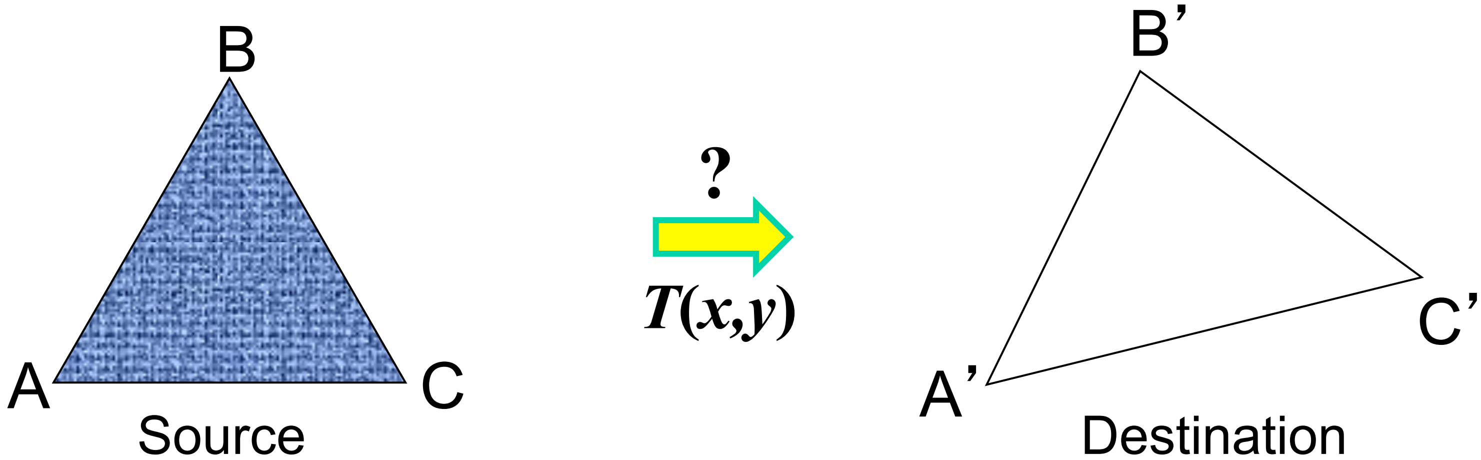
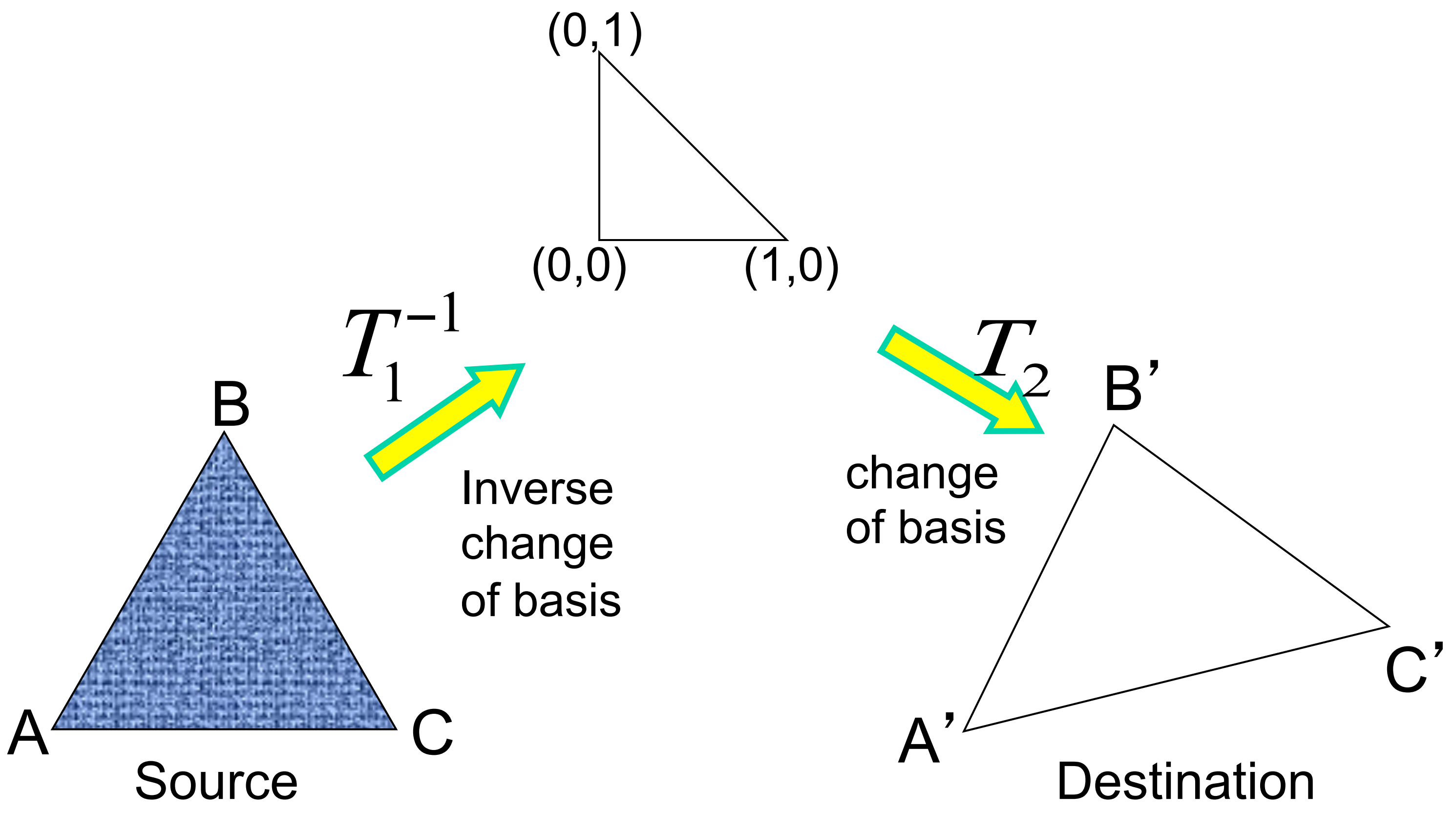
Example: warping triangles



- Given two triangles: ABC and A'B'C' in 2D (12 numbers)
- Need to find transform T to transfer all pixels from one to the other.
- What kind of transformation is T?
- How can we compute the transformation matrix:

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

warping triangles (Barycentric Coordinaes)



Don't forget to move the origin too!

- Very useful in Graphics…

Warp by triangulation

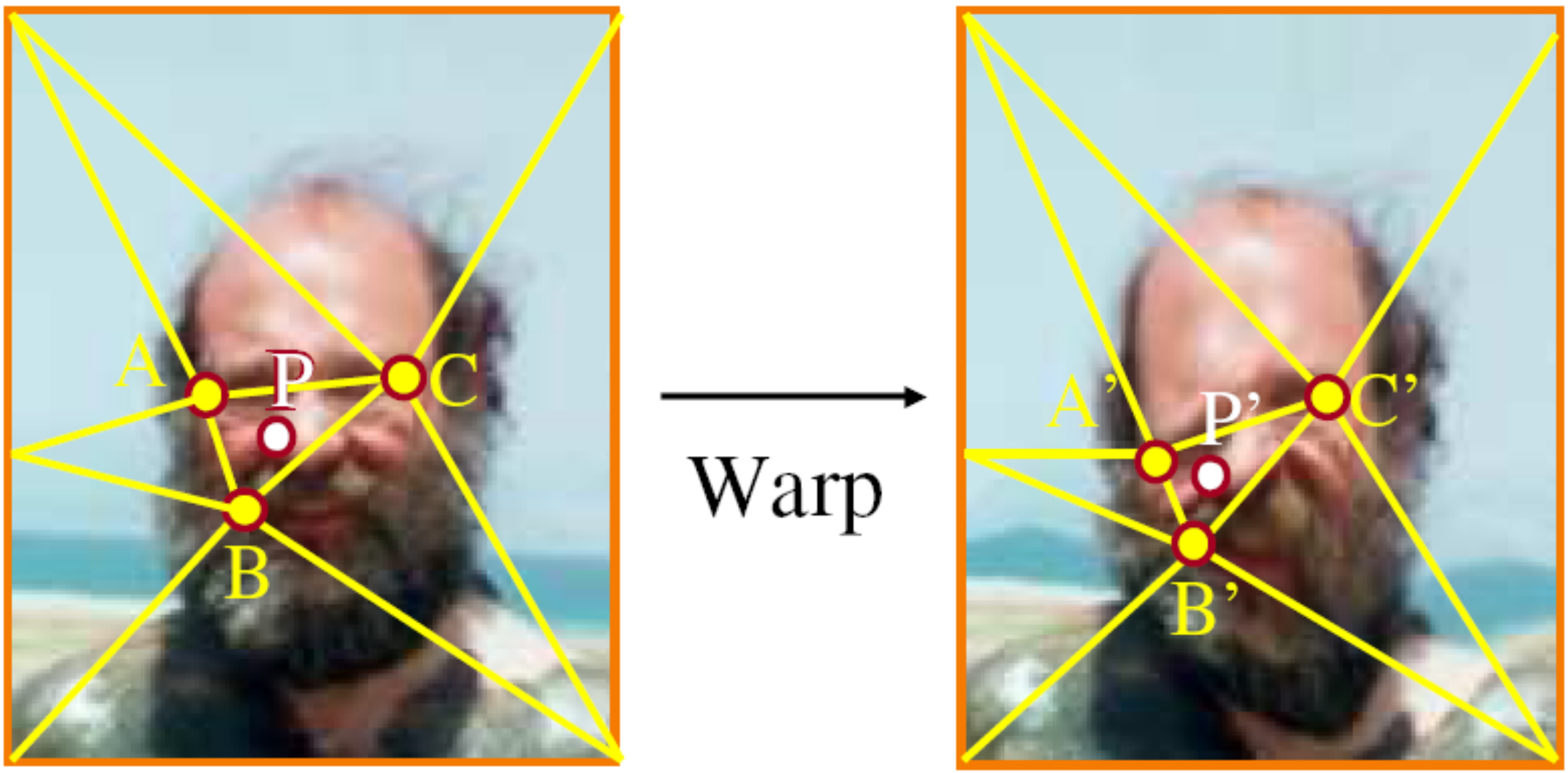


Image morphing

- The goal is to synthesize a fluid transformation from one image to another.
- Cross dissolving is a common transition between cuts, but it is not good for morphing because of the ghosting effects.

image #1

dissolving

image #2

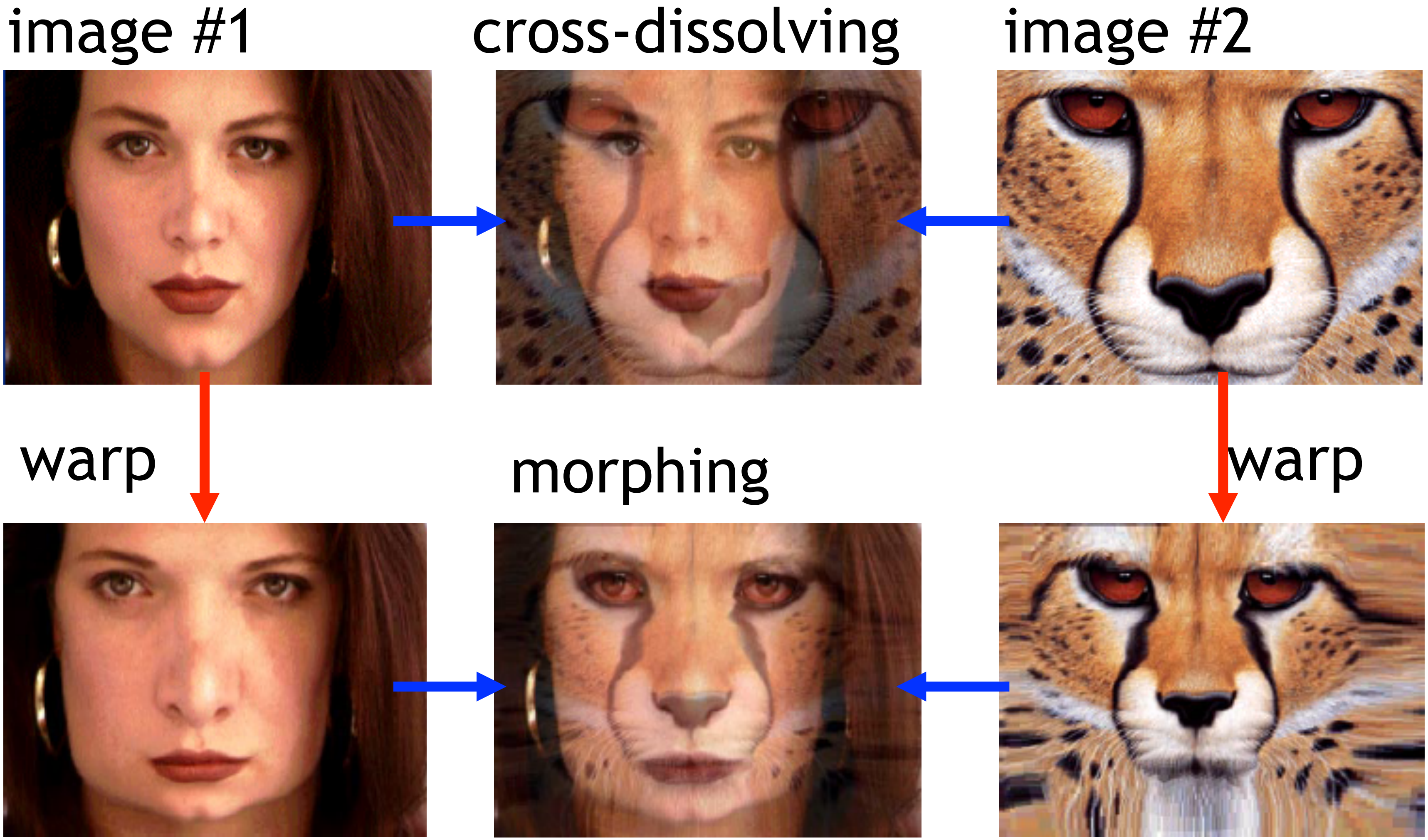
Image morphing

- Why ghosting?
- Morphing = warping + cross-dissolving

shape
(geometric)

color
(photometric)

Image morphing

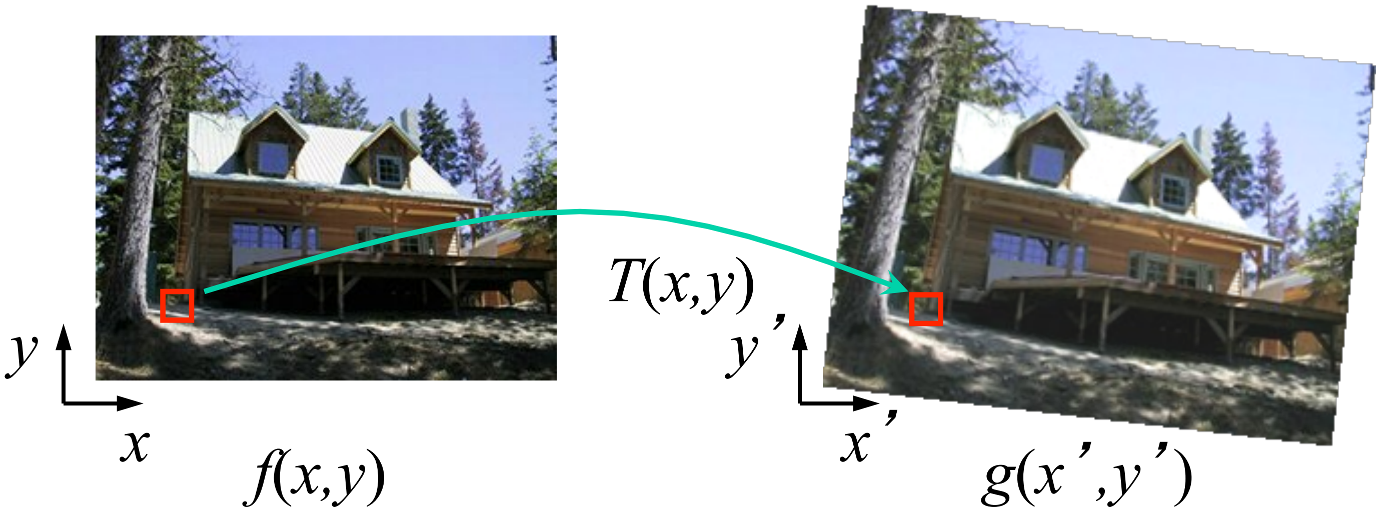


Morphing sequence

Google Picasa Face Movie

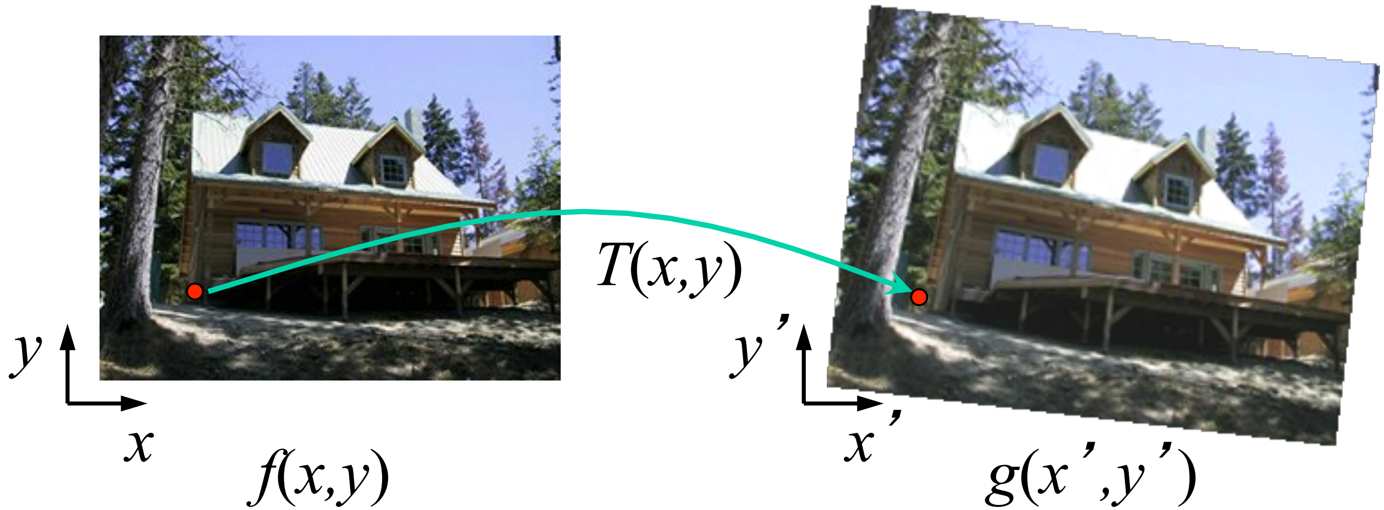
- http://www.youtube.com/watch?v=fLQtssJDMMc

Image warping



- Given a coordinate transform $(x',y') = T(x,y)$ and a source image $f(x,y)$, how do we compute a transformed image $g(x',y') = f(T(x,y))$?

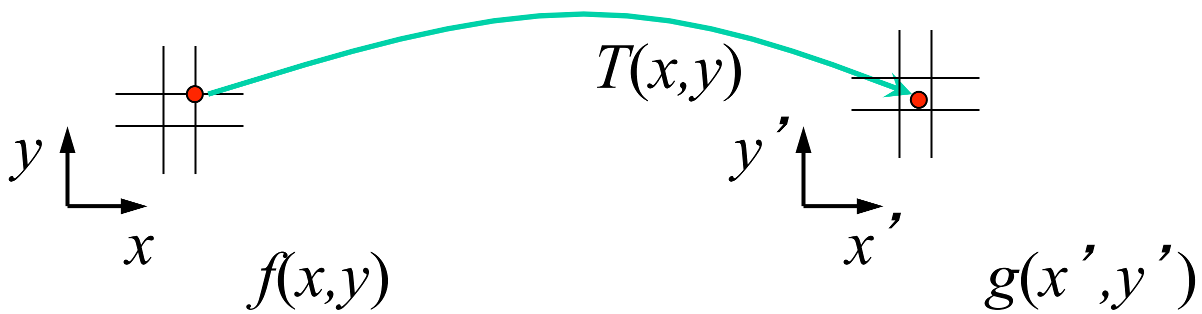
Forward warping



- Send each pixel $f(x,y)$ to its corresponding location
- $(x',y') = T(x,y)$ in the second image

Q: what if pixel lands "between" two pixels?

Forward warping



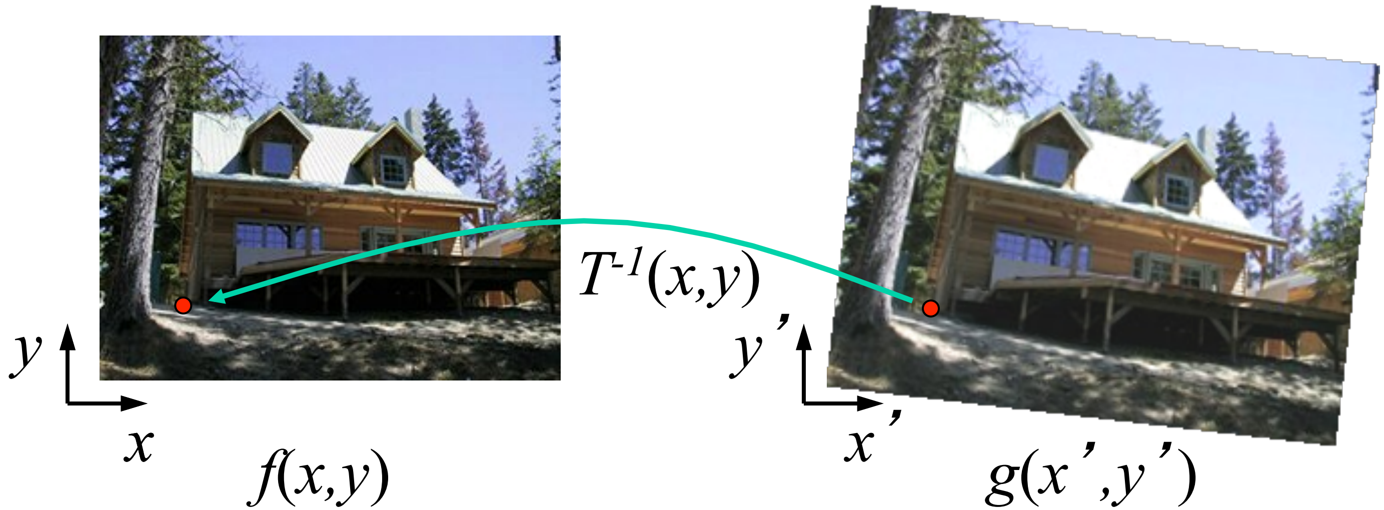
- Send each pixel $f(x,y)$ to its corresponding location
- $(x',y') = T(x,y)$ in the second image

Q: what if pixel lands "between" two pixels?

A: distribute color among neighboring pixels (x',y')

- Known as "splatting"

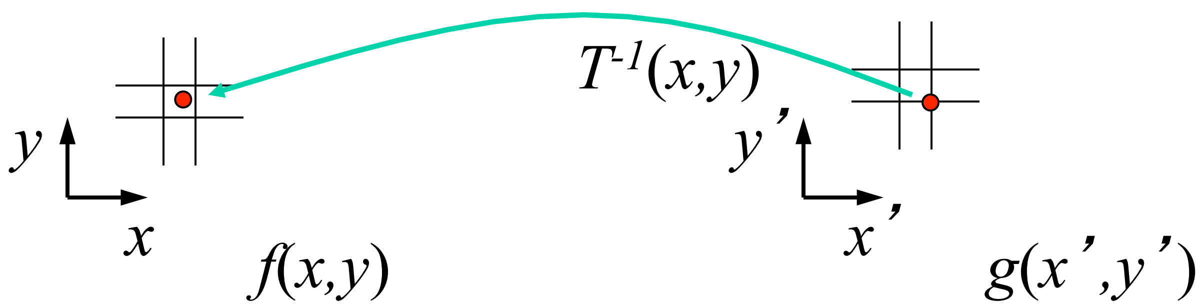
Inverse warping



- Get each pixel $g(x',y')$ from its corresponding location
- $(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

Inverse warping

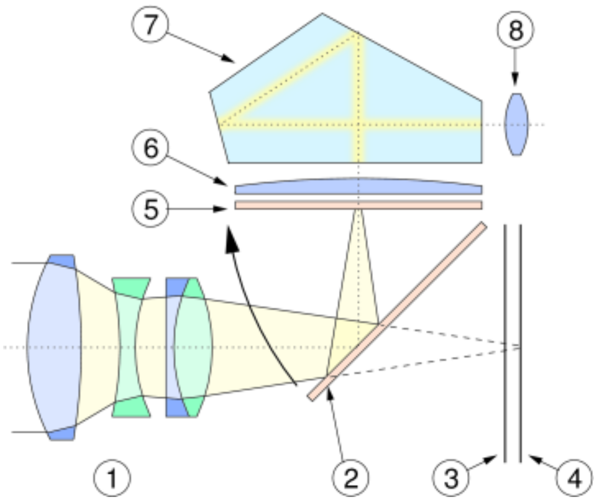


- Get each pixel $g(x',y')$ from its corresponding location
- $(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

A: *Interpolate* color value from neighbors
- nearest neighbor, bilinear, Gaussian, bicubic

Last 4 lectures



Camera Structure


HDR

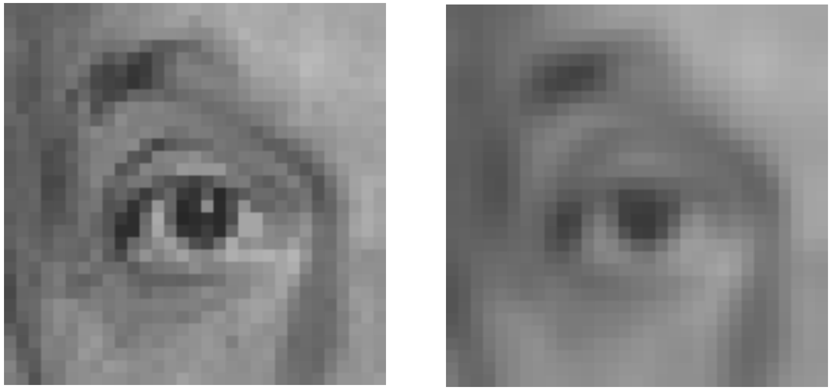


Image Filtering

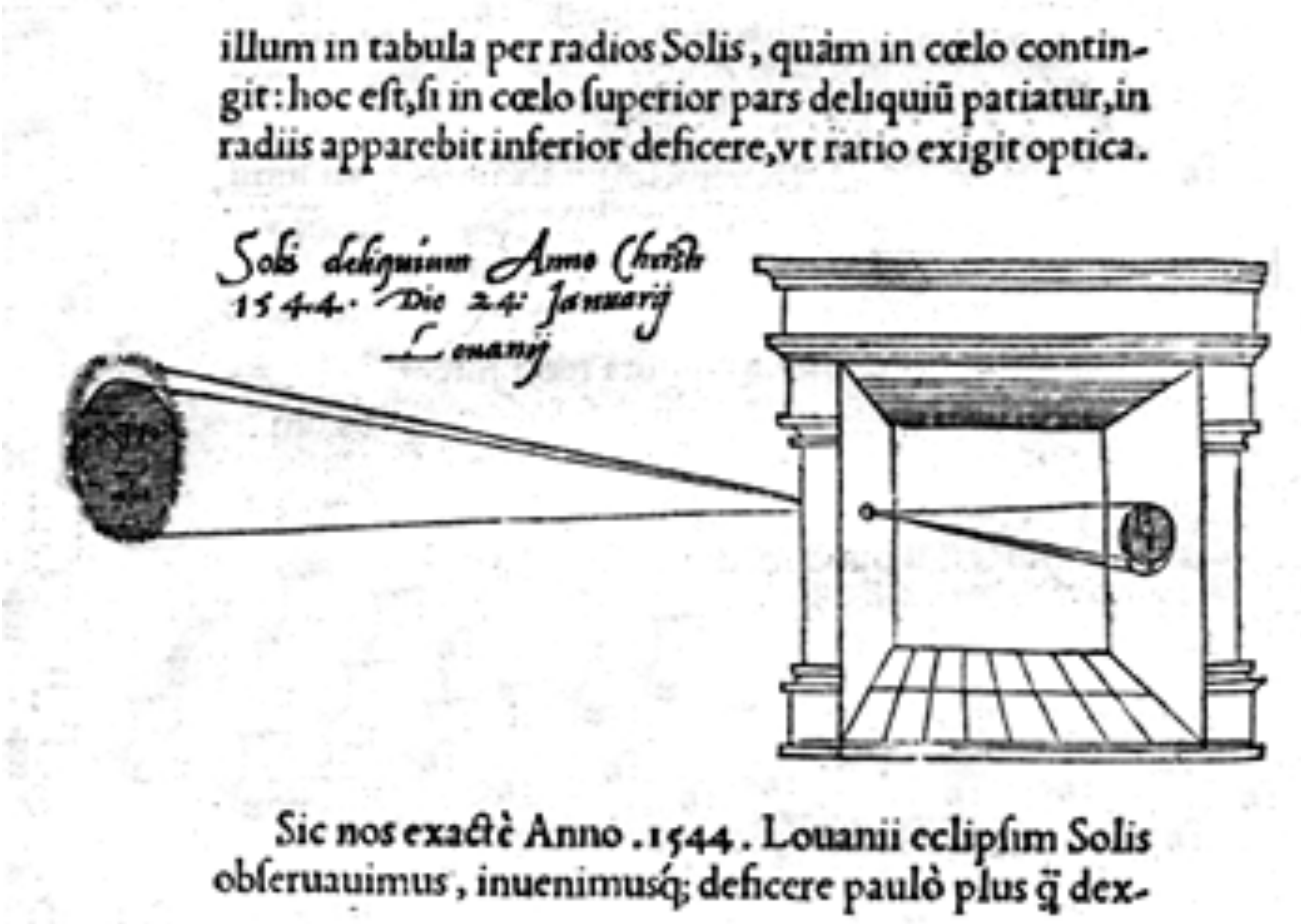

Image Transform

Next

Camera Projection

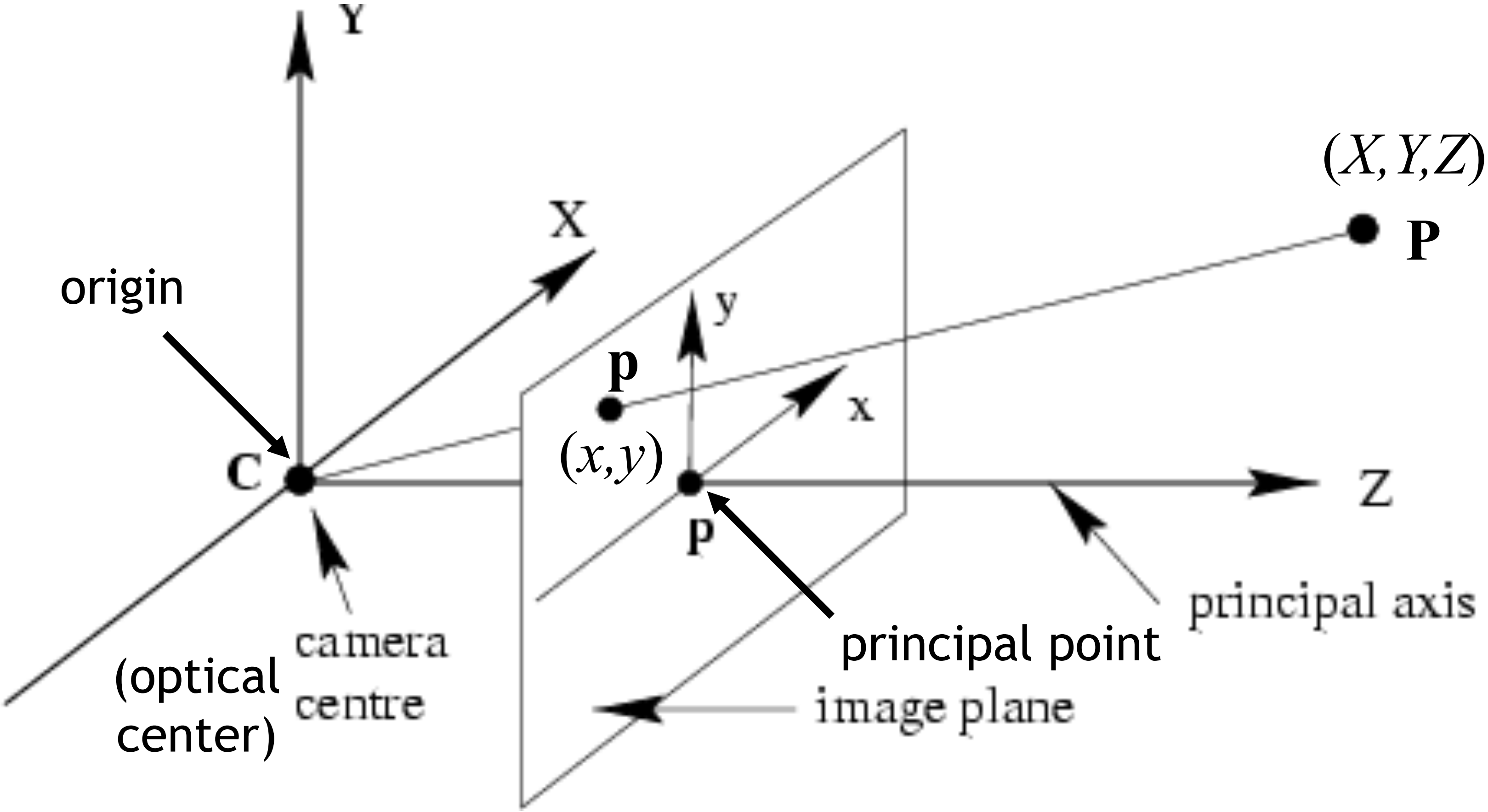
Camera Calibration

Pinhole camera


illum in tabula per radios Solis, quàm in cœlo contingit: hoc eſt,ſi in cœlo ſuperior pars deliquiũ patiatur,in radiis apparebit inferior deficere,vt ratio exigit optica.

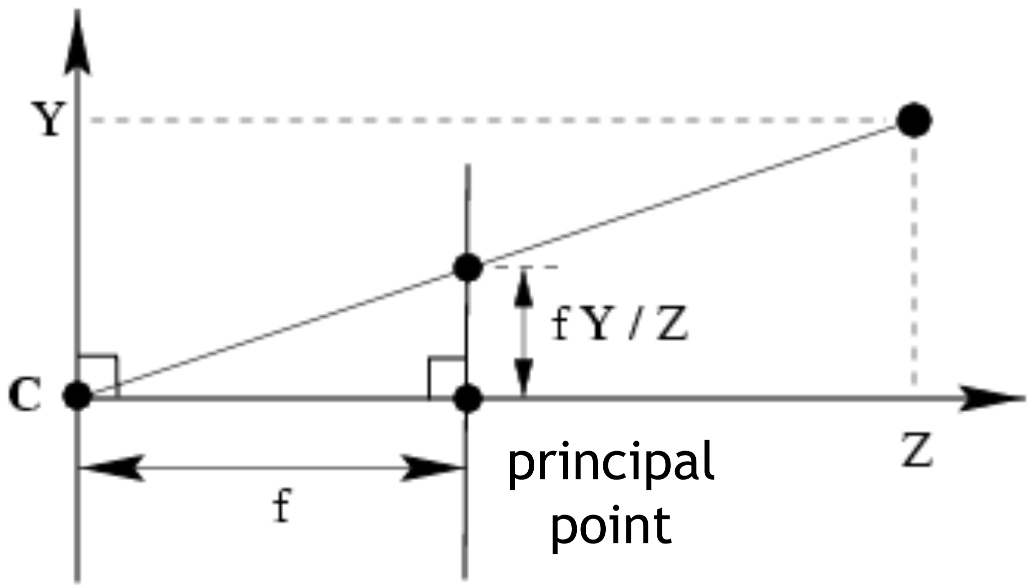
Sic nos exactè Anno .1544. Louanii eclipſim Solis obſeruauimus , inuenimusq́; deficere paulò plus q̄ dex-

Pinhole camera model



- The coordinate system
  - We will use the pin-hole model as an approximation
  - Put the optical center (**C**enter **O**f **P**rojection) at the origin
  - Put the image plane (**P**rojection **P**lane) *in front* of the COP (Why?)

Pinhole camera model

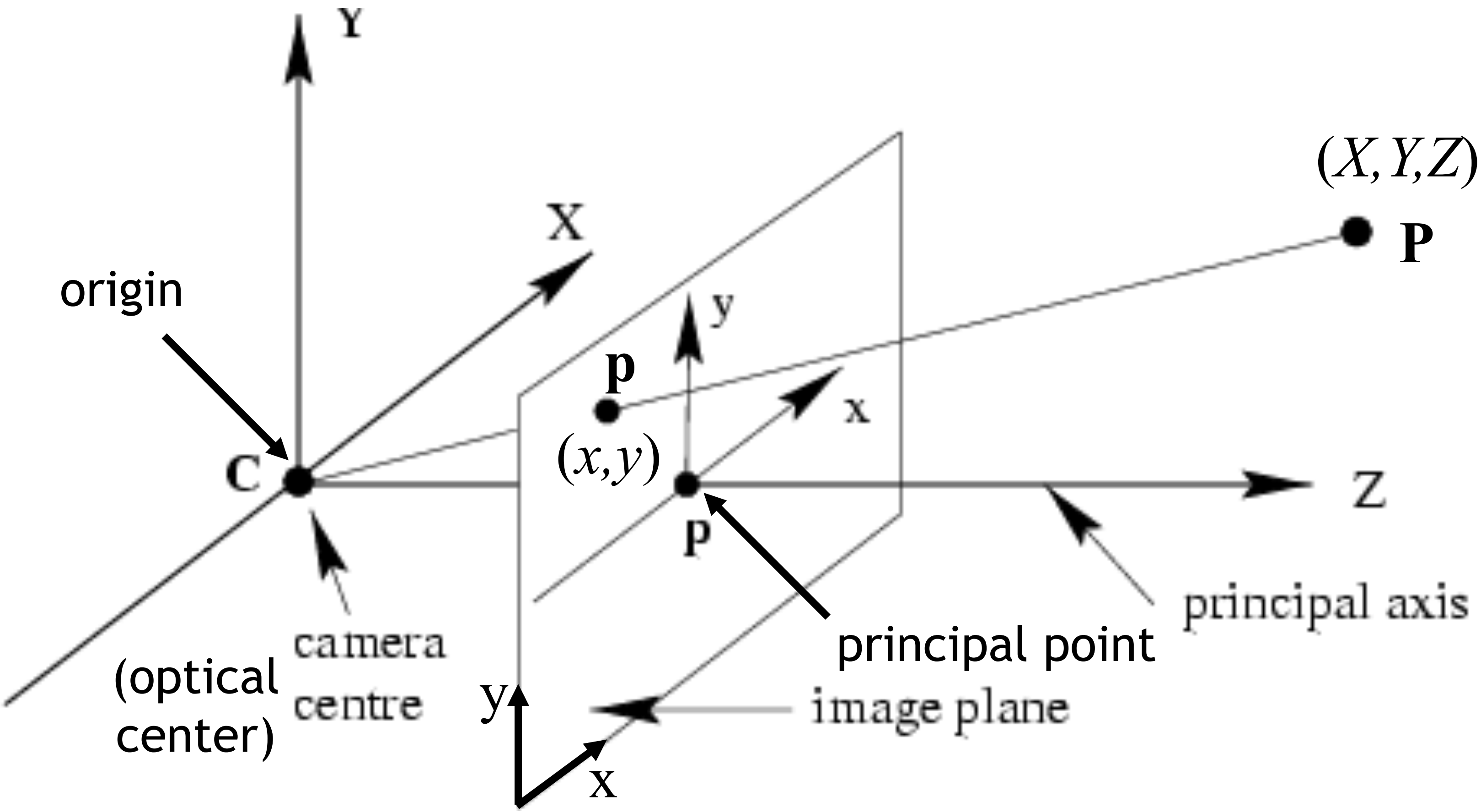


$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

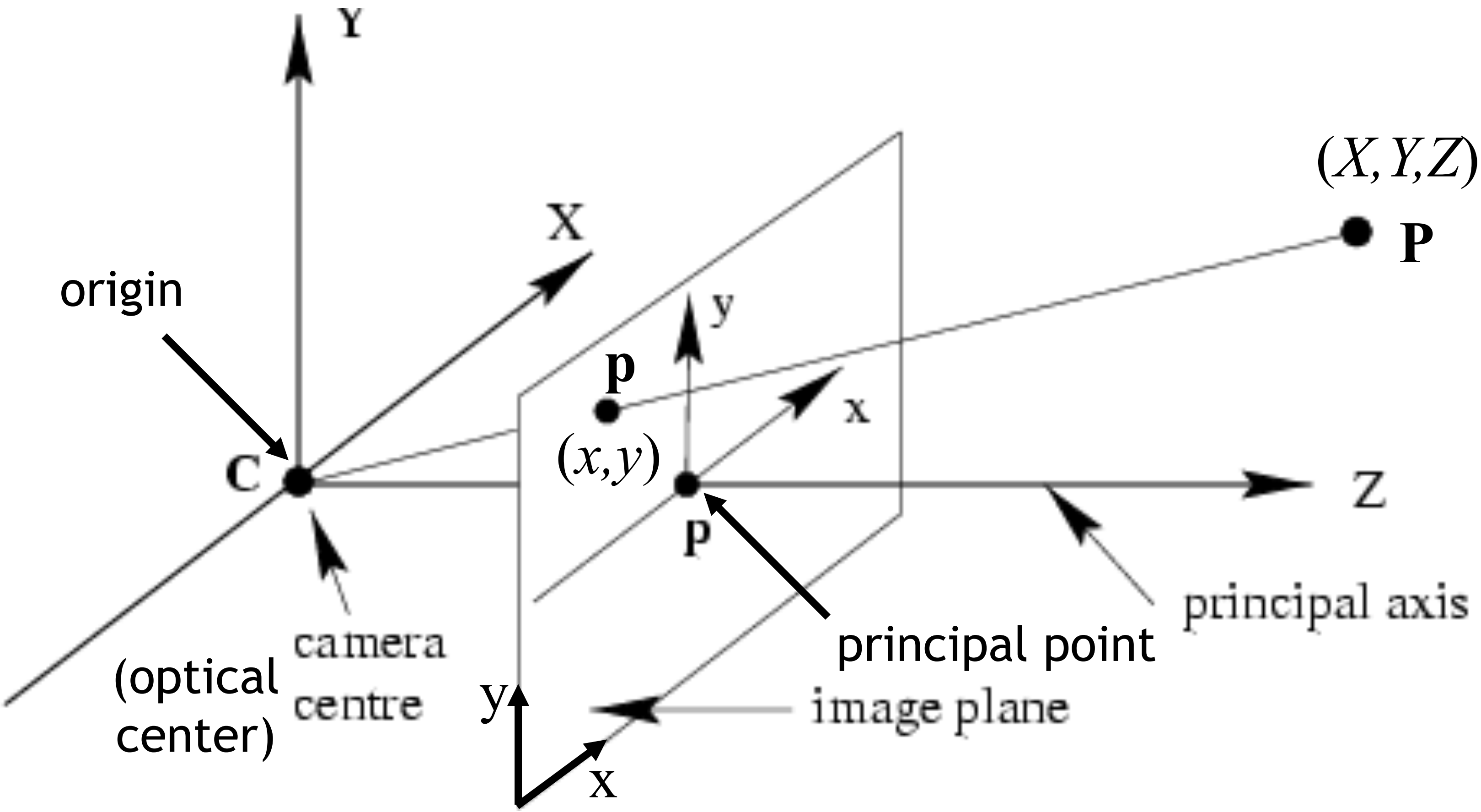
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Pinhole camera model



$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Pinhole camera model



$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Intrinsic matrix

Is this form of K good enough?

$$\mathbf{K} = \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

- non-square pixels (digital video)

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

Intrinsic matrix

Is this form of K good enough?

$$\mathbf{K} = \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

- non-square pixels (digital video)
- skew

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

Intrinsic matrix

Is this form of K good enough?

$$\mathbf{K} = \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

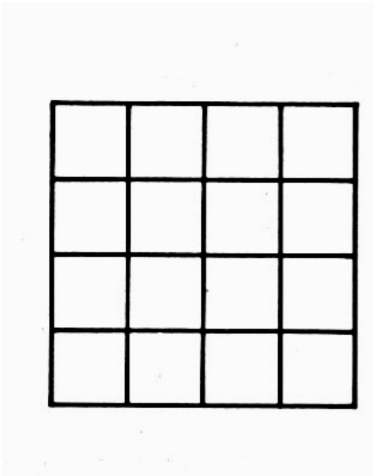
- non-square pixels (digital video)
- skew
- radial distortion

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

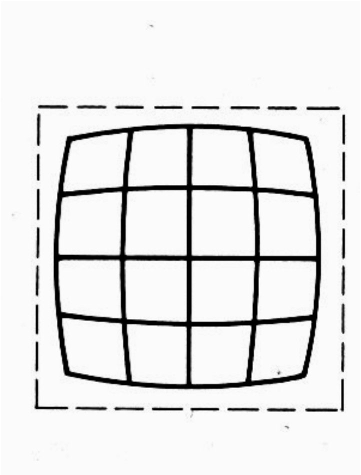
Distortion

- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens

Barrel Distortion



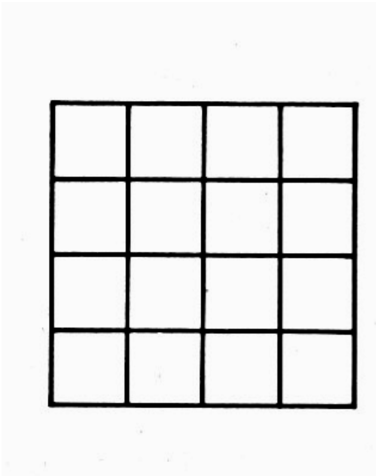
No distortion



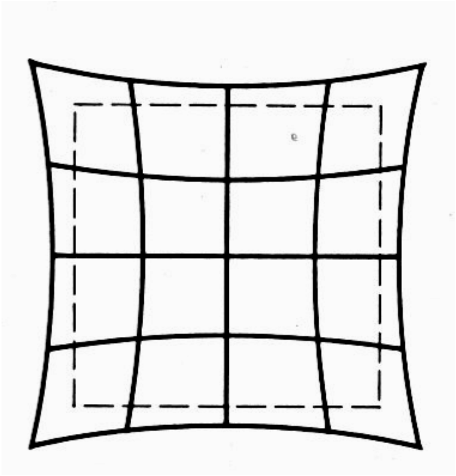
Barrel

Wide Angle Lens

Pin Cushion Distortion



No distortion



Pin cushion

Telephoto lens

Modeling distortion

Distortion-Free:

$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

With Distortion:

1. Project (X, Y, Z) to “normalized” image coordinates

$$x_n = \frac{X}{Z}$$

$$y_n = \frac{Y}{Z}$$

$$r^2 = x_n^2 + y_n^2$$

2. Apply radial distortion

$$x_d = x_n\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)$$

$$y_d = y_n\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)$$

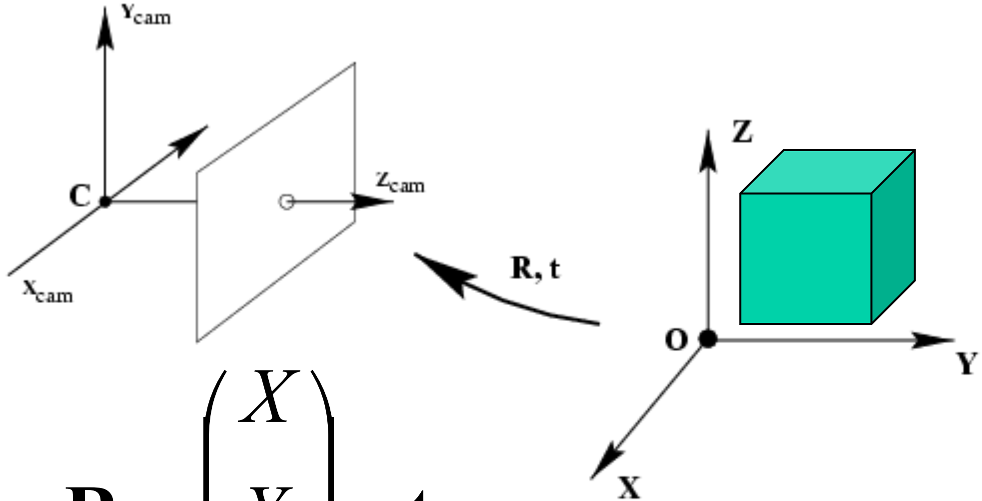
3. Apply focal length translate image center

$$x = f x_d + x_c$$

$$y = f y_d + y_c$$

- To model lens distortion
  - Use above projection operation instead of standard projection matrix multiplication

Camera rotation and translation



$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \mathbf{R}_{3\times 3} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{t}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} \,|\, \mathbf{t} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

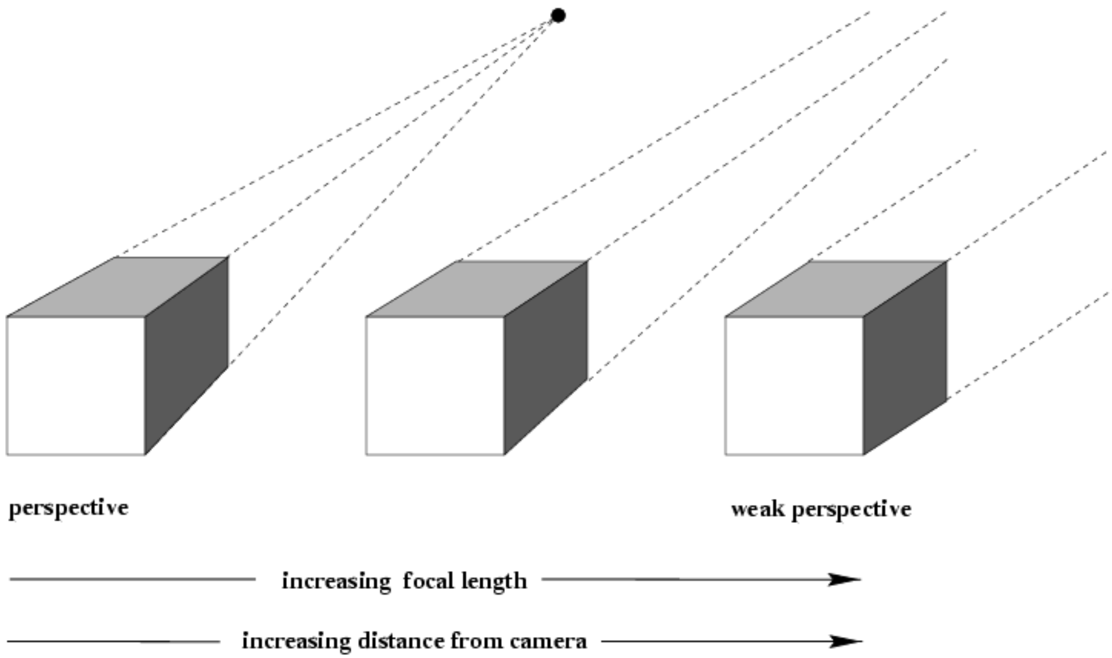
$$\mathbf{x} \sim \mathbf{K} \underbrace{\begin{bmatrix} \mathbf{R} \,|\, \mathbf{t} \end{bmatrix}}_{\text{extrinsic matrix}} \mathbf{X}$$

Two kinds of parameters

- *internal* or *intrinsic* parameters: focal length, optical center, skew
- *external* or *extrinsic* (pose): rotation and translation:

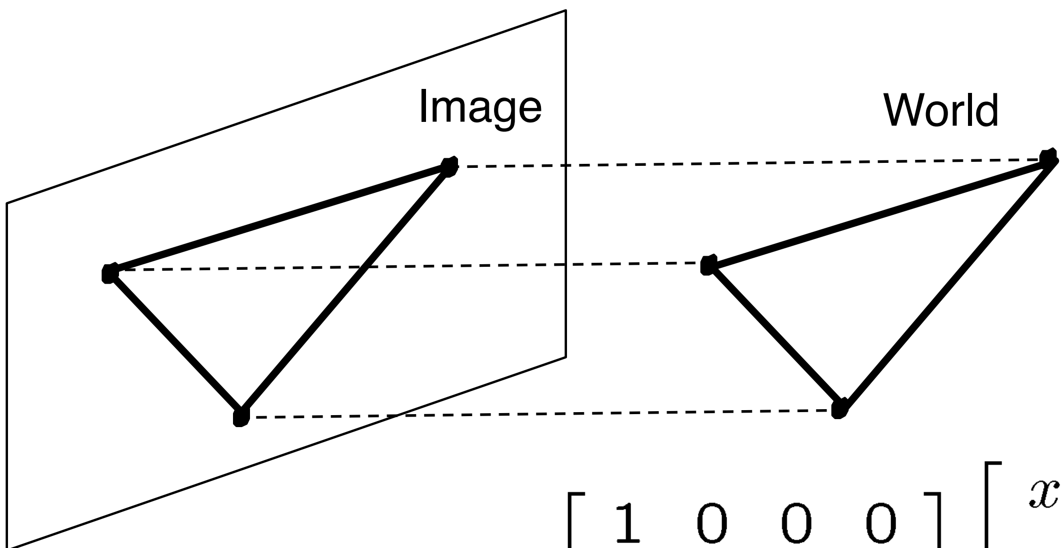
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} \left[\mathbf{R} | \mathbf{t} \right] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Other projection models



Orthographic projection

- Special case of perspective projection
  - Distance from the COP to the PP is infinite



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

  - Also called “parallel projection”: (x, y, z) → (x, y)

Other types of projections

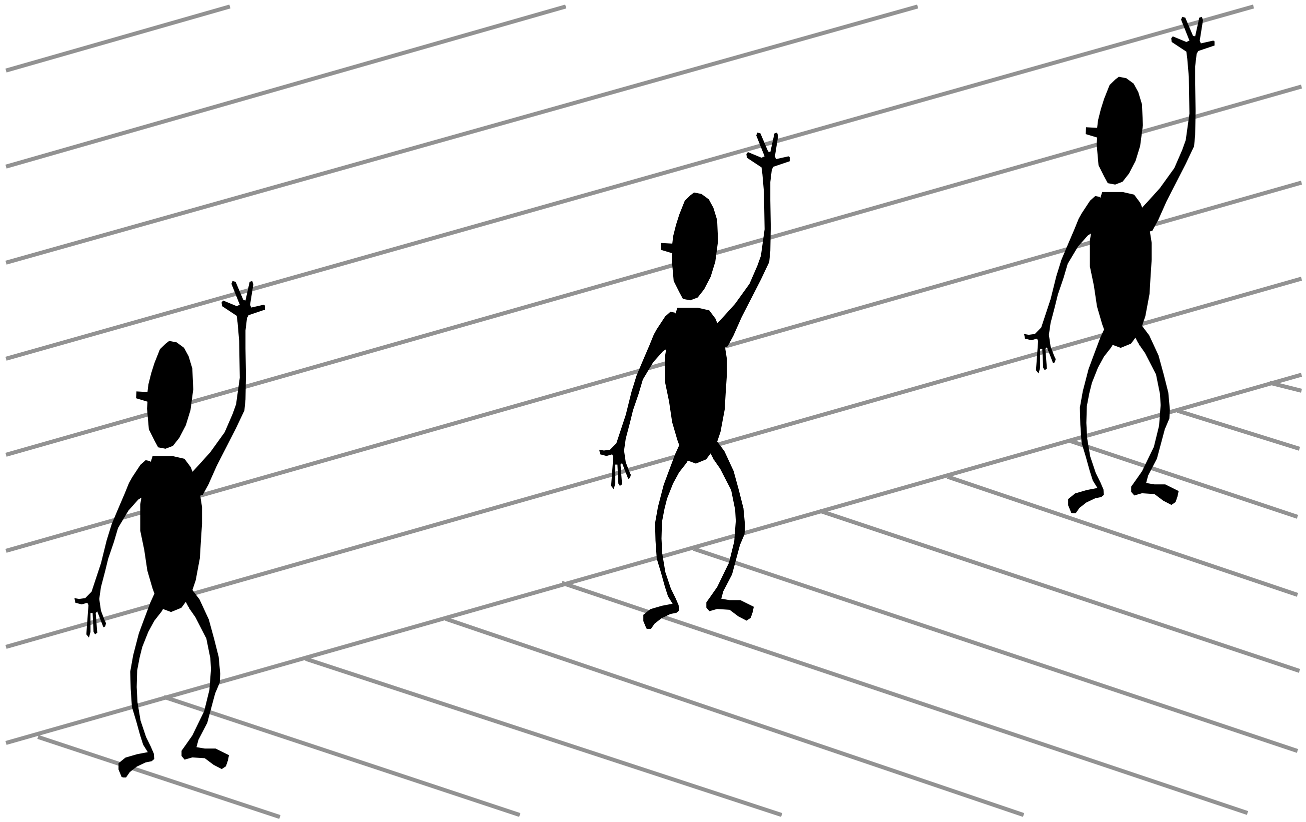
- Scaled orthographic
  - Also called “weak perspective”

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1/d \end{bmatrix} \Rightarrow (dx, dy)$$

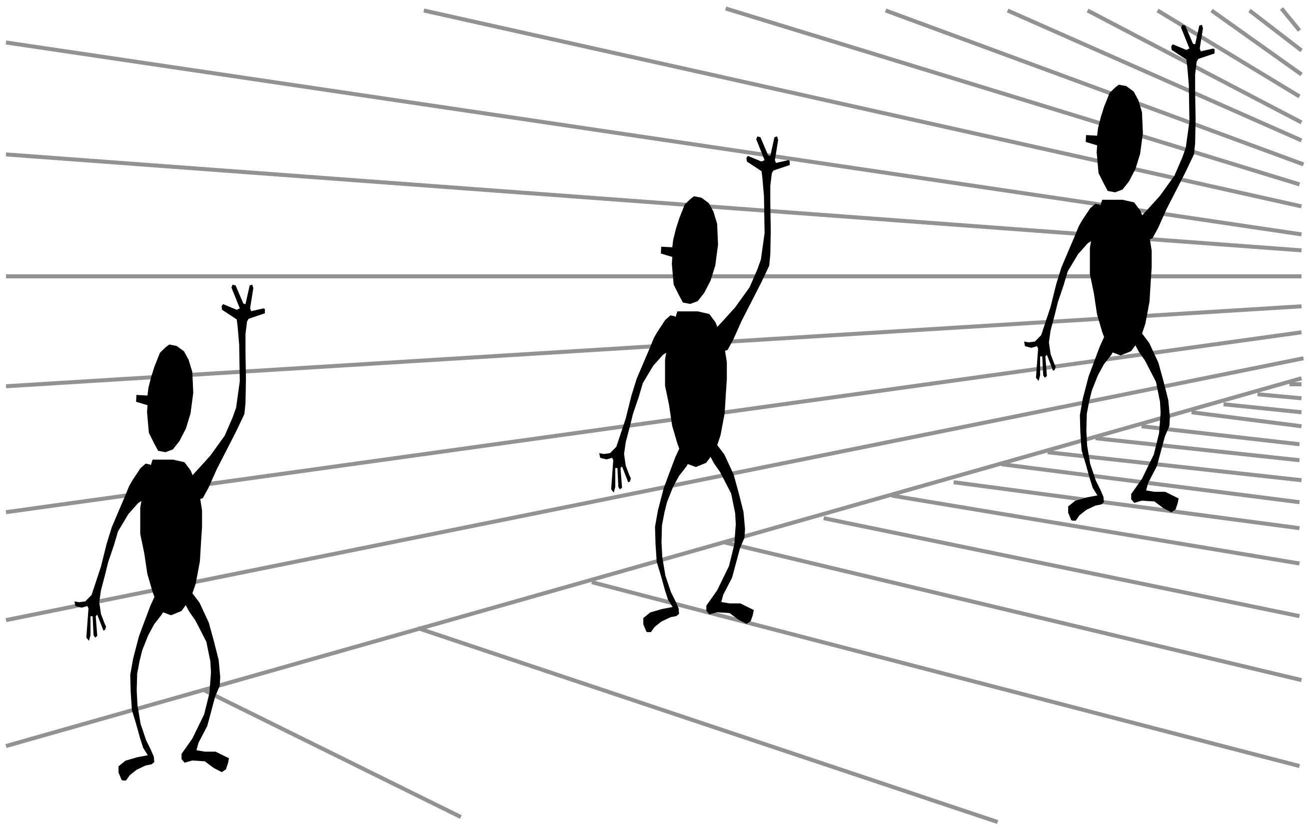
- Affine projection
  - Also called “paraperspective”

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

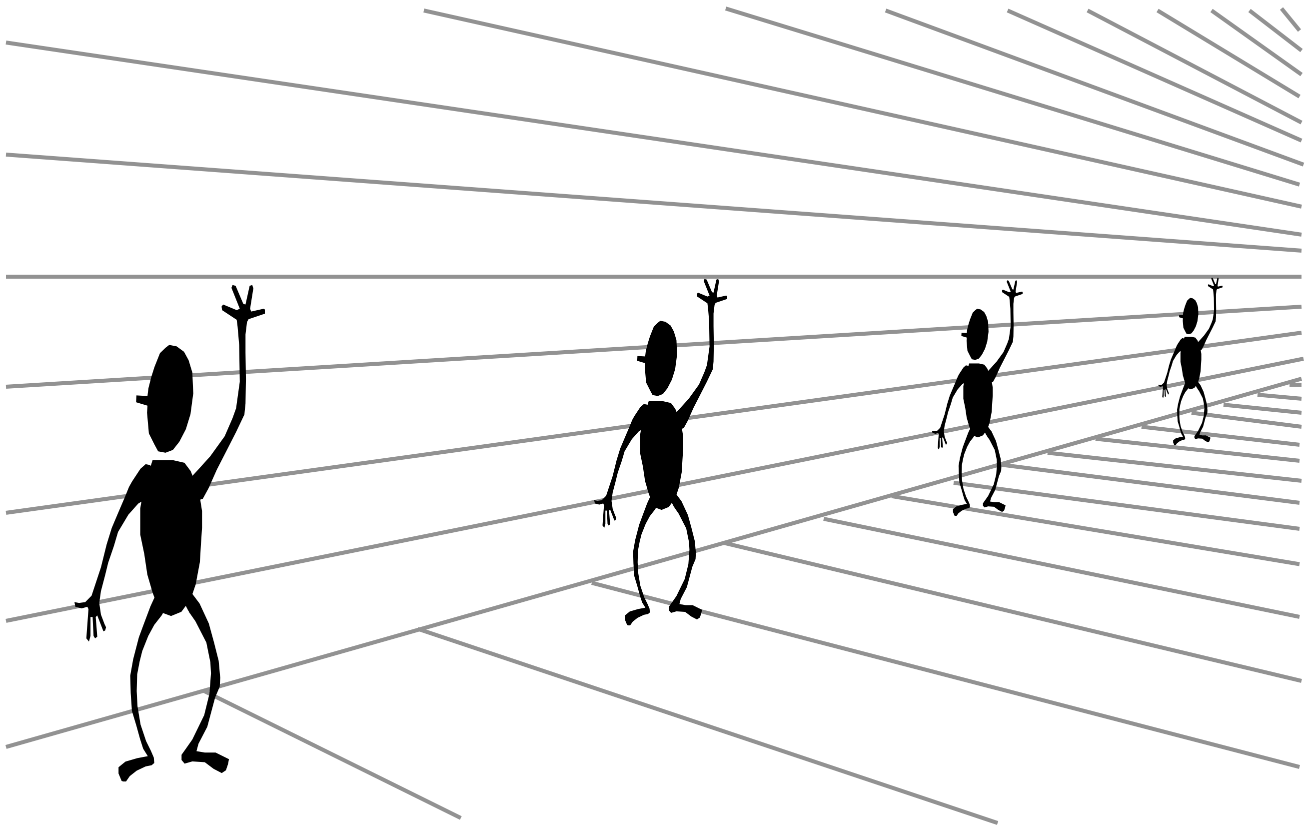
Fun with perspective



Perspective cues

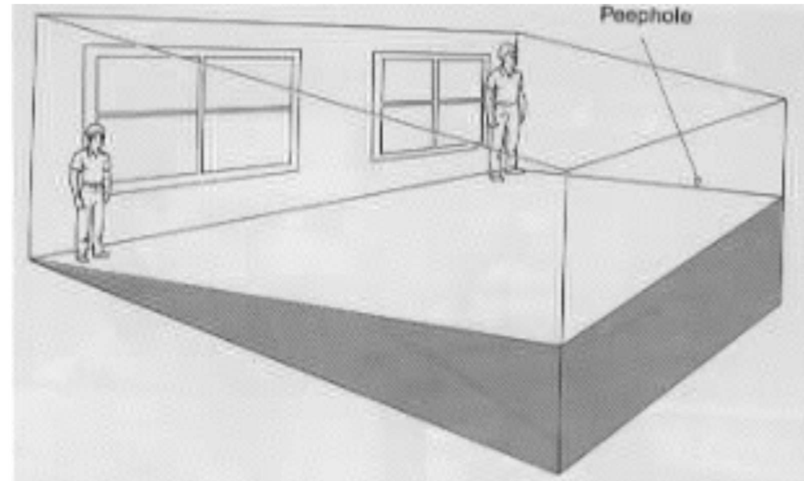


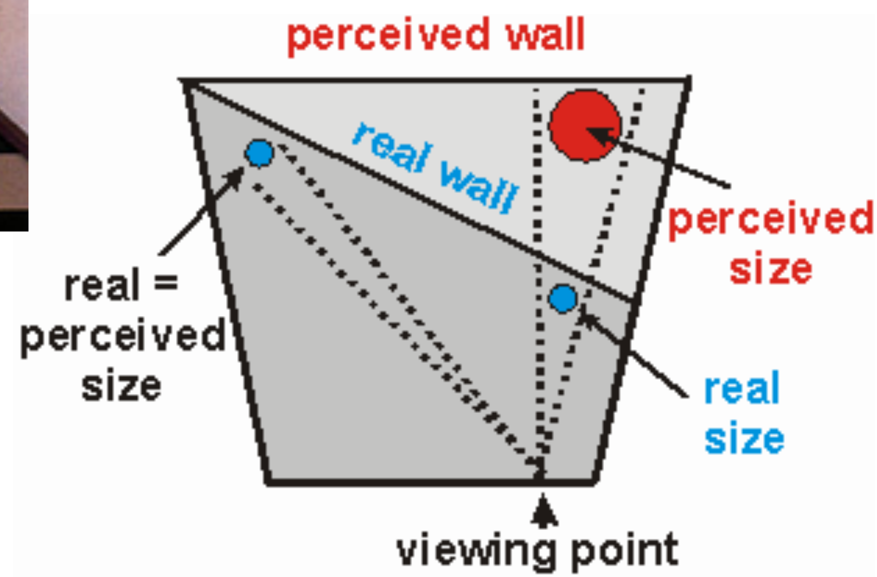
Perspective cues



Fun with perspective

Ames room





Forced perspective in LOTR

Elijah Wood: 5' 6" (1.68 m)	Ian McKellen 5' 11" (1.80 m)