CS4670/5670: Computer Vision

Kavita Bala

Lec 24: Single View Modeling 2

Announcements

- PA 3 is out
- Prelim grades out
  - Solutions on CMS

Points and lines

- Intersection of two lines
  - u1 = (a1, b1, c1), u2 = (a2, b2, c2)
  - p = (b1 c2 –b2c1, a2c1-a1c2, a1b2-a2b1)
  - p = u1 x u2
  - If u1 parallel to u2
    - p = (b1c2-b2c1,a2c1-a1c2,0)
- Given two points p1 and p2
  - Line through them u = p1 x p2

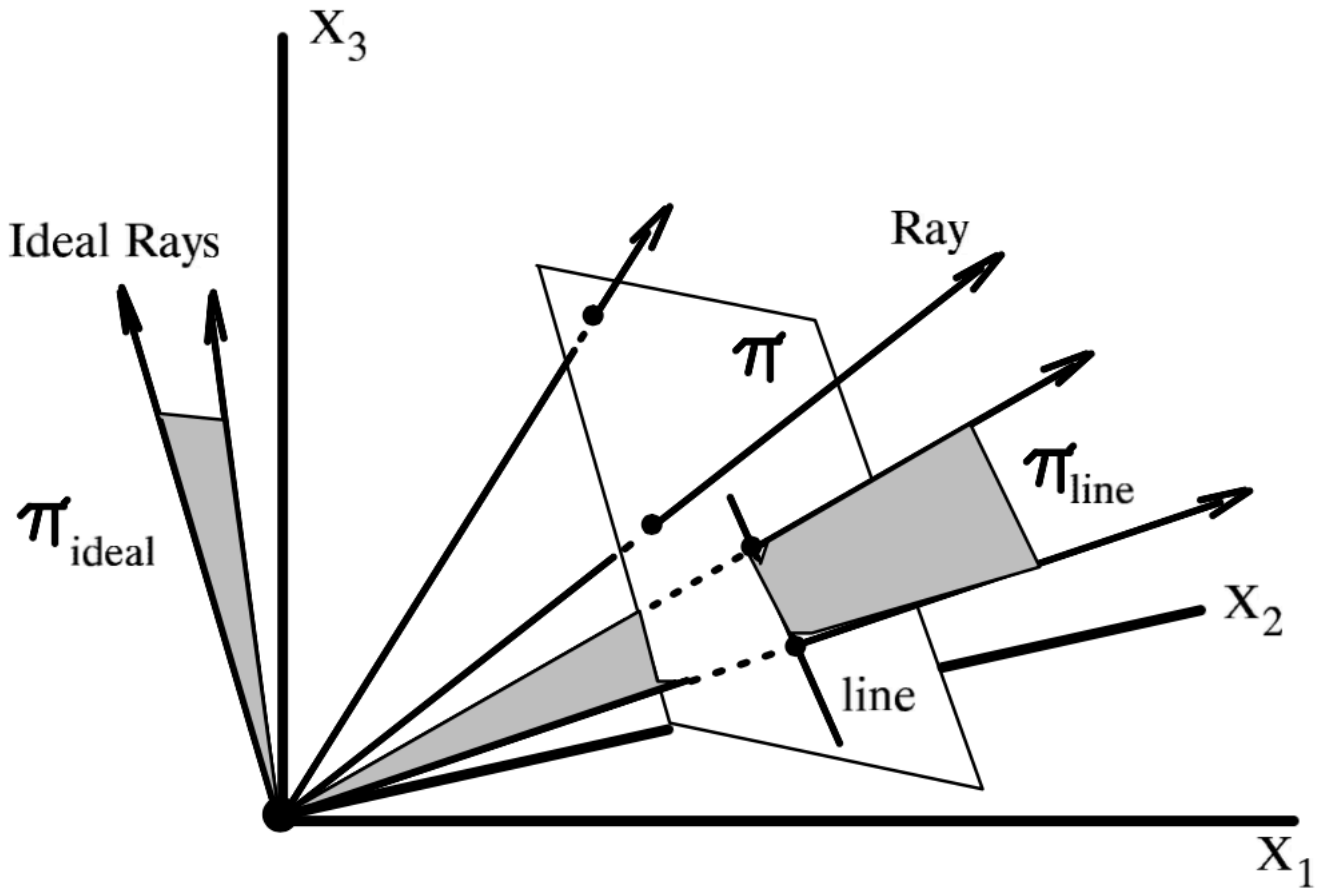
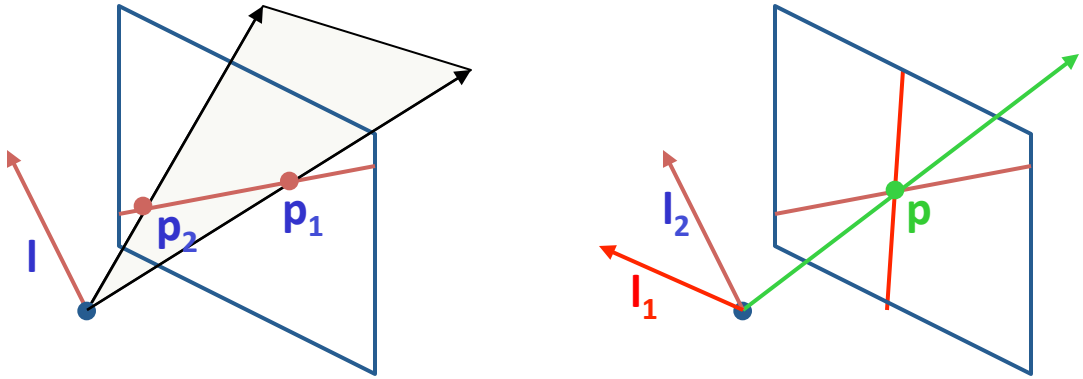


Figure 23.11
A model for the projective plane can be constructed by rays in 3D space. The rays correspond to points in the projective plane. Two rays through the origin define a unique plane though the origin. Any plane through the origin corresponds to a projective line.

Point and line duality

- A line **l** is a homogeneous 3-vector



What is the line **l** spanned by rays $\mathbf{p}_1$ and $\mathbf{p}_2$?

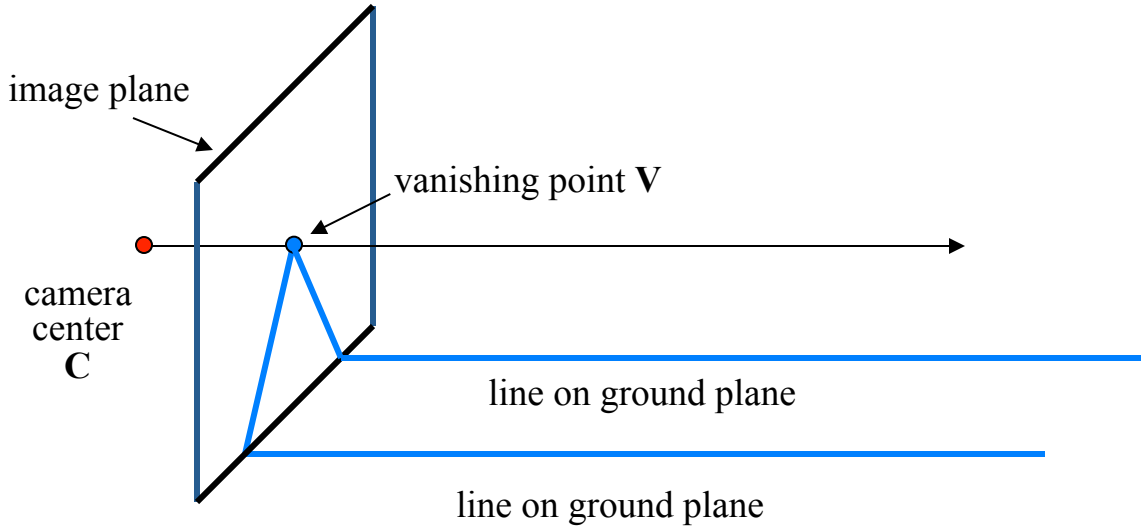
- **l** is $\perp$ to $\mathbf{p}_1$ and $\mathbf{p}_2$ $\Rightarrow$ $\mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2$
- **l** can be interpreted as a *plane normal*

What is the intersection of two lines $\mathbf{l}_1$ and $\mathbf{l}_2$?

- **p** is $\perp$ to $\mathbf{l}_1$ and $\mathbf{l}_2$ $\Rightarrow$ $\mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2$

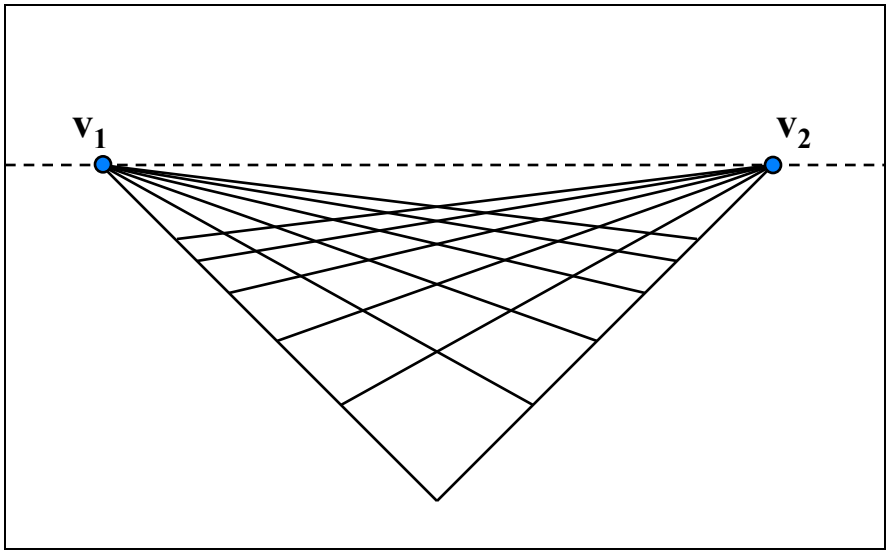
Points and lines are *dual* in projective space

Vanishing points



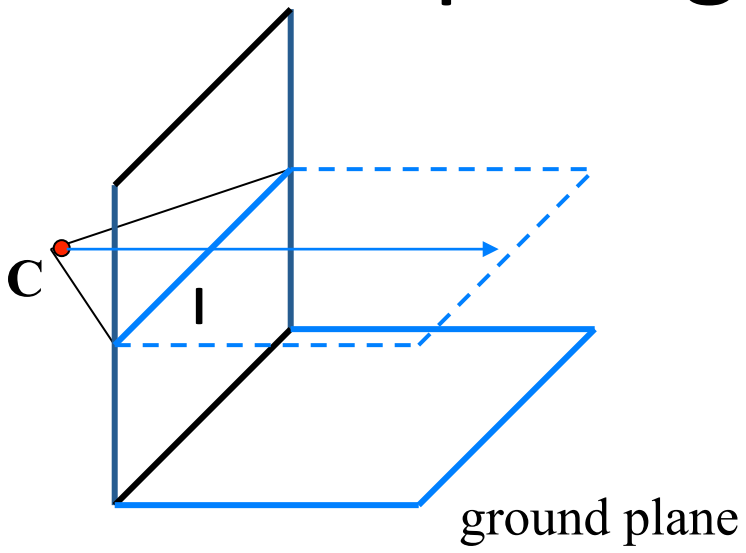
- Properties
  - Any two parallel lines (in 3D) have the same vanishing point **v**
  - The ray from **C** through **v** is parallel to the lines
  - An image may have more than one vanishing point
    - in fact, every image point is a potential vanishing point

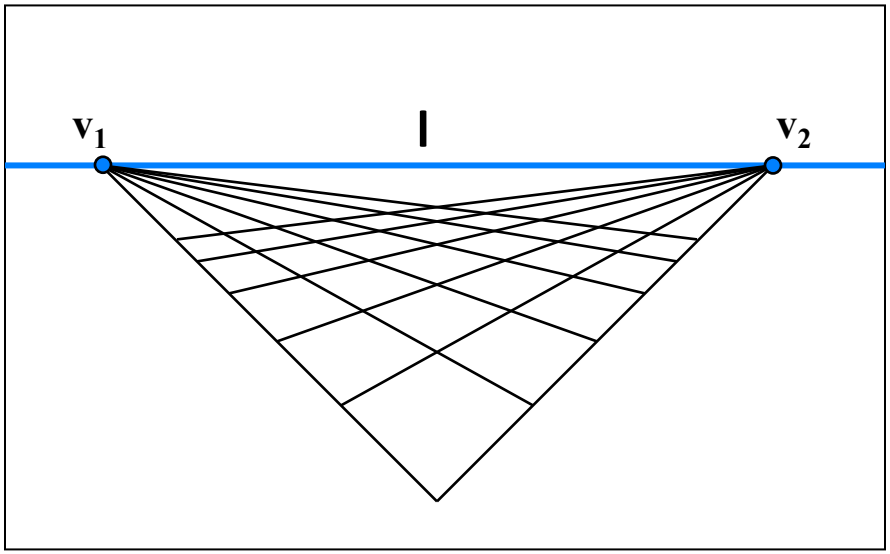
Vanishing lines



- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the *horizon line*
    - also called *vanishing line*
  - Note that different planes (can) define different vanishing lines

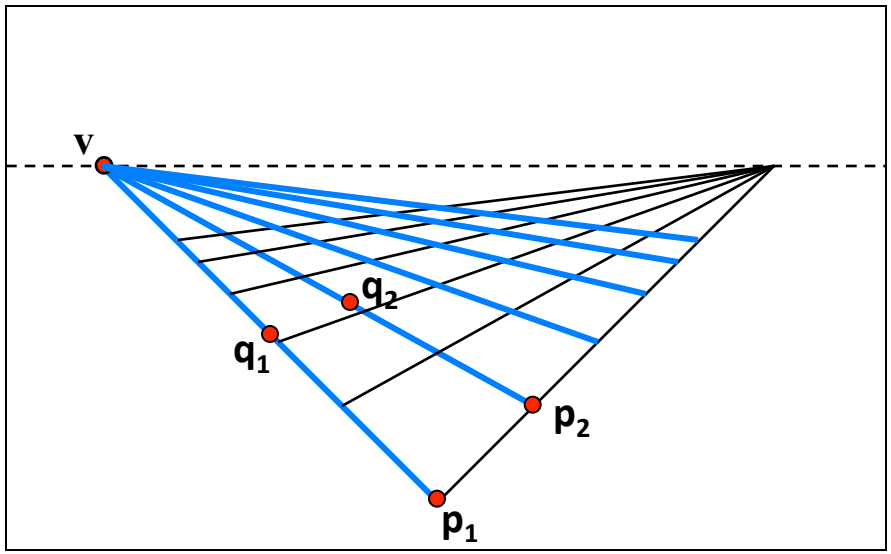
Computing vanishing lines





- Properties
  - **l** is intersection of horizontal plane through **C** with image plane
  - Compute **l** from two sets of parallel lines on ground plane
  - All points at same height as **C** project to **l**
    - points higher than C project above l
  - Provides way of comparing height of objects in the scene

Computing vanishing points (from lines)



- Intersect p_1q_1 with p_2q_2

$$v = (p_1 \times q_1) \times (p_2 \times q_2)$$

Least squares version

- Better to use more than two lines and compute the "closest" point of intersection
- See notes by Bob Collins for one good way of doing this:
  - http://www-2.cs.cmu.edu/~ph/869/www/notes/vanishing.txt

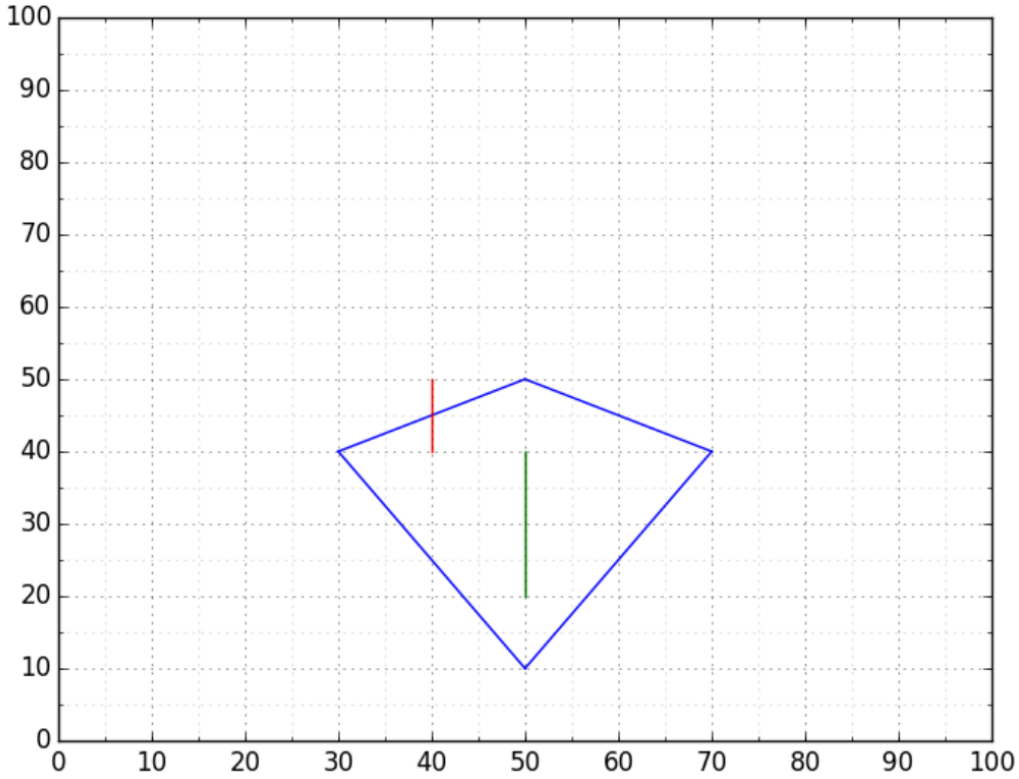


Figure 1: A square viewed under perspective projection with two objects.

if you only have two lines, l1 and l2, you can compute a homogeneous coordinate vector V representing their point of intersection as the cross product of these two line vectors

$$V = l1 \times l2$$

scaling V so that the last coordinate is 1, i.e. (Vx, Vy, 1), and you have Vx and Vy as the point in the image that is the vanishing point. It is better to leave the vanishing point as a homogeneous coordinate vector, however, because the vanishing point could be very far off the image, or even at infinity (in which case the third component of V is 0, and you get a divide by zero when you try to scale).

One word about numerical conditioning is in order. If you just use image pixel coordinates directly, the problem will be horribly poorly conditioned. There is a good paper by Hartley on this topic, title something like "In defense of the Eight-point algorithm." If your lines are spread out throughout the image, you can get approximately the kind of well-conditioned approach he mentions by doing the following.

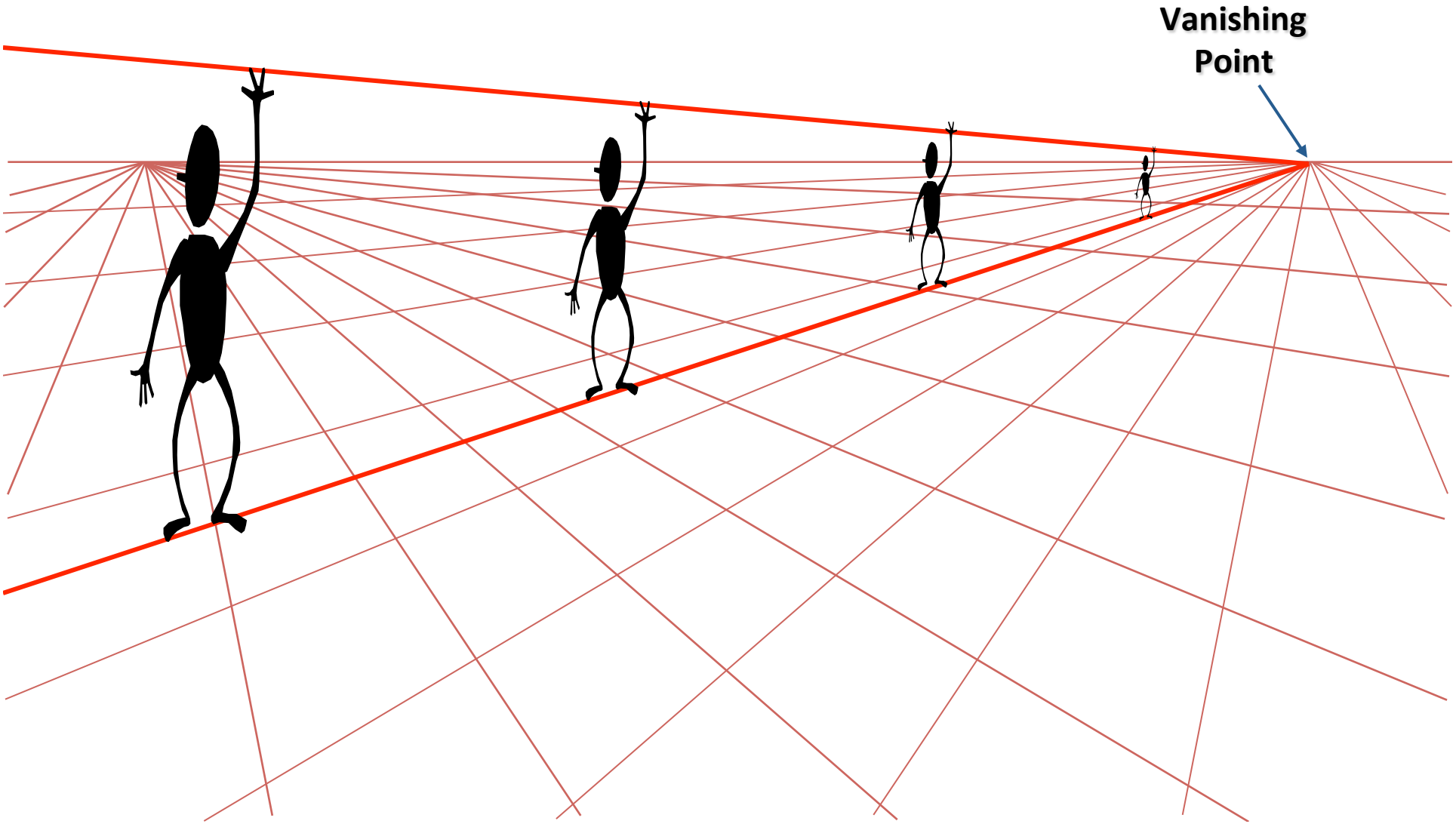
1) translate by (-imageX/2, -imageY/2) so that (0,0) is in the center of the image [I think Hartley's conditioning method would take the center of mass of the line endpoints as the origin].

2) Scale coordinates so that the magnitude of all image point homogeneous coordinates is roughly 1 [Hartley says how to do this exactly, in his paper]. I used to do it approximately by just setting the constant w, mentioned during the first step of this algorithm, to be equal to half the image size in pixels. So if you had a 256x256 image, then you would have w = 128. IF the image width and height aren't the same, just take the average or something, before dividing by two.

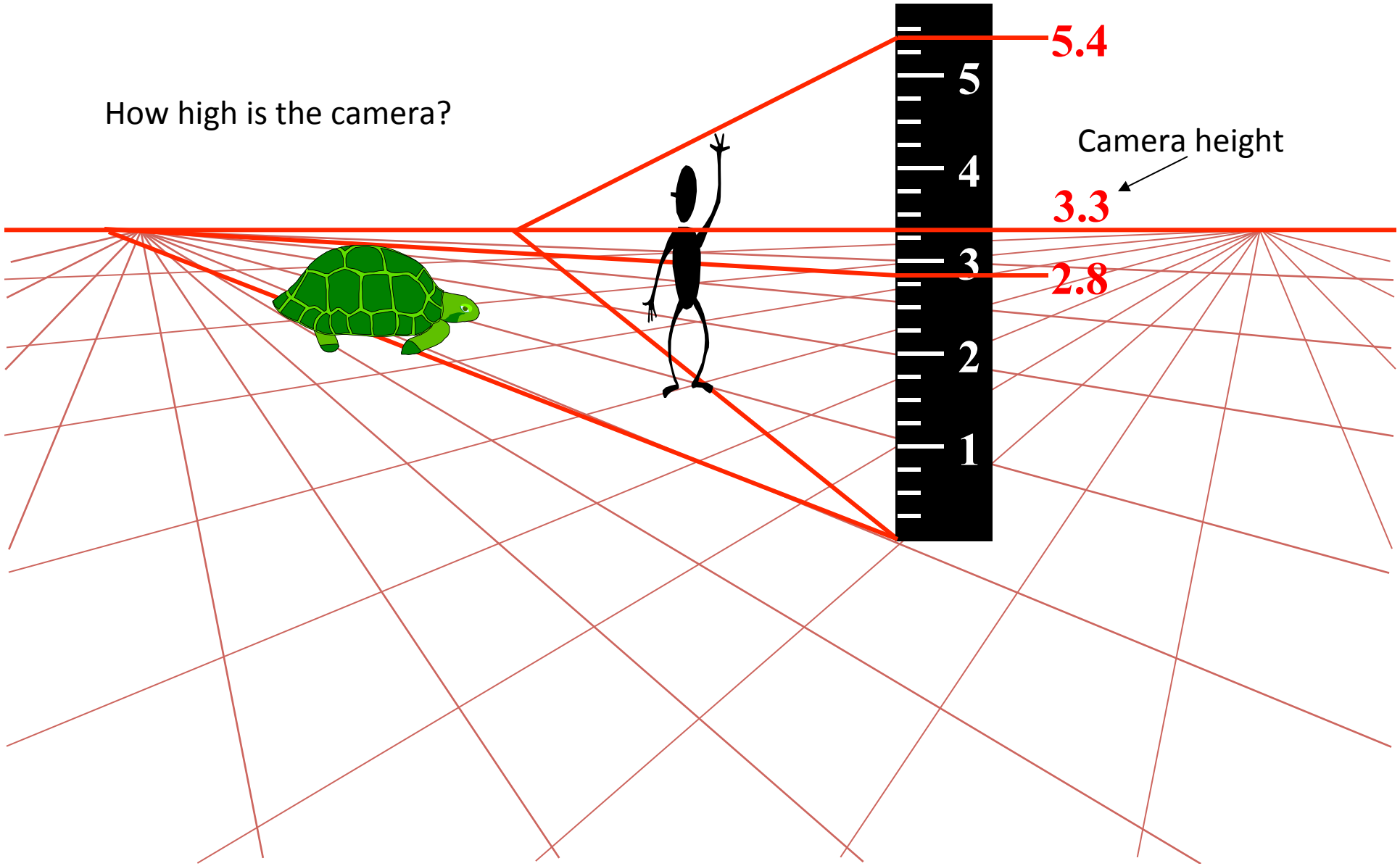
Vanishing points are useful

- Recover size
- Camera calibration
- …

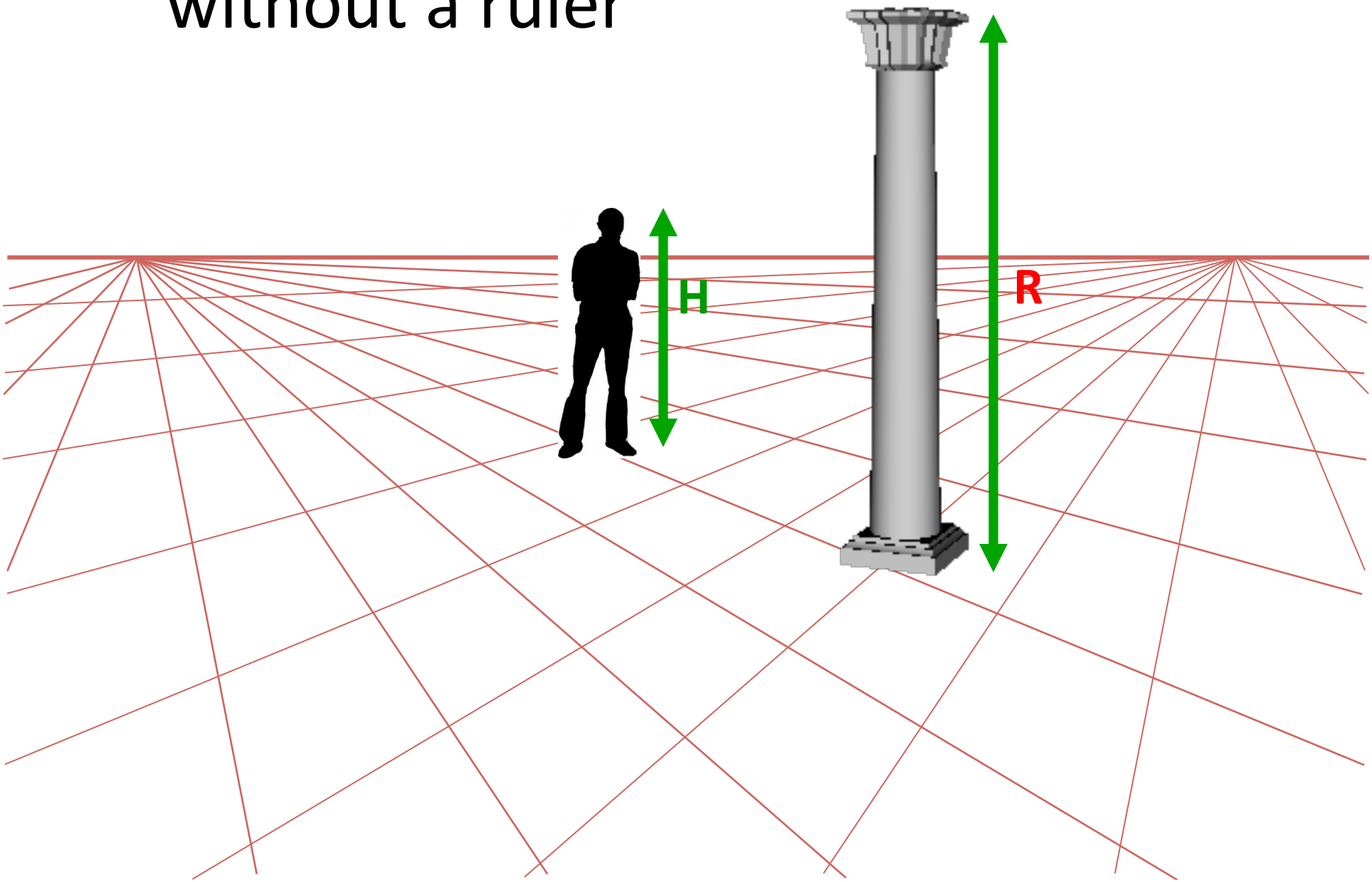
Comparing heights



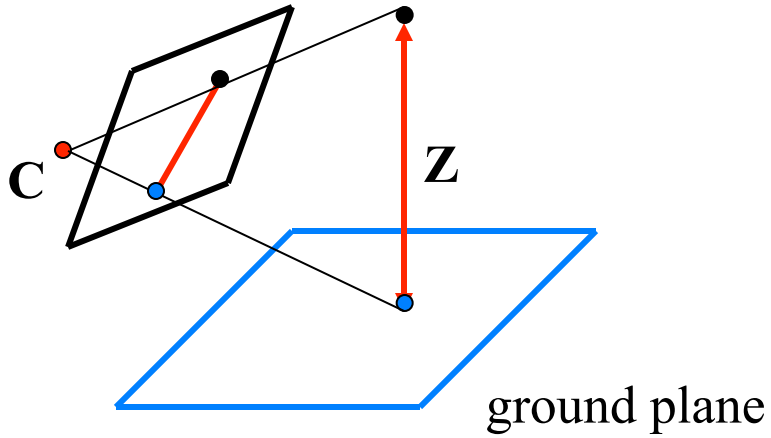
Measuring height



Measuring height without a ruler



Measuring height without a ruler



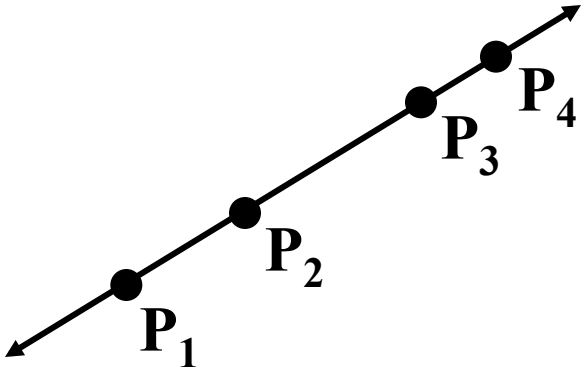
Compute Z from image measurements

Actually get a scaled version

The cross ratio

- A Projective Invariant
  - Something that does not change under projective transformations (including perspective projection)

The *cross-ratio* of 4 collinear points: ratio of ratios



$$\frac{\|\mathbf{P}_3 - \mathbf{P}_1\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_3 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_1\|}$$

$$\mathbf{P}_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

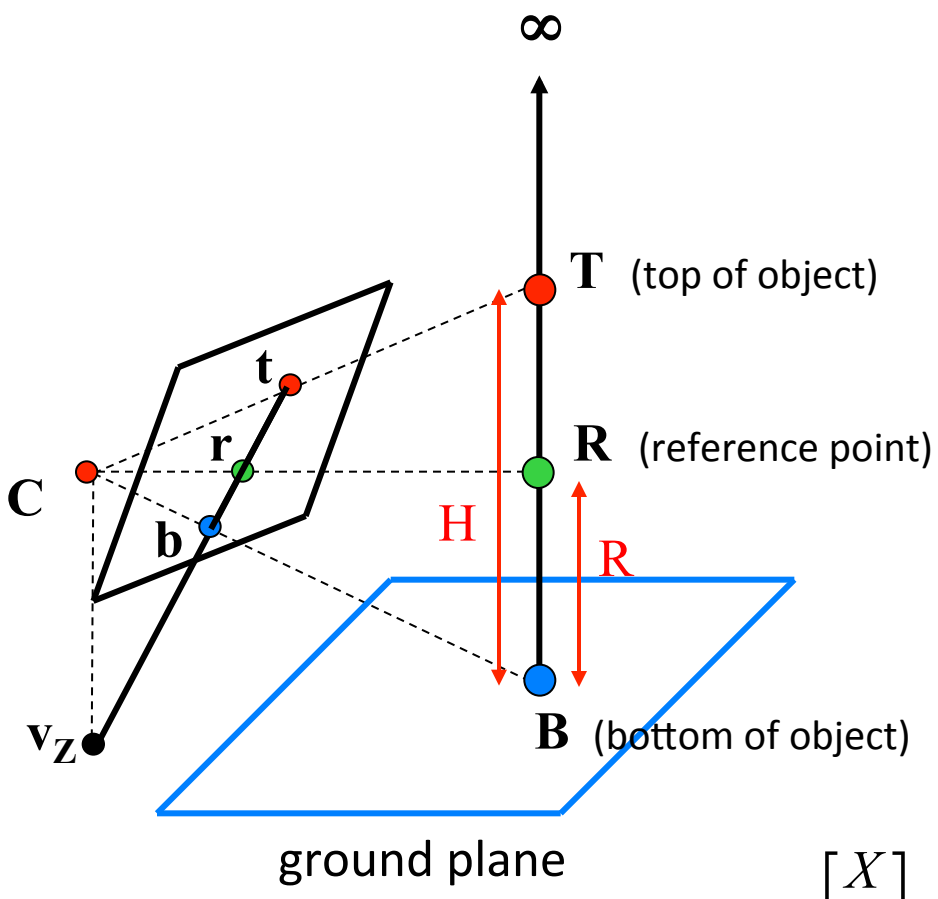
Can permute the point ordering

- 4! = 24 different orders (but only 6 distinct values)

$$\frac{\|\mathbf{P}_1 - \mathbf{P}_3\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_1 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_3\|}$$

This is the fundamental invariant of projective geometry

Measuring height



$$\frac{\|\mathbf{T}-\mathbf{B}\|\,\|\infty-\mathbf{R}\|}{\|\mathbf{R}-\mathbf{B}\|\,\|\infty-\mathbf{T}\|}=\frac{H}{R}$$

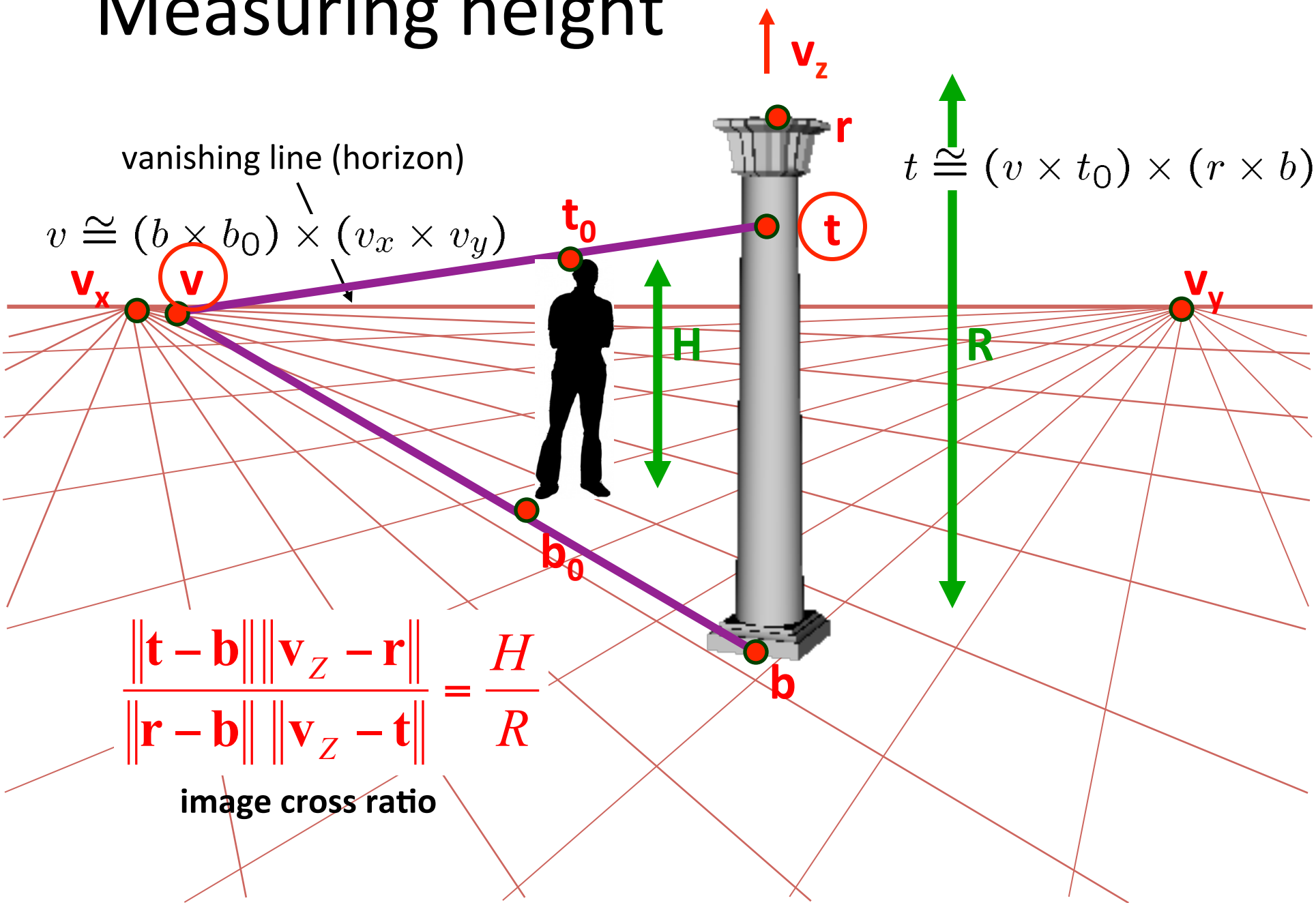
scene cross ratio

$$\frac{\|\mathbf{t}-\mathbf{b}\|\|\mathbf{v}_Z-\mathbf{r}\|}{\|\mathbf{r}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{t}\|}=\frac{H}{R}$$

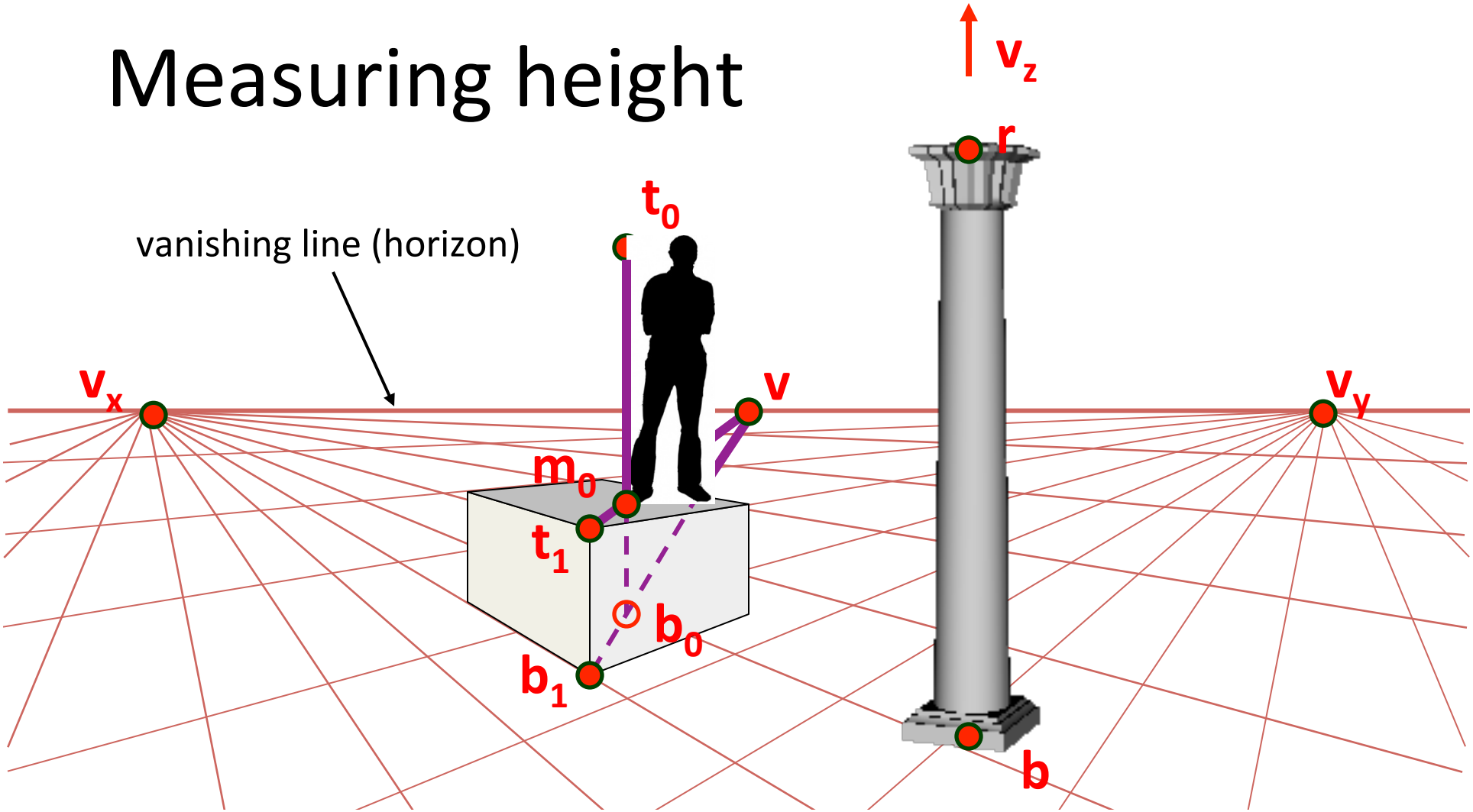
image cross ratio

scene points represented as $\mathbf{P}=\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$ image points as $\mathbf{p}=\begin{bmatrix}x\\y\\1\end{bmatrix}$

Measuring height



Measuring height



What if the point on the ground plane $\mathbf{b_0}$ is not known?

- Here the guy is standing on the box, height of box is known
- Use one side of the box to help find $\mathbf{b_0}$ as shown above

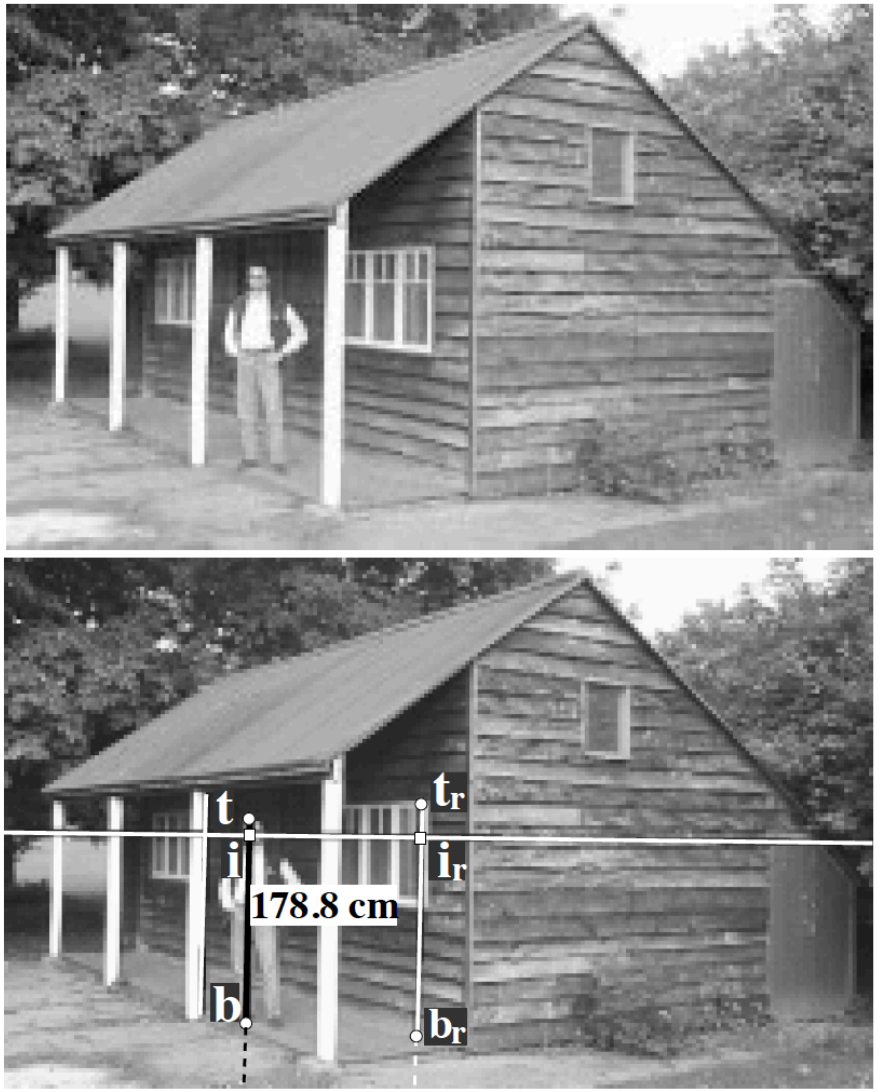
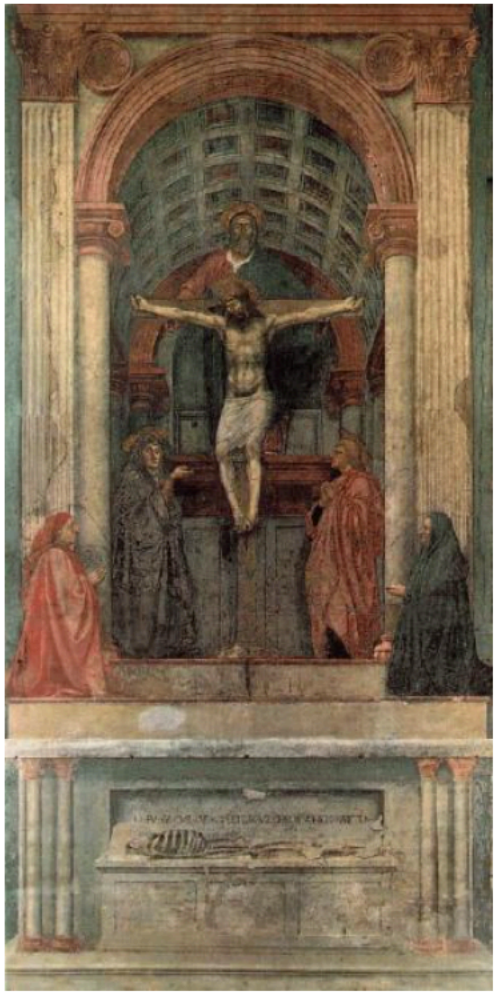


Figure 3: Measuring the height of a person: (top) original image; (bottom) the height of the person is computed from the image as 178.8cm (the true height is 180cm, but note that the person is leaning down a bit on his right foot). The vanishing line is shown in white and the reference height is the segment $(\mathbf{t}_r, \mathbf{b}_r)$. The vertical vanishing point is not shown since it lies well below the image. $\mathbf{t}$ is the top of the head and $\mathbf{b}$ is the base of the feet of the person while $\mathbf{i}$ is the intersection with the vanishing line.

The following example shows the reconstruction of a chapel depicted in one of the earliest and most famous Renaissance frescoes: **La Trinita' (The Trinity)** (1427) by Masaccio (1401-1428).

original fresco



***La Trinita'* (1427)**
by Masaccio

images of the reconstructed 3D model

3D Modeling from a photograph

Flagellation, Piero della Francesca

3D Modeling from a photograph

video by Antonio Criminisi

3D Modeling from a photograph

3D Modeling from a photograph

St. Jerome in his Study, H. Steenwick

3D Modeling from a photograph

Some Related Techniques

- Image-Based Modeling and Photo Editing
  - Mok et al., SIGGRAPH 2001
- Single View Modeling of Free-Form Scenes
  - Zhang et al., CVPR 2001
- Tour Into The Picture
  - Anjyo et al., SIGGRAPH 1997

Camera calibration

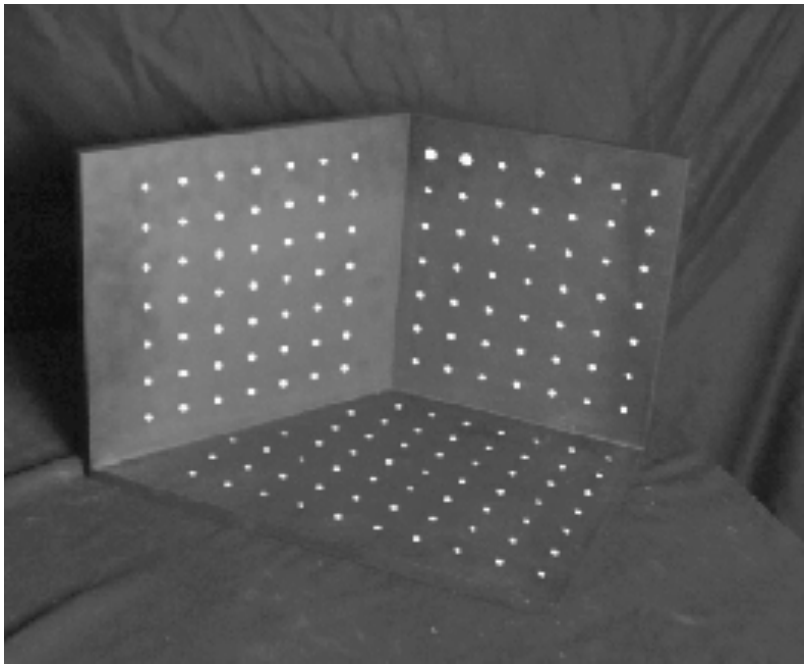
- Goal: estimate the camera parameters
  - Version 1: solve for projection matrix

$$\mathbf{x} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi X}$$

  - Version 2: solve for camera parameters separately
    - intrinsics (focal length, principal point, pixel size)
    - extrinsics (rotation angles, translation)
    - radial distortion

Calibration using a reference object

- Place a known object in the scene
  - identify correspondence between image 2D and scene 3D
  - compute mapping from scene to image

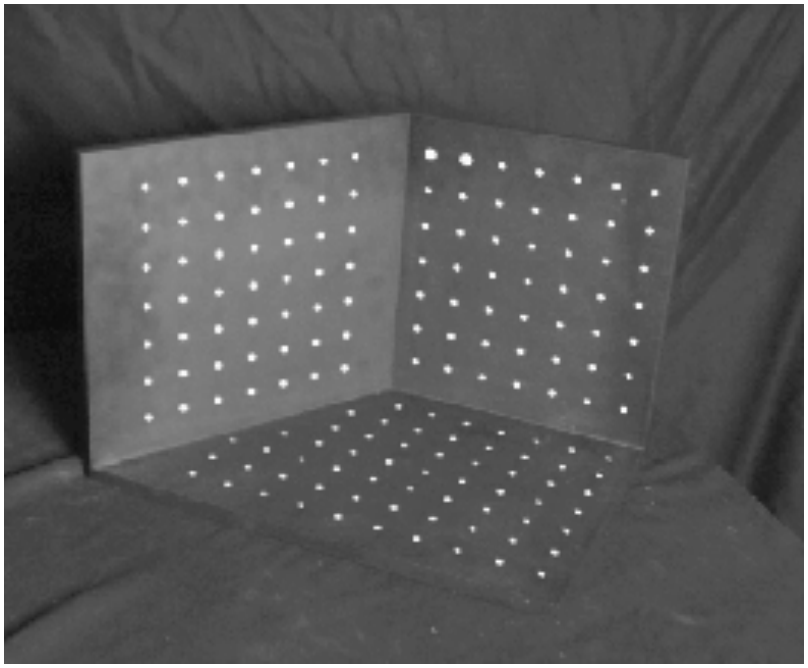


Issues

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

Estimating the projection matrix

- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image



$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Direct linear calibration

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$
$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Direct linear calibration

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Can solve for m_{ij} by linear least squares

- use eigenvector trick that we used for homographies. A x = 0

Defines a least squares problem: minimize $\|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$

- Since **h** is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

Direct linear calibration

- Advantage:
  - Very simple to formulate and solve
- Disadvantages:
  - Doesn't directly tell you the camera parameters
  - Doesn't model radial distortion

Nonlinear *methods* are preferred

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

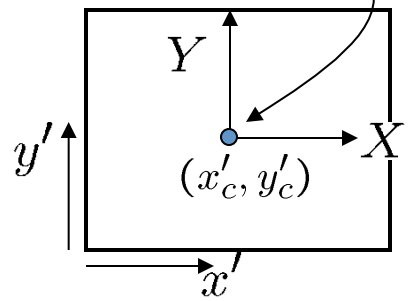
Summary

- Known correspondences
  - (ui, vi) and (Xi, Yi, Zi)
- Compute mij solving system of linear equations
  - May use this to initialize non linear error minimization problem to recover more accurate mij

Camera parameters

A camera is described by several parameters

- Translation T of the optical center from the origin of world coords
- Rotation R of the image plane
- focal length f, principal point (x'_c, y'_c), pixel size (s_x, s_y)
- blue parameters are called "extrinsics," red are "intrinsics"



Projection equation

$$\mathbf{x} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \Pi \mathbf{X}$$

- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

identity matrix

$$\mathbf{\Pi} = \begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}$$

intrinsics projection rotation translation

- The definitions of these parameters are **not** completely standardized
  - especially intrinsics—varies from one book to another

Calibration from vanishing points