
Unit 3

Camera Calibration

Outline

- Geometric camera projection model
- Camera calibration
- Plane projective transformation
- Vanishing points
- Cross-ratio – projective invariant

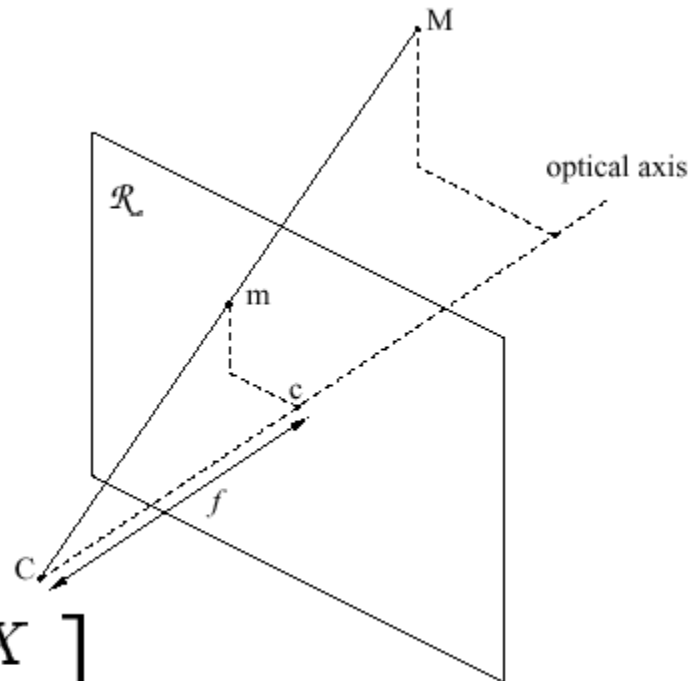
Review Camera Model

- Simple Model: Pinhole
- Image plane at $Z=1$ ($f=1$)
- Projection process:

$$x = \frac{X}{Z} \quad y = \frac{Y}{Z}$$

- Homogeneous Coord, representation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



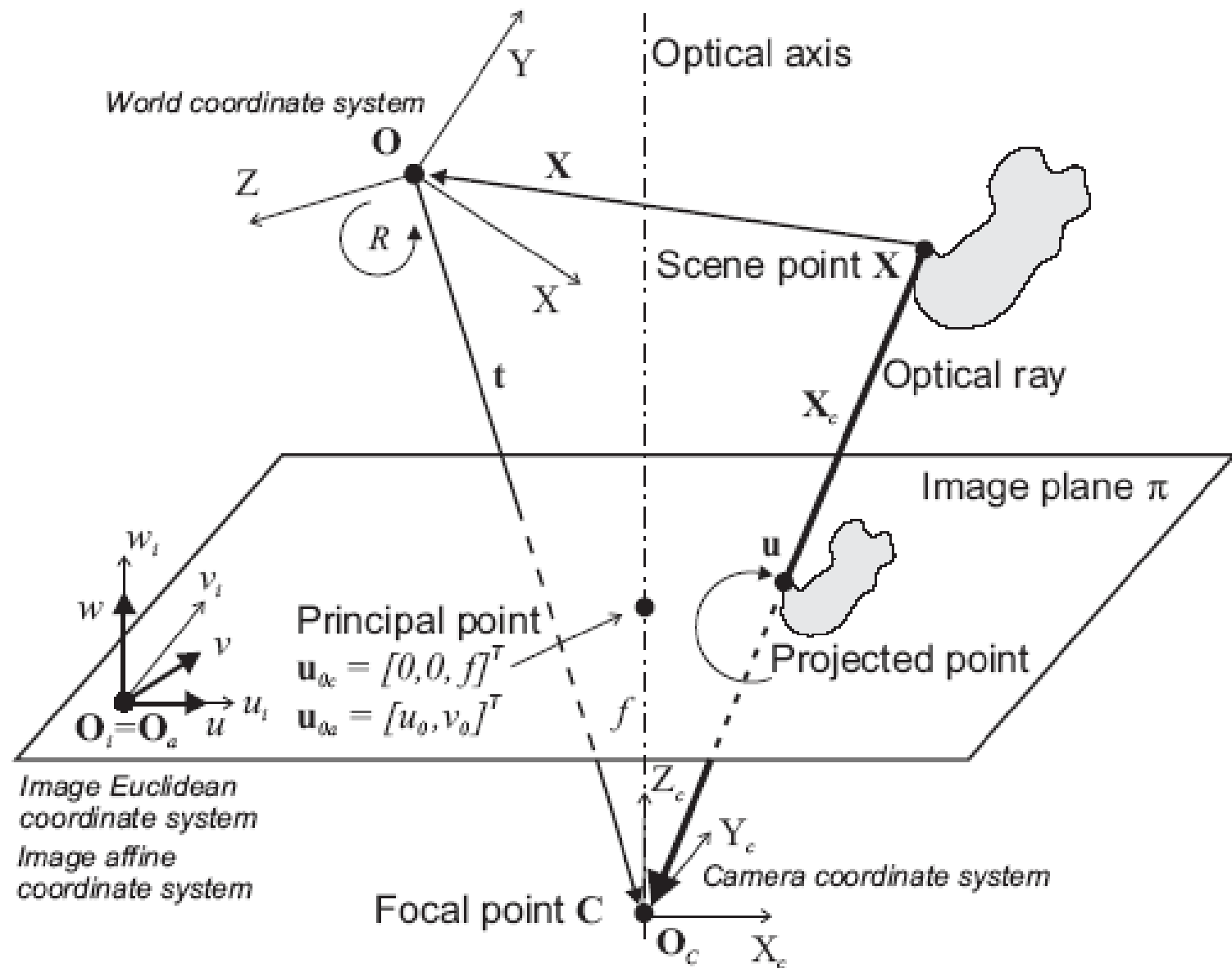
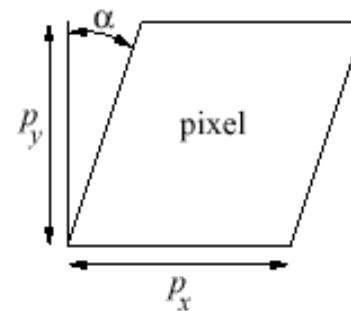
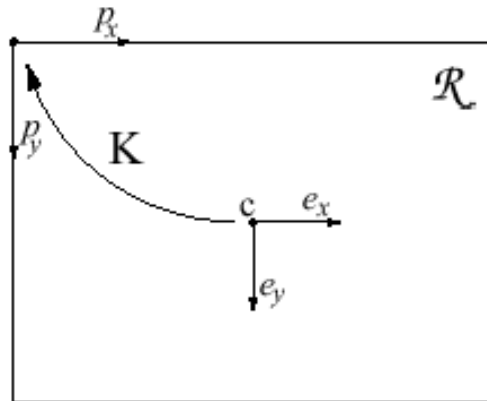


Figure 11.7: The geometry of a linear perspective camera.

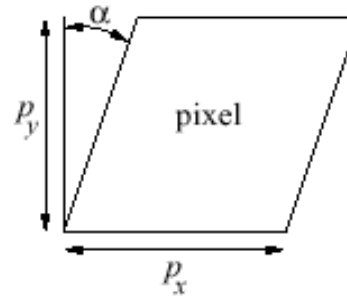
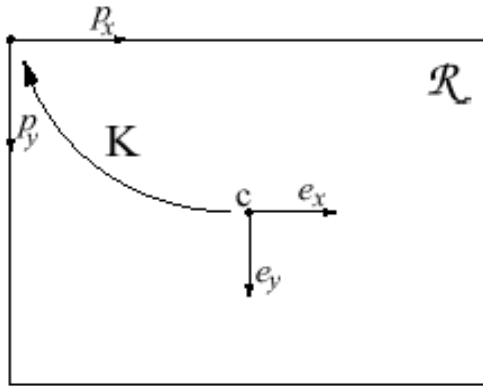
Intrinsic Parameters

- Perspective projection, focal length f .
- Camera frame (pixel coordinates, pixel size and shape).
- Geometric distortion introduced by optics.



From retinal coordinates to image coordinates

Intrinsic Parameters ctd.



From retinal coordinates to image coordinates

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{p_x} & (\tan \alpha) \frac{f}{p_y} & c_x \\ & \frac{f}{p_y} & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} x_{\mathcal{R}} \\ y_{\mathcal{R}} \\ 1 \end{bmatrix}$$

- f : focal length
- p_x, p_y width and height of pixels
- $[c_x, c_y]^T$ principal point (retinal coordinates)
- α : skew angle

Simplified Notation: Calibration Matrix of Camera

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{p_x} & (\tan \alpha) \frac{f}{p_y} & c_x \\ & \frac{f}{p_y} & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} x_{\mathcal{R}} \\ y_{\mathcal{R}} \\ 1 \end{bmatrix}$$



Simplified notation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} x_{\mathcal{R}} \\ y_{\mathcal{R}} \\ 1 \end{bmatrix}$$

$$\mathbf{m} = \mathbf{K} \mathbf{m}_{\mathcal{R}}$$

Combined Projection Matrix

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \underbrace{\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic K}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix}}_{\text{extrinsic}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{m} \sim \mathbf{K} [\mathbf{R} \mid \mathbf{t}] \mathbf{M}$$

$$\mathbf{m} \sim \mathbf{P} \mathbf{M}$$

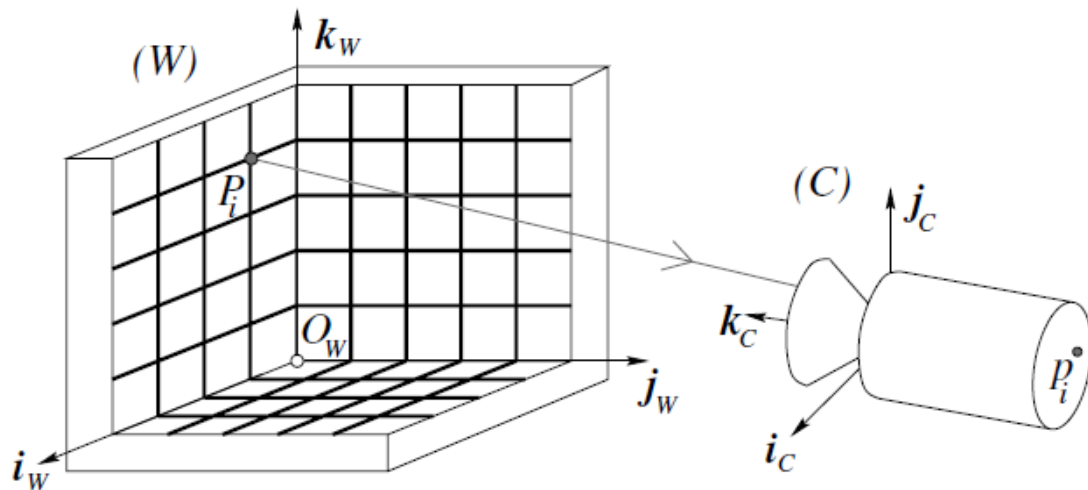
P: 3x4 matrix, camera projection matrix.

Projection Matrix: #Parameters

- Intrinsic: 5 ($f_x, f_y, c_x, c_y, \{s\}$)
- Extrinsic: 6 (R, T)
- **Total: 10-11 DOF**
- Simplification often used for initialization:
 - $(c_x, c_y) \approx$ center of image
 - $s \approx 0$ (rectangular pixels)
- Attention: Intrinsic parameters fixed for fixed optics camera \neq not true for zoom lens!

The Calibration Problem

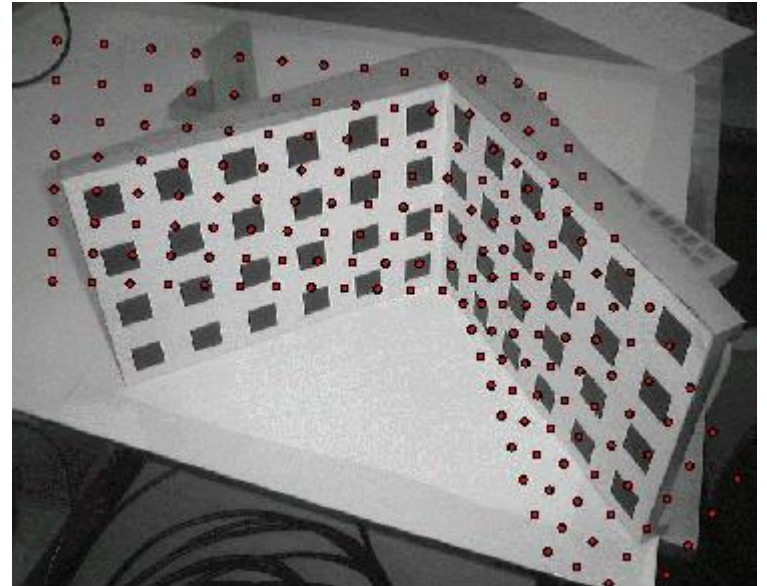
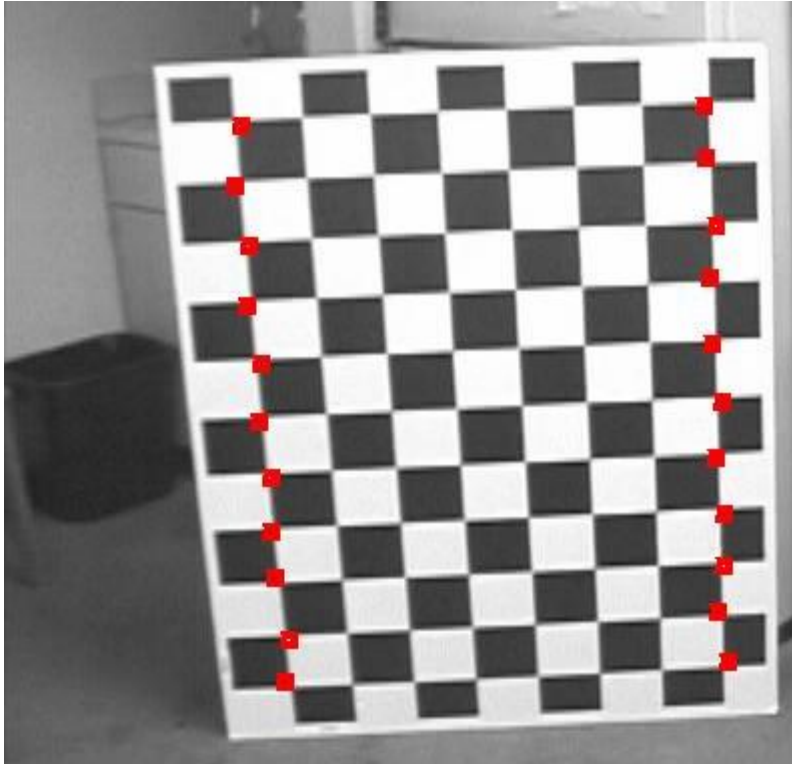
- Given
 - Calibration pattern with N corners
 - M views of this calibration pattern
- Recover the intrinsic and extrinsic parameters
 - Sometimes, we are only interested in calibrating intrinsic or extrinsic parameters.



Camera Calibration

- Issues:
 - what are intrinsic parameters of the camera?
 - what is the camera matrix? (intrinsic+extrinsic)
- General strategy:
 - view calibration object
 - identify image points
 - obtain camera matrix by minimizing error
 - obtain intrinsic parameters from camera matrix
- Error minimization:
 - Linear least squares
 - easy problem numerically
 - solution can be rather bad
 - Minimize image distance
 - more difficult numerical problem
 - solution usually rather good,
 - start with linear least squares

Example Calibration Pattern



Calibration Pattern: Object with features of known size/geometry

Camera Calibration (DLT)

Problem Statement:

Given n correspondences $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$, where \mathbf{X}_i is a scene point and \mathbf{x}_i its image:

Compute

$P = K [R | \mathbf{t}]$ such that $\mathbf{x}_i = P\mathbf{X}_i$.

The algorithm for camera calibration has two parts:

- (i) Compute the matrix P from a set of point correspondences.
- (ii) Decompose P into K , R and \mathbf{t} via the QR decomposition.

Algorithm step 1: Compute the matrix P (DLT)

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

Each correspondence generates two equations

$$x_i = \frac{p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} \quad y_i = \frac{p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}}$$

Multiplying out gives equations **linear** in the matrix elements of P

$$\begin{aligned} x_i(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}) &= p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14} \\ y_i(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}) &= p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24} \end{aligned}$$

These equations can be rearranged as

$$\begin{bmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -x \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -yX & -yY & -yZ & -y \end{bmatrix} \mathbf{p} = \mathbf{0}$$

with $\mathbf{p} = (p_{11}, p_{12}, p_{13}, p_{14}, p_{21}, p_{22}, p_{23}, p_{24}, p_{31}, p_{32}, p_{33}, p_{34})^\top$ a 12-vector.

Algorithm step 1 continued

Solving for P

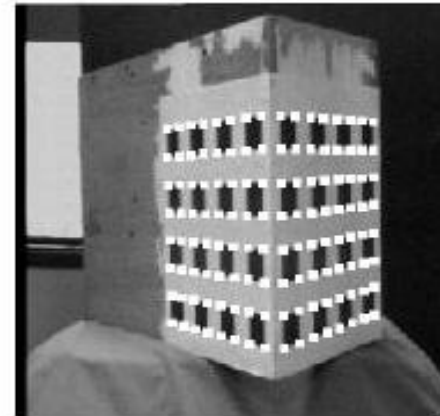
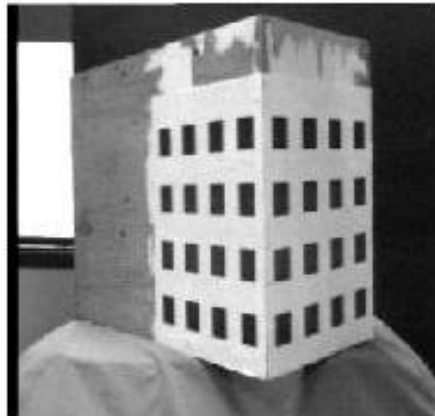
(i) Concatenate the equations from ($n \geq 6$) correspondences to generate $2n$ simultaneous equations, which can be written: $A\mathbf{p} = 0$, where A is a $2n \times 12$ matrix.

(ii) In general this will not have an exact solution, but a (linear) solution which minimises $\|A\mathbf{p}\|$, subject to $\|\mathbf{p}\| = 1$ is obtained from the eigenvector with least eigenvalue of $A^T A$. Or equivalently from the vector corresponding to the smallest singular value of the SVD of A .

(iii) This linear solution is then used as the starting point for a non-linear minimisation of the difference between the measured and projected point:

$$\min_{\mathbf{P}} \sum_i ((x_i, y_i) - P(x_i, y_i, Z_i, 1))^2$$

Example – Calibration Object



Determine accurate corner positions by

- (i) Extract and link edges using Canny edge operator.
- (ii) Fit lines to edges using orthogonal regression.
- (iii) Intersect lines to obtain corners to sub-pixel accuracy.

The final error between measured and projected points is typically less than 0.02 pixels.

Algorithm step 2: Decompose P into K,R and t

The first 3×3 submatrix, M, of P is the product ($M = KR$) of an upper triangular and rotation matrix.

(i) Factor M into KR using the QR matrix decomposition. This determines K and R.

(ii) Then

$$\mathbf{t} = K^{-1}(p_{14}, p_{24}, p_{34})^T$$

Note, this produces a matrix with an extra **skew** parameter s

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

with $s = \tan \theta$, and θ the angle between the image axes.

Another Solution

- Camera projection matrix $P = [A \ b]$, R is formed by r_1, r_2 & r_3

$$\rho(A \ b) = \mathcal{K}(\mathcal{R} \ t) \iff \rho \begin{pmatrix} a_1^T \\ a_2^T \\ a_3^T \end{pmatrix} = \begin{pmatrix} \alpha r_1^T - \alpha \cot \theta r_2^T + x_0 r_3^T \\ \frac{\beta}{\sin \theta} r_2^T + y_0 r_3^T \\ r_3^T \end{pmatrix}$$

$$\mathcal{K} = \begin{pmatrix} \mathcal{K}_2 & p_0 \\ 0^T & 1 \end{pmatrix}, \quad \text{where } \mathcal{K}_2 \stackrel{\text{def}}{=} \begin{pmatrix} \alpha & -\alpha \cot \theta \\ 0 & \frac{\beta}{\sin \theta} \end{pmatrix} \quad \text{and } p_0 \stackrel{\text{def}}{=} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

- Using the fact that the rows of a rotation matrix have unit length and are perpendicular to each other yields

$$\begin{cases} \rho = \varepsilon / \|a_3\|, & \text{where } \varepsilon = \mp 1. \\ r_3 = \rho a_3, \\ x_0 = \rho^2 (a_1 \cdot a_3), \\ y_0 = \rho^2 (a_2 \cdot a_3), \end{cases}$$

$$\left\{ \begin{array}{l} \rho^2(\mathbf{a}_1 \times \mathbf{a}_3) = -\alpha \mathbf{r}_2 - \alpha \cot \theta \mathbf{r}_1, \\ \rho^2(\mathbf{a}_2 \times \mathbf{a}_3) = \frac{\beta}{\sin \theta} \mathbf{r}_1, \end{array} \right. \quad \text{and} \quad \left\{ \begin{array}{l} \rho^2 \|\mathbf{a}_1 \times \mathbf{a}_3\| = \frac{|\alpha|}{\sin \theta}, \\ \rho^2 \|\mathbf{a}_2 \times \mathbf{a}_3\| = \frac{|\beta|}{\sin \theta}, \end{array} \right.$$

thus:

$$\left\{ \begin{array}{l} \cos \theta = -\frac{(\mathbf{a}_1 \times \mathbf{a}_3) \cdot (\mathbf{a}_2 \times \mathbf{a}_3)}{\|\mathbf{a}_1 \times \mathbf{a}_3\| \|\mathbf{a}_2 \times \mathbf{a}_3\|}, \\ \alpha = \rho^2 \|\mathbf{a}_1 \times \mathbf{a}_3\| \sin \theta, \\ \beta = \rho^2 \|\mathbf{a}_2 \times \mathbf{a}_3\| \sin \theta, \end{array} \right.$$

Finally, we have

$$\left\{ \begin{array}{l} \mathbf{r}_1 = \frac{\rho^2 \sin \theta}{\beta} (\mathbf{a}_2 \times \mathbf{a}_3) = \frac{1}{\|\mathbf{a}_2 \times \mathbf{a}_3\|} (\mathbf{a}_2 \times \mathbf{a}_3) \\ \mathbf{r}_2 = \mathbf{r}_3 \times \mathbf{r}_1. \end{array} \right.$$

Pose Estimation

- Estimating an object's 3D pose from a set of 2D point projections
- This pose estimation problem is also known as extrinsic calibration
- Solving P from the following equations with DLT

$$x_i = \frac{p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$
$$y_i = \frac{p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}},$$

- Estimate R & t from P (given K)

Iterative non-linear algorithms

- Projection equation

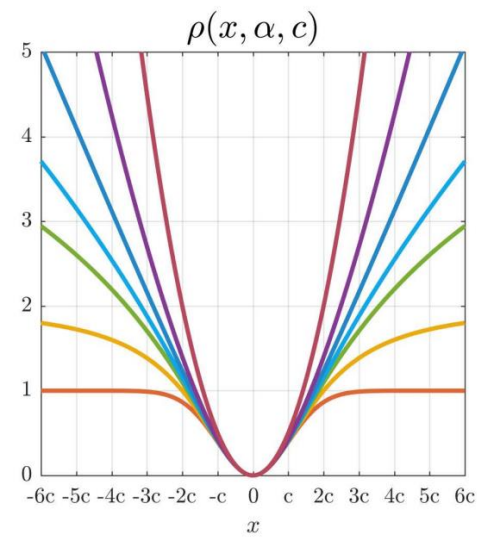
$$\mathbf{x}_i = \mathbf{f}(\mathbf{p}_i; \mathbf{R}, \mathbf{t}, \mathbf{K})$$

- Iteratively minimize the robustified linearized reprojection errors

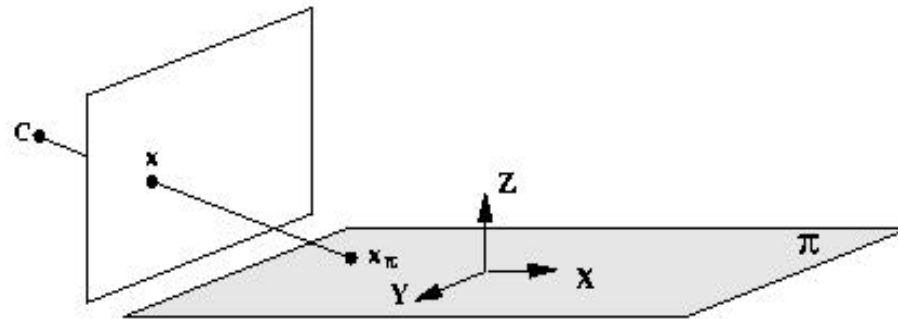
$$E_{\text{NLP}} = \sum_i \rho \left(\frac{\partial \mathbf{f}}{\partial \mathbf{R}} \Delta \mathbf{R} + \frac{\partial \mathbf{f}}{\partial \mathbf{t}} \Delta \mathbf{t} + \frac{\partial \mathbf{f}}{\partial \mathbf{K}} \Delta \mathbf{K} - \mathbf{r}_i \right),$$

where $\mathbf{r}_i = \tilde{\mathbf{x}}_i - \hat{\mathbf{x}}_i$ is the current residual vector (2D error in predicted position), ρ is the robust loss function.

$$f(x, \alpha, c) = \frac{|\alpha - 2|}{\alpha} \left(\left(\frac{(x/c)^2}{|\alpha - 2|} + 1 \right)^{\alpha/2} - 1 \right)$$



Plane Projective Transformations

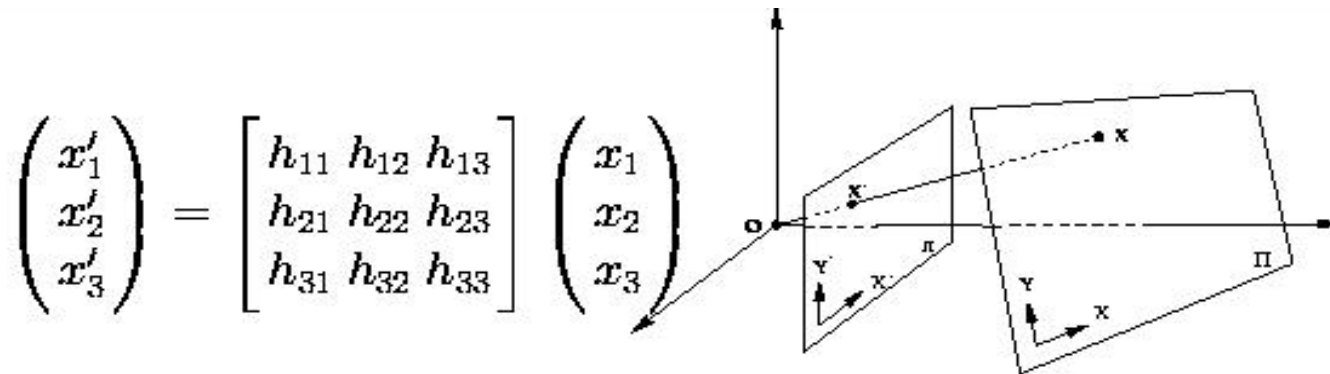


Choose the world coordinate system such that the plane of the points has zero Z coordinate. Then the 3×4 matrix P reduces to

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

which is a 3×3 matrix representing a general plane to plane projective transformation.

Projective transformations continued



or $\mathbf{x}' = \mathbf{H}\mathbf{x}$, where \mathbf{H} is a 3×3 non-singular homogeneous matrix.

- This is the most general transformation between the world and image plane under imaging by a perspective camera.
- It is often only the 3×3 **form** of the matrix that is important in establishing properties of this transformation.
- A projective transformation is also called a “homography” and a “collineation”.
- \mathbf{H} has 8 degrees of freedom.

Four points define a projective transformation

Given n point correspondences $(x, y) \leftrightarrow (x', y')$

Compute H such that $\mathbf{x}'_i = H\mathbf{x}_i$

- Each point correspondence gives two constraints

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \quad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

and multiplying out generates two **linear** equations for the elements of H

$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13}$$

$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23}$$

- If $n \geq 4$ (no three points collinear), then H is determined uniquely.
- The converse of this is that it is possible to transform any four points in general position to any other four points in general position by a projectivity.

Example 1: Removing Perspective Distortion

Given: the coordinates of four points on the scene plane

Find: a projective rectification of the plane



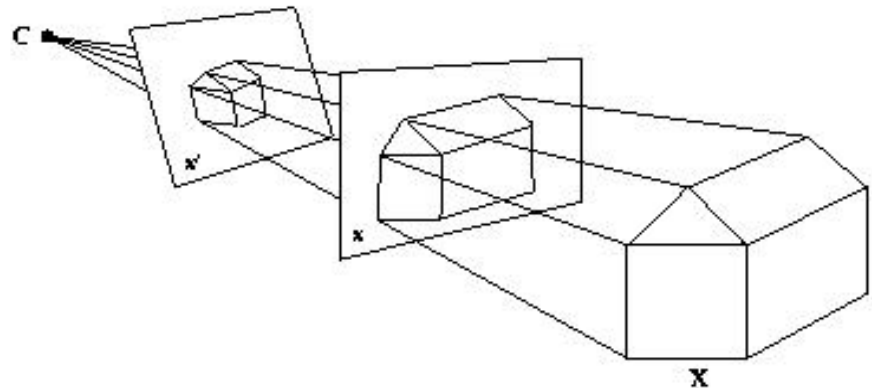
- This **rectification** does not require knowledge of **any** of the camera's parameters or the pose of the plane.
- It is not always necessary to know coordinates for four points.

The Cone of Rays

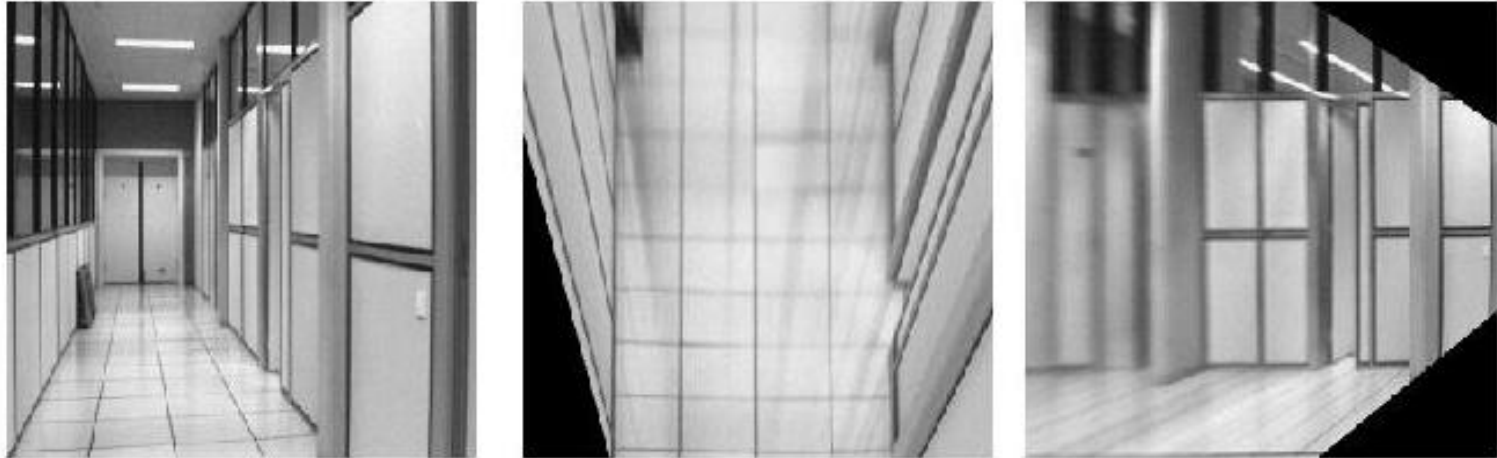
An image is the intersection of a plane with the cone of rays between points in 3-space and the optical centre. Any two such “images” (with the same camera centre) are related by a planar projective transformation.

$$\mathbf{x}' = \mathbf{H}\mathbf{x}$$

e.g. rotation about the camera centre

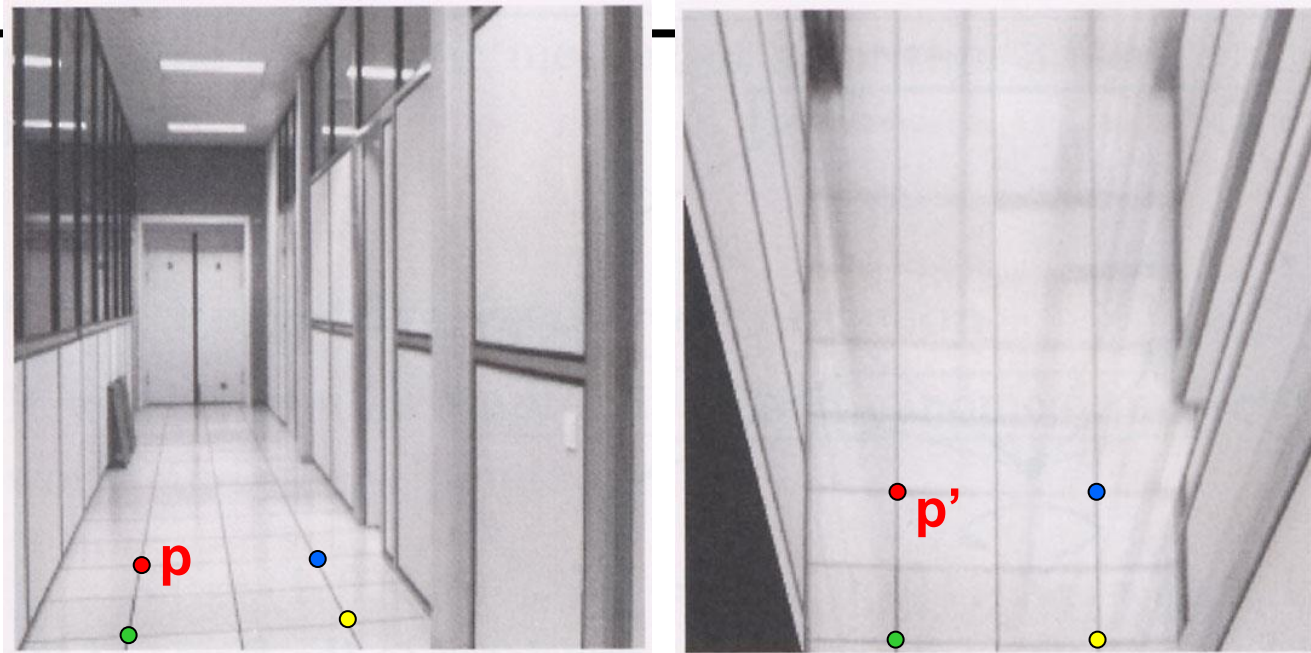


Example 2: Synthetic Rotations



The synthetic images are produced by projectively warping the original image so that four corners of an imaged rectangle map to the corners of a rectangle. Both warpings correspond to a synthetic rotation of the camera about the (fixed) camera centre.

Image rectification



To unwarp (rectify) an image

- solve for homography \mathbf{H} given \mathbf{p} and \mathbf{p}'
- solve equations of the form: $w\mathbf{p}' = \mathbf{H}\mathbf{p}$
 - linear in unknowns: w and coefficients of \mathbf{H}
 - \mathbf{H} is defined up to an arbitrary scale factor
 - how many points are necessary to solve for \mathbf{H} ?

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$x'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$

$$y'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies

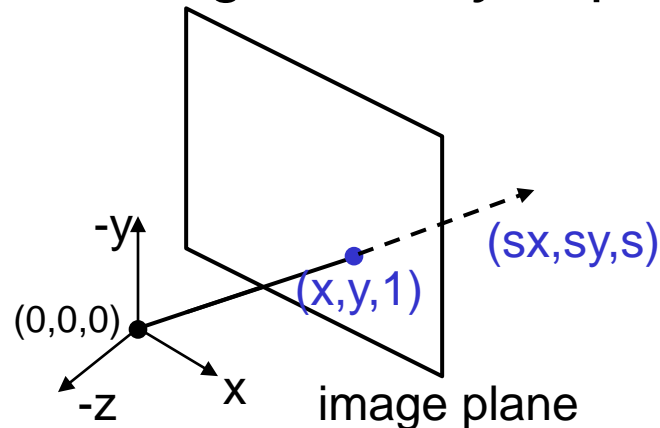
$$\begin{array}{c}
 \begin{bmatrix}
 x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\
 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\
 & & & & & \vdots & & & \\
 x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\
 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n
 \end{bmatrix}
 \begin{bmatrix}
 h_{00} \\
 h_{01} \\
 h_{02} \\
 h_{10} \\
 h_{11} \\
 h_{12} \\
 h_{20} \\
 h_{21} \\
 h_{22}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 0 \\
 \vdots \\
 0 \\
 0
 \end{bmatrix}
 \\
 \mathbf{A} \qquad \qquad \mathbf{h} \qquad \qquad \mathbf{0} \\
 2n \times 9 \qquad \qquad 9 \qquad \qquad 2n
 \end{array}$$

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

- Since \mathbf{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

The projective plane

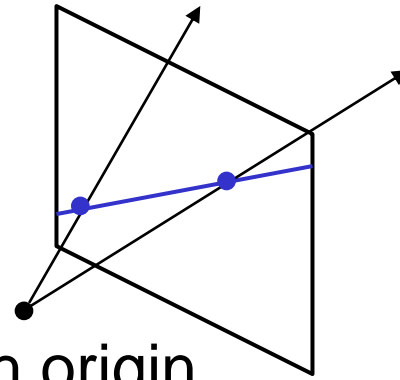
- Why do we need homogeneous coordinates?
 - represent points at infinity, homographies, perspective projection, multi-view relationships
- What is the geometric intuition?
 - a point in the image is a *ray* in projective space



- Each *point* (x,y) on the plane is represented by a *ray* (sx,sy,s)
 - all points on the ray are equivalent: $(x, y, 1) \cong (sx, sy, s)$

Projective lines

- What does a line in the image correspond to in projective space?



- A line is a *plane* of rays through origin
 - all rays (x,y,z) satisfying: $ax + by + cz = 0$

in vector notation :

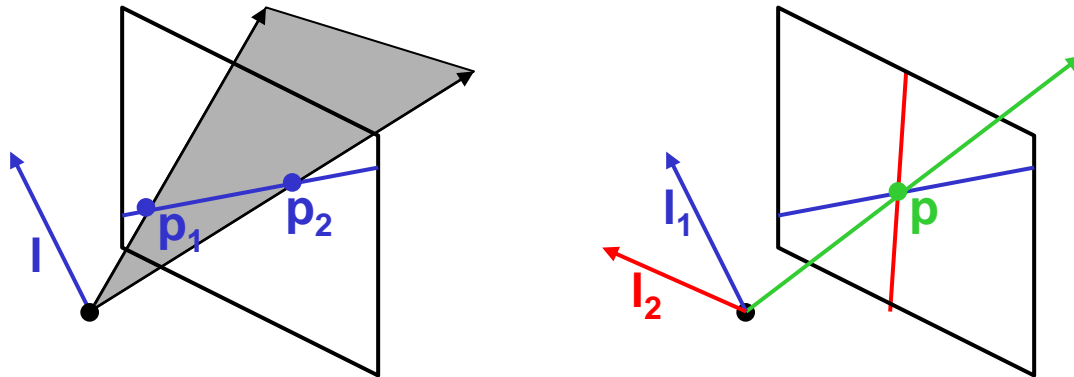
$$0 = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

l p

- A line is also represented as a homogeneous 3-vector **l**

Point and line duality

- A line \mathbf{l} is a homogeneous 3-vector
- It is \perp to every point (ray) \mathbf{p} on the line: $\mathbf{l} \cdot \mathbf{p} = 0$



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

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

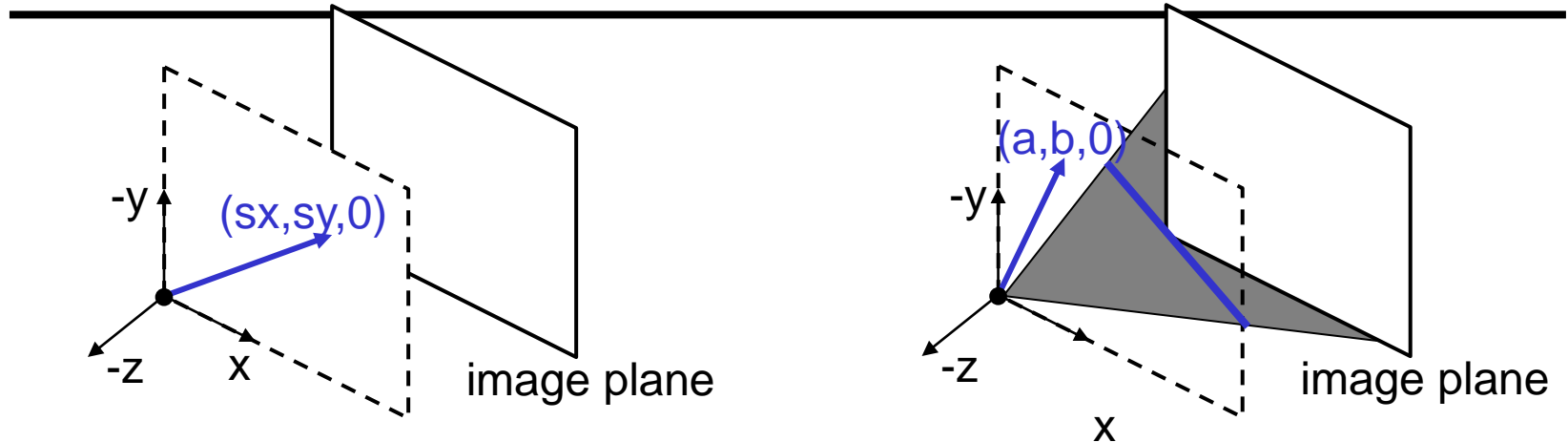
What is the intersection of two lines \mathbf{l}_1 and \mathbf{l}_2 ?

- \mathbf{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

- given any formula, can switch the meanings of points and lines to get another formula

Ideal points and lines



- Ideal point (“point at infinity”)
 - $p \cong (x, y, 0)$ – parallel to image plane
 - It has infinite image coordinates

Ideal line

- $l \cong (a, b, 0)$ – parallel to image plane
- Corresponds to a line in the image (finite coordinates)

Homographies of points and lines

- Computed by 3x3 matrix multiplication
 - To transform a point: $\mathbf{p}' = \mathbf{H}\mathbf{p}$
 - To transform a line: $\mathbf{l}\mathbf{p}=0 \rightarrow \mathbf{l}'\mathbf{p}'=0$
 - $0 = \mathbf{l}\mathbf{p} = \mathbf{l}\mathbf{H}^{-1}\mathbf{H}\mathbf{p} = \mathbf{l}\mathbf{H}^{-1}\mathbf{p}' \Rightarrow \mathbf{l}' = \mathbf{l}\mathbf{H}^{-1}$
 - lines are transformed by post-multiplication of \mathbf{H}^{-1}

3D projective geometry

- These concepts generalize naturally to 3D
 - Homogeneous coordinates
 - Projective 3D points have four coords: $\mathbf{P} = (X, Y, Z, W)$
 - Duality
 - A plane \mathbf{N} is also represented by a 4-vector
 - Points and planes are dual in 3D: $\mathbf{N} \mathbf{P} = 0$
 - Projective transformations
 - Represented by 4x4 matrices \mathbf{T} : $\mathbf{P}' = \mathbf{T} \mathbf{P}$, $\mathbf{N}' = \mathbf{N} \mathbf{T}^{-1}$

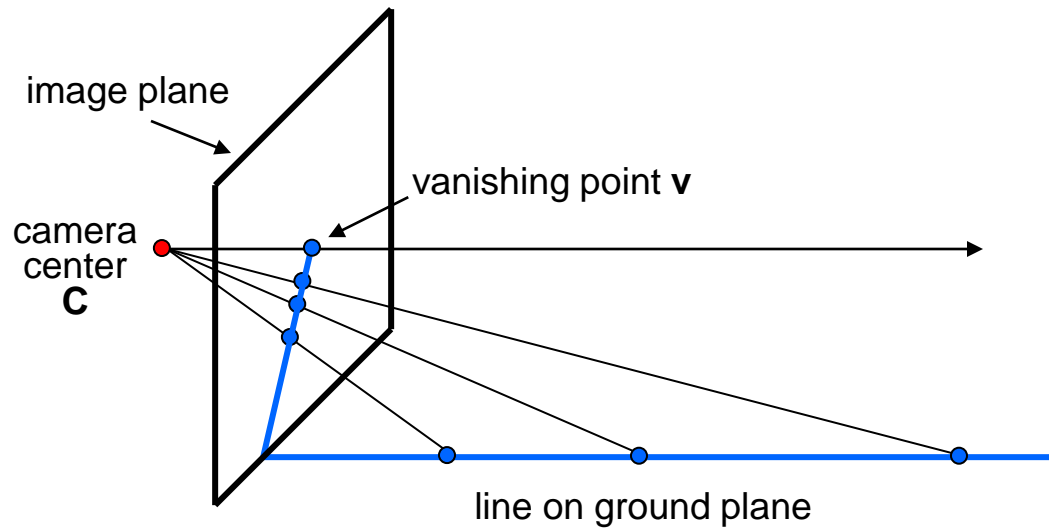
3D to 2D: “perspective” projection

- Matrix Projection:
$$\mathbf{p} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi P}$$

What is *not* preserved under perspective projection?

What is preserved?

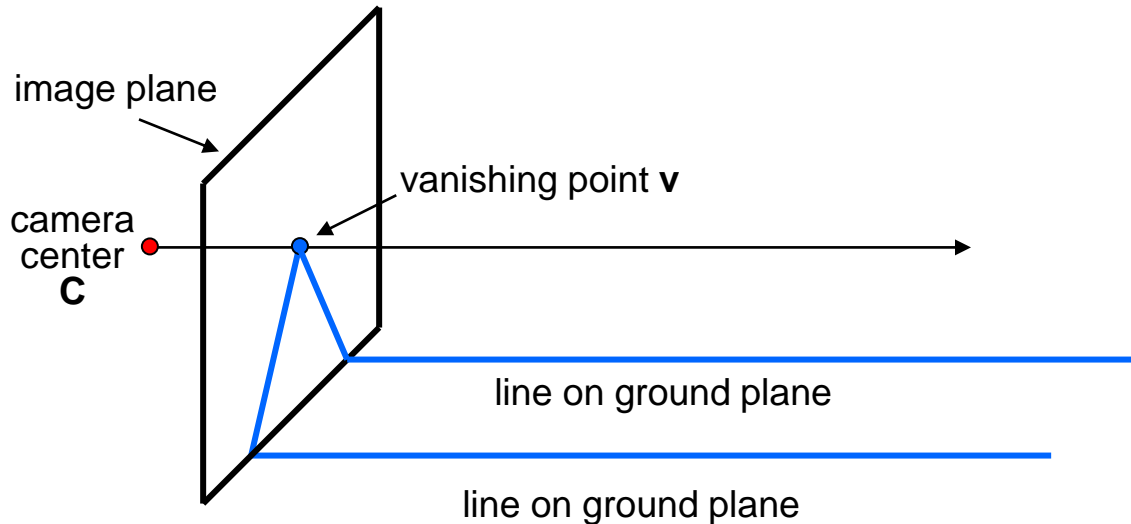
Vanishing points



Vanishing point

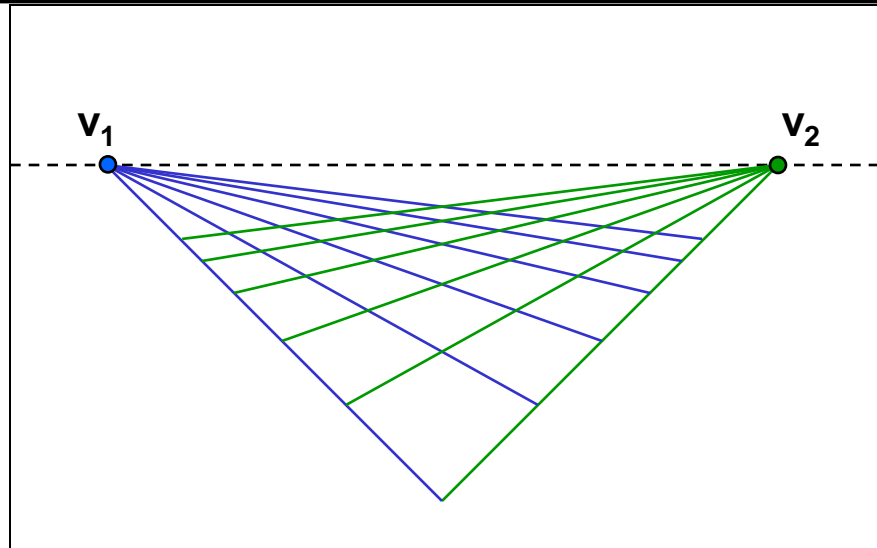
- projection of a point at infinity

Vanishing points



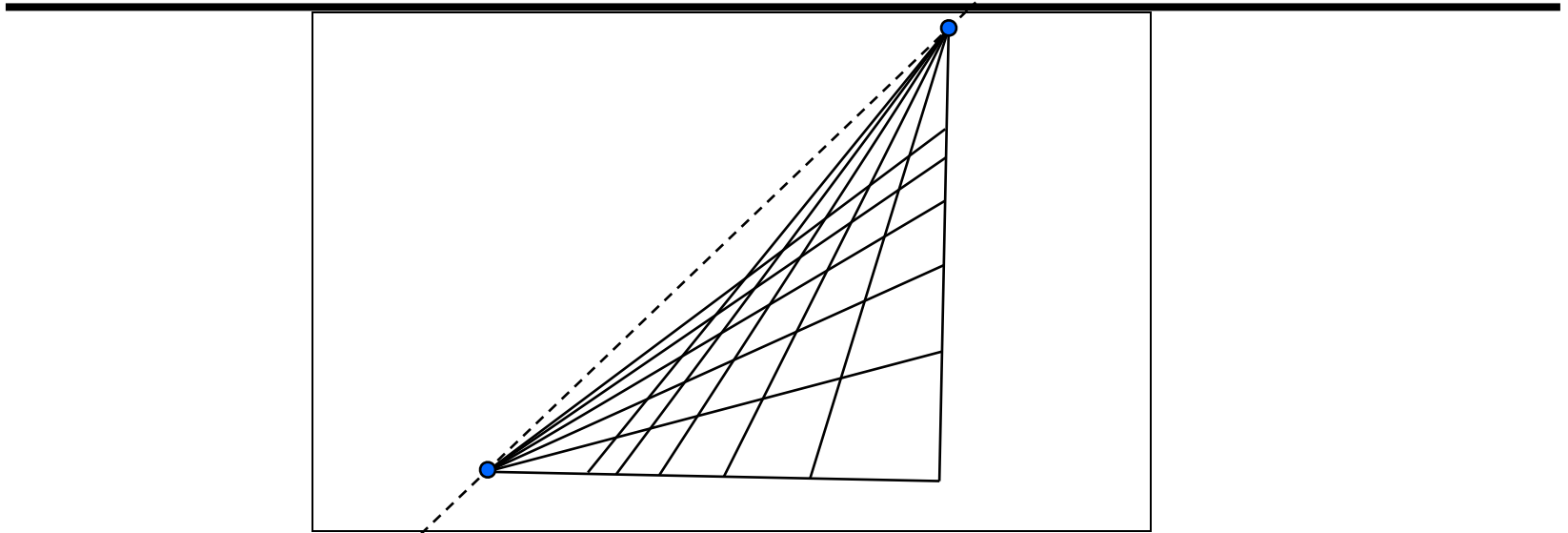
- Properties
 - Any two parallel lines have the same vanishing point \mathbf{v}
 - The ray from \mathbf{C} through \mathbf{v} is parallel to the lines
 - An image may have more than one vanishing point
 - in fact every pixel is a potential vanishing point

Vanishing lines



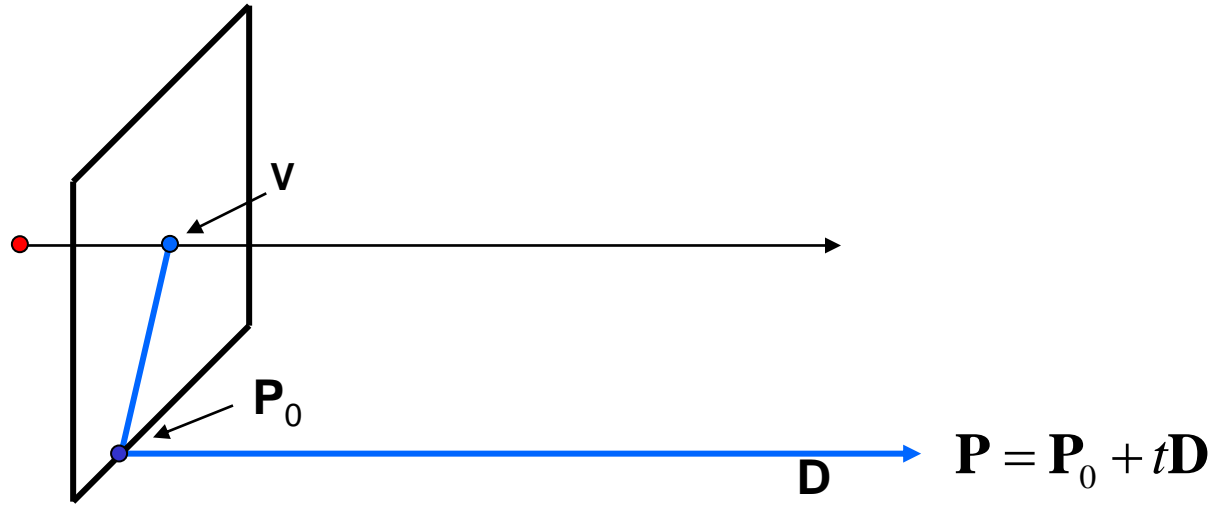
- Multiple Vanishing Points
 - Any set of parallel lines on the plane define a vanishing point
 - The union of all of vanishing points from lines on the same plane is the *vanishing line*
 - For the ground plane, this is called the *horizon*

Vanishing lines



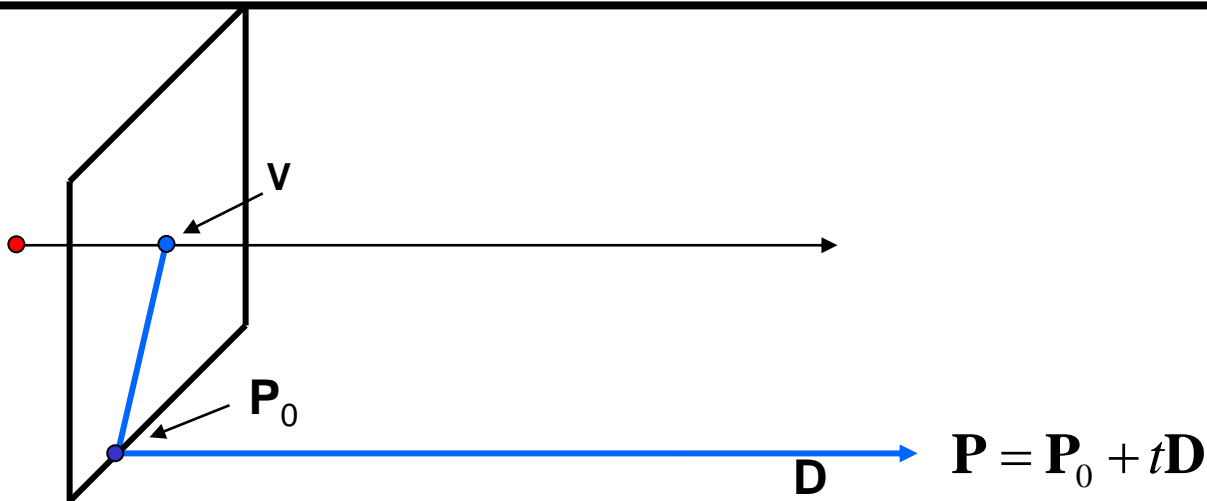
- Multiple Vanishing Points
 - Different planes define different vanishing lines

Computing vanishing points



$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix}$$

Computing vanishing points

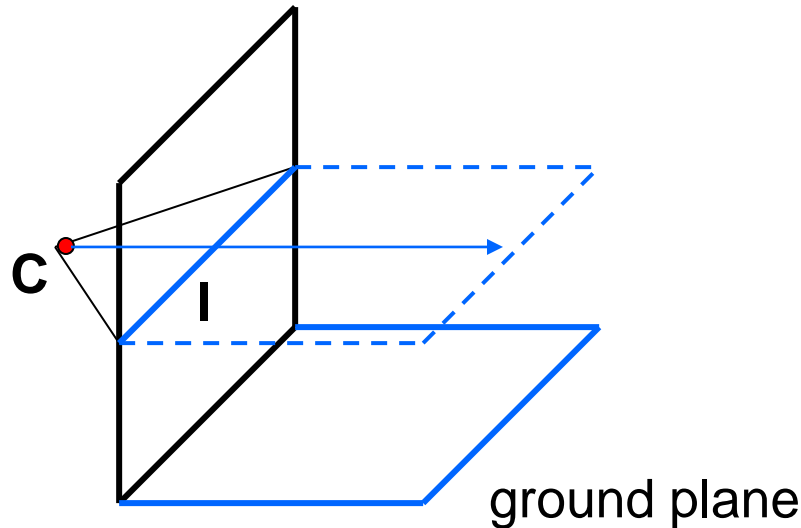


$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_X / t + D_X \\ P_Y / t + D_Y \\ P_Z / t + D_Z \\ 1/t \end{bmatrix} \quad t \rightarrow \infty \quad \mathbf{P}_\infty \cong \begin{bmatrix} D_X \\ D_Y \\ D_Z \\ 0 \end{bmatrix} \quad \mathbf{v} = \mathbf{\Pi} \mathbf{P}_\infty$$

• Properties

- \mathbf{P}_∞ is a point at *infinity*, \mathbf{v} is its projection
- They depend only on line *direction*
- Parallel lines $\mathbf{P}_0 + t\mathbf{D}$, $\mathbf{P}_1 + t\mathbf{D}$ intersect at \mathbf{P}_∞

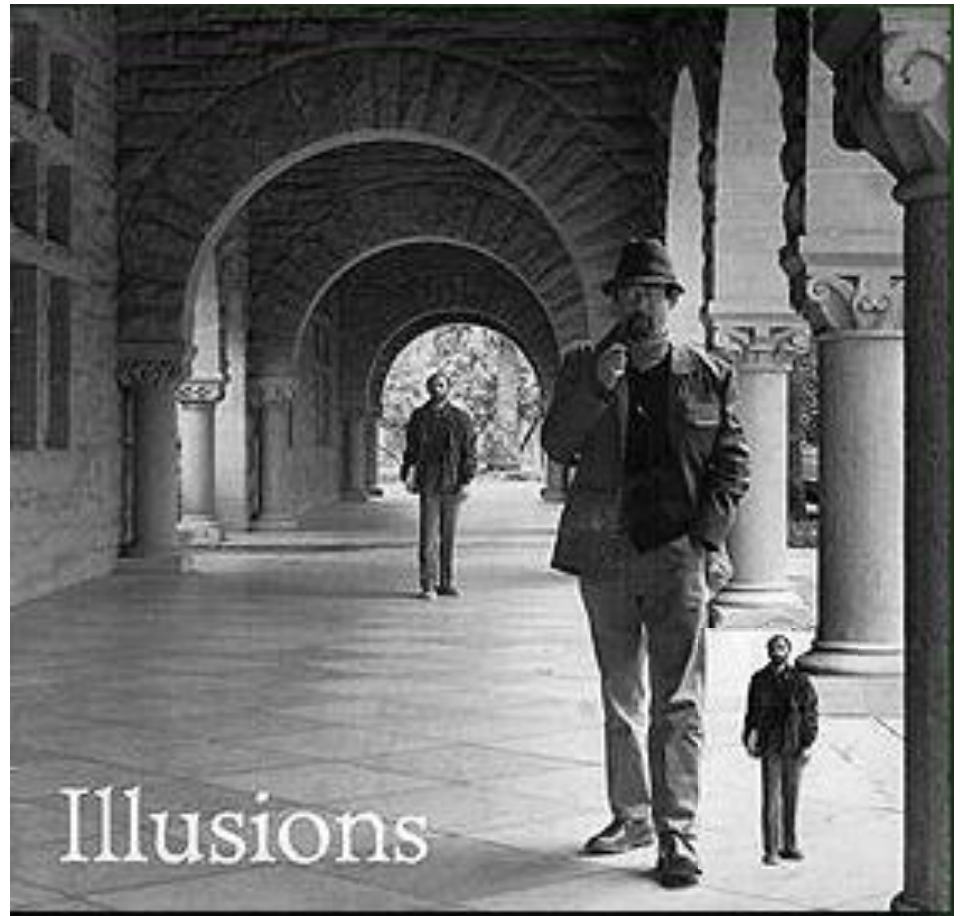
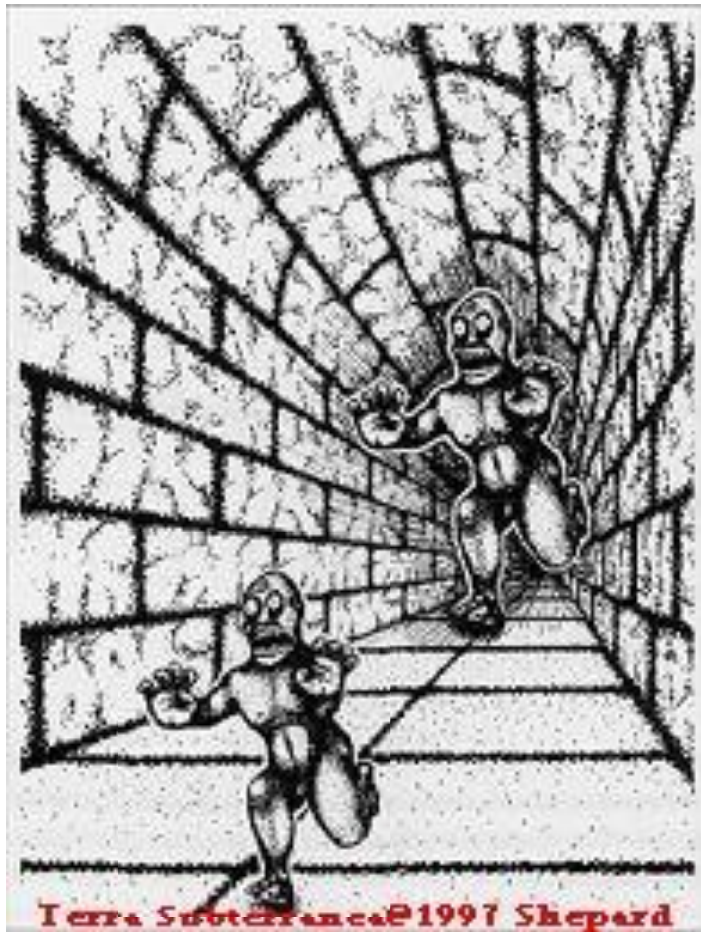
Computing the horizon



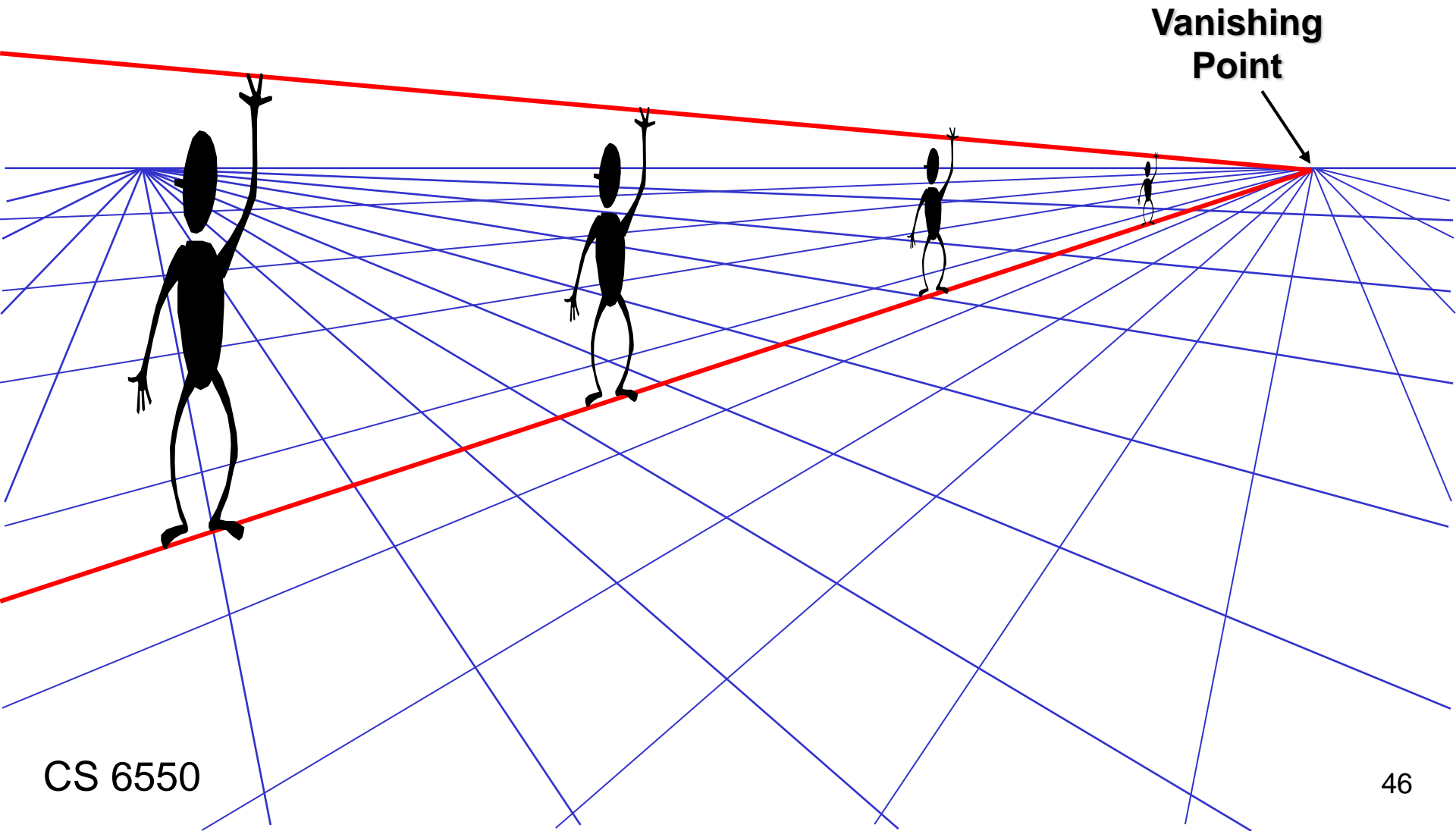
- **Properties**

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

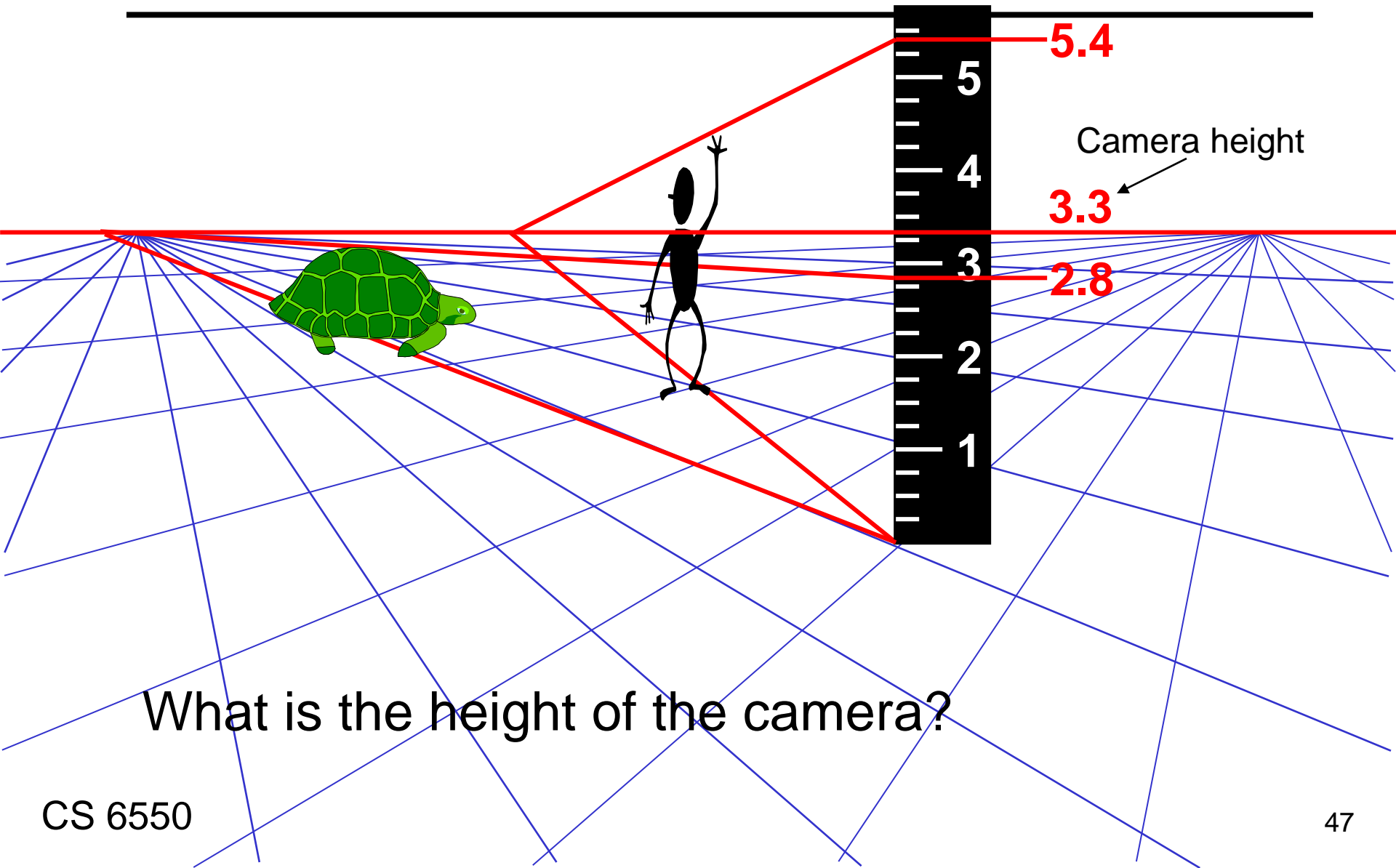
Fun with vanishing points



Comparing heights

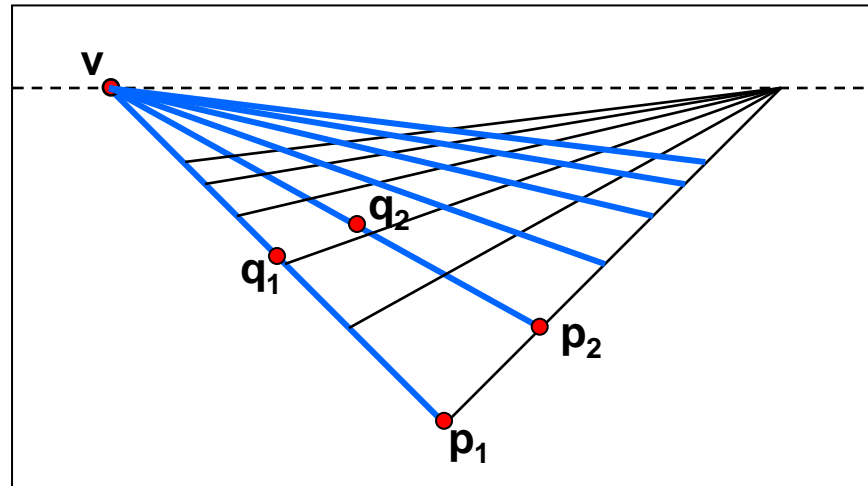


Measuring height



What is the height of the camera?

Computing vanishing points (from lines)



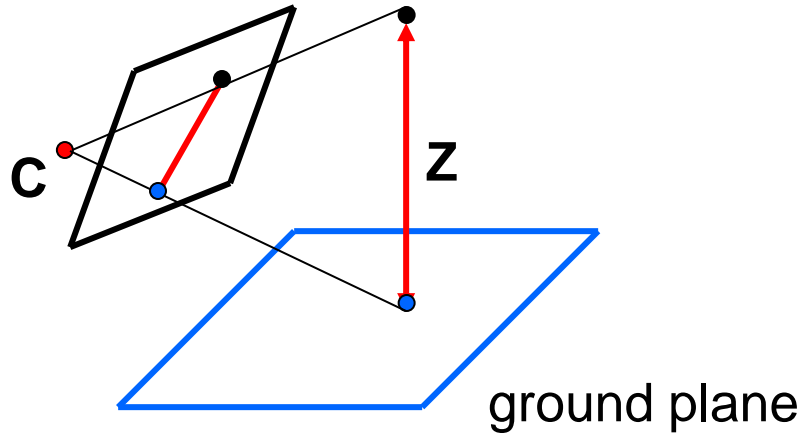
- Intersect p_1q_1 with p_2q_2

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

Least squares version

- Better to use more than two lines and compute the “closest” point of intersection

Measuring height from image



Compute Z from image measurements

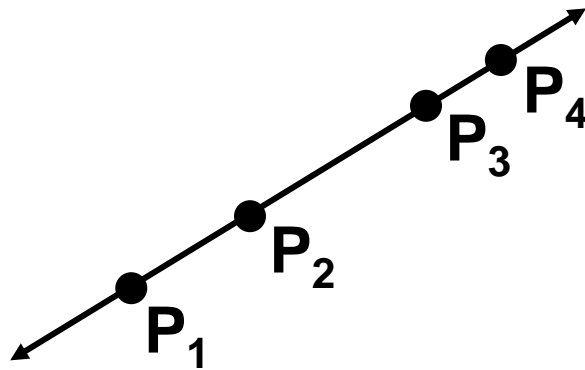
- Need more than vanishing points to do this

Cross ratio

- A Projective Invariant

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

The cross-ratio of 4 collinear points



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

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

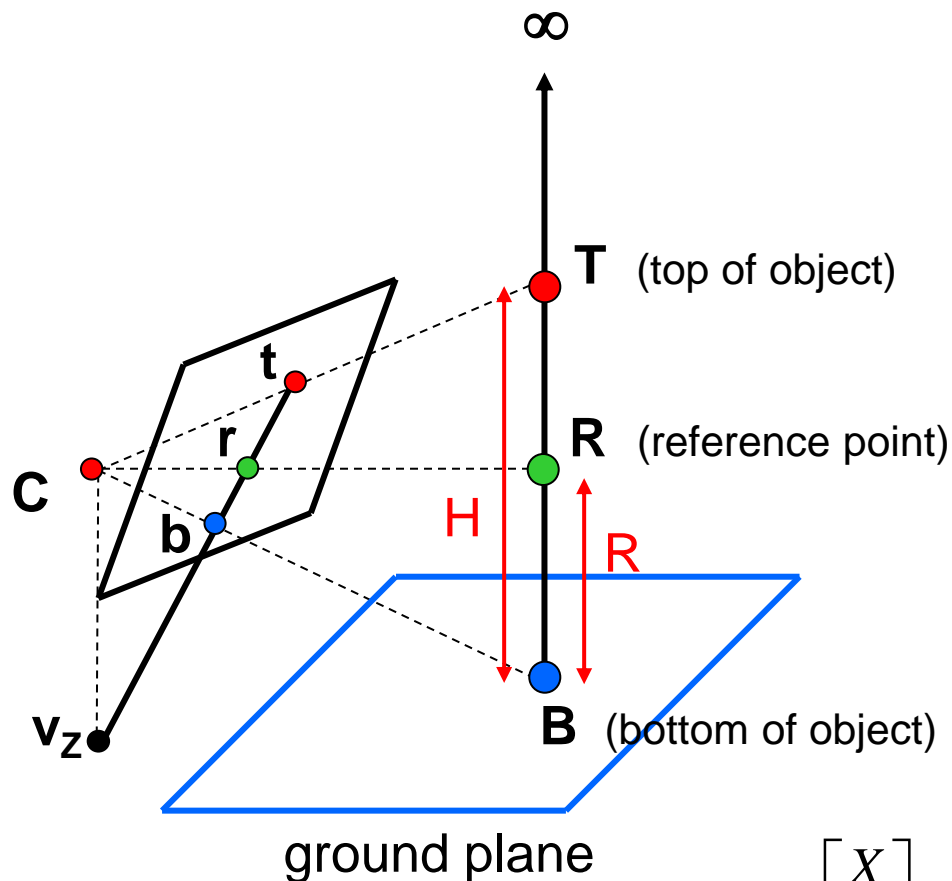
$$\frac{\|\mathbf{P}_1 - \mathbf{P}_3\| \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_1 - \mathbf{P}_2\| \|\mathbf{P}_4 - \mathbf{P}_3\|}$$

Can permute the point ordering

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

This is the fundamental invariant of projective geometry

Measuring height



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

scene cross ratio

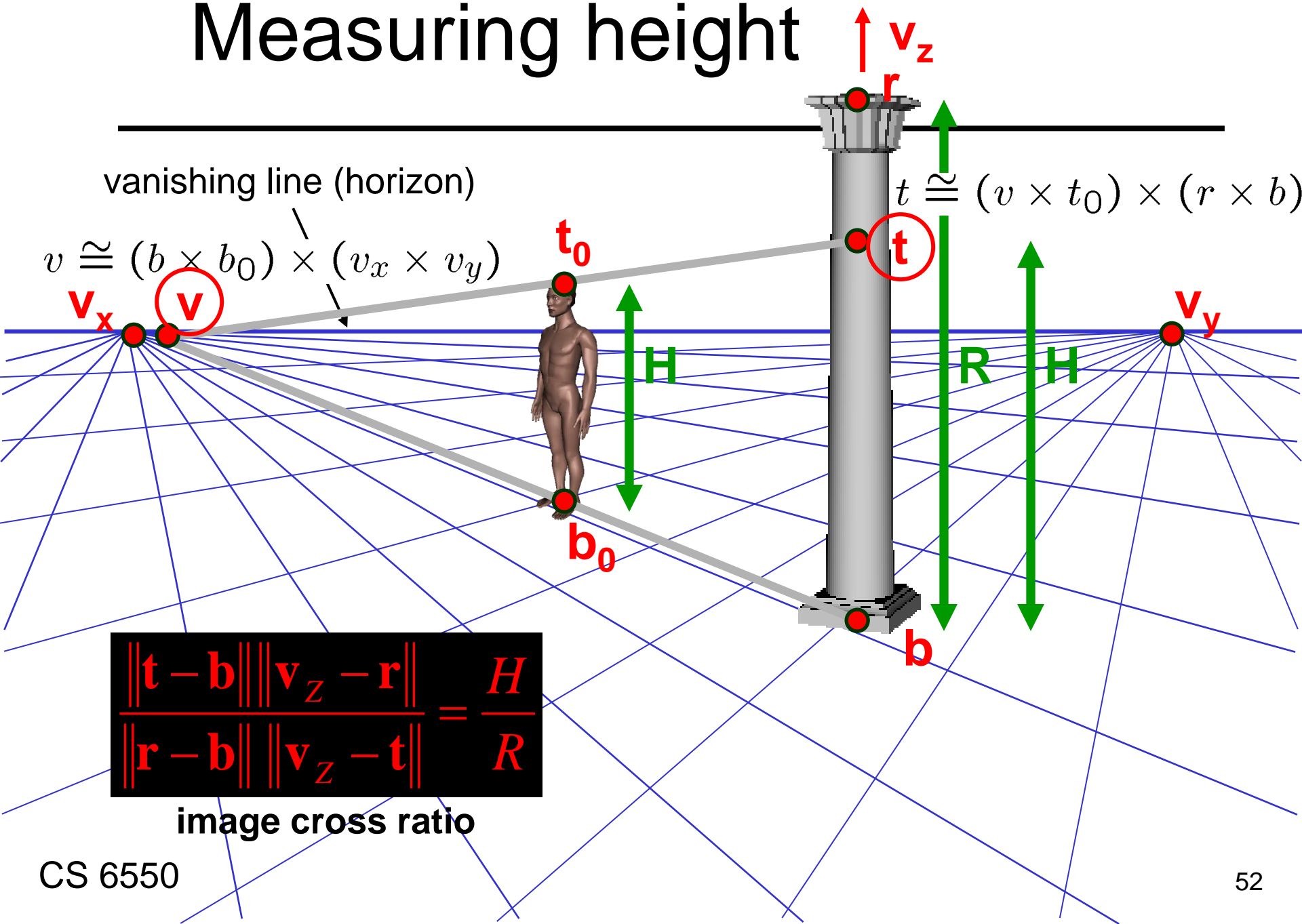
$$\frac{\|t - b\| \|v_z - r\|}{\|r - b\| \|v_z - t\|} = \frac{H}{R}$$

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



Summary

- Camera calibration
 - DLT
- Homography transformation
- Homogeneous coordinates
 - Vanishing points, vanishing lines
- Cross ratio – projective invariant