera is a mapping from the 3D world to a 2D image

What are the dimensions of each variable?

The camera as a coordinate transformation

$$\mathbf{x} = \mathbf{P}\mathbf{X}$$

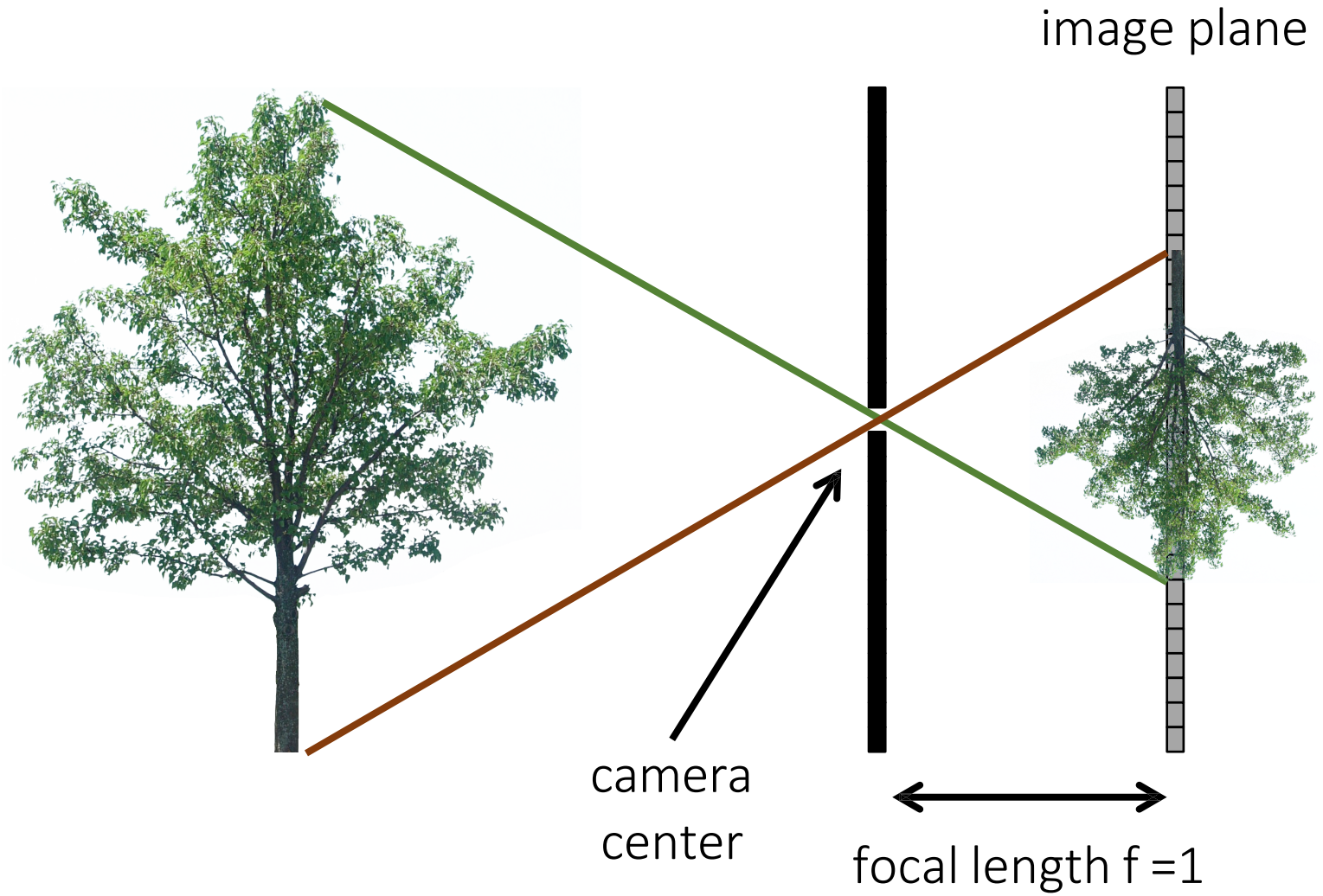
$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

homogeneous image coordinates 3 x 1

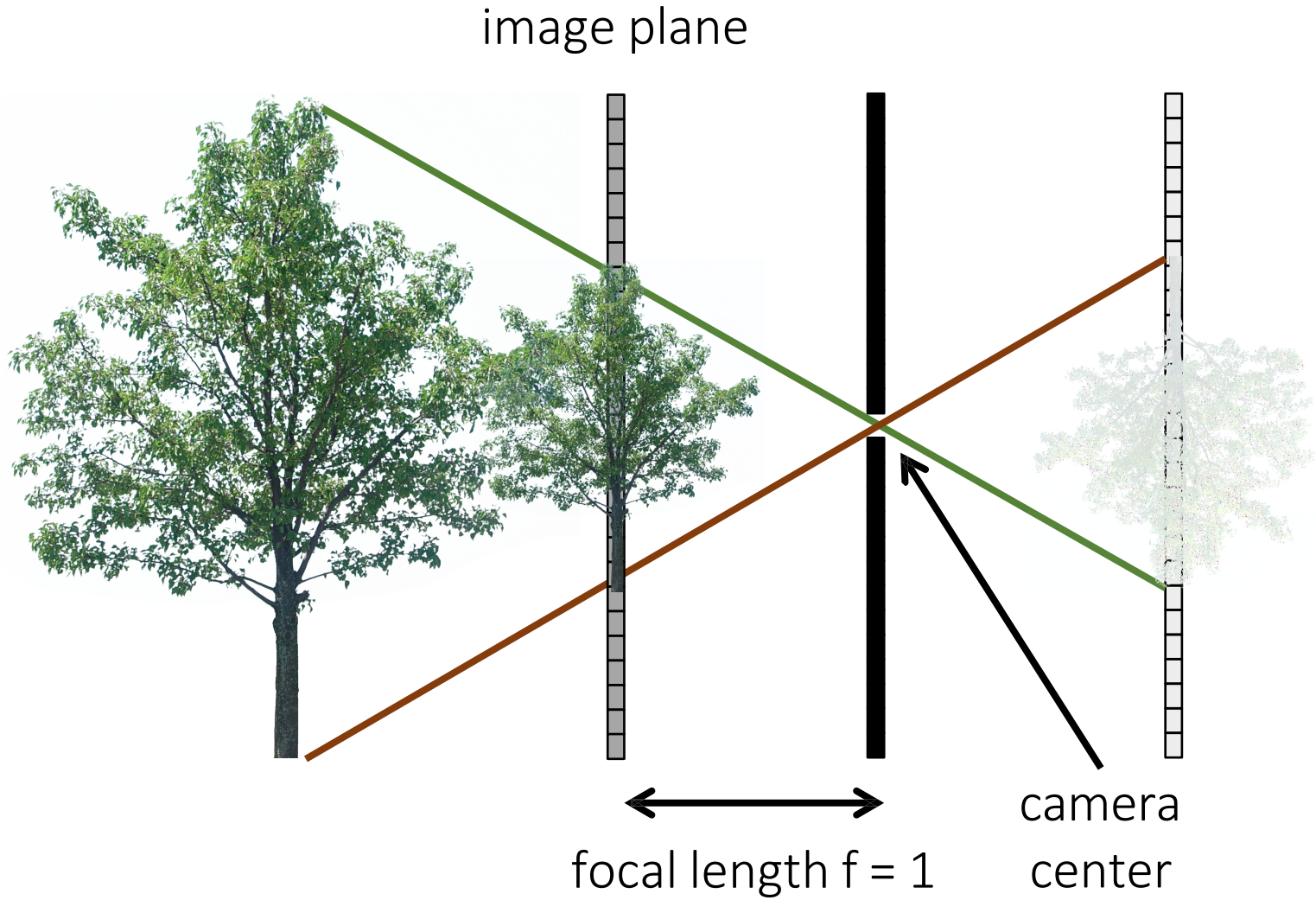
camera matrix 3 x 4

homogeneous world coordinates 4 x 1

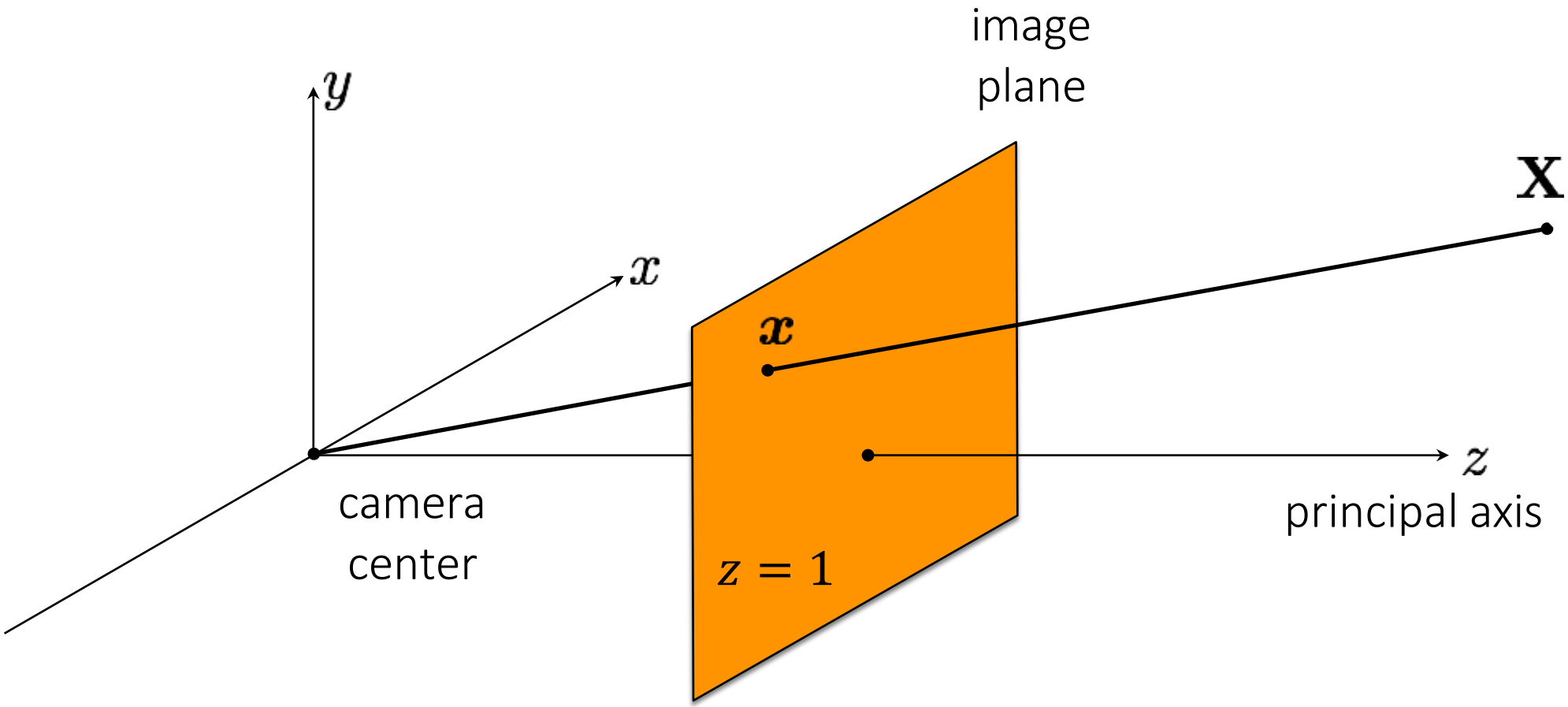
The pinhole camera



The (rearranged) pinhole camera

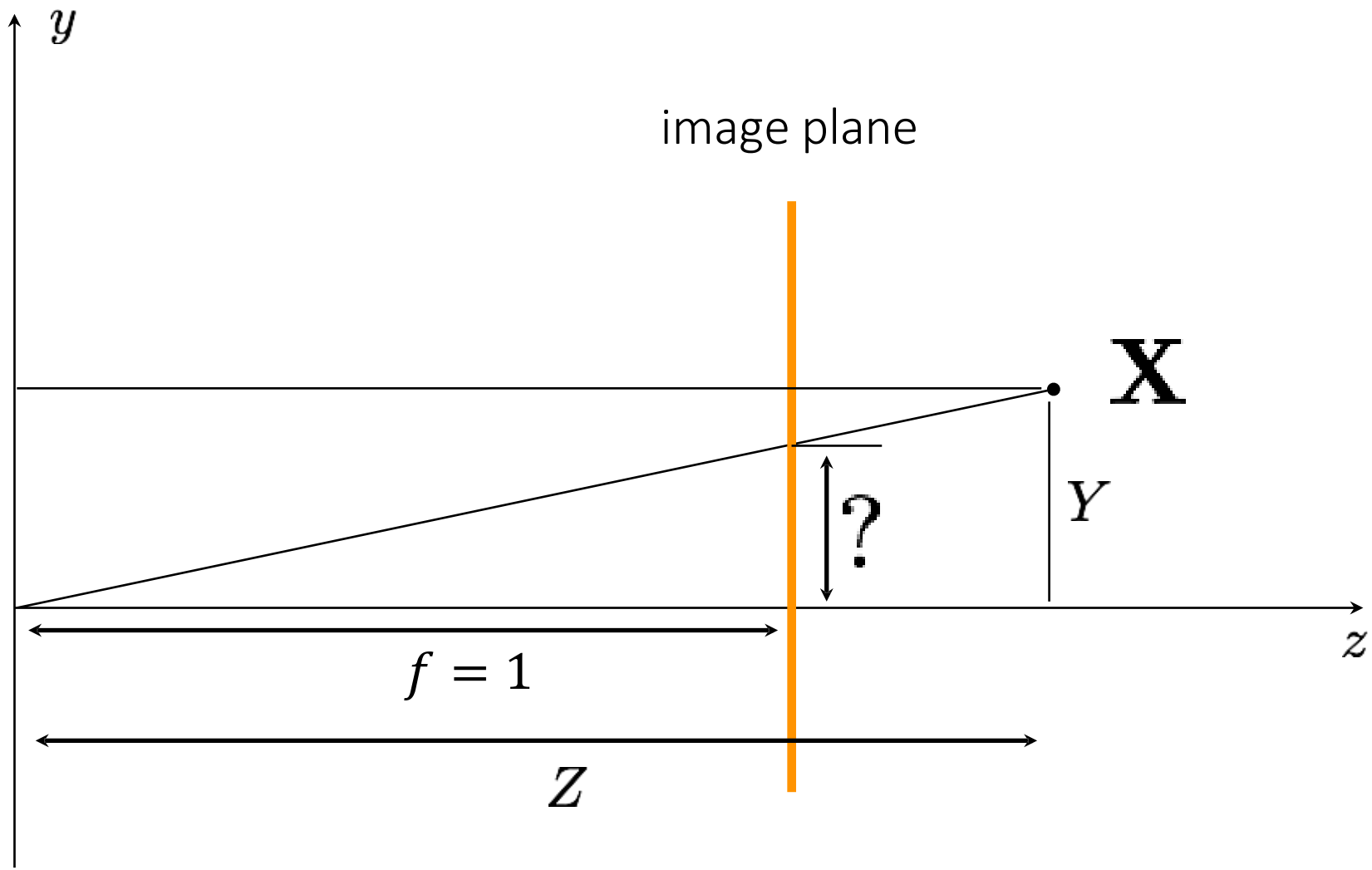


The (rearranged) pinhole camera



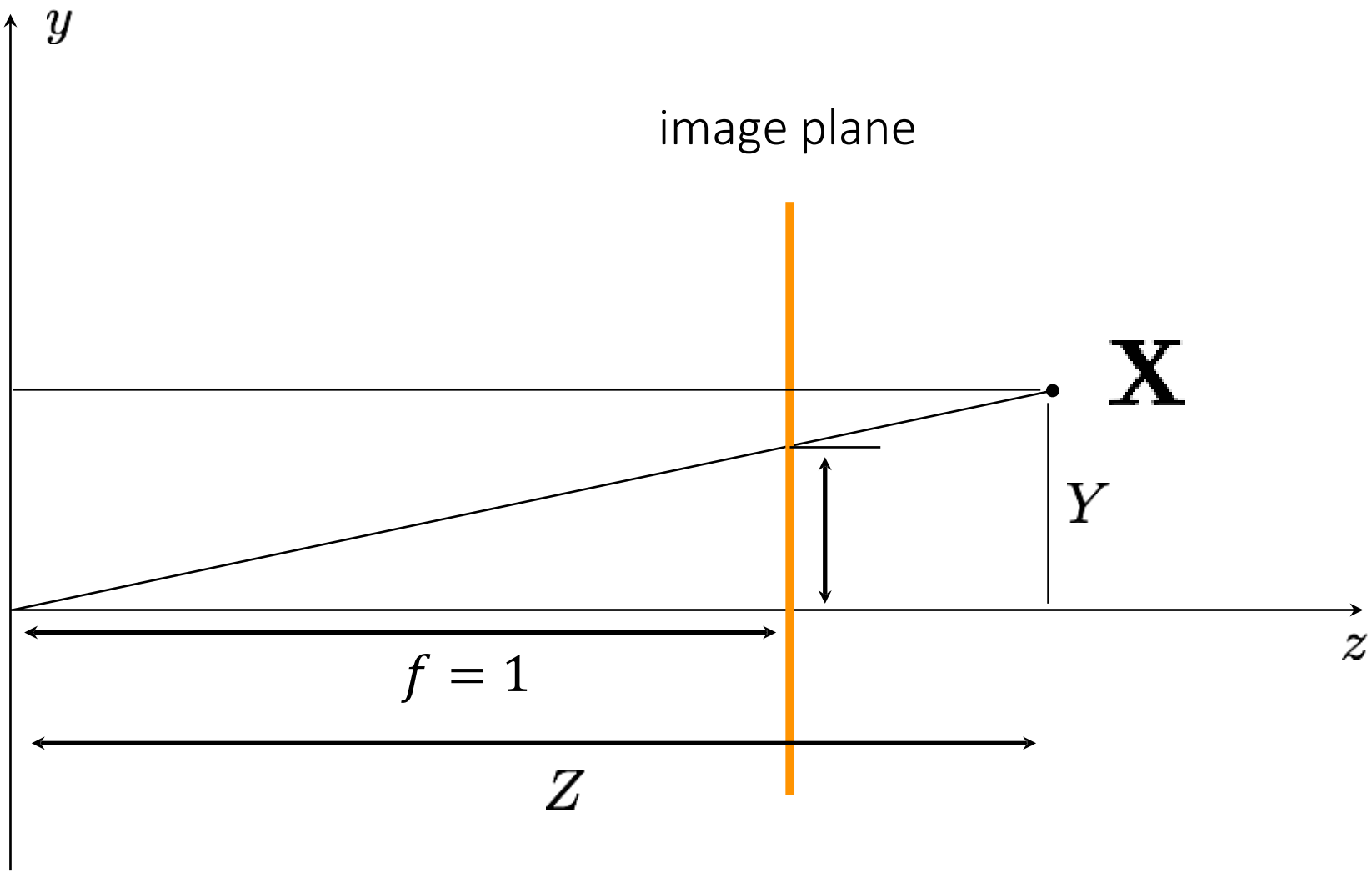
What is the equation for image coordinate x in terms of X?

The 2D view of the (rearranged) pinhole camera



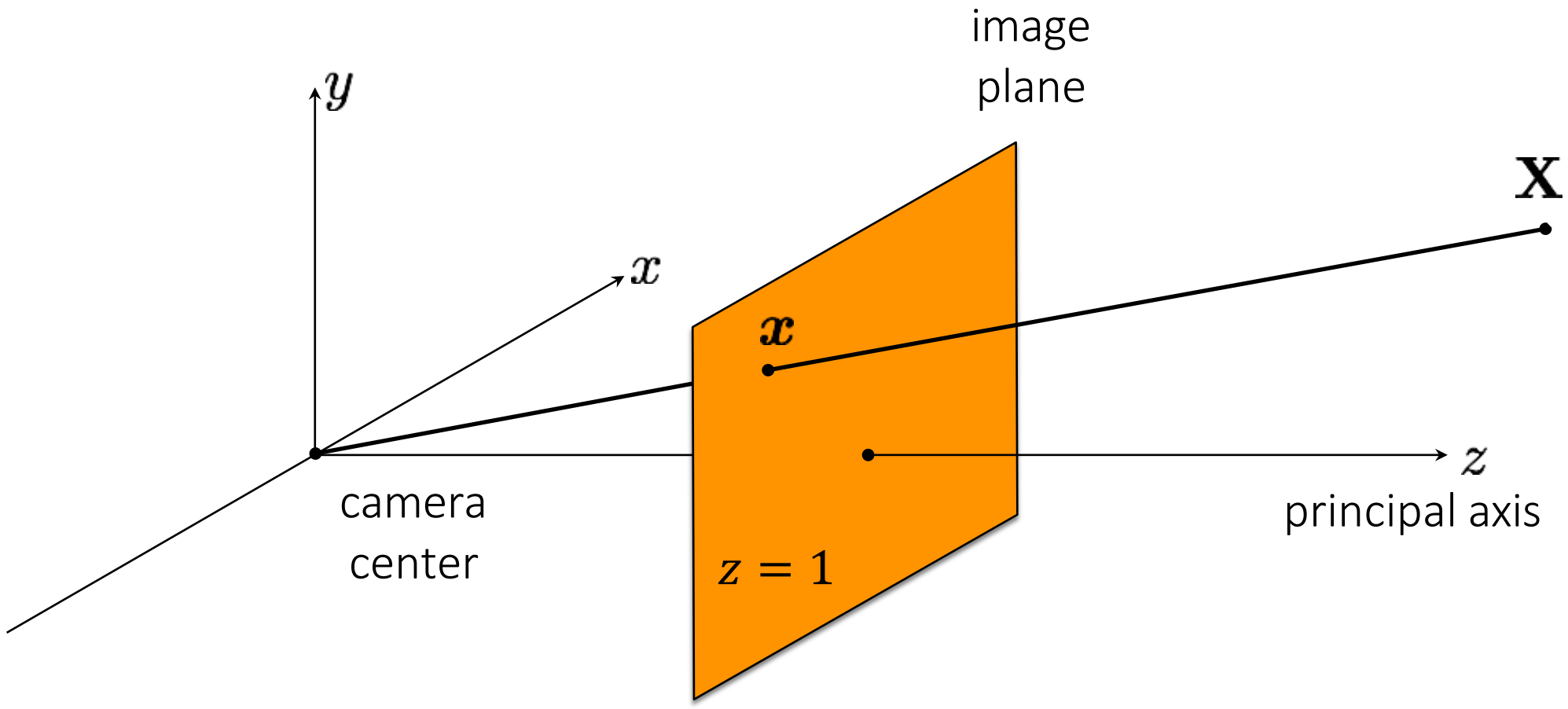
What is the equation for image coordinate x in terms of X?

The 2D view of the (rearranged) pinhole camera



$$[X \quad Y \quad Z] \rightarrow [X/Z \quad Y/Z]$$

The (rearranged) pinhole camera



What is the camera matrix P for a pinhole camera?

$$\boldsymbol{x} = \mathbf{P}\mathbf{X}$$

The pinhole camera matrix

Relationship from similar triangles:

$$[X \quad Y \quad Z]^T \rightarrow [X/Z \quad Y/Z]$$

General camera model in homogeneous coordinates:

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

What does the pinhole camera projection look like?

$$\mathbf{P} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$

The pinhole camera matrix

Relationship from similar triangles:

$$[X \quad Y \quad Z]^T \rightarrow [X/Z \quad Y/Z]$$

General camera model in homogeneous coordinates:

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

What does the pinhole camera projection look like?

The perspective projection matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The pinhole camera matrix

Relationship from similar triangles:

$$[X \quad Y \quad Z]^T \rightarrow [X/Z \quad Y/Z]$$

General camera model in homogeneous coordinates:

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

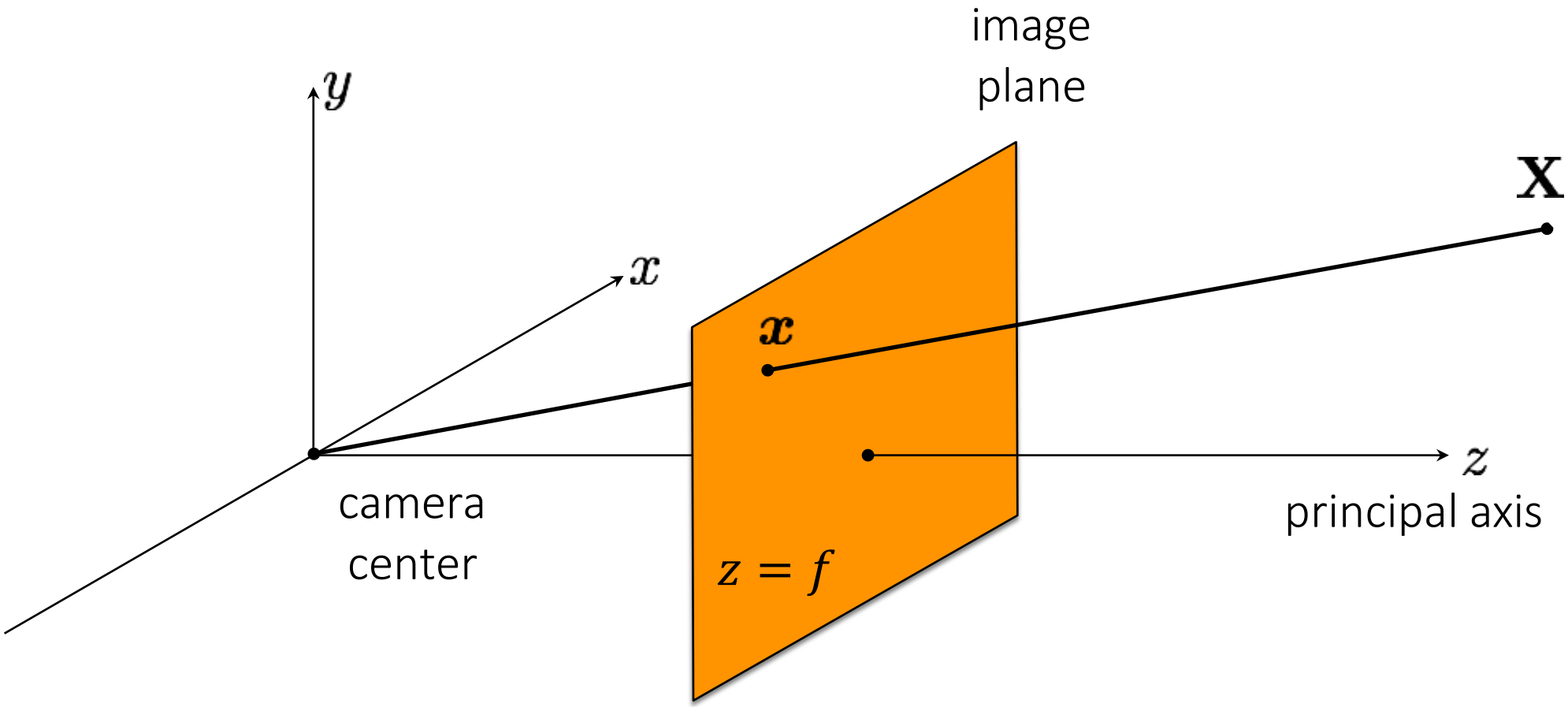
What does the pinhole camera projection look like?

The perspective projection matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] = \mathbf{I} \mid \mathbf{0}$$

alternative way to write the same thing

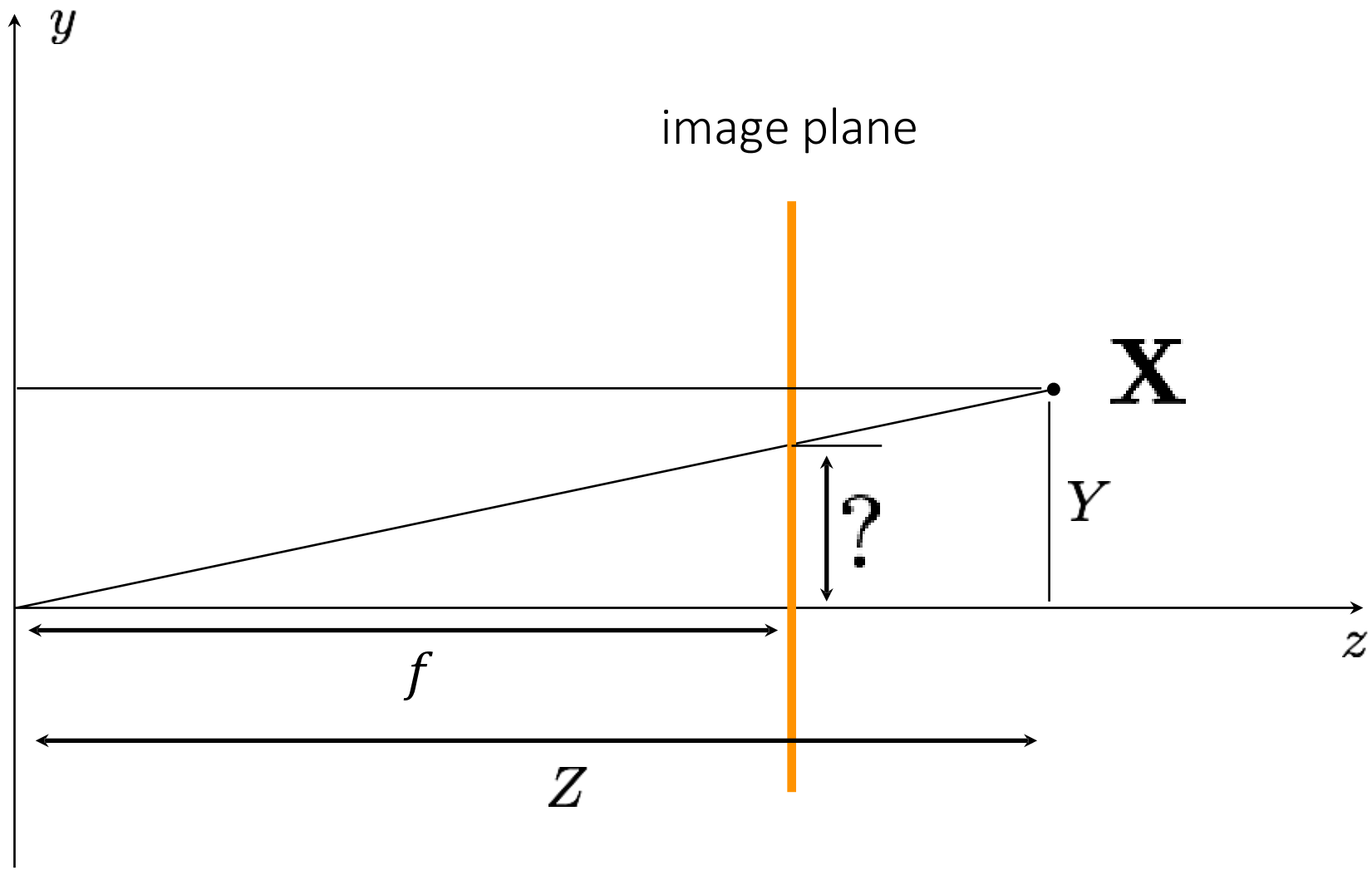
More general case: arbitrary focal length



What is the camera matrix P for a pinhole camera?

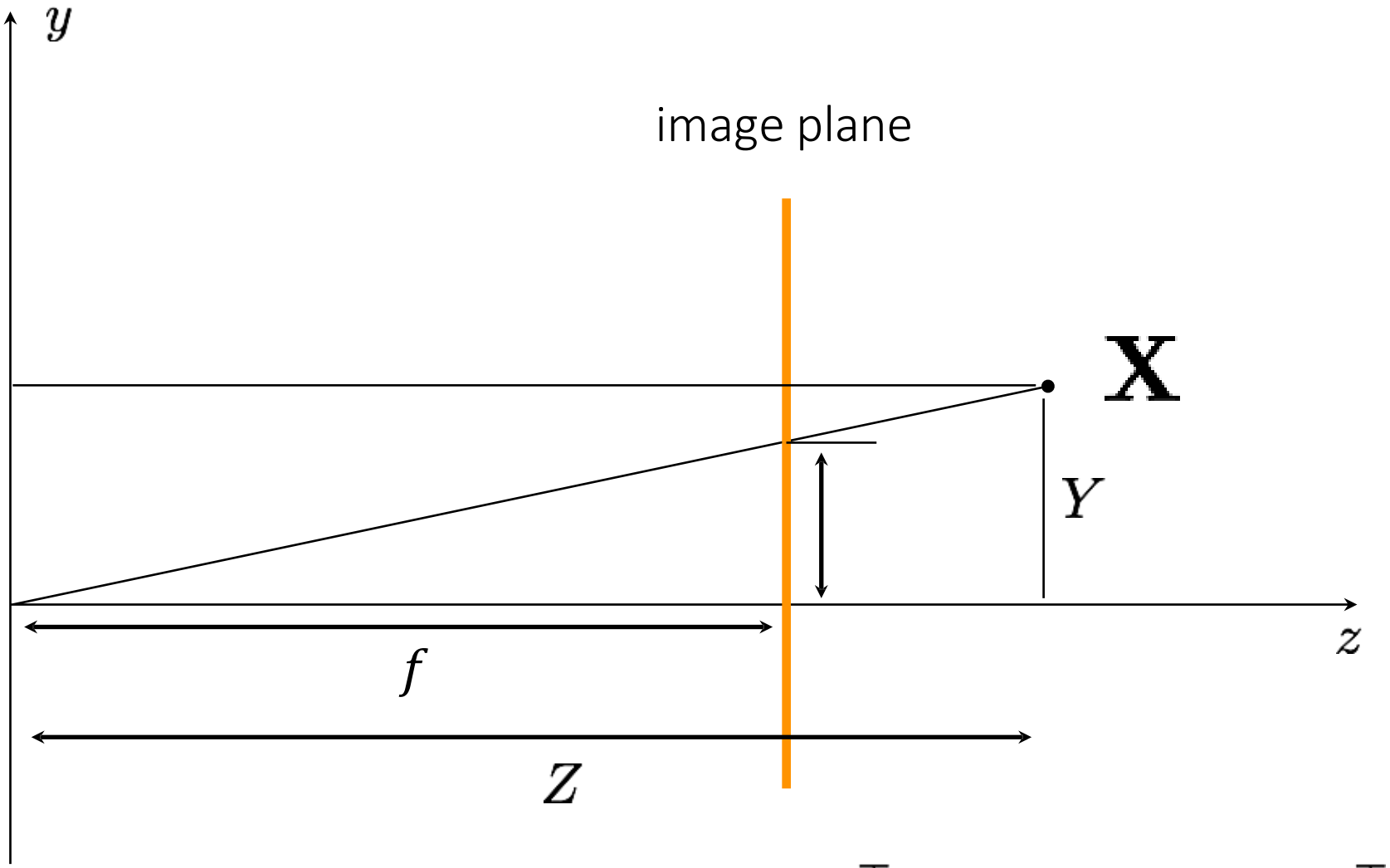
$$\boldsymbol{x} = \mathbf{P}\mathbf{X}$$

More general (2D) case: arbitrary focal length



What is the equation for image coordinate x in terms of X?

More general (2D) case: arbitrary focal length



$$[X \quad Y \quad Z]^\top \mapsto [fX/Z \quad fY/Z]^\top$$

The pinhole camera matrix for arbitrary focal length

Relationship from similar triangles:

$$[X \quad Y \quad Z]^\top \mapsto [fX/Z \quad fY/Z]^\top$$

General camera model *in homogeneous coordinates*:

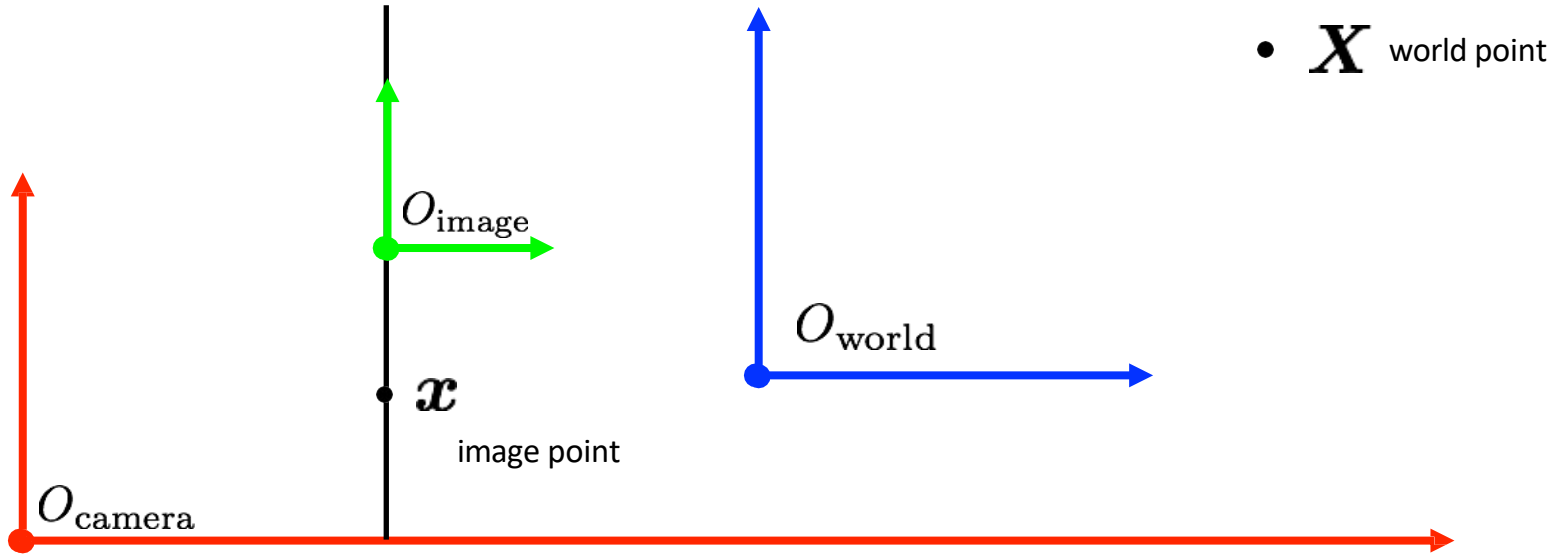
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

What does the pinhole camera projection look like?

$$\mathbf{P} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

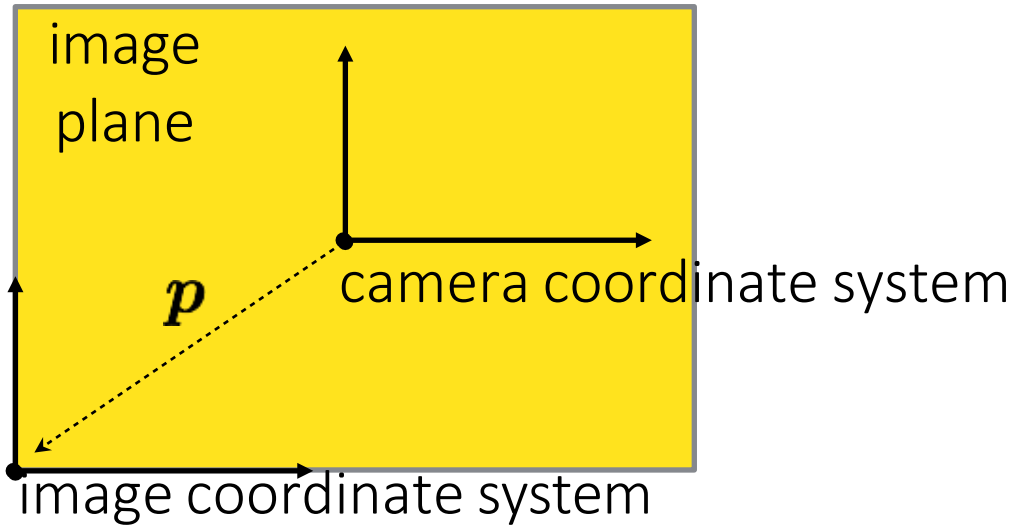
Generalizing the camera matrix

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



Generalizing the camera matrix

In particular, the camera origin and image origin may be different:

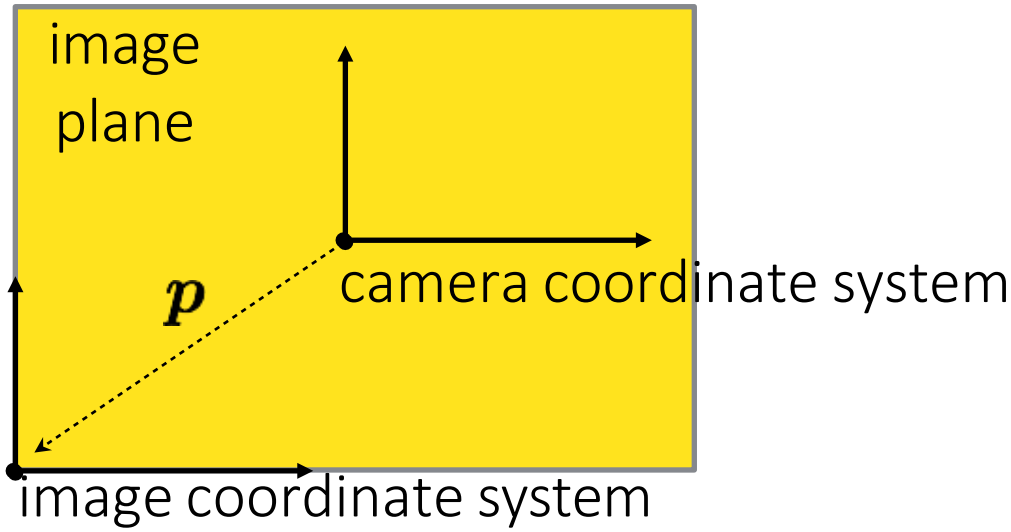


How does the camera matrix change?

$$\mathbf{P} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Generalizing the camera matrix

In particular, the camera origin and image origin may be different:



How does the camera matrix change?

shift vector transforming camera origin to image origin

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

Camera matrix decomposition

We can decompose the camera matrix like this:

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

What does each part of the matrix represent?

Camera matrix decomposition

We can decompose the camera matrix like this:

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

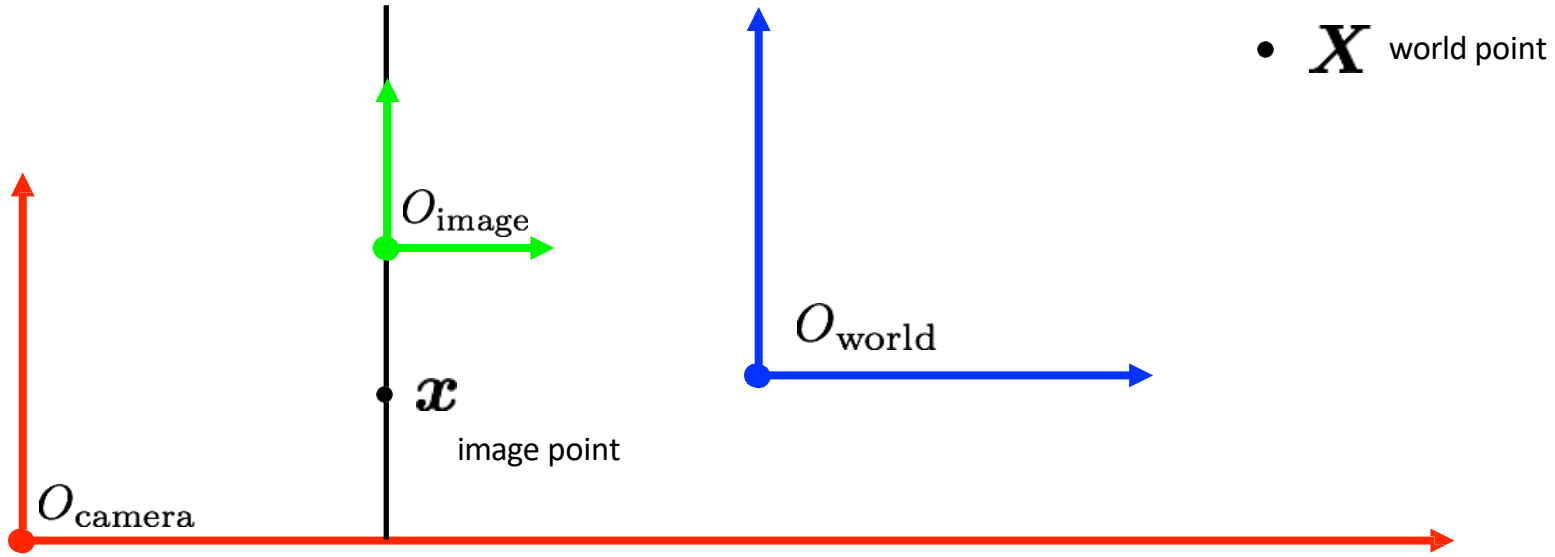
(homogeneous) transformation from 2D to 2D, accounting for not unit focal length and origin shift

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

Also written as: $\mathbf{P} = \mathbf{K}[\mathbf{I}|\mathbf{0}]$ where $\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$

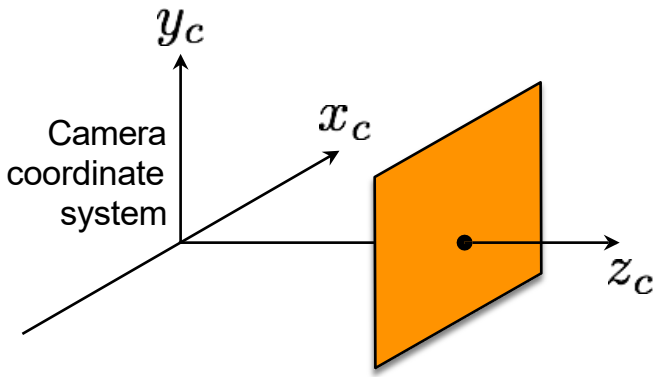
Generalizing the camera matrix

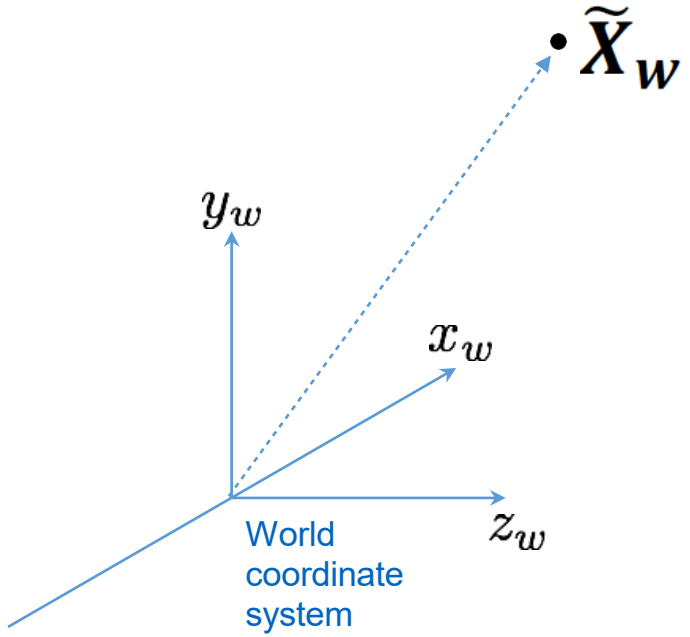
In general, there are ***three***, generally different, coordinate systems:



We need to know the transformations between them.

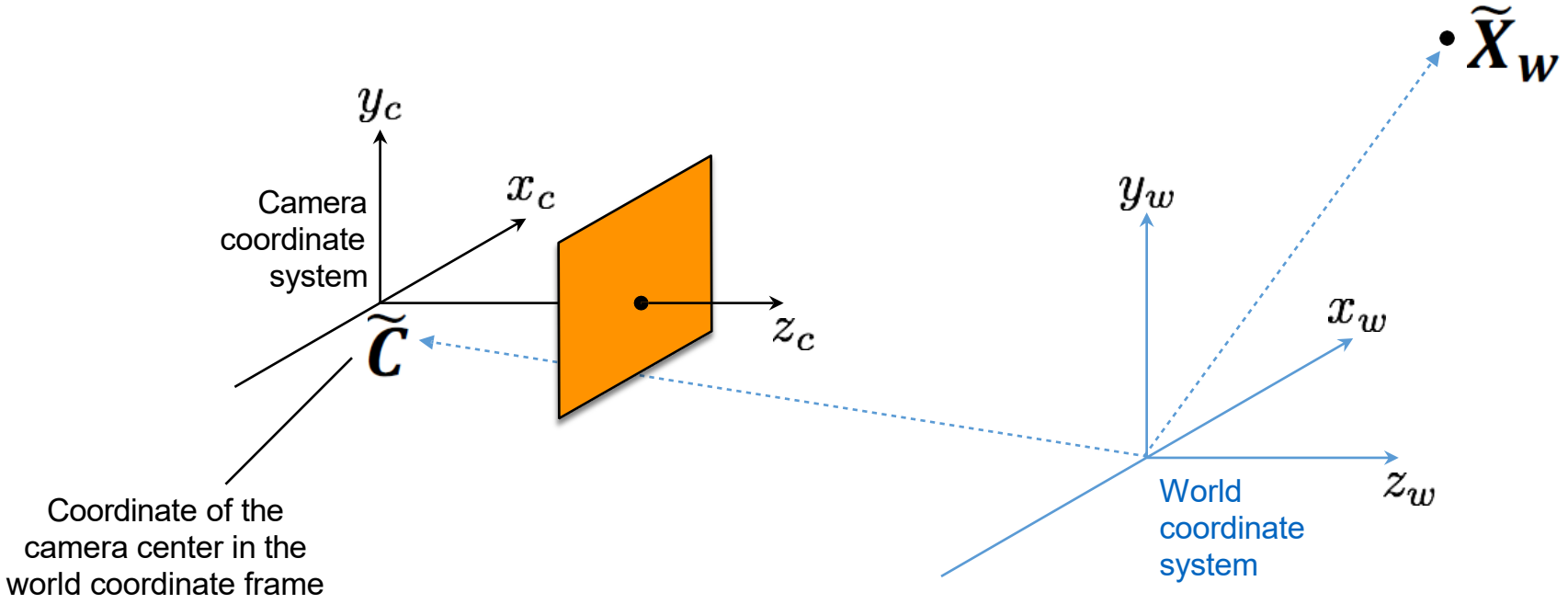
World-to-camera coordinate system transformation



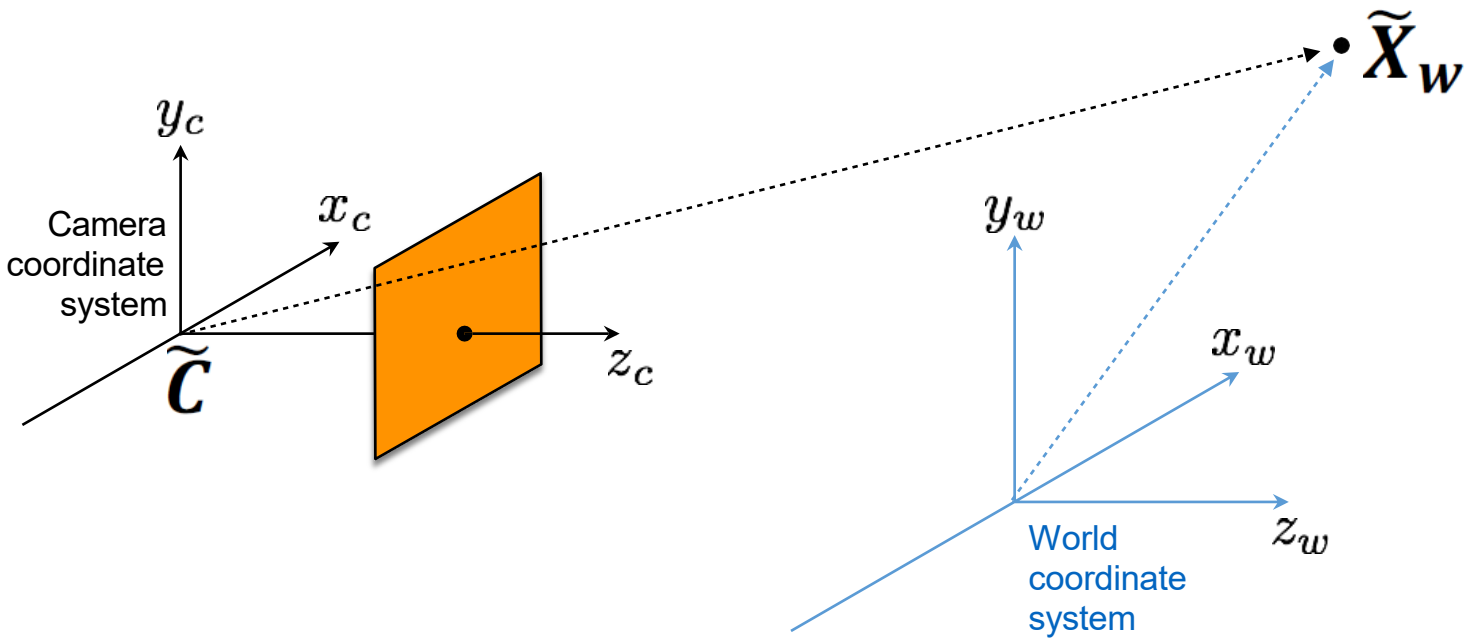


tilde means *heterogeneous* coordinates

World-to-camera coordinate system transformation



World-to-camera coordinate system transformation



points now coincide

$$R \cdot \left(\widetilde{X}_w - \widetilde{C}\right)$$

rotate translate

Modeling the coordinate system transformation

In heterogeneous coordinates, we have:

$$\tilde{\mathbf{X}}_{\mathbf{c}} = \mathbf{R} \cdot \left(\tilde{\mathbf{X}}_{\mathbf{w}} - \tilde{\mathbf{C}} \right)$$

How do we write this transformation in homogeneous coordinates?

Modeling the coordinate system transformation

In heterogeneous coordinates, we have:

$$\tilde{\mathbf{X}}_\mathbf{c} = \mathbf{R} \cdot \left(\tilde{\mathbf{X}}_\mathbf{w} - \tilde{\mathbf{C}} \right)$$

In homogeneous coordinates, we have:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad \text{or} \quad \mathbf{X_c} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{X_w}$$

Incorporating the transform in the camera matrix

The previous camera matrix is for homogeneous 3D coordinates in camera coordinate system:

$$\mathbf{x} = \mathbf{P}\mathbf{X}_\mathbf{c} = \mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{X}_\mathbf{c}$$

We also just derived:

$$\mathbf{X}_\mathbf{c} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{X}_\mathbf{w}$$

Putting it all together

We can write everything into a single projection:

$$\mathbf{x} = \mathbf{P}\mathbf{X}_\mathbf{w}$$

The camera matrix now looks like:

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{I} \mid \mathbf{0}] \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ \mathbf{0} & 1 \end{bmatrix}$$

intrinsic parameters (3 x 3): correspond to camera internals (image-to-image transformation)

perspective projection (3 x 4): maps 3D to 2D points (camera-to-image transformation)

extrinsic parameters (4 x 4): correspond to camera externals (world-to-camera transformation)

Putting it all together

We can write everything into a single projection:

$$\mathbf{x} = \mathbf{P}\mathbf{X}_\mathbf{w}$$

The camera matrix now looks like:

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\mathbf{R} \;\middle|\; -\mathbf{R}\mathbf{C} \right]$$

intrinsic parameters (3 x 3):
correspond to camera internals
(sensor not at f = 1 and origin shift)

extrinsic parameters (3 x 4):
correspond to camera externals
(world-to-image transformation)

General pinhole camera matrix

We can decompose the camera matrix like this:

$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}|-\mathbf{C}]$$

(translate first then rotate)

Another way to write the mapping:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

where $\mathbf{t} = -\mathbf{R}\mathbf{C}$

(rotate first then translate)

General pinhole camera matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{array}\right]$$

intrinsic parameters

extrinsic parameters

$$\mathbf{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \qquad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

3D rotation

3D translation

Recap

What is the size and meaning of each term in the camera matrix?

$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}|-\mathbf{C}]$$

? ? ? ?

Recap

What is the size and meaning of each term in the camera matrix?

$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}|-\mathbf{C}]$$

3x3 intrinsics

?

?

?

Recap

What is the size and meaning of each term in the camera matrix?

$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}|-\mathbf{C}]$$

3x3 intrinsics

3x3 3D rotation

?

?

Recap

What is the size and meaning of each term in the camera matrix?

$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}|-\mathbf{C}]$$

3x3 intrinsics

3x3 3D rotation

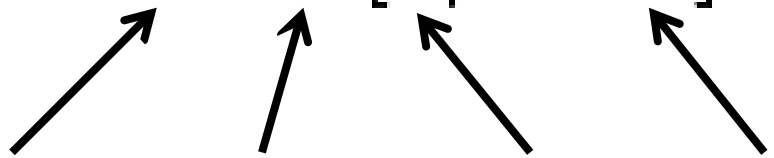
3x3 identity

?

Recap

What is the size and meaning of each term in the camera matrix?

$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}|-\mathbf{C}]$$



3x3	3x3	3x3	3x1
intrinsics	3D rotation	identity	3D translation

Quiz

The camera matrix relates what two quantities?

Quiz

The camera matrix relates what two quantities?

$$\boldsymbol{x} = \mathbf{PX}$$

homogeneous 3D points to 2D image points

Quiz

The camera matrix relates what two quantities?

$$\boldsymbol{x} = \mathbf{PX}$$

homogeneous 3D points to 2D image points

The camera matrix can be decomposed into?

Quiz

The camera matrix relates what two quantities?

$$\boldsymbol{x} = \mathbf{P}\mathbf{X}$$

homogeneous 3D points to 2D image points

The camera matrix can be decomposed into?

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

intrinsic and extrinsic parameters

More general camera matrices

The following is the standard camera matrix we saw.

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \Big[\mathbf{R} \;\Big|\; -\mathbf{RC}\Big]$$

More general camera matrices

CCD camera: pixels may not be square.

$$\mathbf{P} = \begin{bmatrix} \alpha_x & 0 & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\mathbf{R} \mid -\mathbf{RC} \right]$$

How many degrees of freedom?

More general camera matrices

CCD camera: pixels may not be square.

$$\mathbf{P} = \begin{bmatrix} \alpha_x & 0 & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\mathbf{R} \mid -\mathbf{RC} \right]$$

How many degrees of freedom?

10 DOF

More general camera matrices

Finite projective camera: sensor be skewed.

$$\mathbf{P} = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\mathbf{R} \mid -\mathbf{RC} \right]$$

How many degrees of freedom?

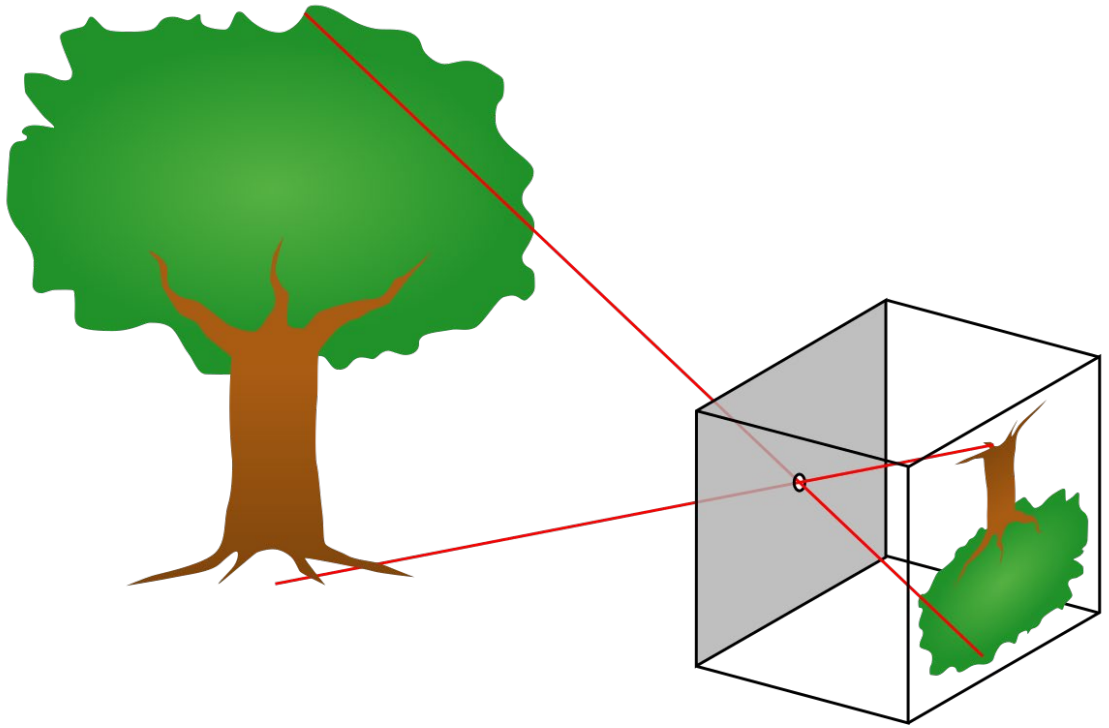
More general camera matrices

Finite projective camera: sensor be skewed.

$$\mathbf{P} = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\mathbf{R} \mid -\mathbf{RC} \right]$$

How many degrees of freedom?

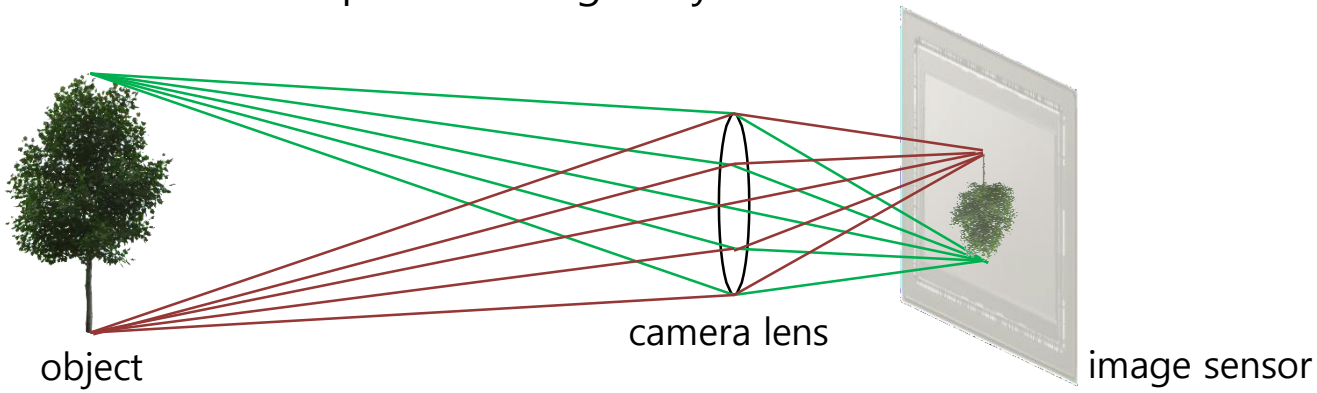
11 DOF



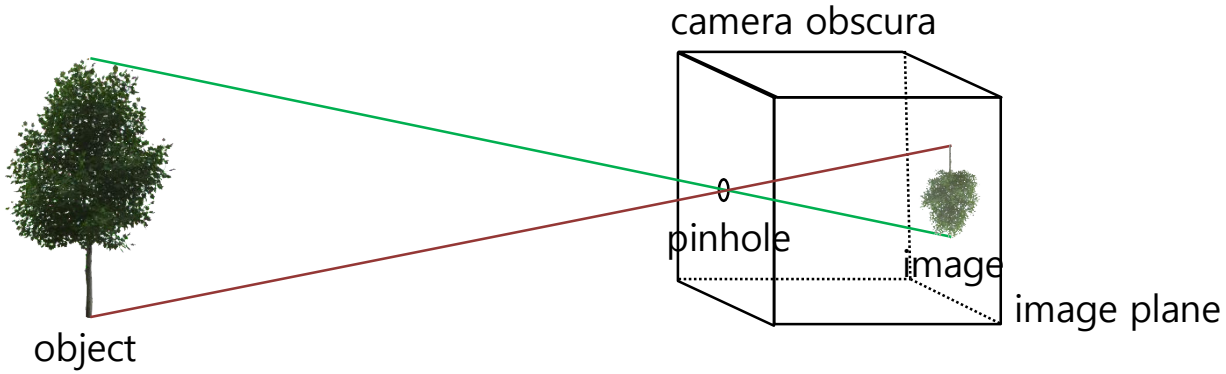
Recap

Pinhole Camera Model

- **Real camera with a lens**
  - Q: Why does a camera use a lens? To acquire more light rays

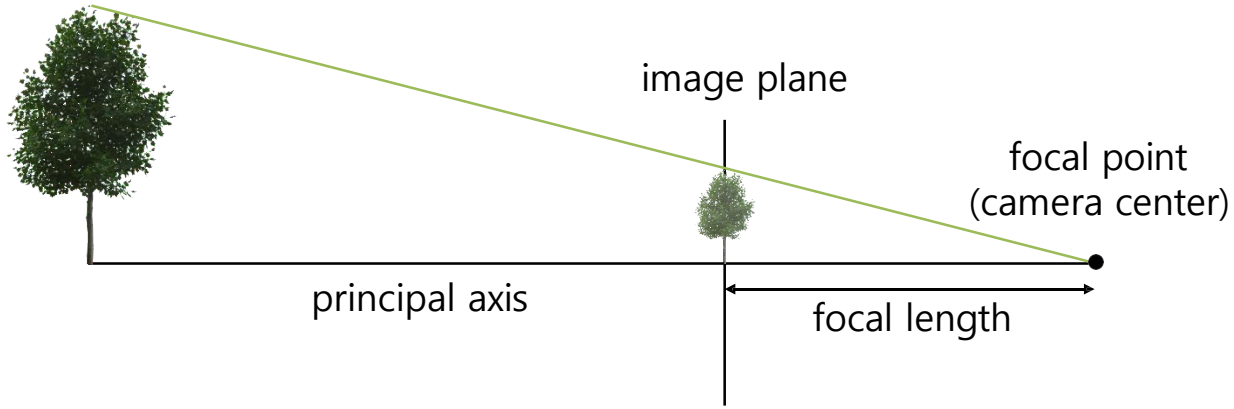


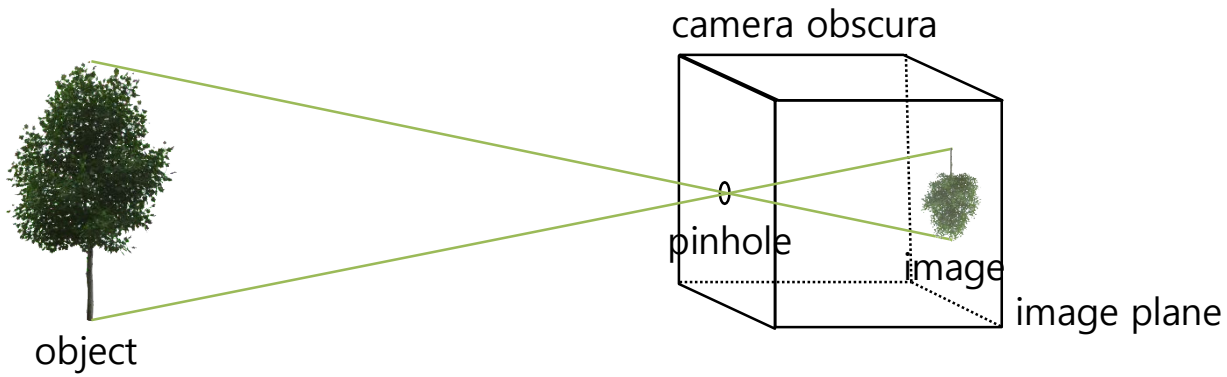
- **Pinhole camera model**



Pinhole Camera Model

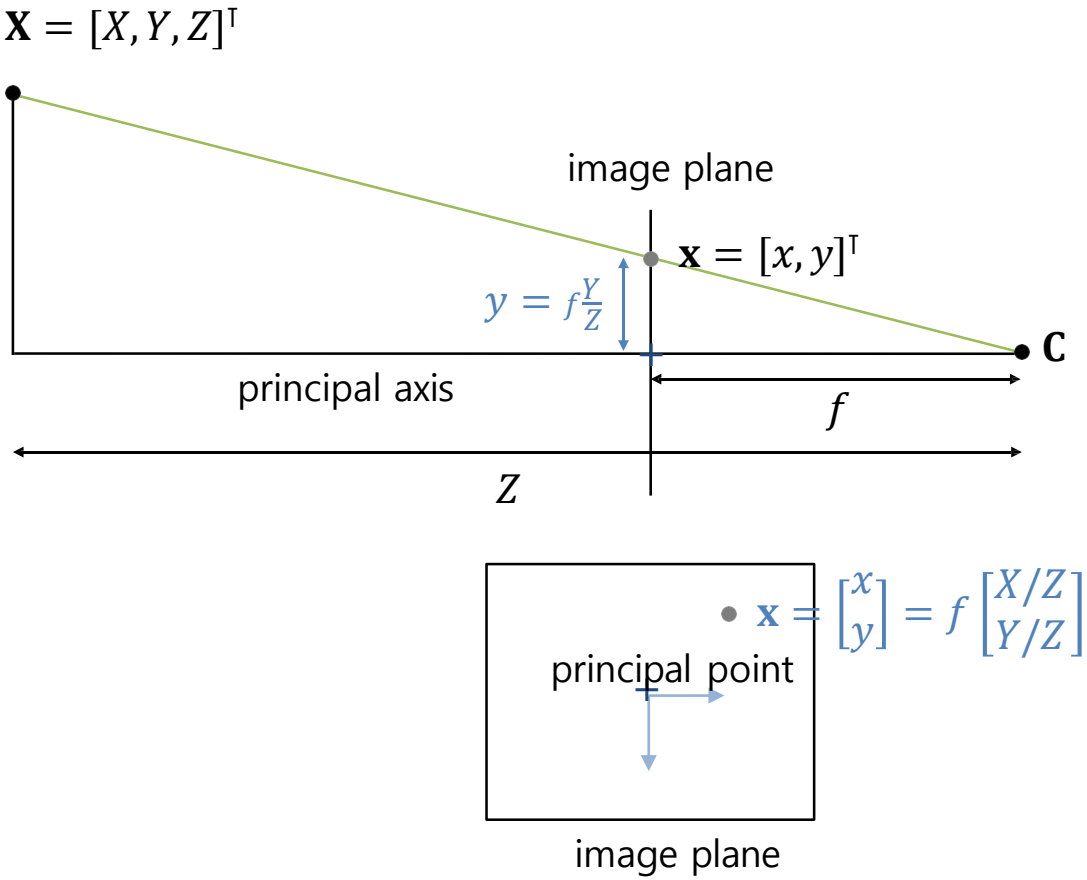
- **Pinhole camera model**
  - In conclusion (without lens distortion), $\mathbf{x} = \mathrm{P}\mathbf{X} \quad (\mathrm{P} = \mathrm{K}[\mathrm{R} \mid \mathbf{t}])$





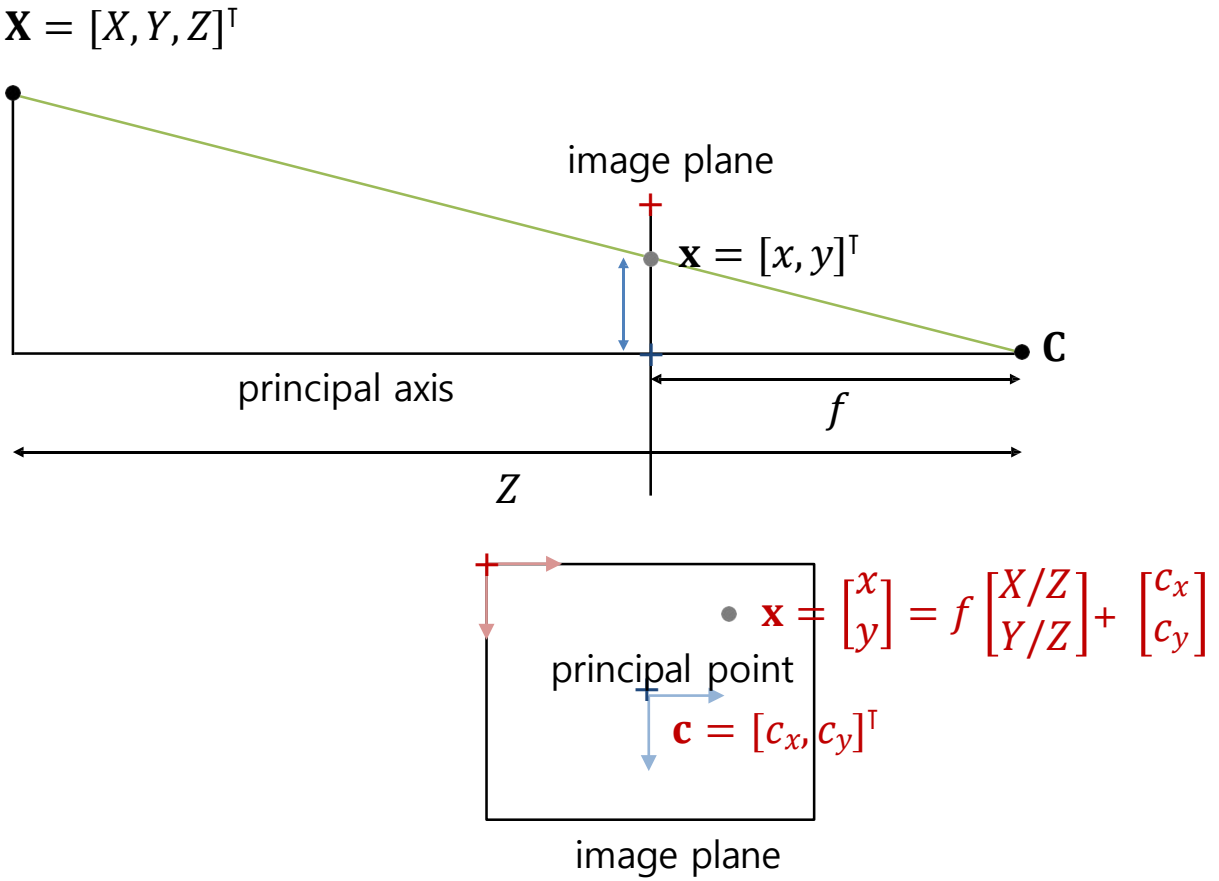
Pinhole Camera Model

- Pinhole camera model



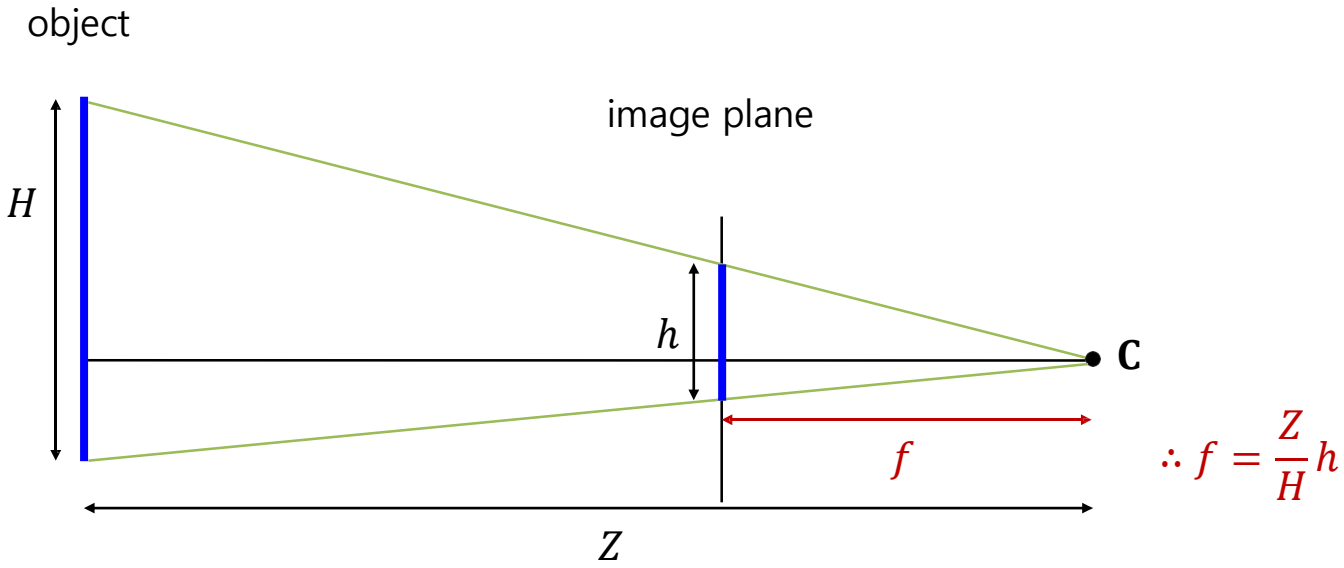
Pinhole Camera Model

- Pinhole camera model



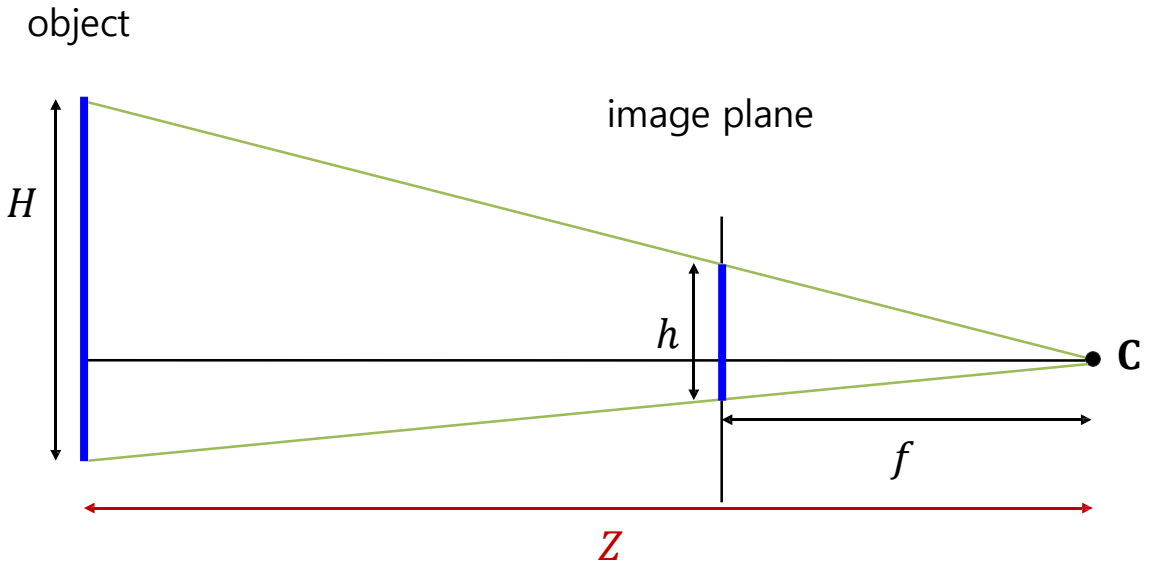
Pinhole Camera Model

- Example: **Simple camera calibration**
  - Unknown: Focal length (f) of the camera (unit: [pixel])
  - Given: The observed object height (h) on the image plane (unit: [pixel])
  - Assumptions
    - The object height (H) and distance (Z) from the camera are known.
    - The object is aligned with the image plane.



Pinhole Camera Model

- Example: **Simple depth estimation** (object localization)
  - Unknown: Object distance (Z) from the camera (unit: [m])
  - Given: The observed object height (h) on the image plane (unit: [pixel])
  - Assumptions
    - The object height (H) and focal length (f) are known.
    - The object is aligned with the image plane.

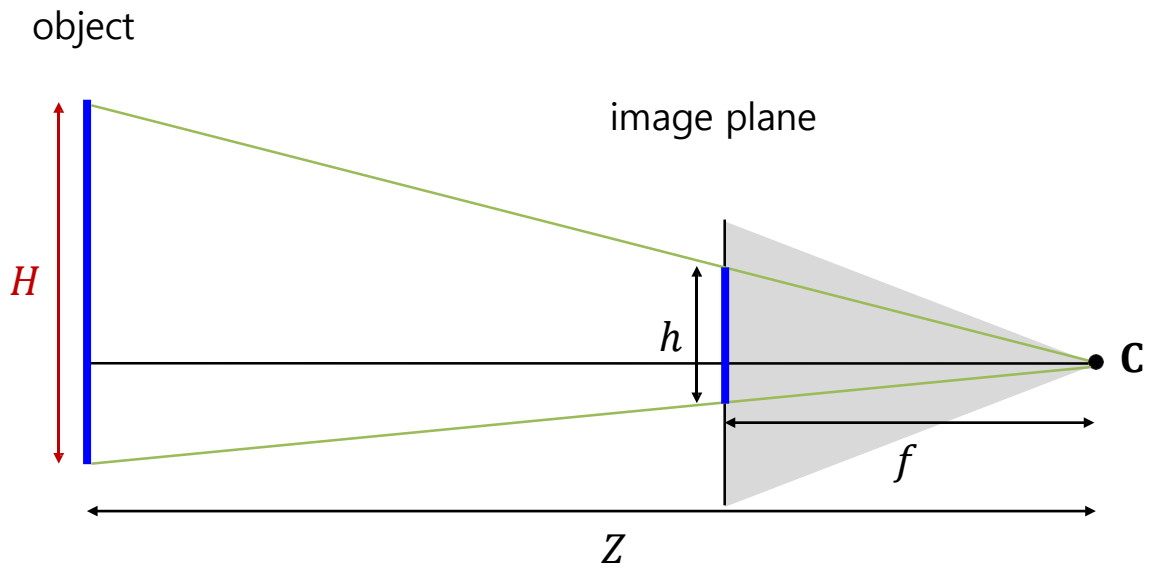


$$\therefore Z = \frac{f}{h} H$$

Pinhole Camera Model

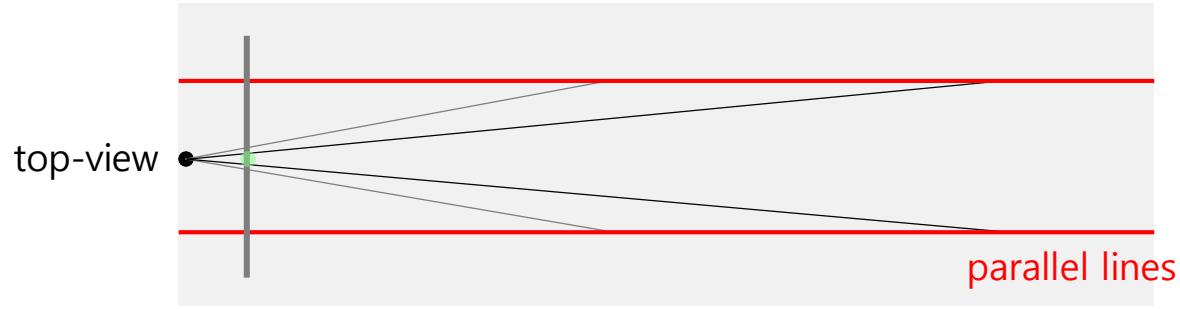
- Example: **Simple object measurement**
  - Unknown: Object height (H) (unit: [m])
  - Given: The observed object height (h) on the image plane (unit: [pixel])
  - Assumptions
    - The object distance (Z) from the camera and focal length (f) are known.
    - The object is aligned with the image plane.

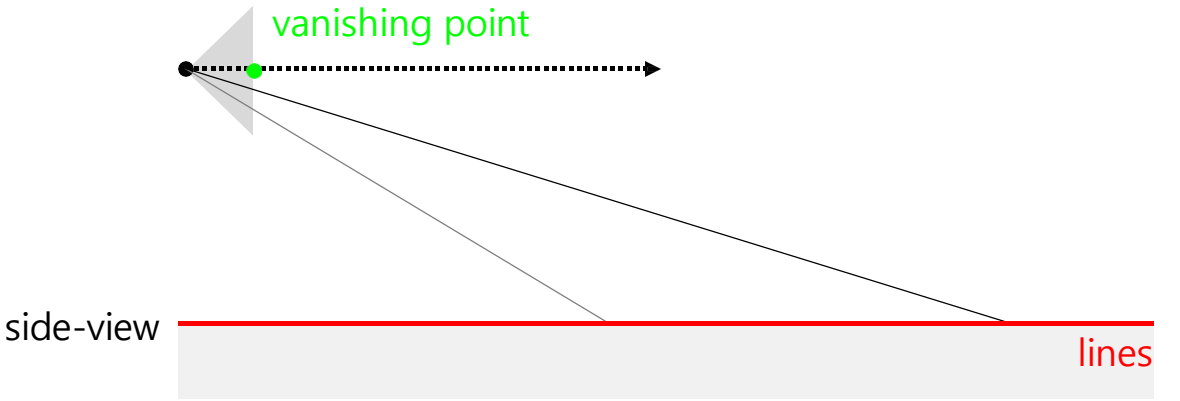
$$\therefore H = \frac{h}{f} Z$$



Pinhole Camera Model

- **Vanishing points**
  - A point on the image plane where mutually parallel lines (in 3D space) intersect
    - A vector to the vanishing point is parallel to the lines.
    - A vector to the vanishing point is parallel to the reference plane made by the lines.





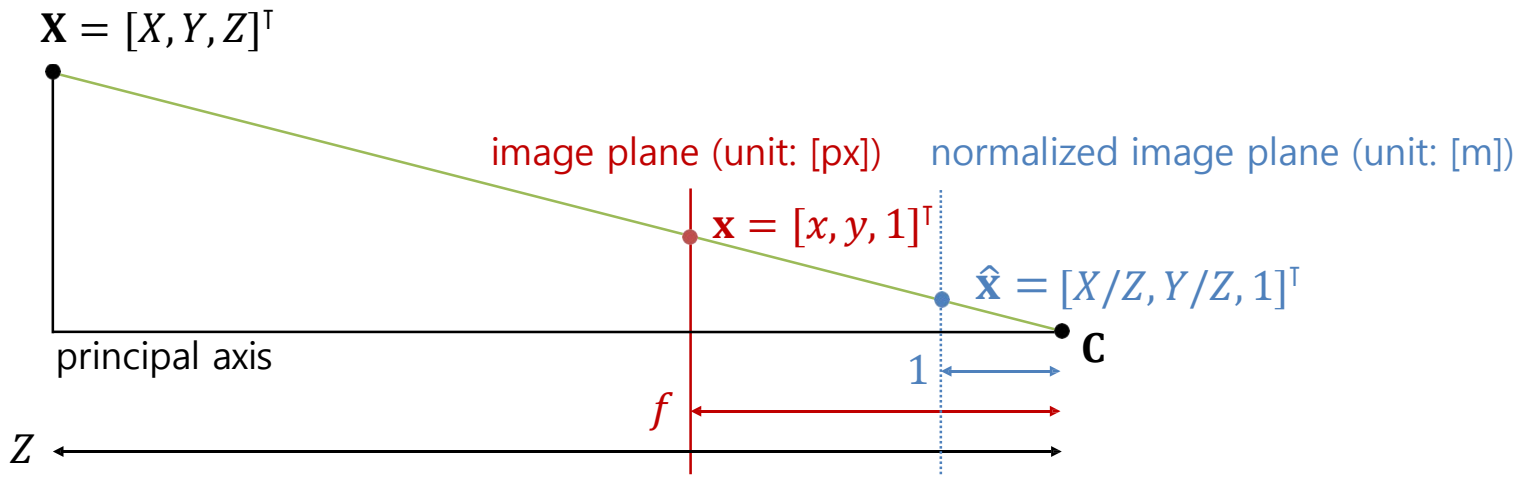
Pinhole Camera Model

- **Camera matrix** K

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = f \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \rightarrow \mathbf{x} = \mathrm{K}\hat{\mathbf{x}} \text{ where } \mathrm{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \text{ and } \hat{\mathbf{x}} = \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix}$$

$$\text{Simplified as } \mathrm{K} = \begin{bmatrix} f & 0 & w/2 \\ 0 & f & h/2 \\ 0 & 0 & 1 \end{bmatrix} \quad (w\text{: image width, } h\text{: image height})$$

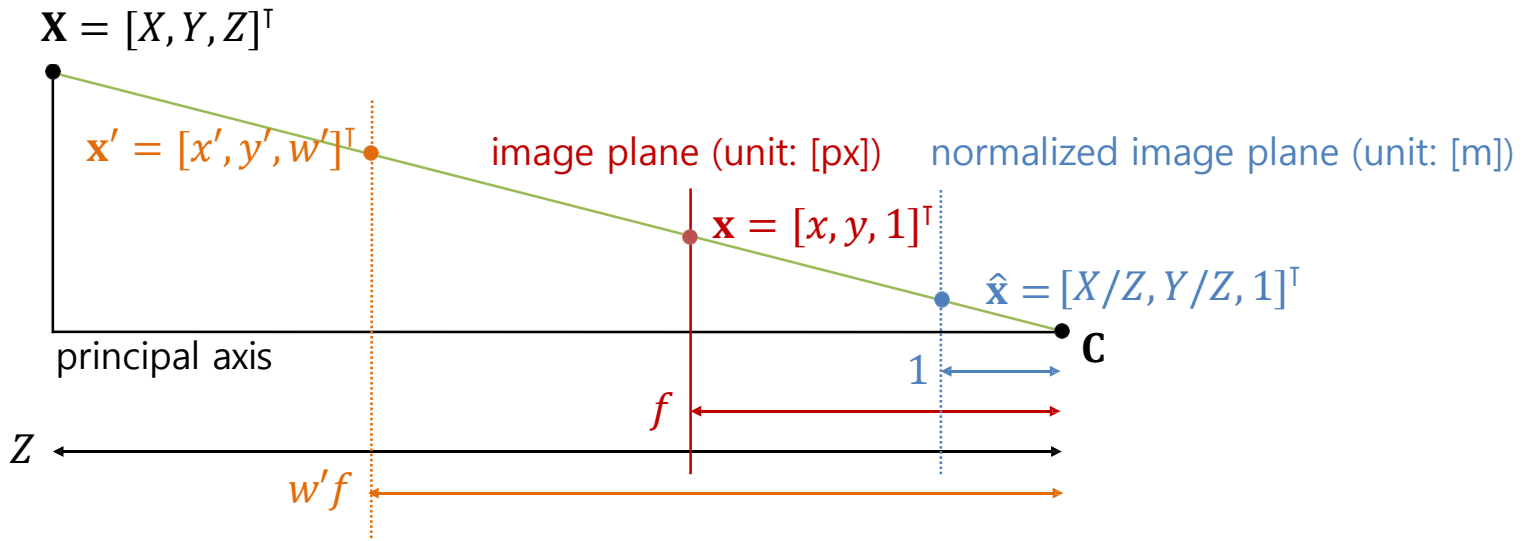
$$\text{Generalized as } \mathrm{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (s\text{: skew parameter})$$



Pinhole Camera Model

- **Homogeneous coordinates** (a.k.a. projective coordinates)
  - It describes n-dimensional project space as $n+1$-dimensional coordinate system.
  - It hold non-conventional equivalence relationship: $(x_1, x_2, \ldots, x_{n+1}) \sim (\lambda x_1, \lambda x_2, \ldots, \lambda x_{n+1})$ such that $(0 \neq \lambda \in \mathbb{R})$.
    - e.g. (5, 12) is written as (5, 12, 1) which is also equal to (10, 24, 2) or (15, 36, 3) or …
  - On the previous slide, $\mathbf{x} = \mathrm{K}\hat{\mathbf{x}}$ where $\mathrm{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, and $\hat{\mathbf{x}} = \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix}$

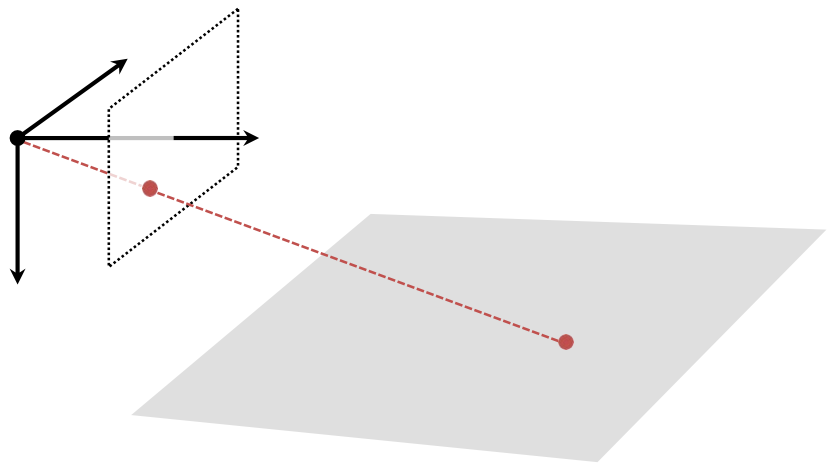
$$\mathbf{x}' = \mathrm{K}\mathbf{X} \quad \text{where} \quad \mathrm{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{x}' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix}, \text{ and } \quad \mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (\text{Note: } \mathbf{x} = \tfrac{1}{w'}\mathbf{x}')$$

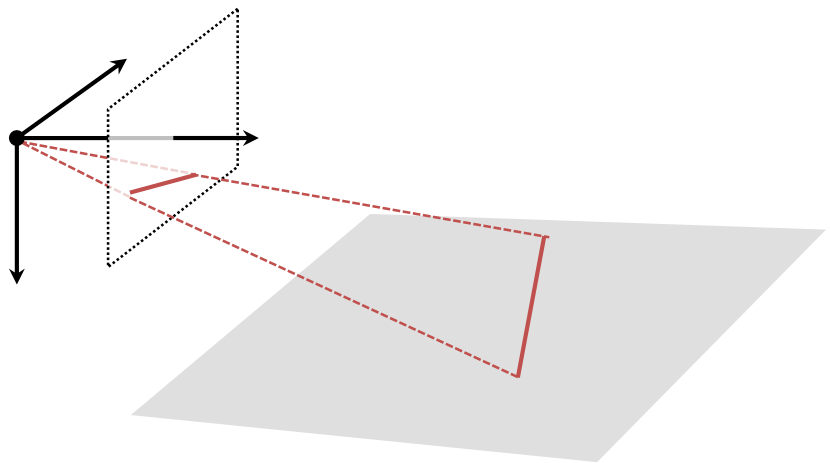


Pinhole Camera Model

- **Why homogeneous coordinates?**
  - An affine transformation ($\mathbf{y} = \mathrm{A}\mathbf{x} + \mathbf{b}$) is formulated by a single matrix multiplication.
  - A point at infinity (a.k.a. ideal point) is numerically represented by $w = 0$.
  - A point and line ($ax + by + c = 0$) are described beautifully as like $\mathbf{l}^{\mathrm{T}}\mathbf{x} = 0$ or $\mathbf{x}^{\mathrm{T}}\mathbf{l} = 0$ ($\mathbf{l} = [a, b, c]^{\mathrm{T}}$).
    - Intersection of two lines: $\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2$
    - A line by two points: $\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2$

    } *Duality* of a point and line
  - A light ray (line at the camera center) is observed as a point on the image plane.
    - A plane at the camera center is observed as a line on the image plane.
    - A conic whose peak is at the camera center is observed as a conic section on the image plane.

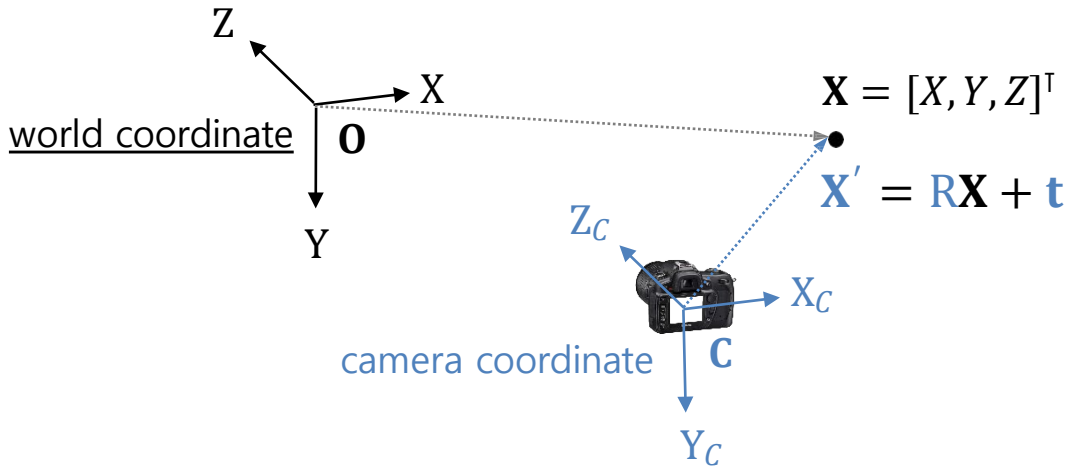




Pinhole Camera Model

- **Projection matrix** P
  - Generally, a point $\mathbf{X}$ is not based on the camera coordinate so that it need be transformed to the camera coordinate.

$$\mathbf{X}' = \mathrm{R}\mathbf{X} + \mathbf{t} \;\rightarrow\; \mathbf{X}' = [\mathrm{R} \mid \mathbf{t}] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$



  - The whole camera projection from the world coordinate to the image coordinate:

$$\mathbf{x} = \mathrm{K}\mathbf{X}' = \mathrm{K}(\mathrm{R}\mathbf{X} + \mathbf{t}) = \mathrm{K}\,[\mathrm{R} \mid \mathbf{t}] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

→ $\mathbf{x} = \mathrm{P}\mathbf{X}$ where $\mathrm{P} = \mathrm{K}\,[\mathrm{R} \mid \mathbf{t}]$ (3x4 matrix), $\mathbf{x}$ and $\mathbf{X}$ in homogenous coordinates

  - Note) The camera pose (R^{T} and $-\mathrm{R}^{\mathrm{T}}\mathbf{t}$) can be derived from the inverse of point transformation (R and $\mathbf{t}$).

$$\mathrm{T}^{-1} = \begin{bmatrix} \mathrm{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathrm{R}^{\mathrm{T}} & -\mathrm{R}^{\mathrm{T}}\mathbf{t} \\ 0 & 1 \end{bmatrix}$$

Pinhole Camera Model

- **Camera parameters** ~ projection matrix P

$$\mathrm{P} = \mathrm{K}\,[\mathrm{R} \mid \mathbf{t}]$$

  - **Intrinsic parameters** ~ camera matrix K
    - e.g. Focal length, principle point, skew, distortion coefficient, …
  - **Extrinsic parameters** ~ point transformation R and $\mathbf{t}$
    - e.g. Rotation and translation