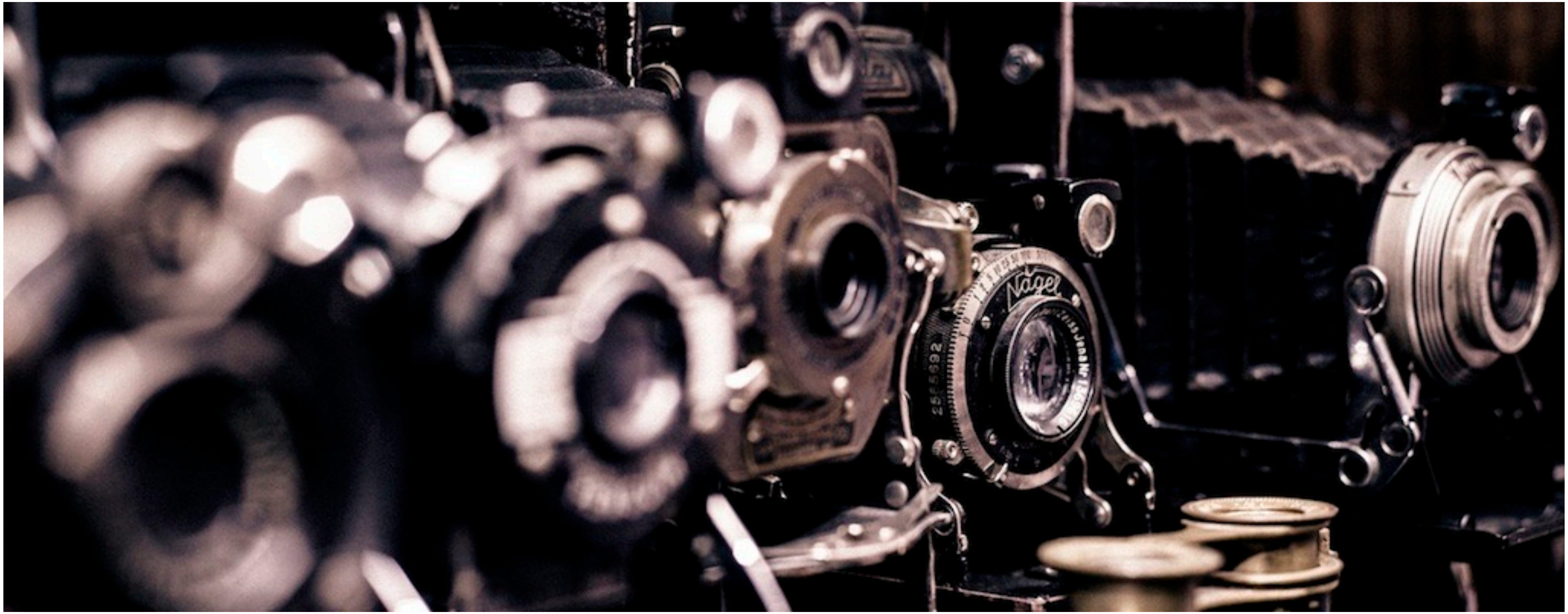
Geometric camera models (cont.)



16-385 Computer Vision
Fall 2022, Lecture 10

http://16385.courses.cs.cmu.edu/

Overview of today’s lecture

- Review of camera matrix.
- Perspective.
- Other camera models.
- Pose estimation.

Slide credits

Most of these slides were adapted from:

- Kris Kitani (15-463, Fall 2016).

Some slides inspired from:

- Fredo Durand (MIT).

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

Perspective distortion

Finite projective camera

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

What does this matrix look like if the camera and world have the same coordinate system?

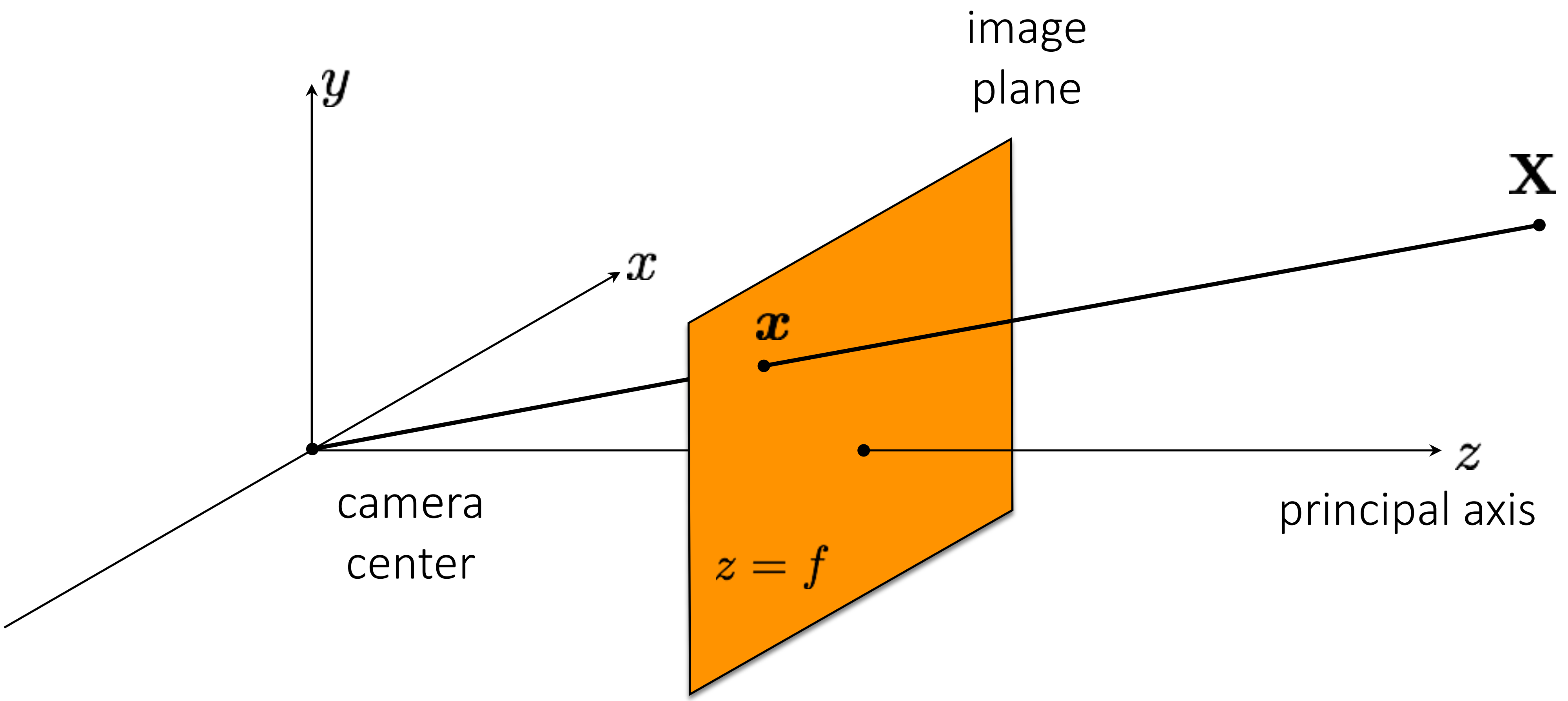
Finite projective camera

The pinhole camera and all of the more general cameras we have seen so far have "*perspective distortion*".

$$\mathbf{P} = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Perspective projection from (homogeneous) 3D to 2D coordinates

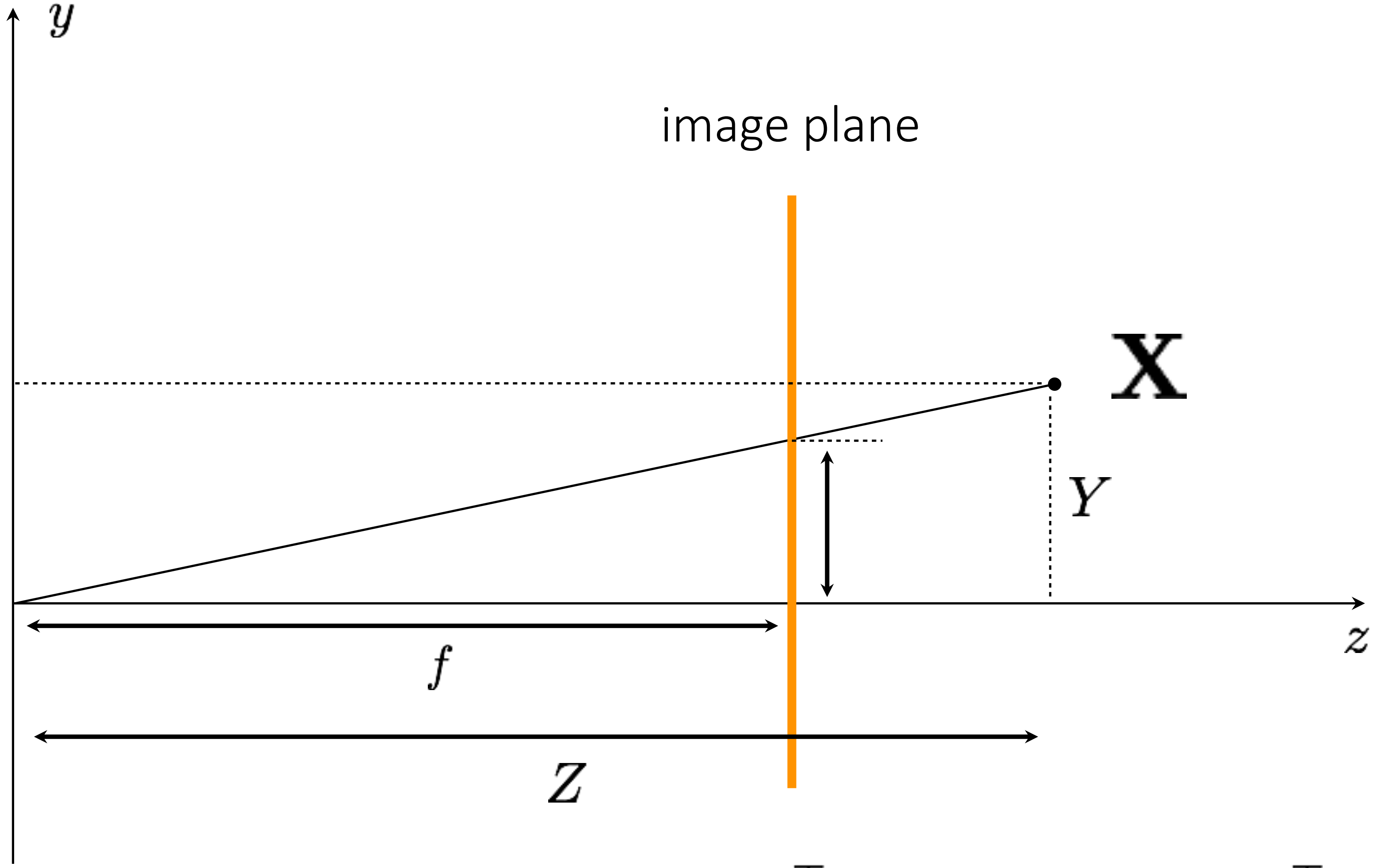
The (rearranged) pinhole camera



Perspective projection in 3D

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

The 2D view of the (rearranged) pinhole camera



Perspective distortion: magnification changes with depth

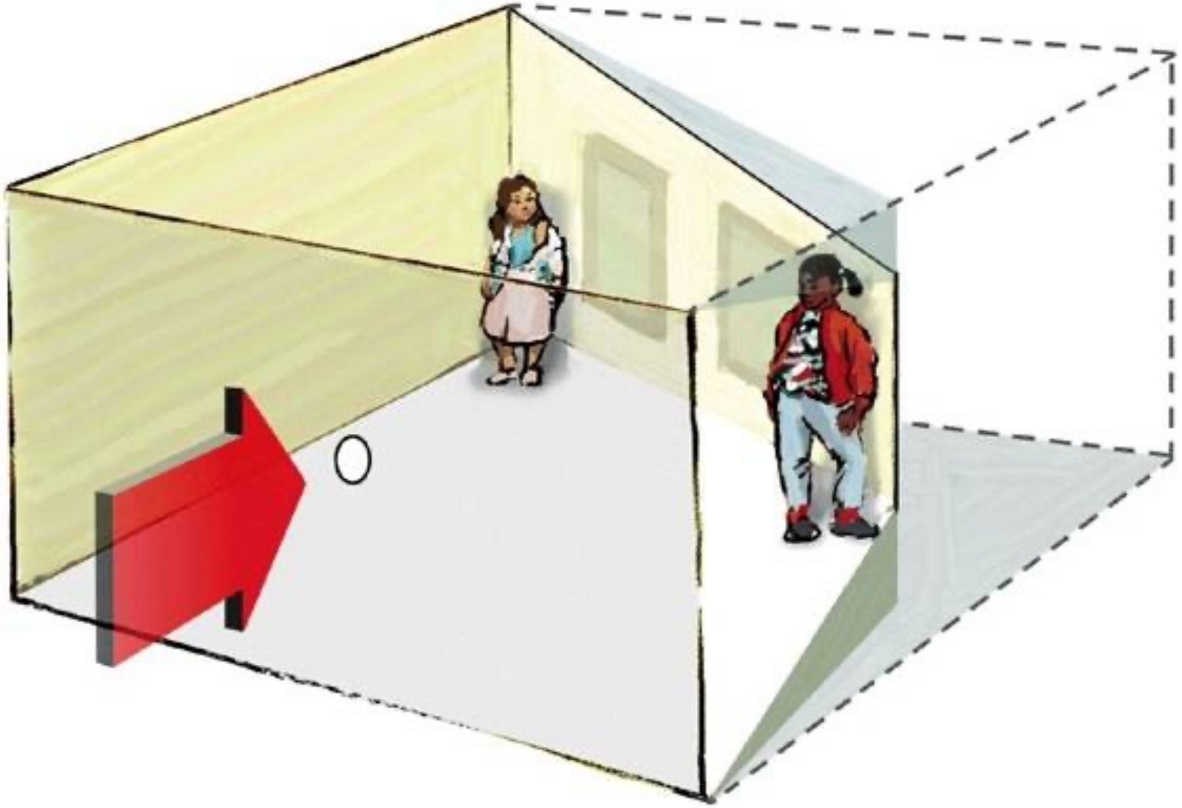
Perspective projection in 2D

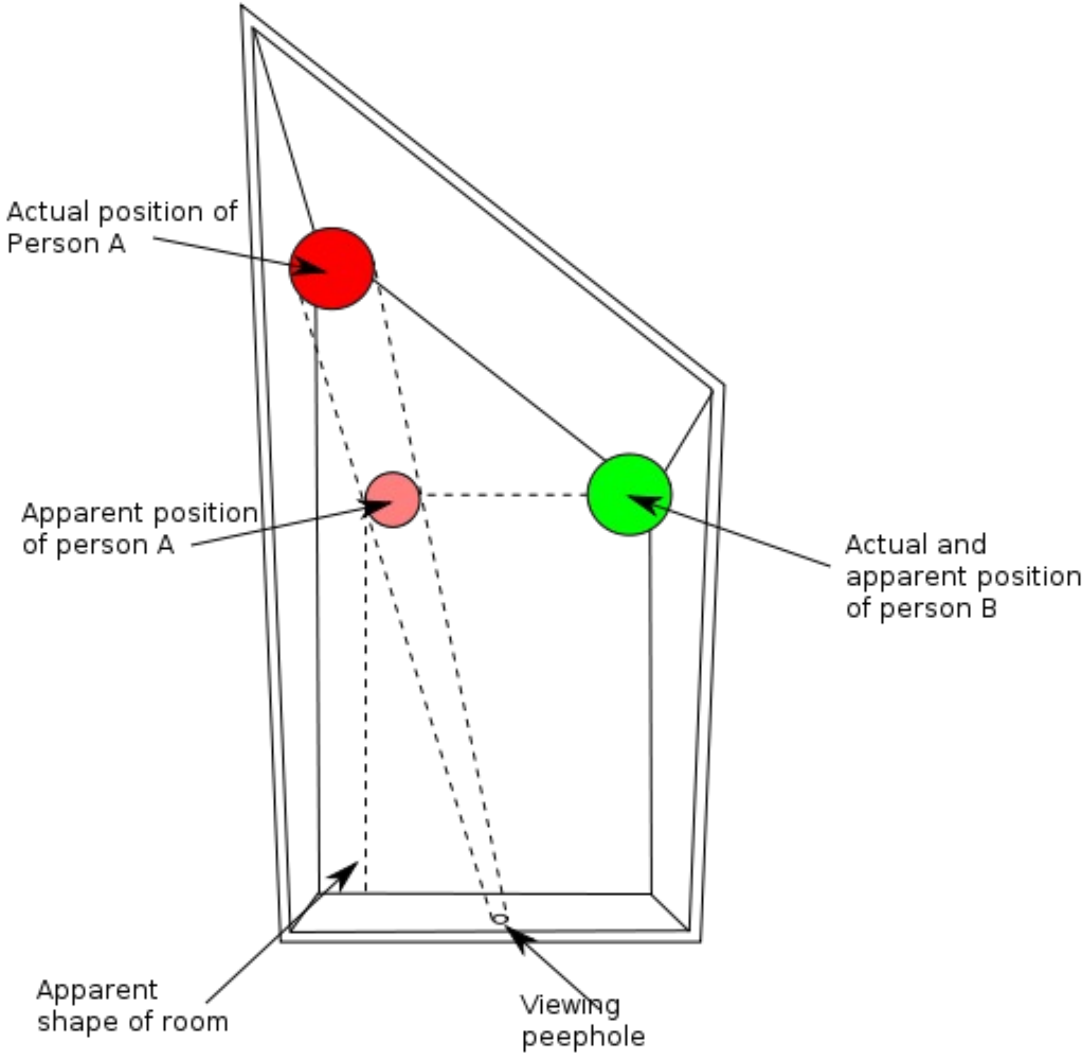
$$[X \quad Y \quad Z]^\top \mapsto [fX/Z \quad fY/Z]^\top$$

Forced perspective

The Ames room illusion

The Ames room illusion





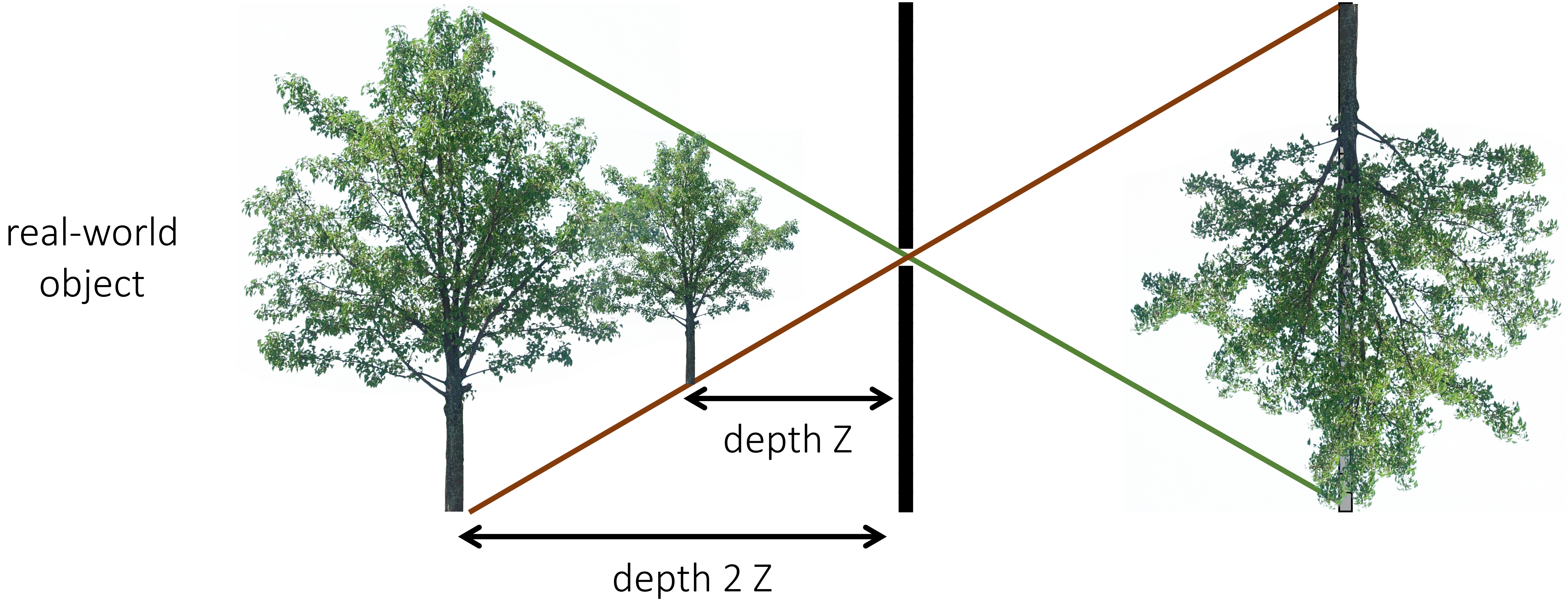
The arrow illusion

Forced Perspective Display

Forced Perspective Display

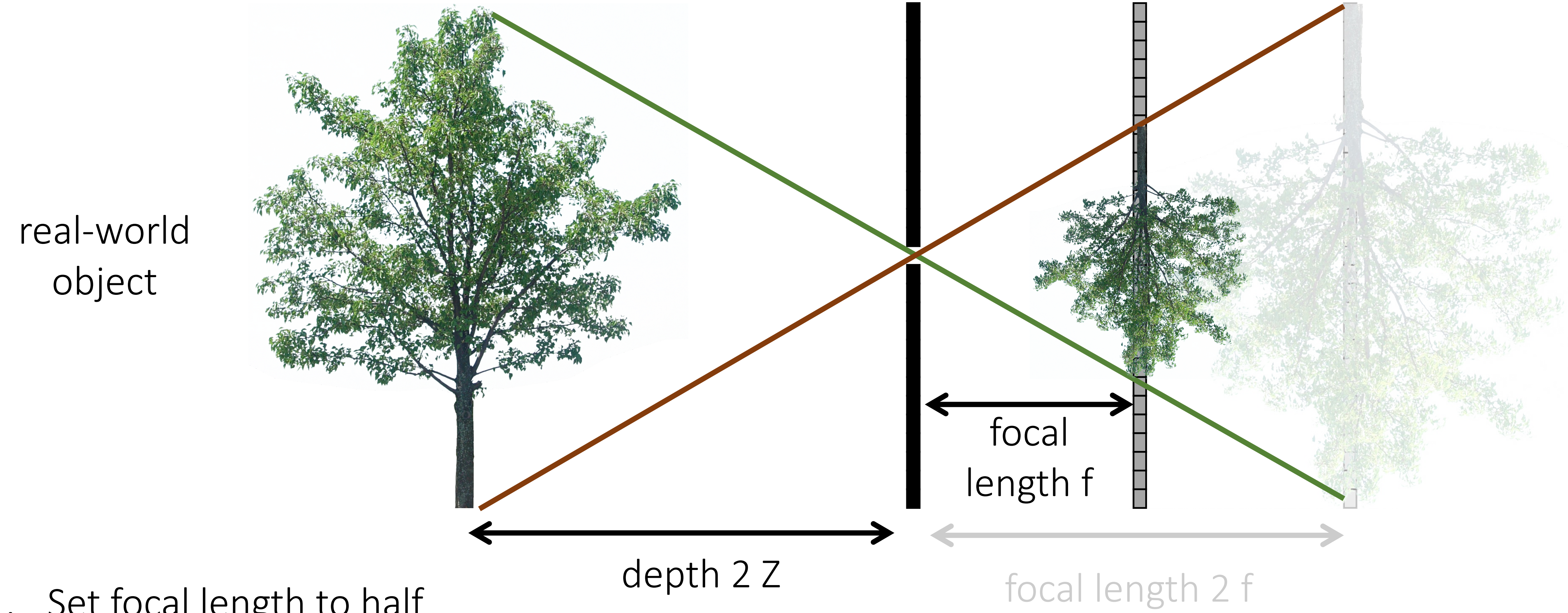
Magnification depends on depth

What happens as we change the focal length?



Magnification depends on focal length

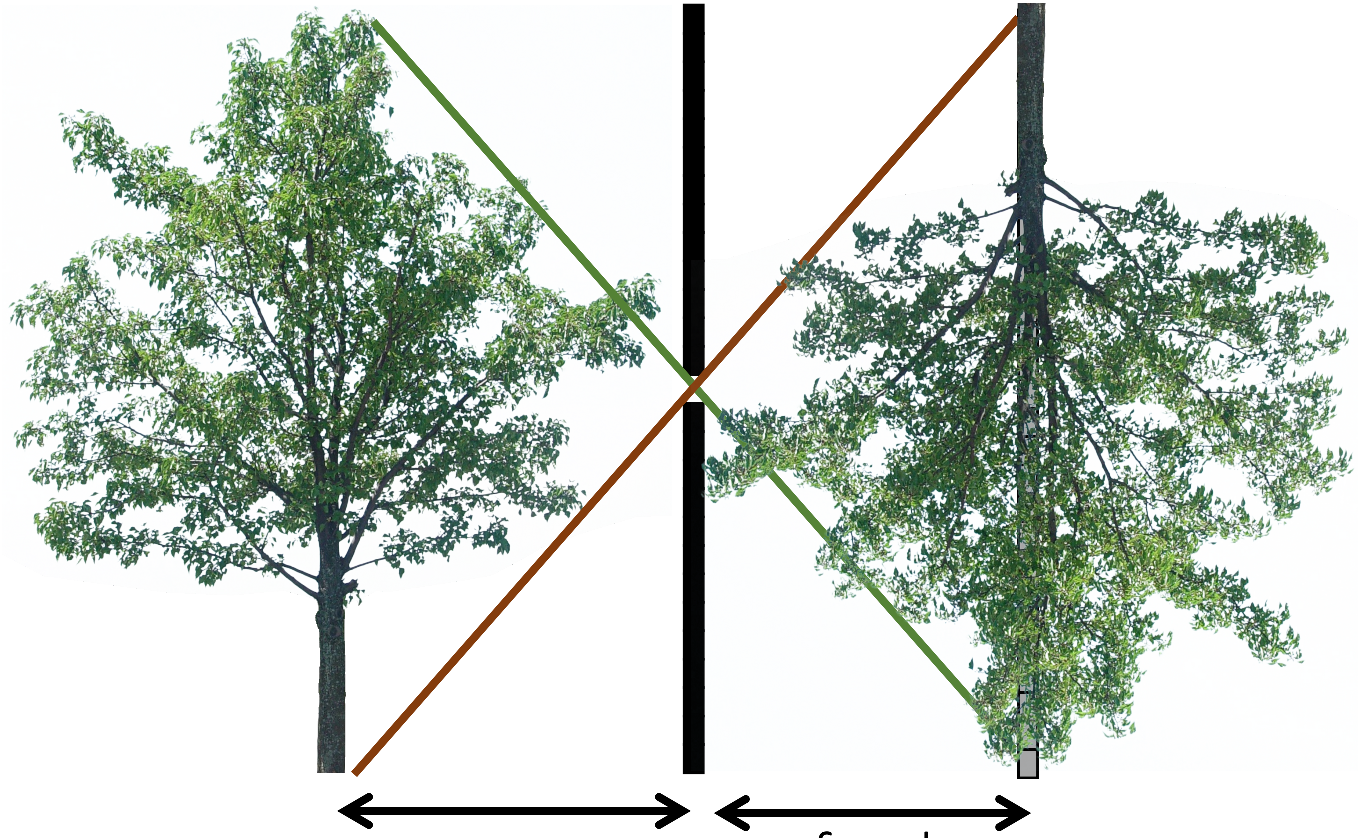
What if...



1. Set focal length to half

What if…

real-world object



Is this the same image as the one I had at focal length 2f and distance 2Z?

1. Set focal length to half
2. Set depth to half

Perspective distortion

long focal length

mid focal length

short focal length

Perspective distortion

Vertigo effect

Named after Alfred Hitchcock's movie

- also known as "dolly zoom"

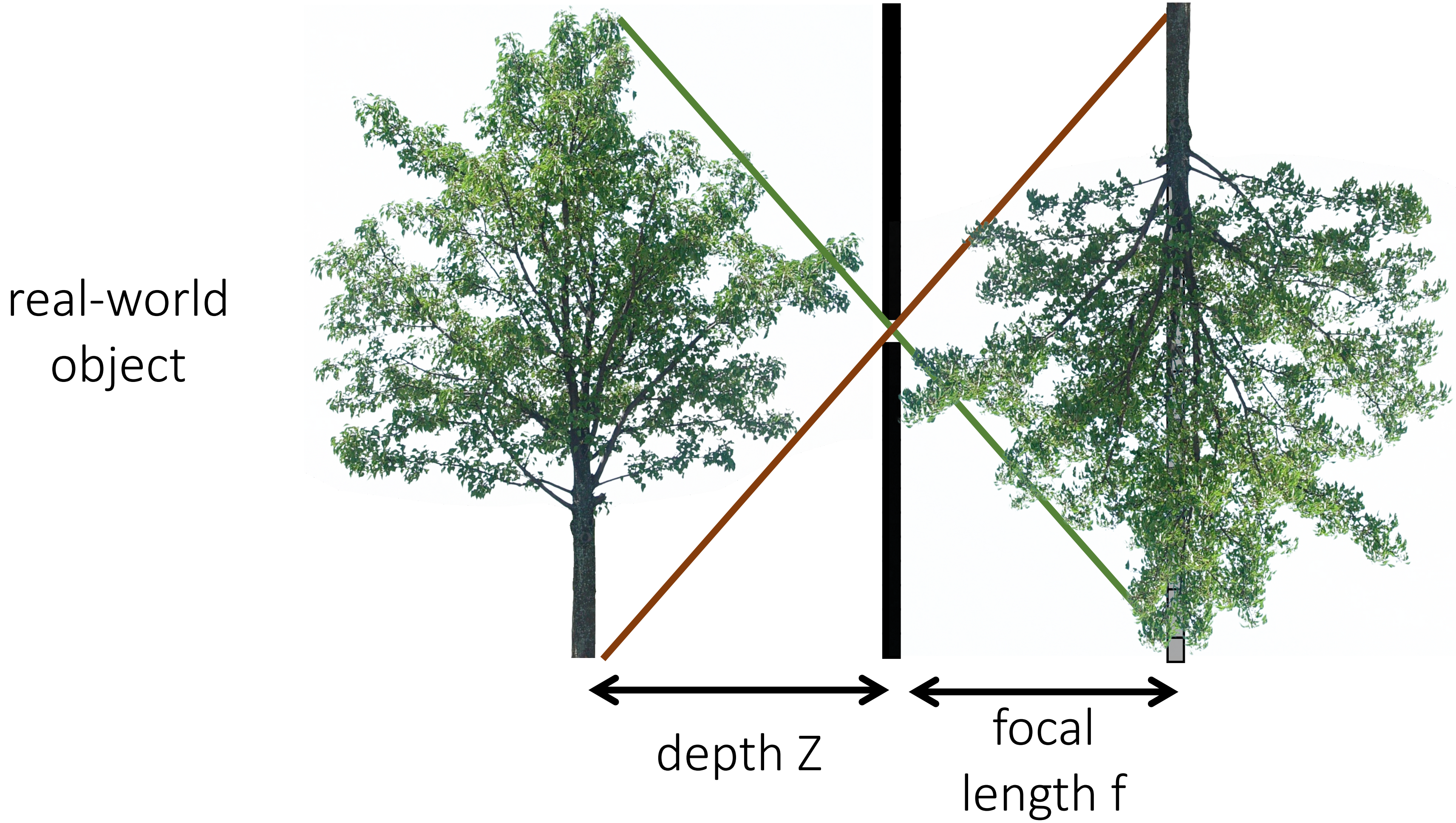
Vertigo effect

How would you create this effect?

Vertigo effect

Other camera models

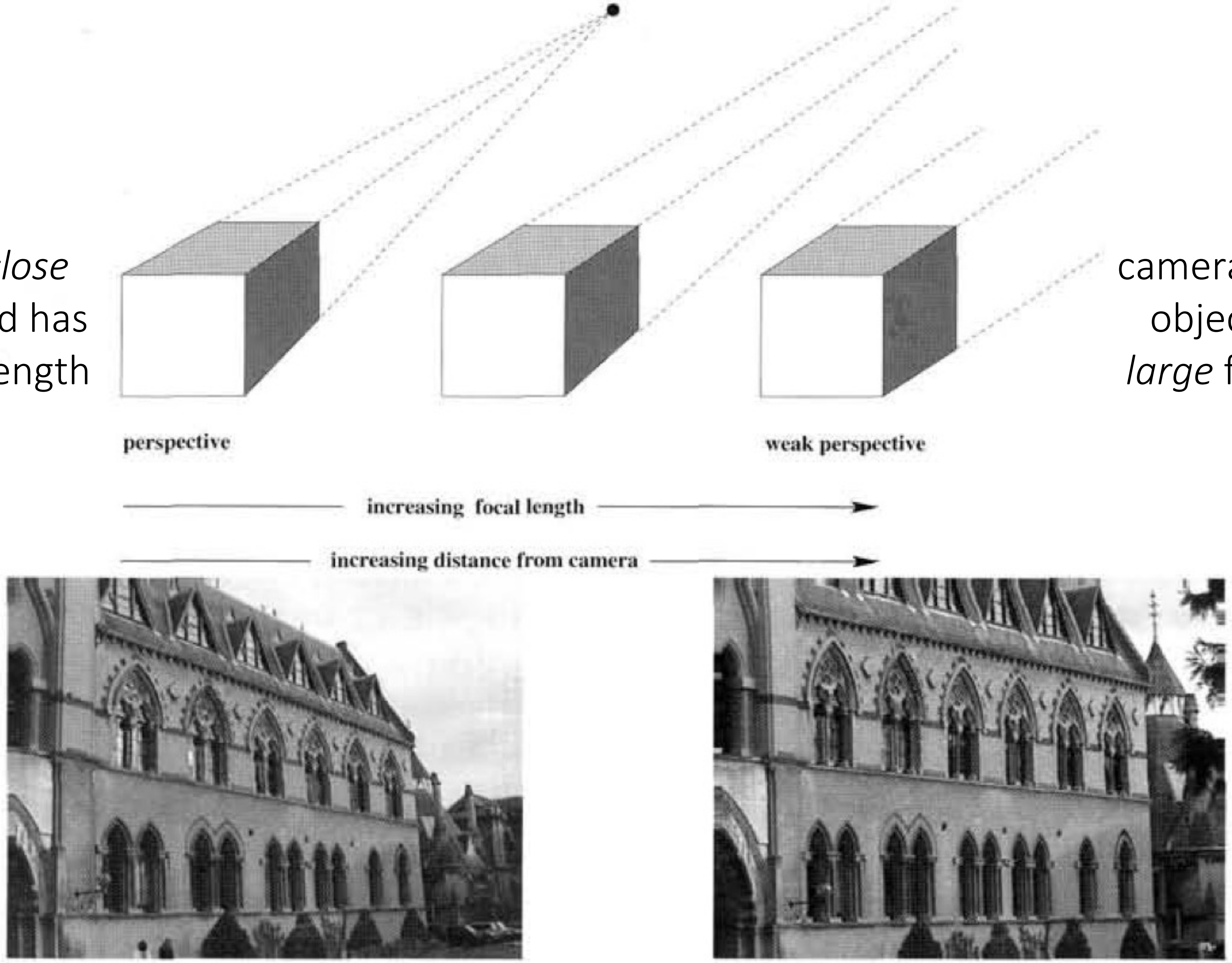
What if…



… we continue increasing Z and f while maintaining same magnification?

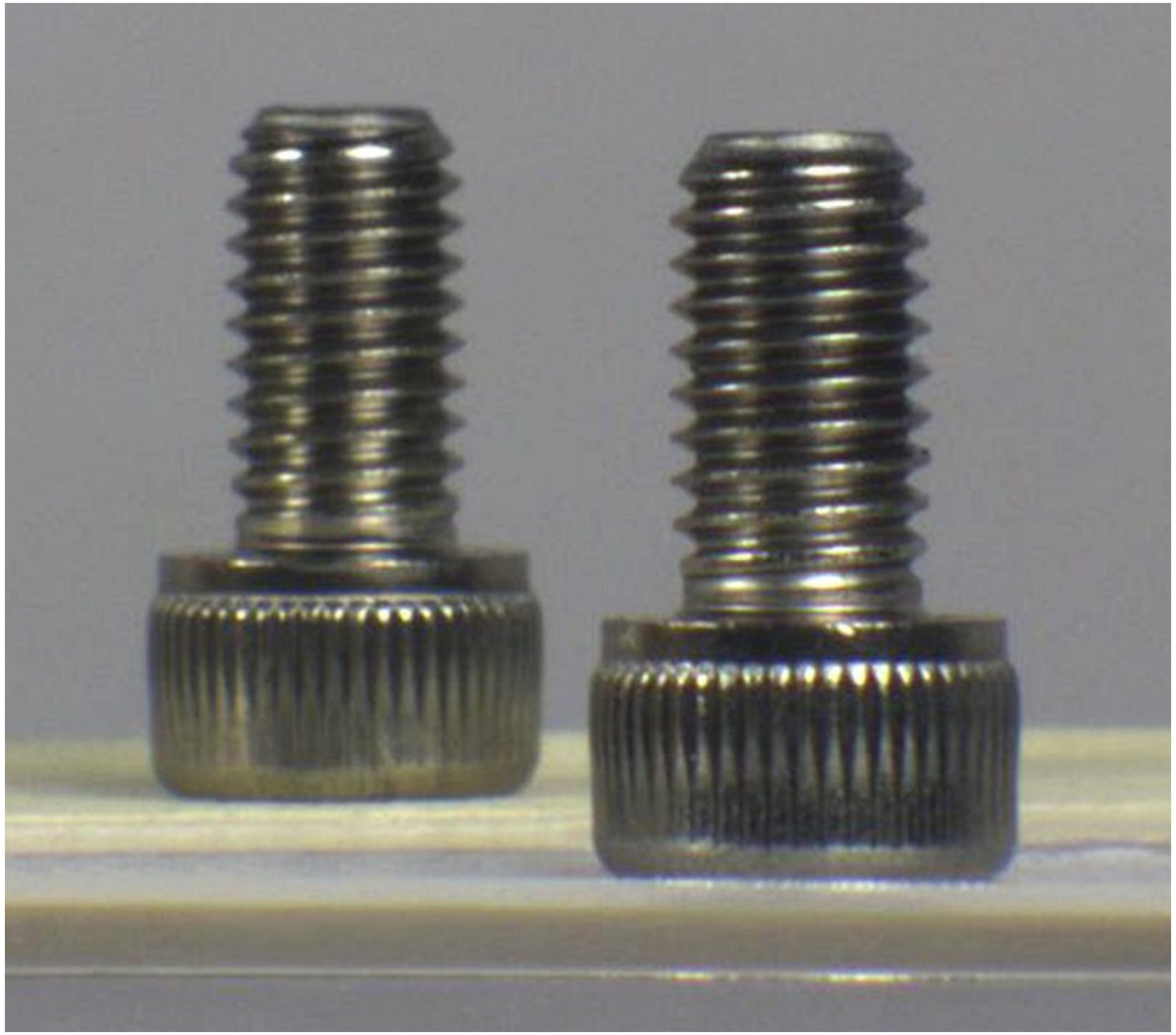
$$f \rightarrow \infty \text{ and } \frac{f}{Z} = \text{constant}$$

camera is *close* to object and has *small* focal length

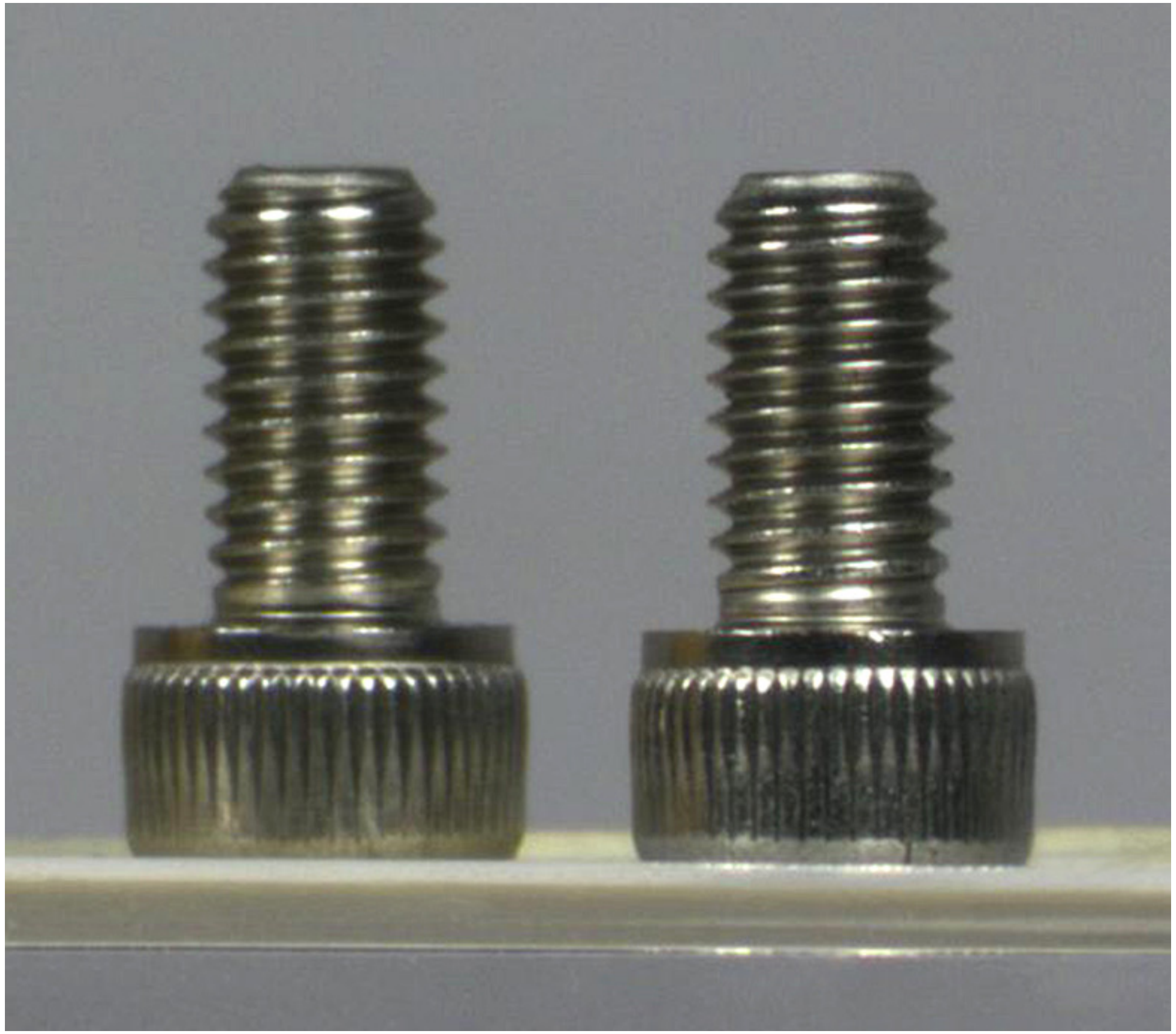


camera is *far* from object and has *large* focal length

Different cameras



perspective camera

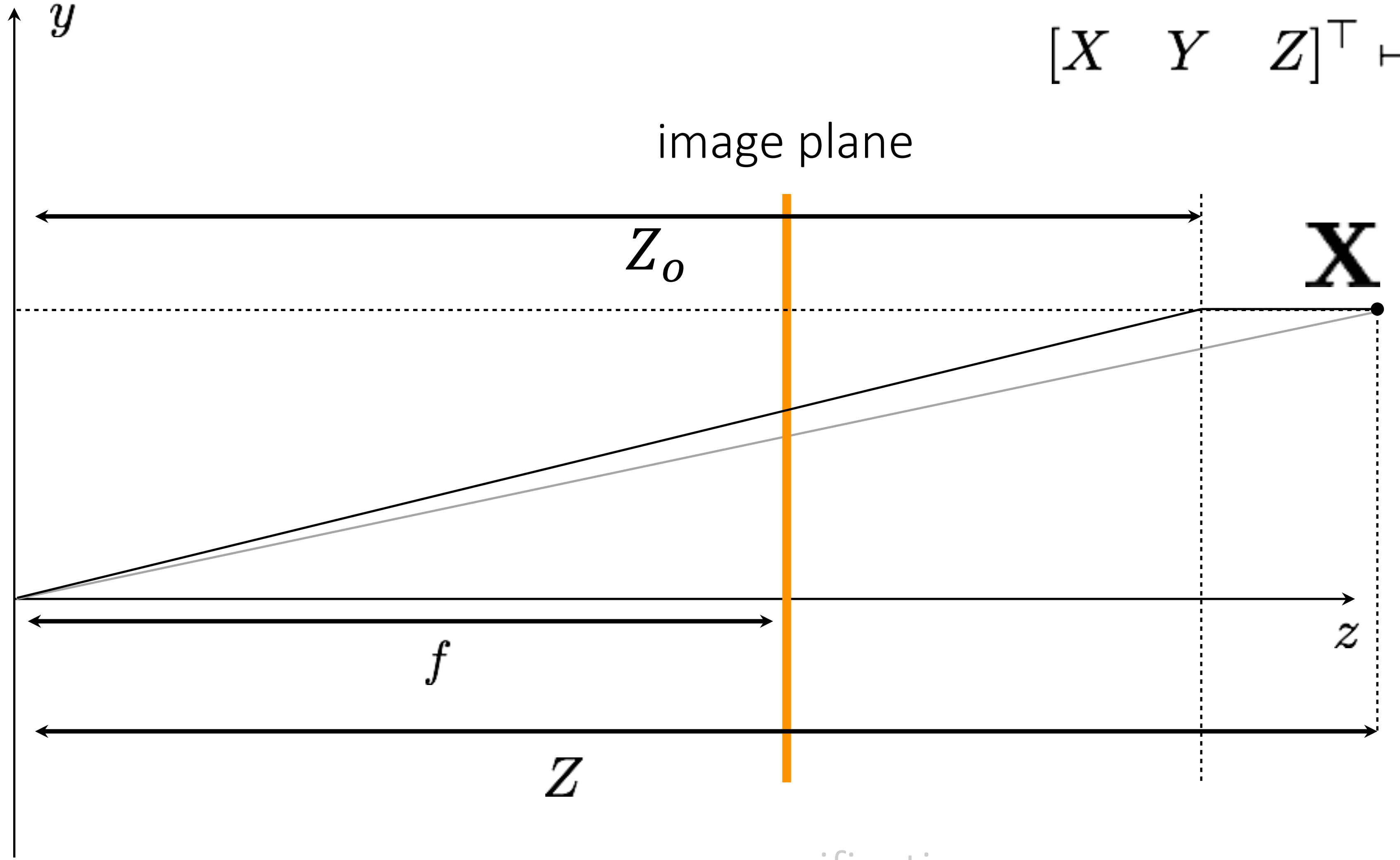


weak perspective camera

Weak perspective vs perspective camera

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

- magnification does not change with depth
- *constant* magnification depending on f and Z_o



magnification changes with depth

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

Comparing camera matrices

Let's assume that the world and camera coordinate systems are the same.

- The *perspective* camera matrix can be written as:

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

- What would the matrix of the weak perspective camera look like?

Comparing camera matrices

Let's assume that the world and camera coordinate systems are the same.

- The *perspective* camera matrix can be written as:

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

- The *weak perspective* camera matrix can be written as:

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

Comparing camera matrices

Let's assume that the world and camera coordinate systems are the same.

- The *finite projective* camera matrix can be written as:

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

where we now have the more general intrinsic matrix

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

- The *affine* camera matrix can be written as:

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

In both cameras, we can incorporate extrinsic parameters same as we did before.

When can we assume a weak perspective camera?

When can we assume a weak perspective camera?

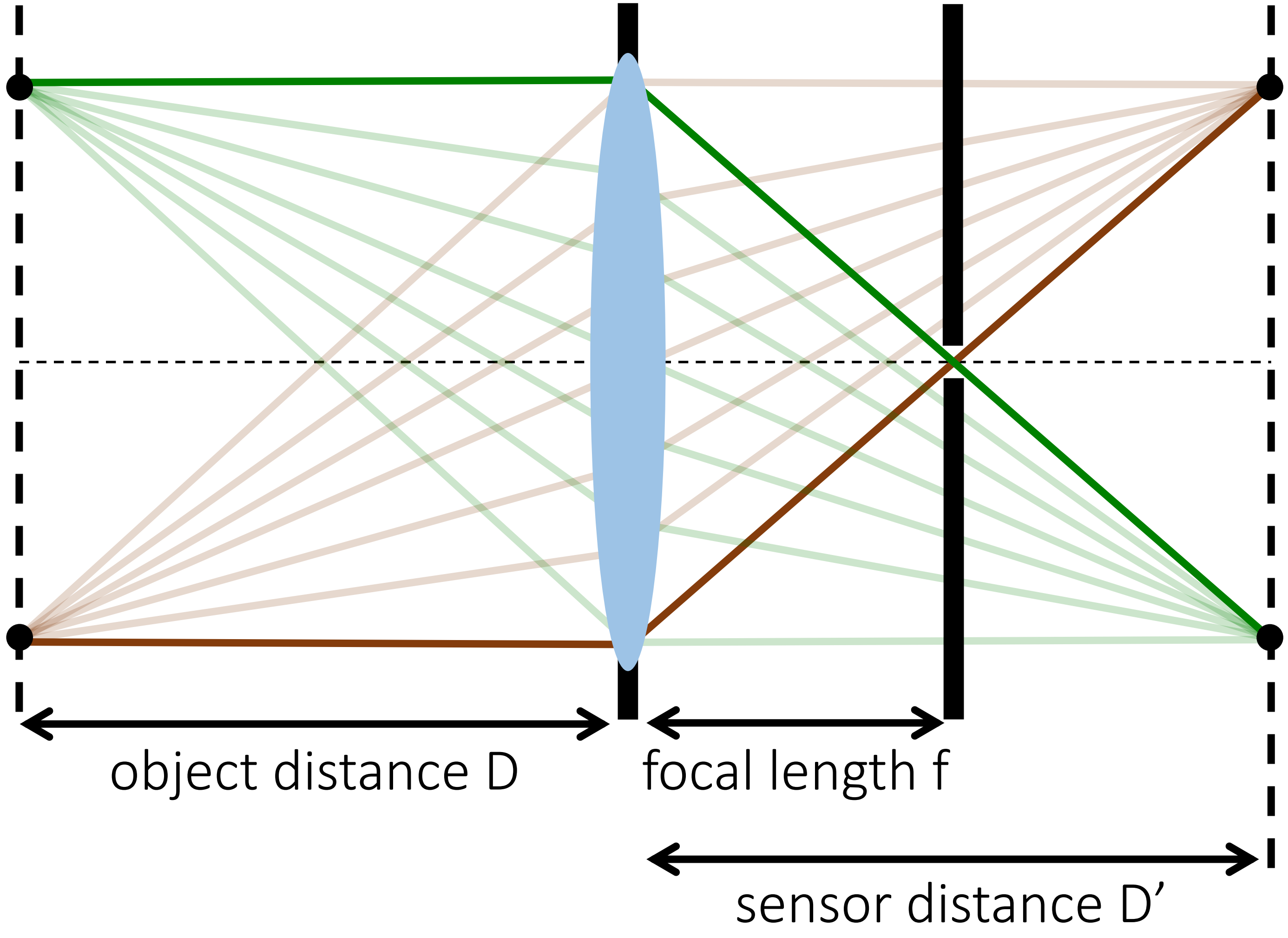
1. When the scene (or parts of it) is very far away.

Weak perspective projection applies to the mountains.

When can we assume a weak perspective camera?

2. When we use a telecentric lens.

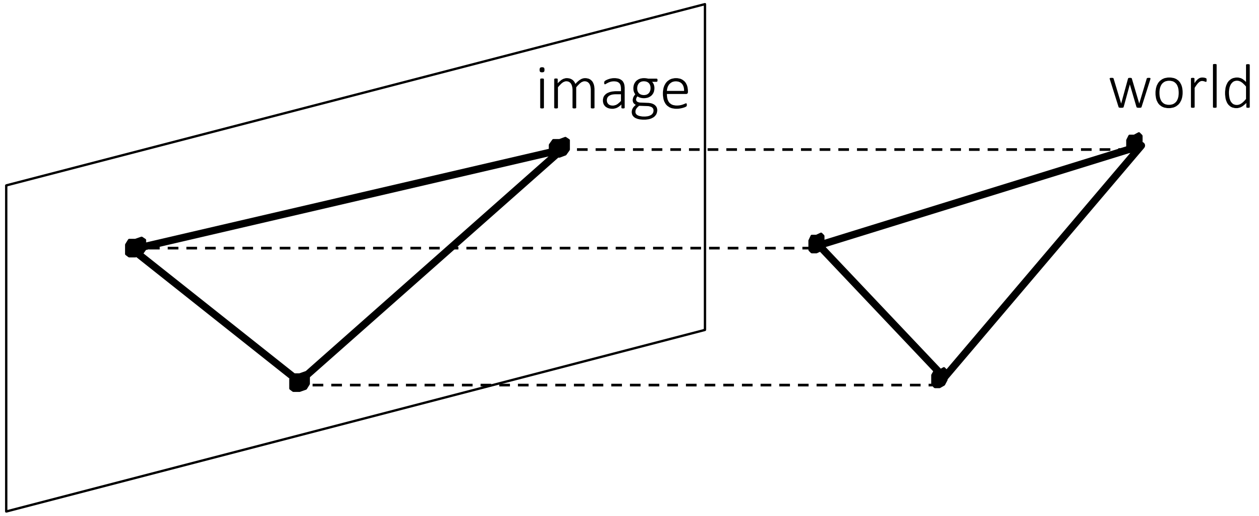
Place a pinhole at focal length, so that only rays parallel to primary ray pass through.



Orthographic camera

Special case of weak perspective camera where:

- constant magnification is equal to 1.
- there is no shift between camera and image origins.
- the world and camera coordinate systems are the same.

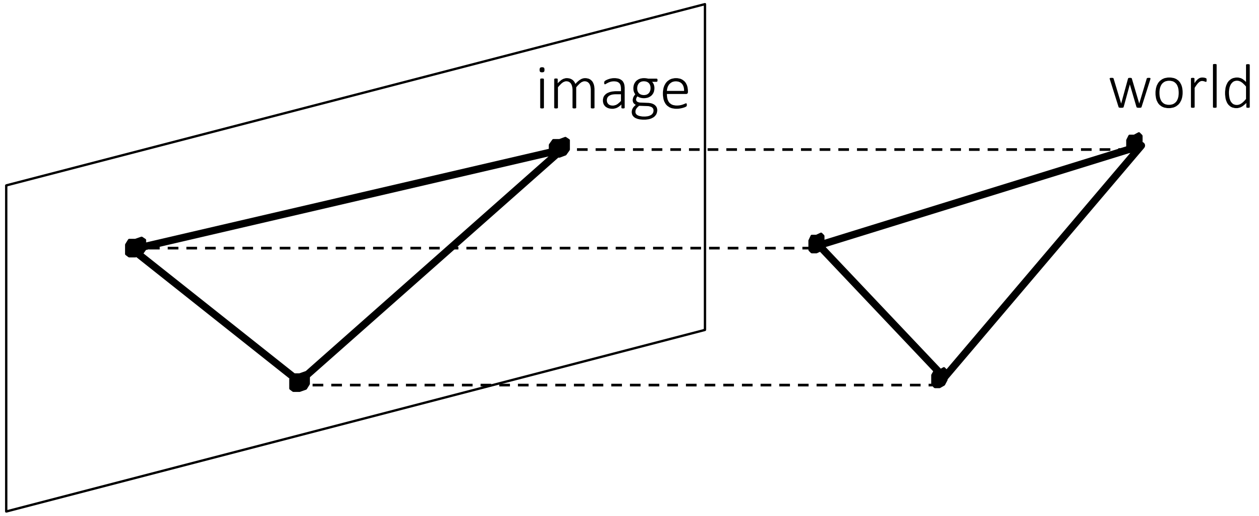


What is the camera matrix in this case?

Orthographic camera

Special case of weak perspective camera where:

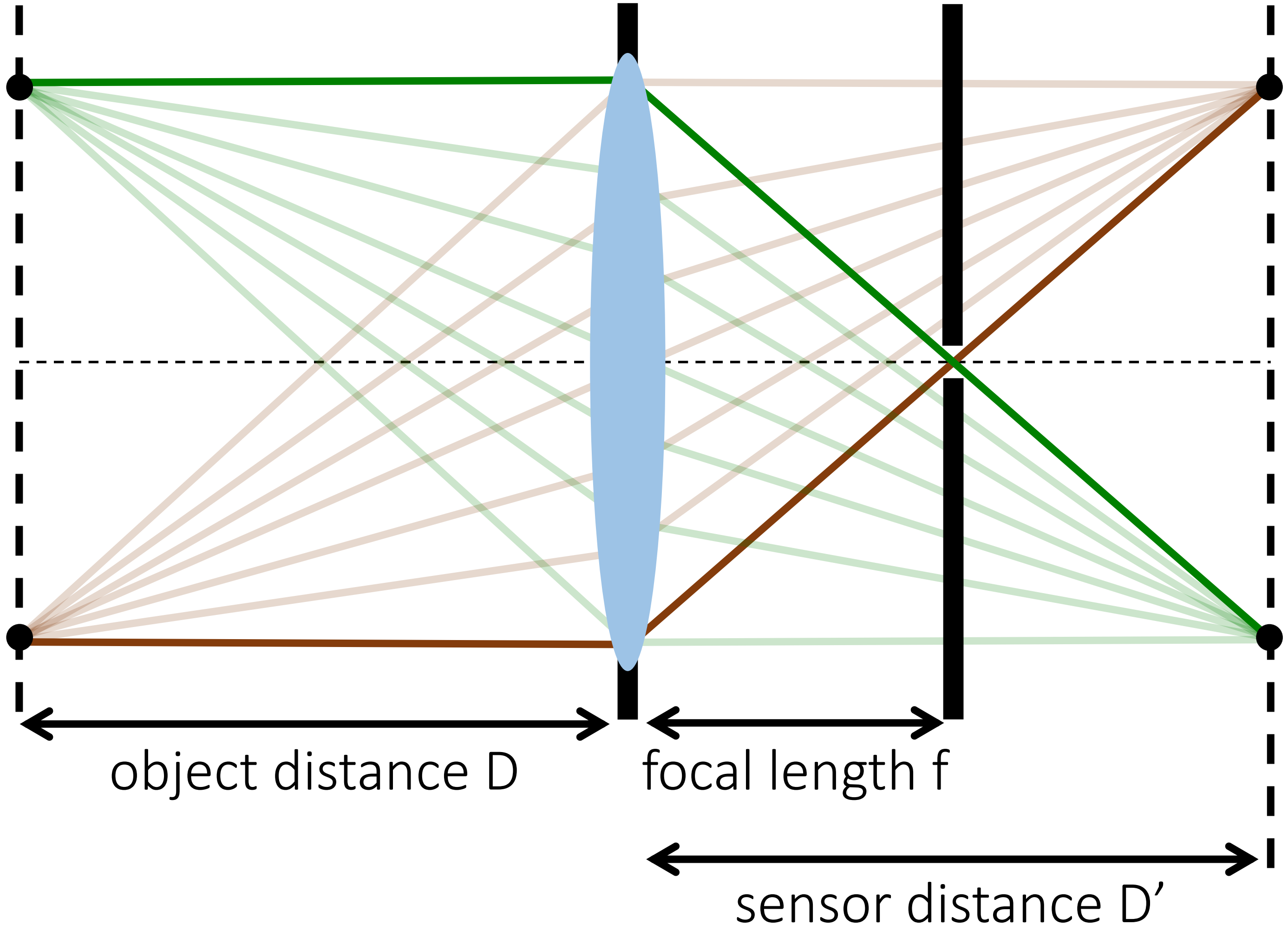
- constant magnification is equal to 1.
- there is no shift between camera and image origins.
- the world and camera coordinate systems are the same.



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

Orthographic projection using a telecentric lens

How do we make the telecentric lens act as an orthographic camera?



We set the sensor distance as:

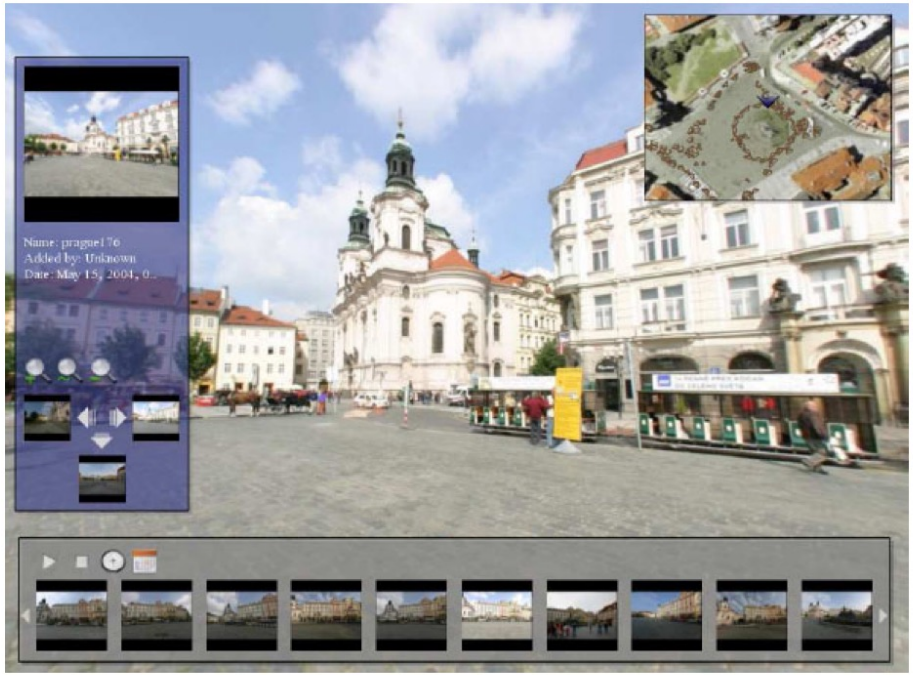
$$D' = 2f$$

in order to achieve unit magnification.

Geometric camera calibration

	Structure (scene geometry)	Motion (camera geometry)	Measurements
Camera Calibration (a.k.a. Pose Estimation)	known	estimate	3D to 2D correspondences
Triangulation	estimate	known	2D to 2D coorespondences
Reconstruction	estimate	estimate	2D to 2D coorespondences

Pose Estimation



Given a single image,
estimate the exact position of the photographer

Geometric camera calibration

Given a set of matched points

$$\{\mathbf{X}_i, \boldsymbol{x}_i\}$$

point in 3D space — point in the image

and camera model

$$\boldsymbol{x} = \boldsymbol{f}(\mathbf{X}; \boldsymbol{p}) = \mathbf{P}\mathbf{X}$$

projection model — parameters — Camera matrix

Find the (pose) estimate of

$$\mathbf{P}$$

We'll use a **perspective** camera model for pose estimation

Same setup as homography estimation

(slightly different derivation here)

Mapping between 3D point and image points

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

What are the unknowns?

Mapping between 3D point and image points

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

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} — & \boldsymbol{p}_1^\top & — \\ — & \boldsymbol{p}_2^\top & — \\ — & \boldsymbol{p}_3^\top & — \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{X} \\ | \end{bmatrix}$$

Heterogeneous coordinates

$$x' = \frac{\boldsymbol{p}_1^\top \boldsymbol{X}}{\boldsymbol{p}_3^\top \boldsymbol{X}} \qquad y' = \frac{\boldsymbol{p}_2^\top \boldsymbol{X}}{\boldsymbol{p}_3^\top \boldsymbol{X}}$$

(non-linear relation between coordinates)

How can we make these relations linear?

How can we make these relations linear?

$$x' = \frac{\boldsymbol{p}_1^\top \boldsymbol{X}}{\boldsymbol{p}_3^\top \boldsymbol{X}} \qquad y' = \frac{\boldsymbol{p}_2^\top \boldsymbol{X}}{\boldsymbol{p}_3^\top \boldsymbol{X}}$$

Make them linear with algebraic manipulation…

$$\boldsymbol{p}_2^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} y' = 0$$

$$\boldsymbol{p}_1^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} x' = 0$$

Now we can setup a system of linear equations
with multiple point correspondences

$$\boldsymbol{p}_2^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} y' = 0$$

$$\boldsymbol{p}_1^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} x' = 0$$

How do we proceed?

$$\boldsymbol{p}_2^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} y' = 0$$

$$\boldsymbol{p}_1^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} x' = 0$$

In matrix form …
$$\begin{bmatrix} \boldsymbol{X}^\top & \boldsymbol{0} & -x'\boldsymbol{X}^\top \\ \boldsymbol{0} & \boldsymbol{X}^\top & -y'\boldsymbol{X}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix} = \boldsymbol{0}$$

How do we proceed?

$$\boldsymbol{p}_2^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} y' = 0$$

$$\boldsymbol{p}_1^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} x' = 0$$

In matrix form …

$$\begin{bmatrix} \boldsymbol{X}^\top & \boldsymbol{0} & -x'\boldsymbol{X}^\top \\ \boldsymbol{0} & \boldsymbol{X}^\top & -y'\boldsymbol{X}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix} = \boldsymbol{0}$$

For N points …

$$\begin{bmatrix} \boldsymbol{X}_1^\top & \boldsymbol{0} & -x'\boldsymbol{X}_1^\top \\ \boldsymbol{0} & \boldsymbol{X}_1^\top & -y'\boldsymbol{X}_1^\top \\ \vdots & \vdots & \vdots \\ \boldsymbol{X}_N^\top & \boldsymbol{0} & -x'\boldsymbol{X}_N^\top \\ \boldsymbol{0} & \boldsymbol{X}_N^\top & -y'\boldsymbol{X}_N^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix} = \boldsymbol{0}$$

How do we solve this system?

Solve for camera matrix by

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\mathbf{A}\boldsymbol{x}\|^2 \text{ subject to } \|\boldsymbol{x}\|^2 = 1$$

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{X}_1^\top & \mathbf{0} & -x'\boldsymbol{X}_1^\top \\ \mathbf{0} & \boldsymbol{X}_1^\top & -y'\boldsymbol{X}_1^\top \\ \vdots & \vdots & \vdots \\ \boldsymbol{X}_N^\top & \mathbf{0} & -x'\boldsymbol{X}_N^\top \\ \mathbf{0} & \boldsymbol{X}_N^\top & -y'\boldsymbol{X}_N^\top \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix}$$

SVD!

Solve for camera matrix by

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\mathbf{A}\boldsymbol{x}\|^2 \text{ subject to } \|\boldsymbol{x}\|^2 = 1$$

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{X}_1^\top & \mathbf{0} & -x'\boldsymbol{X}_1^\top \\ \mathbf{0} & \boldsymbol{X}_1^\top & -y'\boldsymbol{X}_1^\top \\ \vdots & \vdots & \vdots \\ \boldsymbol{X}_N^\top & \mathbf{0} & -x'\boldsymbol{X}_N^\top \\ \mathbf{0} & \boldsymbol{X}_N^\top & -y'\boldsymbol{X}_N^\top \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix}$$

Solution **x** is the column of **V** corresponding to smallest singular value of

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

Solve for camera matrix by

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\mathbf{A}\boldsymbol{x}\|^2 \text{ subject to } \|\boldsymbol{x}\|^2 = 1$$

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{X}_1^\top & \mathbf{0} & -x'\boldsymbol{X}_1^\top \\ \mathbf{0} & \boldsymbol{X}_1^\top & -y'\boldsymbol{X}_1^\top \\ \vdots & \vdots & \vdots \\ \boldsymbol{X}_N^\top & \mathbf{0} & -x'\boldsymbol{X}_N^\top \\ \mathbf{0} & \boldsymbol{X}_N^\top & -y'\boldsymbol{X}_N^\top \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix}$$

Equivalently, solution ***x*** is the Eigenvector corresponding to smallest Eigenvalue of $\mathbf{A}^\top\mathbf{A}$

Now we have:

$$\mathbf{P} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix}$$

Are we done?

Almost there …

$$\mathbf{P} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix}$$

How do you get the intrinsic and extrinsic parameters from the projection matrix?

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array} \right]$$

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

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

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array} \right]$$

$$\begin{aligned} \mathbf{P} &= \mathbf{K}[\mathbf{R}|\mathbf{t}] \\ &= \mathbf{K}[\mathbf{R}| - \mathbf{Rc}] \\ &= [\mathbf{M}| - \mathbf{Mc}] \end{aligned}$$

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

$$\begin{aligned} \mathbf{P} &= \mathbf{K}[\mathbf{R}|\mathbf{t}] \\ &= \mathbf{K}[\mathbf{R}| - \mathbf{R}\mathbf{c}] \\ &= [\mathbf{M}| - \mathbf{M}\mathbf{c}] \end{aligned}$$

Find the camera center **C**

What is the projection of the camera center?

Find intrinsic **K** and rotation **R**

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

$$\begin{aligned} \mathbf{P} &= \mathbf{K}[\mathbf{R}|\mathbf{t}] \\ &= \mathbf{K}[\mathbf{R}| - \mathbf{R}\mathbf{c}] \\ &= [\mathbf{M}| - \mathbf{M}\mathbf{c}] \end{aligned}$$

Find the camera center **C**

$$\mathbf{P}\mathbf{c} = \mathbf{0}$$

How do we compute the camera center from this?

Find intrinsic **K** and rotation **R**

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

$$\begin{aligned} \mathbf{P} &= \mathbf{K}[\mathbf{R}|\mathbf{t}] \\ &= \mathbf{K}[\mathbf{R}| -\mathbf{Rc}] \\ &= [\mathbf{M}| -\mathbf{Mc}] \end{aligned}$$

Find the camera center **C**

$$\mathbf{Pc} = \mathbf{0}$$

SVD of P!

c *is the singular vector corresponding to the smallest singular value*

Find intrinsic **K** and rotation **R**

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

$$\begin{aligned} \mathbf{P} &= \mathbf{K}[\mathbf{R}|\mathbf{t}] \\ &= \mathbf{K}[\mathbf{R}| - \mathbf{Rc}] \\ &= [\mathbf{M}| - \mathbf{Mc}] \end{aligned}$$

Find the camera center **C**

$$\mathbf{Pc} = \mathbf{0}$$

SVD of P!

***c** is the singular vector corresponding to the smallest singular value*

Find intrinsic **K** and rotation **R**

$$\mathbf{M} = \mathbf{KR}$$

Any useful properties of K and R we can use?

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

$$\begin{aligned} \mathbf{P} &= \mathbf{K}[\mathbf{R}|\mathbf{t}] \\ &= \mathbf{K}[\mathbf{R}| - \mathbf{R}\mathbf{c}] \\ &= [\mathbf{M}| - \mathbf{M}\mathbf{c}] \end{aligned}$$

Find the camera center **C**

$$\mathbf{P}\mathbf{c} = \mathbf{0}$$

SVD of P!

***c** is the singular vector corresponding to the smallest singular value*

Find intrinsic **K** and rotation **R**

$$\mathbf{M} = \mathbf{K}\mathbf{R}$$

right upper triangle (K) — orthogonal (R)

How do we find K and R?

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

$$\begin{aligned} \mathbf{P} &= \mathbf{K}[\mathbf{R}|\mathbf{t}] \\ &= \mathbf{K}[\mathbf{R}| - \mathbf{Rc}] \\ &= [\mathbf{M}| - \mathbf{Mc}] \end{aligned}$$

Find the camera center **C**

$$\mathbf{Pc} = \mathbf{0}$$

SVD of P!

***c** is the singular vector corresponding to the smallest singular value*

Find intrinsic **K** and rotation **R**

$$\mathbf{M} = \mathbf{KR}$$

QR decomposition

Geometric camera calibration

Given a set of matched points

$$\{\mathbf{X}_i, \boldsymbol{x}_i\}$$

point in 3D space — point in the image

Where do we get these matched points from?

and camera model

$$\boldsymbol{x} = \boldsymbol{f}(\mathbf{X}; \boldsymbol{p}) = \mathbf{P}\mathbf{X}$$

projection model — parameters — Camera matrix

Find the (pose) estimate of

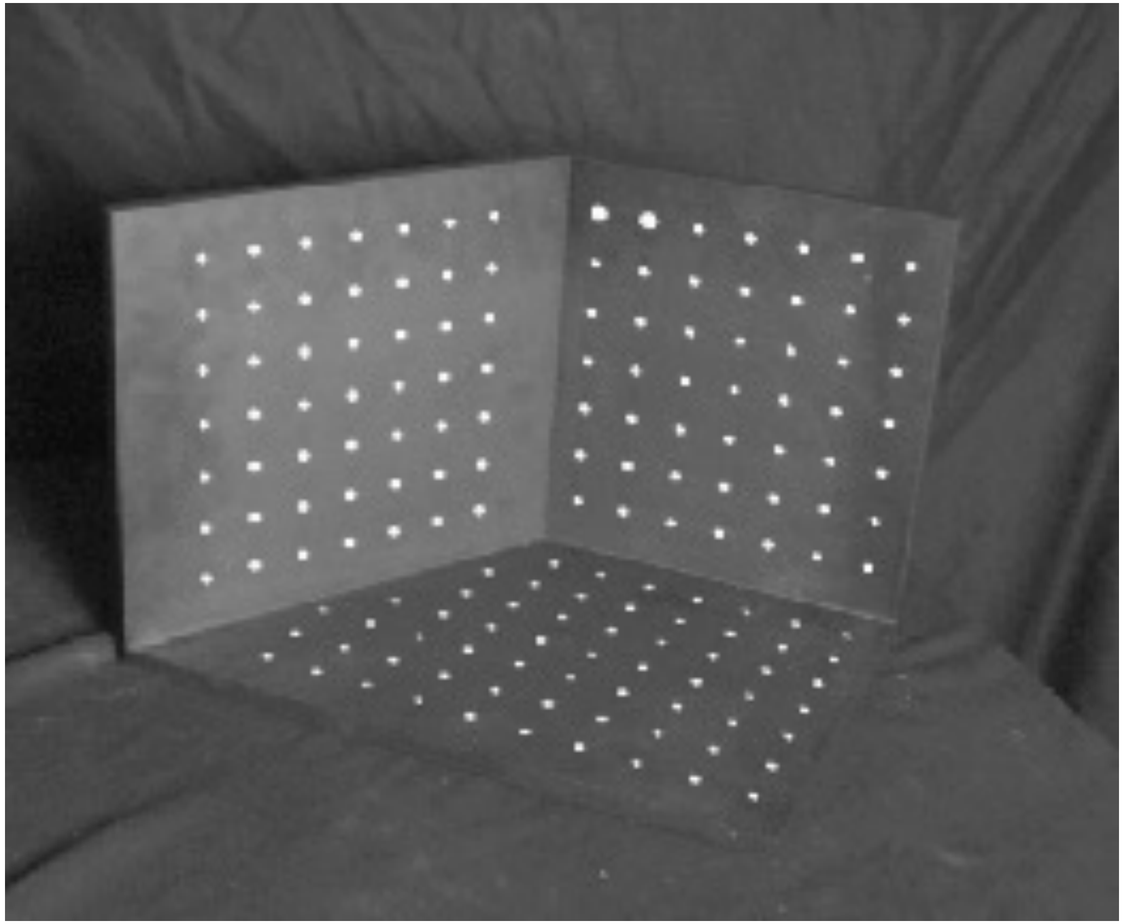
$$\mathbf{P}$$

We'll use a **perspective** camera model for pose estimation

Calibration using a reference object

Place a known object in the scene:

- identify correspondences between image and scene
- compute mapping from scene to image



Issues:

- must know geometry very accurately
- must know 3D->2D correspondence

Geometric camera calibration

Advantages:

- Very simple to formulate.
- Analytical solution.

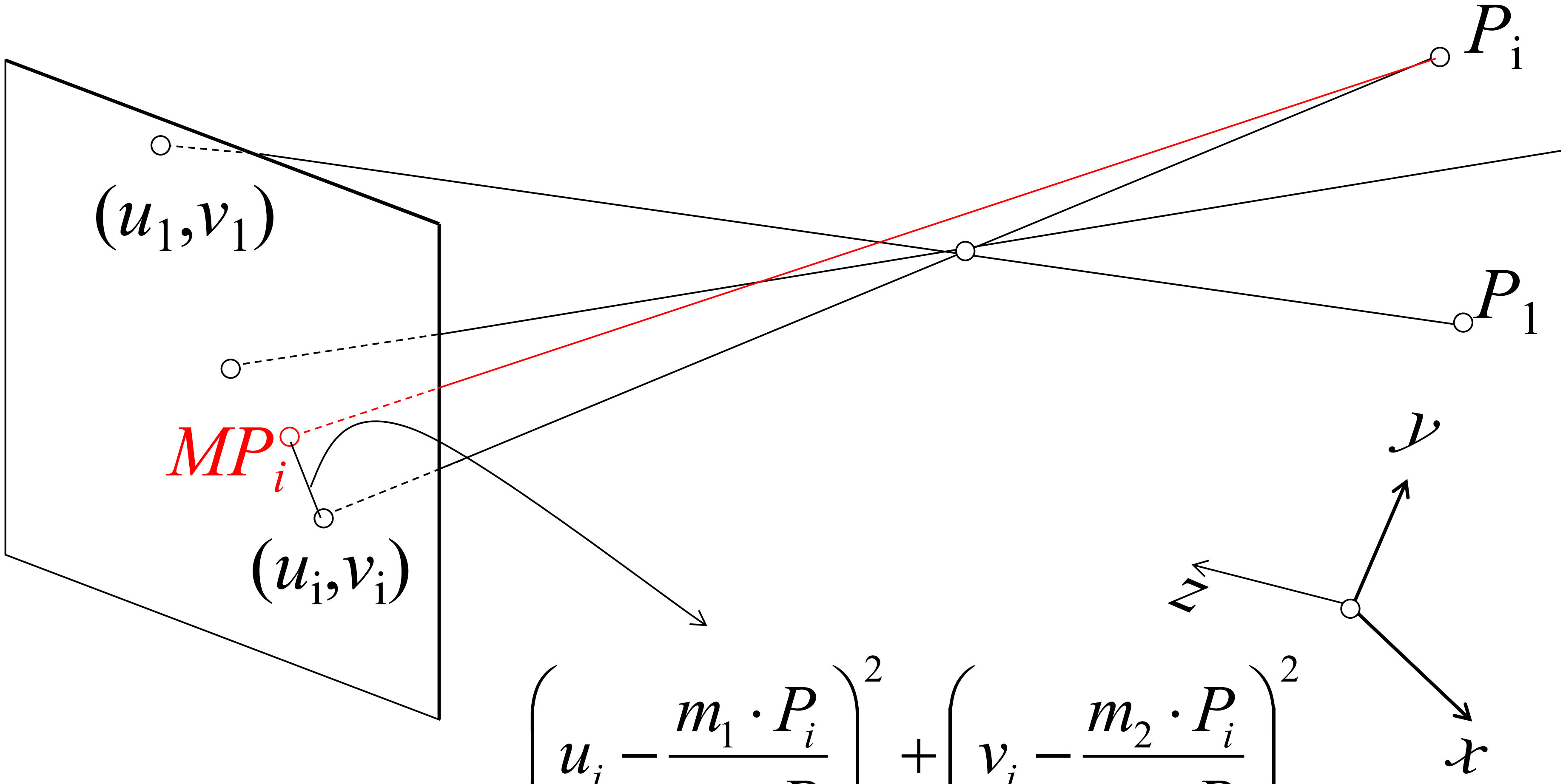
Disadvantages:

- Doesn’t model radial distortion.
- Hard to impose constraints (e.g., known *f*).
- Doesn’t minimize the correct error function.

For these reasons, *nonlinear methods* are preferred

- Define error function E between projected 3D points and image positions
  – E is nonlinear function of intrinsics, extrinsics, radial distortion
- Minimize E using nonlinear optimization techniques

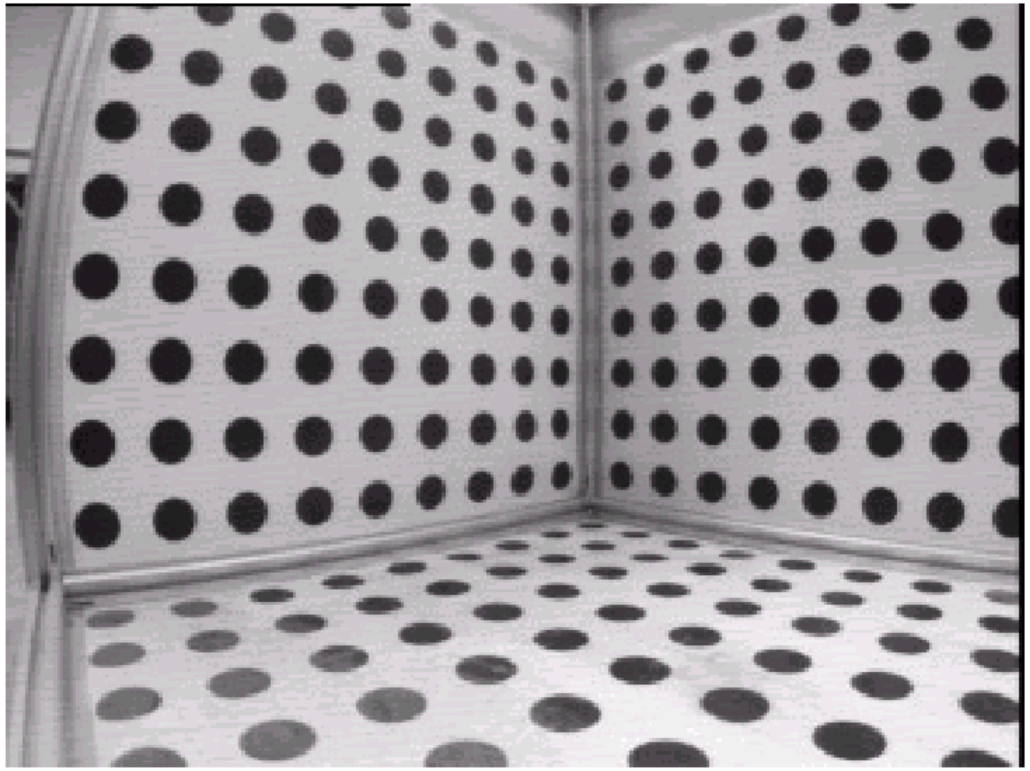
Minimizing reprojection error



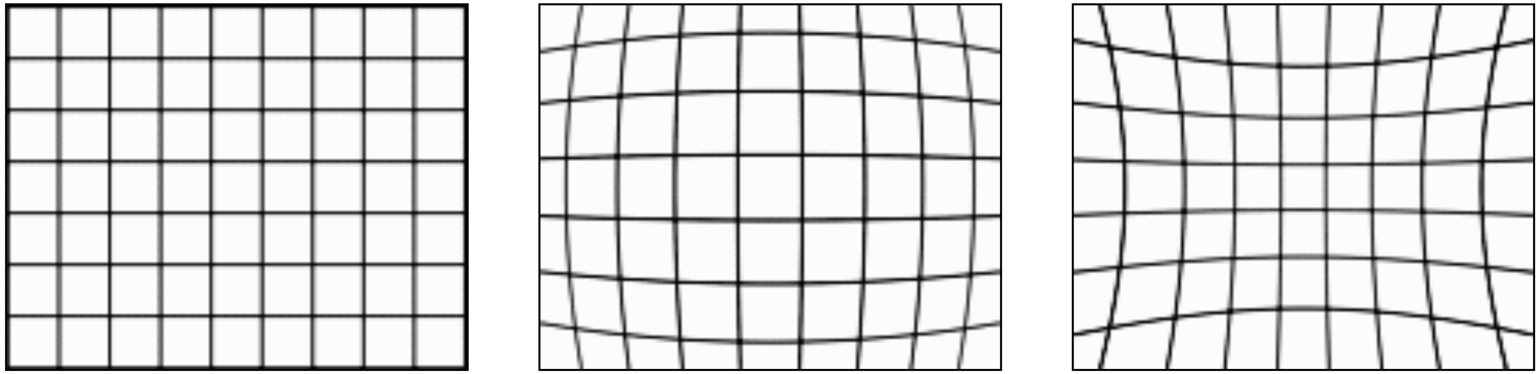
$$\left(u_i - \frac{m_1 \cdot P_i}{m_3 \cdot P_i}\right)^2 + \left(v_i - \frac{m_2 \cdot P_i}{m_3 \cdot P_i}\right)^2$$

Is this equivalent to what we were doing previously?

Radial distortion

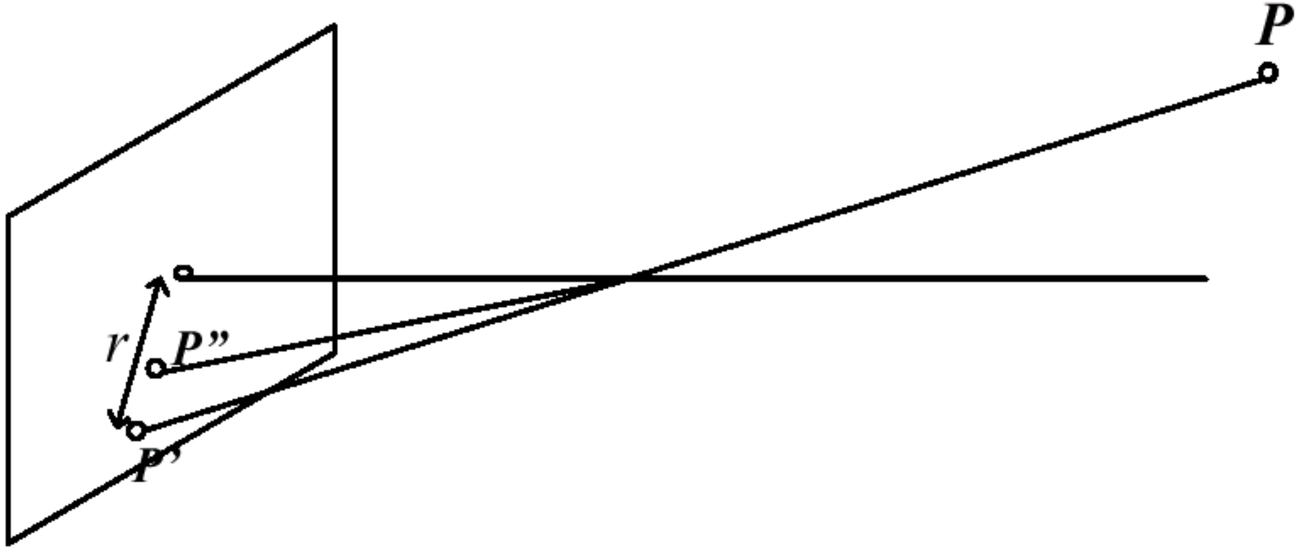


What causes this distortion?



no distortion barrel distortion pincushion distortion

Radial distortion model



Ideal:

$$x' = f\frac{x}{z}$$

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

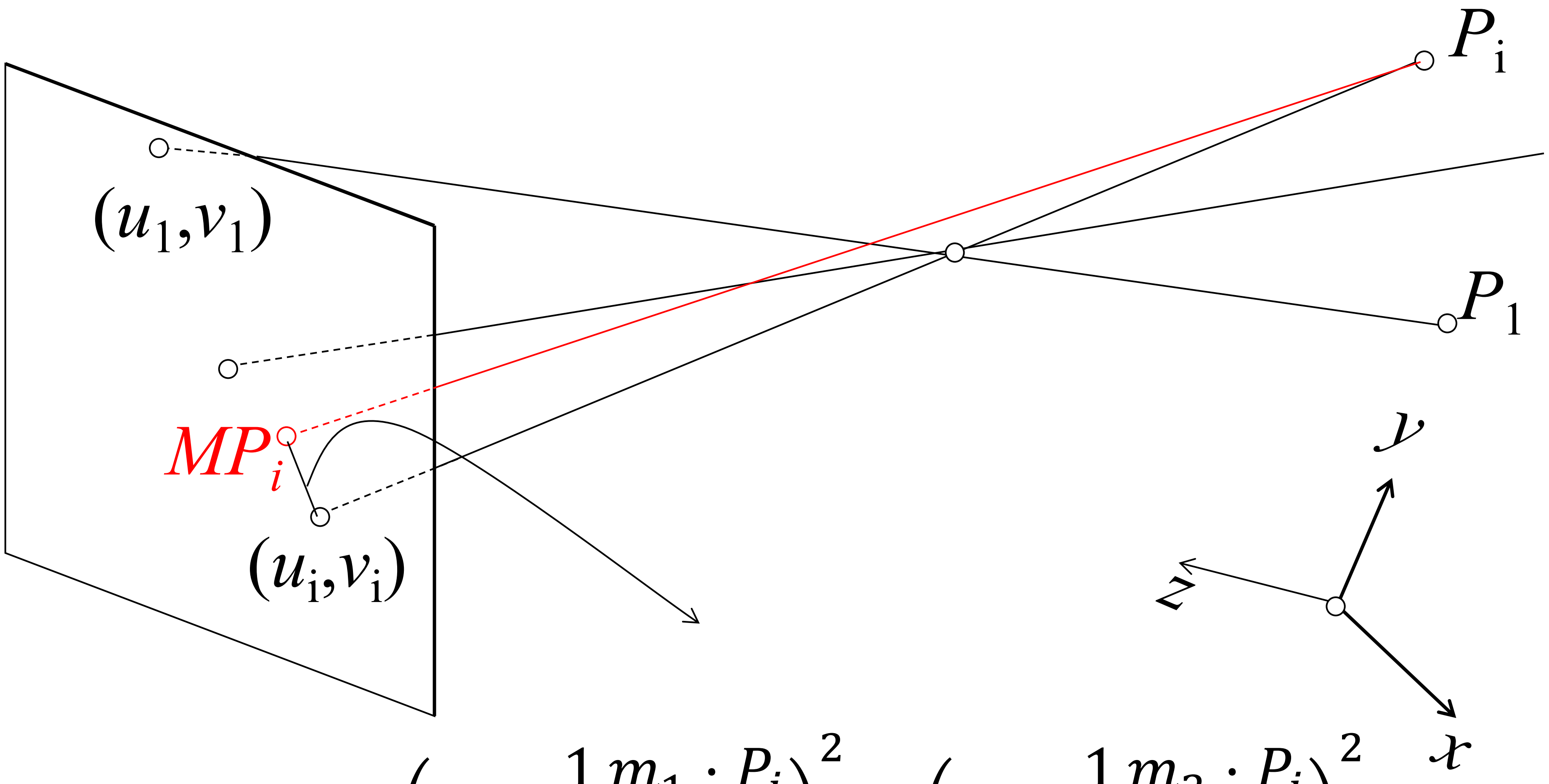
Distorted:

$$x'' = \frac{1}{\lambda}x'$$

$$y'' = \frac{1}{\lambda}y'$$

$$\lambda = 1 + k_1 r^2 + k_2 r^4 + \cdots$$

Minimizing reprojection error with radial distortion



Add distortions to reprojection error:

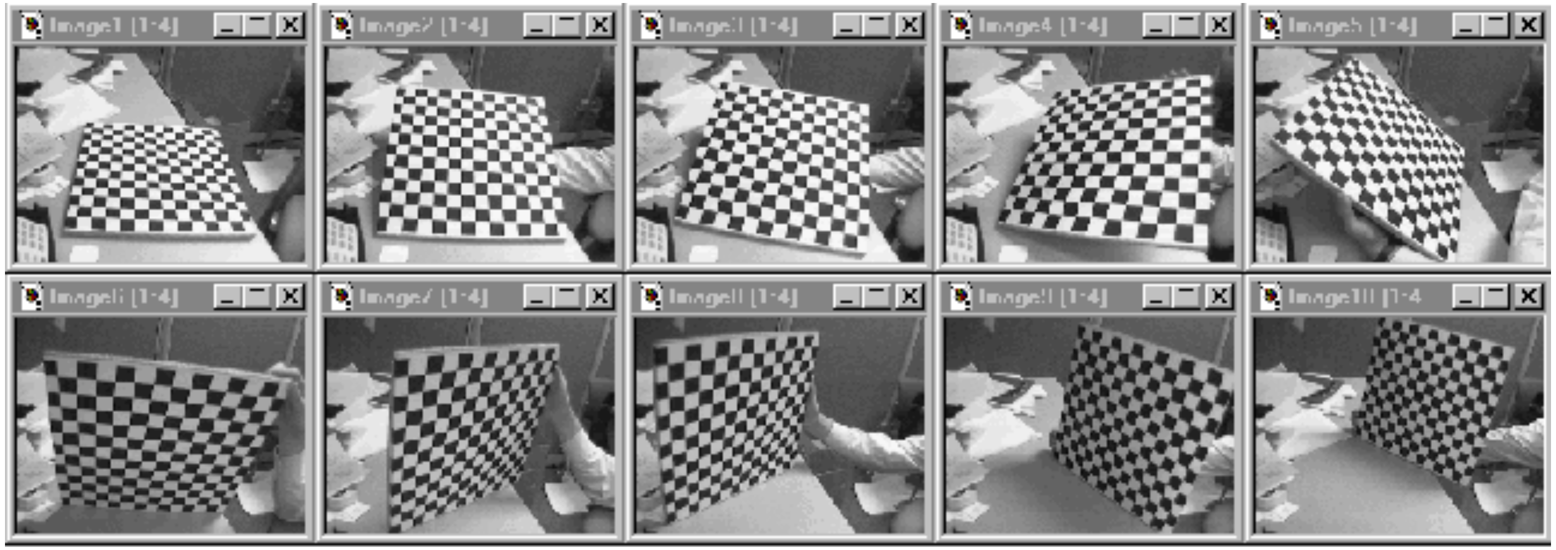
$$\left(u_i - \frac{1}{\lambda}\frac{m_1 \cdot P_i}{m_3 \cdot P_i}\right)^2 + \left(v_i - \frac{1}{\lambda}\frac{m_2 \cdot P_i}{m_3 \cdot P_i}\right)^2$$

Correcting radial distortion

before

after

Alternative: Multi-plane calibration

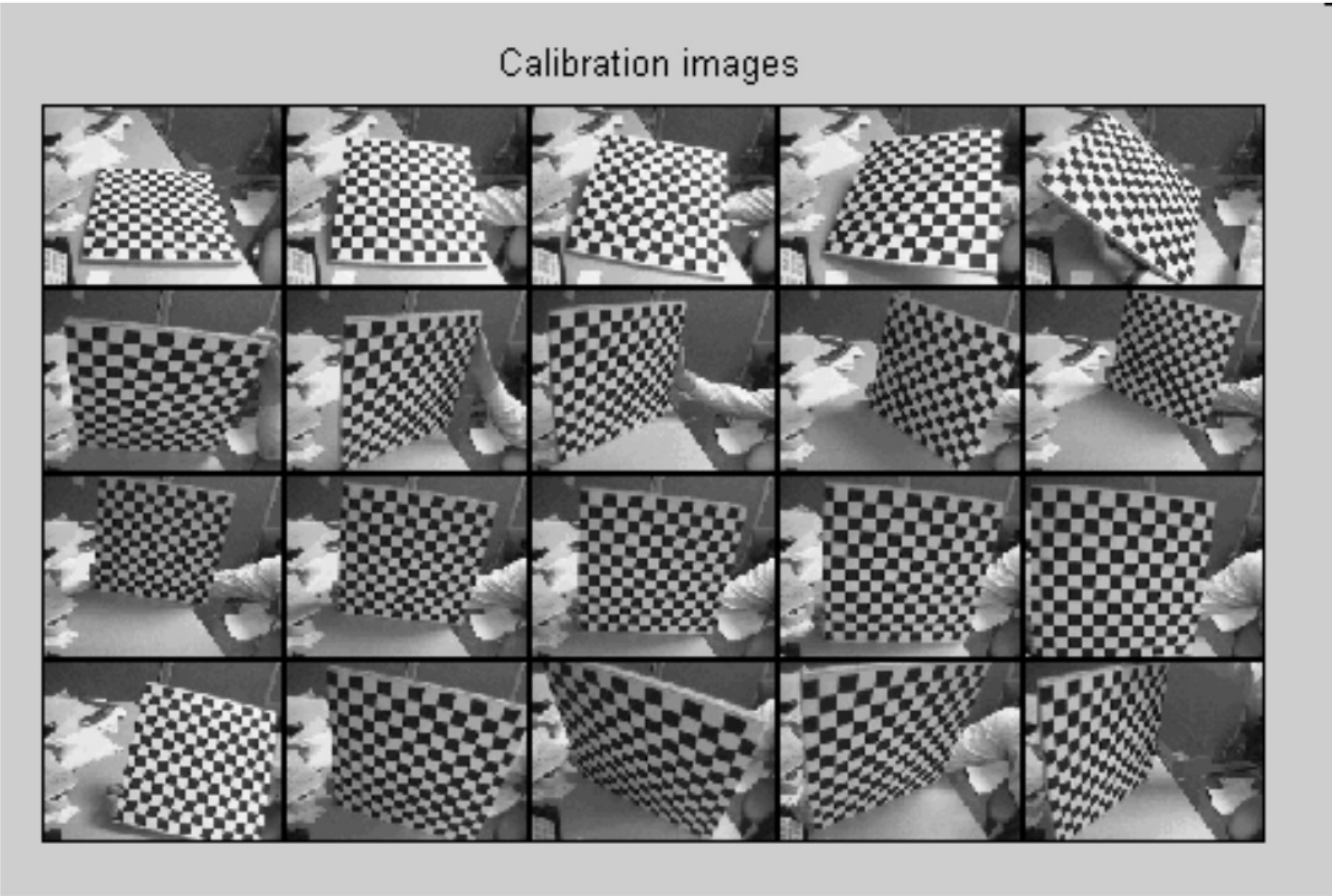


Advantages:

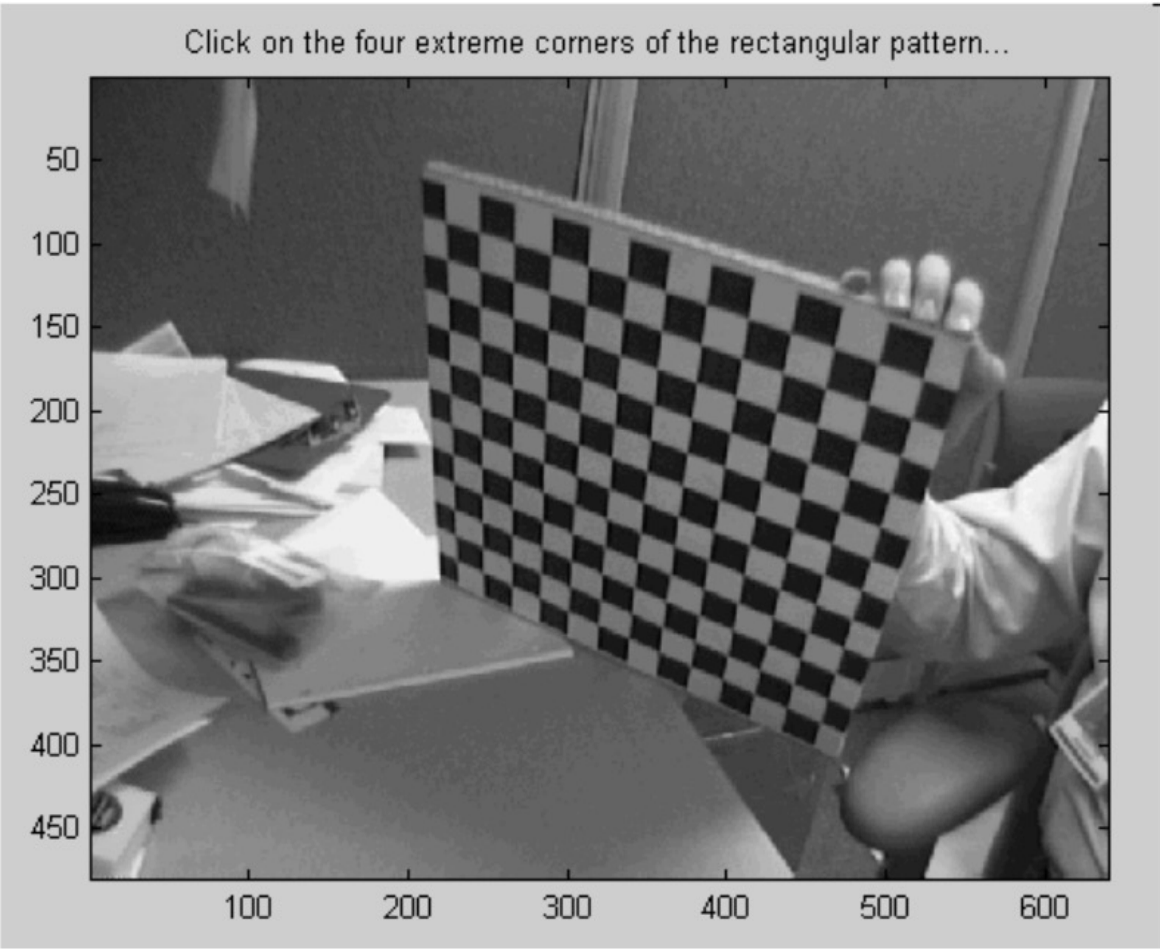
- Only requires a plane
- Don't have to know positions/orientations
- Great code available online!
  - Matlab version: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Also available on OpenCV.

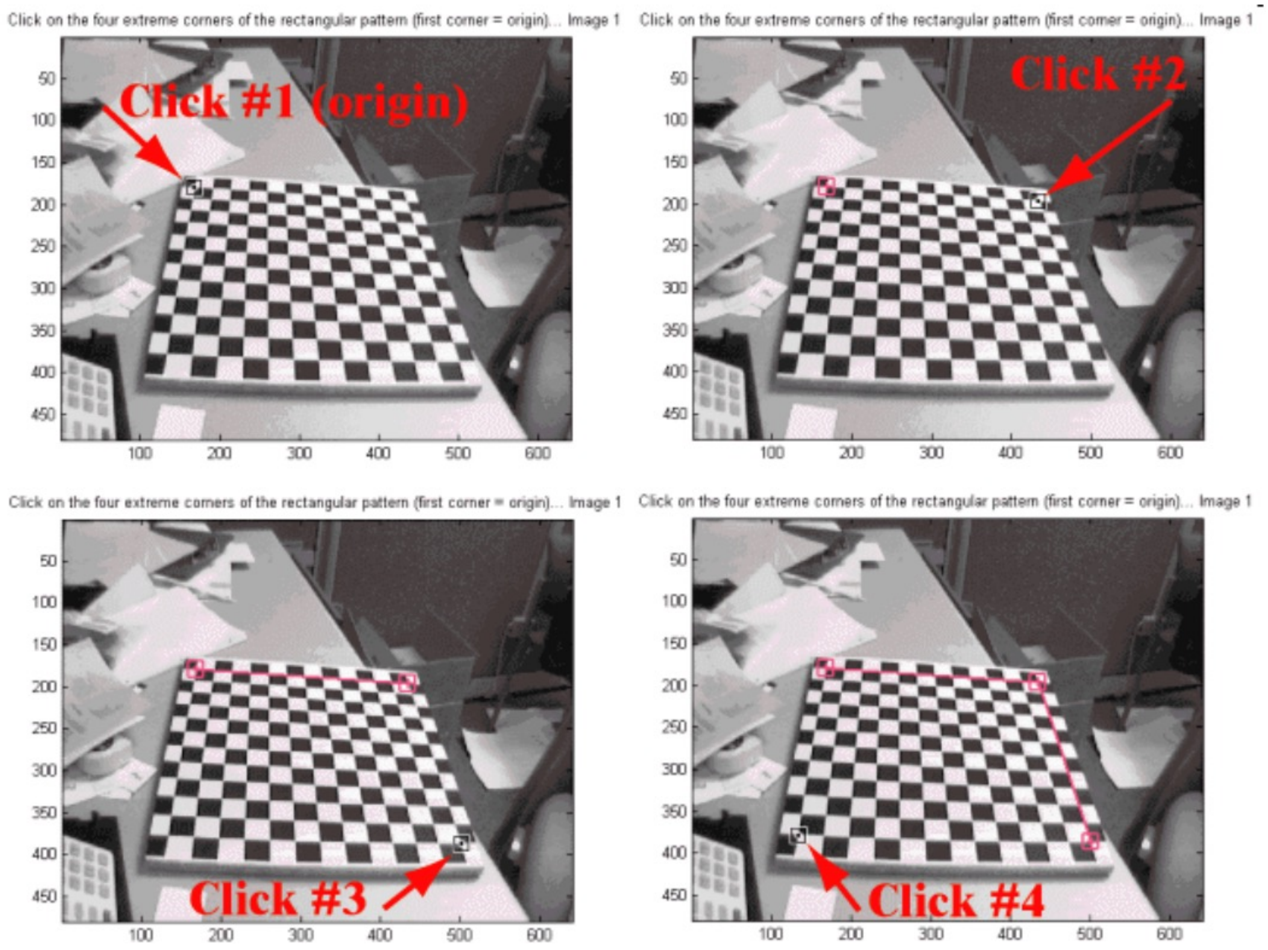
Disadvantage: Need to solve non-linear optimization problem.

Step-by-step demonstration

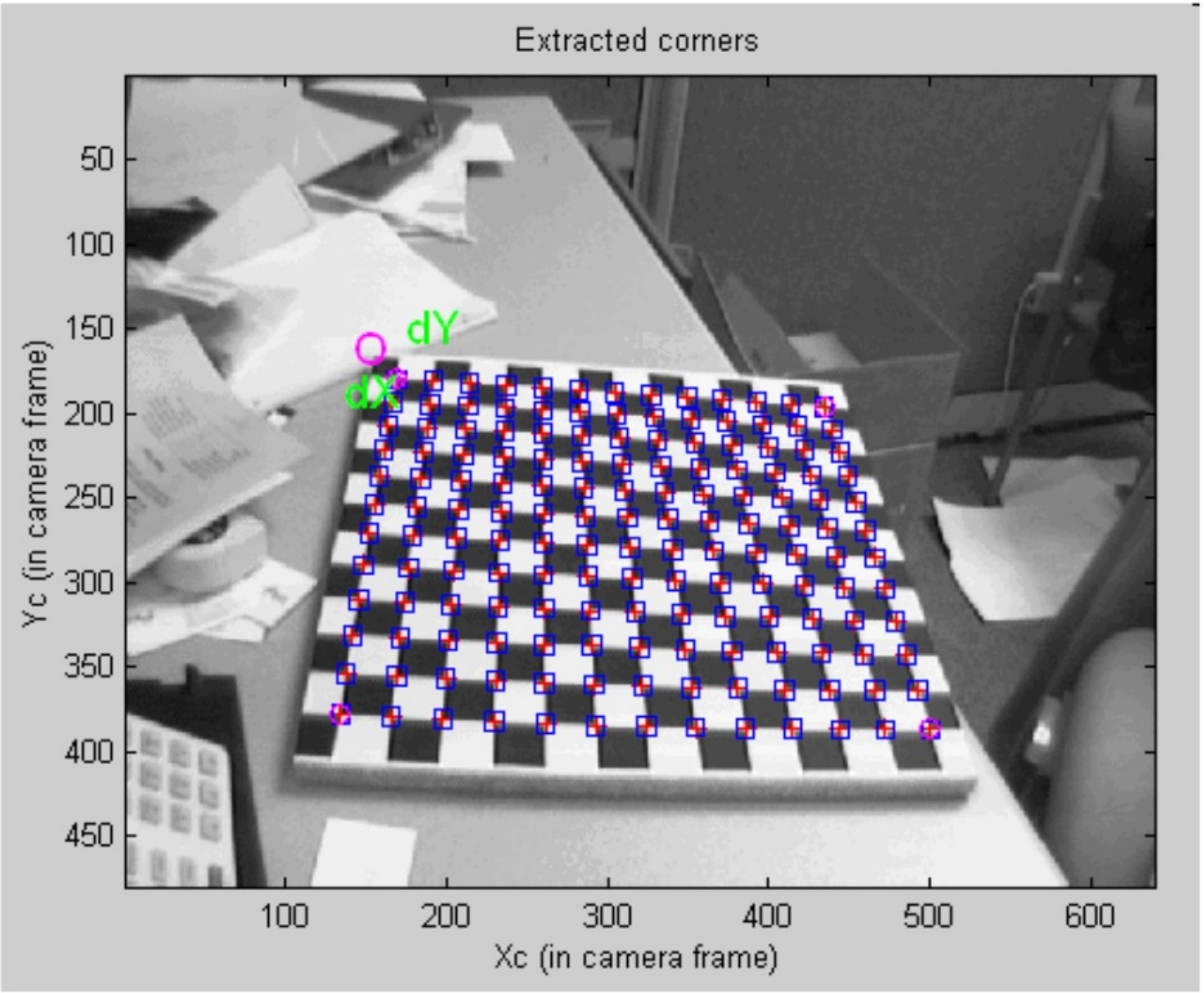


Step-by-step demonstration

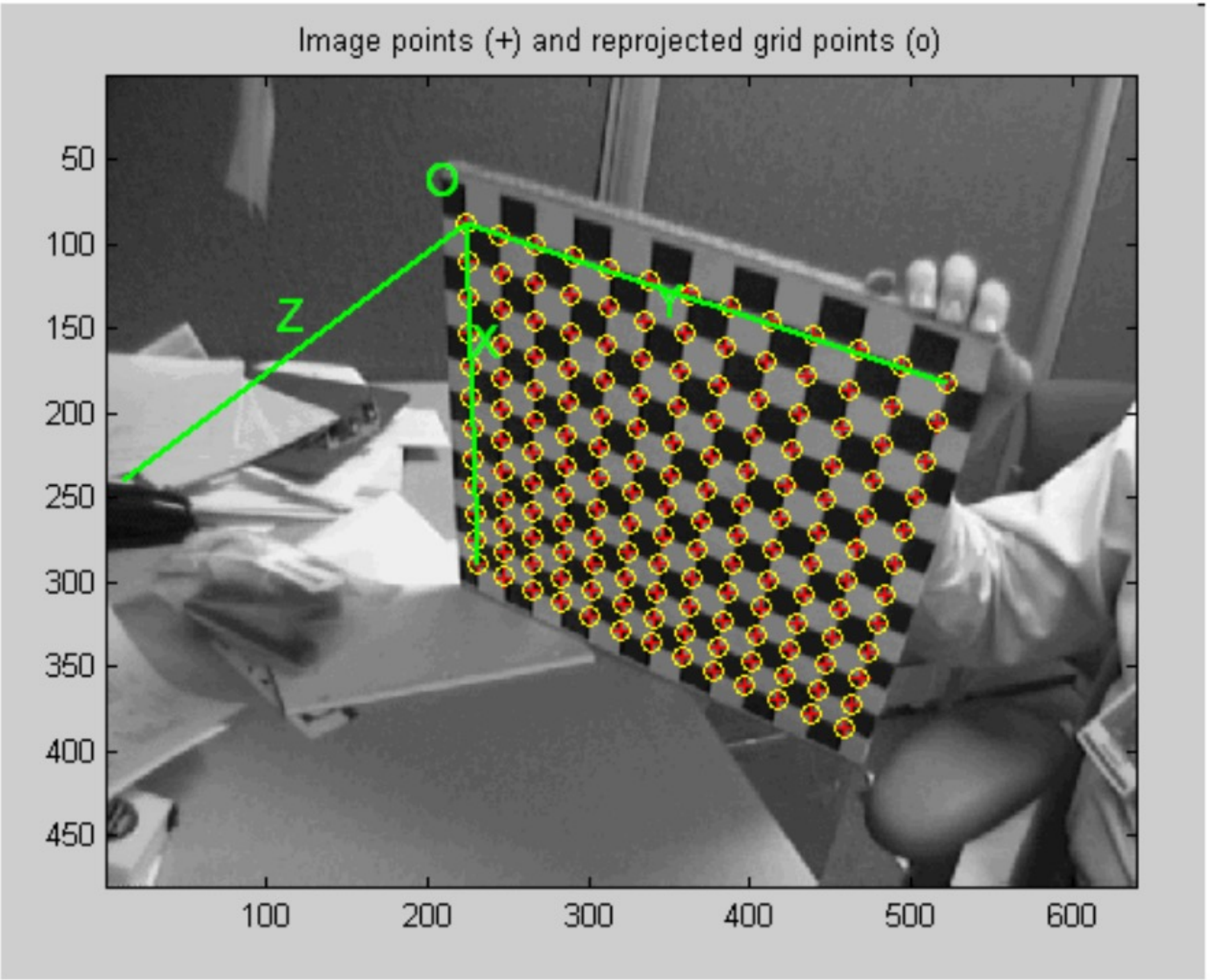




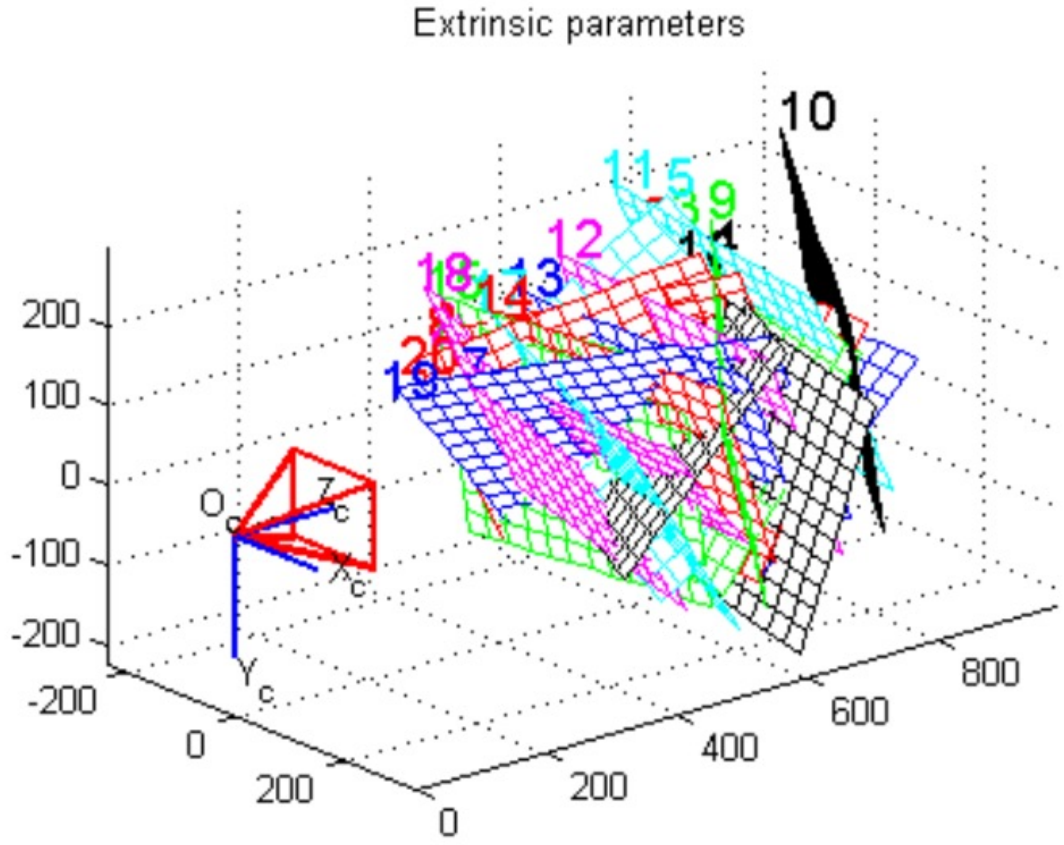
Step-by-step demonstration

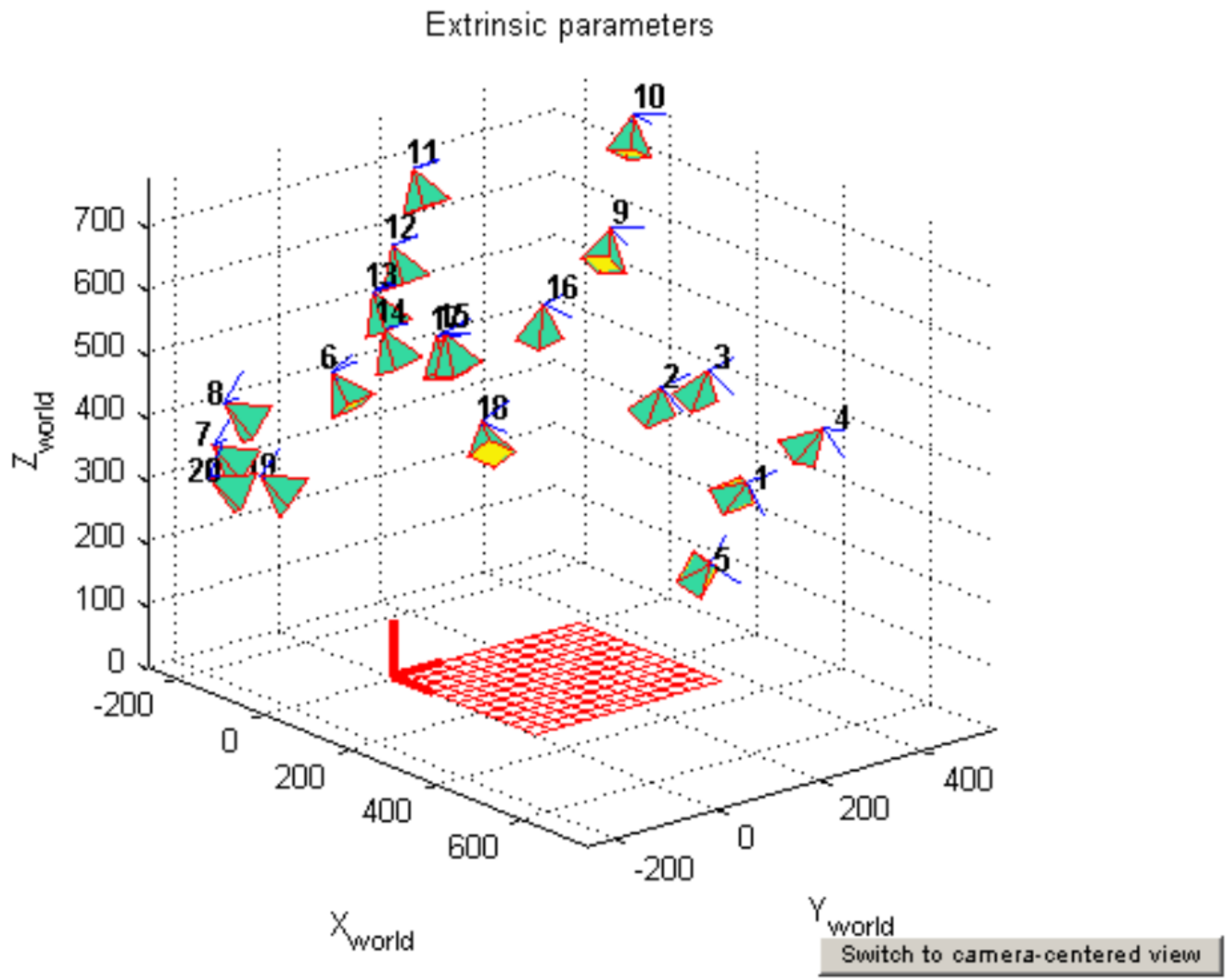


Step-by-step demonstration



Step-by-step demonstration





What does it mean to “calibrate a camera”?

What does it mean to "calibrate a camera"?

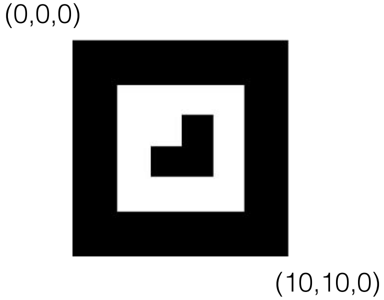
Many different ways to calibrate a camera:

- Radiometric calibration.
- Color calibration.
- Geometric calibration.
- Noise calibration.
- Lens (or aberration) calibration.

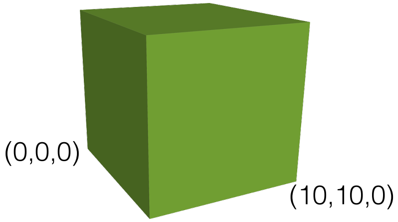
We'll briefly discuss radiometric and color calibration in later lectures. For the rest, see 15-463/663/862.


Tower Health

3D locations of planar marker features are known in advance



3D content prepared in advance



Simple AR program

1. Compute point correspondences (2D and AR tag)
2. Estimate the pose of the camera **P**
3. Project 3D content to image plane using **P**


junaio
SAMSUNG