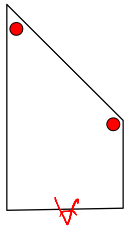
Announcements

- Project 1 Due NOW
- Project 2 out today
  - Help session at end of class

Projective geometry



Ames Room

Readings

- Mundy, J.L. and Zisserman, A., Geometric Invariance in Computer Vision, Appendix: Projective Geometry for Machine Vision, MIT Press, Cambridge, MA, 1992, **(read 23.1 - 23.5, 23.10)**
  - available online: http://www.cs.cmu.edu/~ph/869/papers/zisser-mundy.pdf

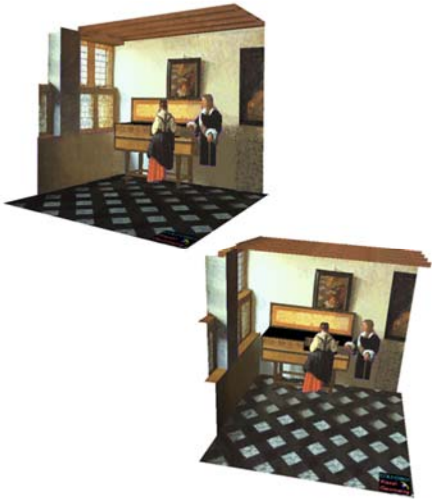
Projective geometry—what's it good for?

Uses of projective geometry

- Drawing
- Measurements
- Mathematics for projection
- Undistorting images
- Focus of expansion
- Camera pose estimation, match move
- Object recognition

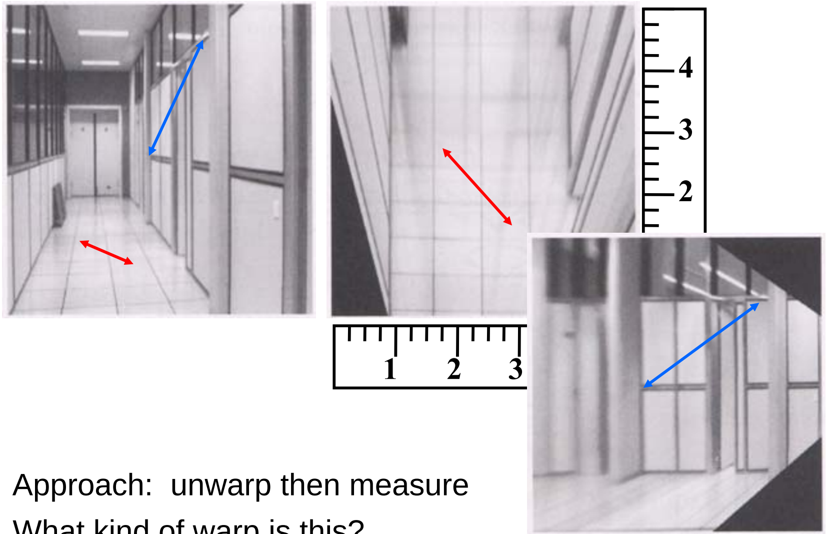
Applications of projective geometry

Vermeer's *Music Lesson*



Reconstructions by Criminisi et al.

Measurements on planes



Approach: unwarp then measure

What kind of warp is this?

Homographies

Perspective projection of a plane

- Lots of names for this:
  - **homography**, texture-map, colineation, planar projective map
- Modeled as a 2D warp using homogeneous coordinates

$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p'}} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$

To apply a homography **H**

- Compute **p'** = **Hp** (regular matrix multiply)
- Convert **p'** from homogeneous to image coordinates
  - divide by w (third) coordinate

Examples

$$\begin{bmatrix} 2x \\ 2y \\ 2 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\Downarrow$

(x, y)

(identity xform)

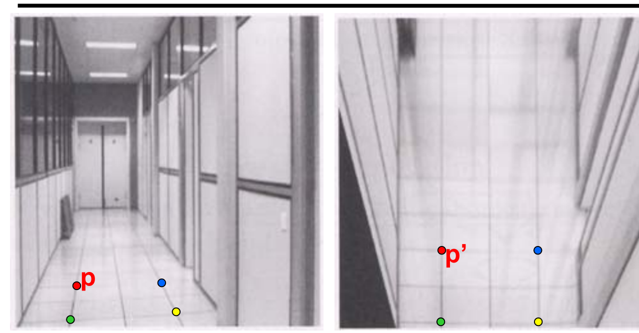
$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

scale

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

rotation

Image rectification



To unwarp (rectify) an image

- solve for homography **H** given **p** and **p'**
- solve equations of the form: w**p'** = **Hp**
- linear in unknowns: w and coefficients of **H**
- **H** is defined up to an arbitrary scale factor
- how many points are necessary to solve for **H**? 4

work out on board

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \cong \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$x' = \frac{ax+by+c}{gx+hy+i} \Rightarrow y'(gx+hy+i) = ax+by+c$$

$$0 = g(x'x) + h(x'y) + ix' - ax - by - c$$

$$y' = \frac{dx+ey+f}{gx+hy+i}$$

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

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

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

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

- 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

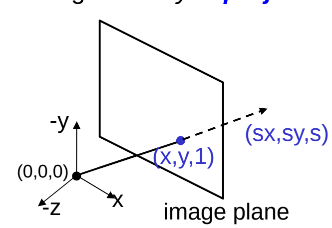
The projective plane

Why do we need homogeneous coordinates?

- represent points at infinity, homographies, perspective projection, multi-view relationships

What is the geometric intuition?

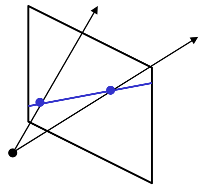
- a point in the image is a *ray* in ***projective space***



- 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) ≅ (sx, sy, s)

Projective lines

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



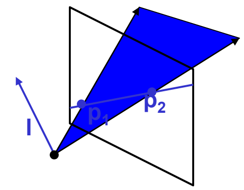
- A line is a *plane* of rays through origin
  - all rays (x,y,z) satisfying: ax + by + cz = 0

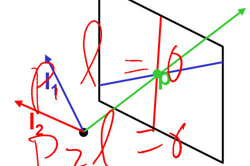
$$\text{in vector notation: } \quad 0 = \underbrace{\begin{bmatrix} a & b & c \end{bmatrix}}_{\mathbf{l}} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{p}}$$

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

Point and line duality

- A line **l** is a homogeneous 3-vector = [a b c]
- It is ⊥ to every point (ray) **p** on the line: **l p**=0





$p_1 \, l = 0$

$p_2 \, l = 0$

What is the line **l** spanned by rays $\mathbf{p_1}$ and $\mathbf{p_2}$?

- **l** is ⊥ to $\mathbf{p_1}$ and $\mathbf{p_2}$ ⇒ $\mathbf{l} = \mathbf{p_1} \times \mathbf{p_2}$

$p_1 \times p_2 \cong l$

- **l** is the plane normal

What is the intersection of two lines $\mathbf{l_1}$ and $\mathbf{l_2}$?

- **p** is ⊥ to $\mathbf{l_1}$ and $\mathbf{l_2}$ ⇒ $\mathbf{p} = \mathbf{l_1} \times \mathbf{l_2}$

$l_1 p = 0$

$l_2 p = 0$

$p \perp l_1$

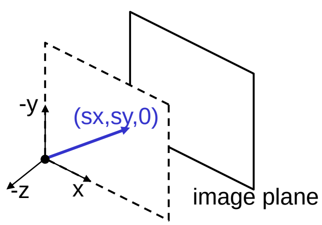
$p \perp l_2$

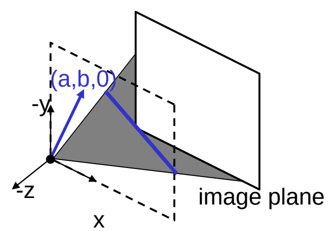
$p \cong l_1 \times l_2$

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 ≅ (x, y, 0) – parallel to image plane
- It has infinite image coordinates

Ideal line

- l ≅ (a, b, 0) – parallel to image plane
  - Corresponds to a line in the image (finite coordinates)
    - goes through image origin (*principle point*)

Homographies of points and lines

Computed by 3x3 matrix multiplication

- To transform a point: **p' = Hp**
- To transform a line: **lp**=0 ⇒ **l'p'**=0
  - 0 = **lp** = **lH^{-1}Hp** = **lH^{-1}p'** ⇒ **l' = lH^{-1}**
  - lines are transformed by postmultiplication of **H^{-1}**

$0 = l H^{-1} \underbrace{(Hp)}_{p'}$

$\underbrace{(l H^{-1})}_{l'} p' = 0$

3D projective geometry

These concepts generalize naturally to 3D

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

3D to 2D: "perspective" projection

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

What is *not* preserved under perspective projection?

angles between lines → parallel lines

size, relative distances, ratios

What IS preserved?

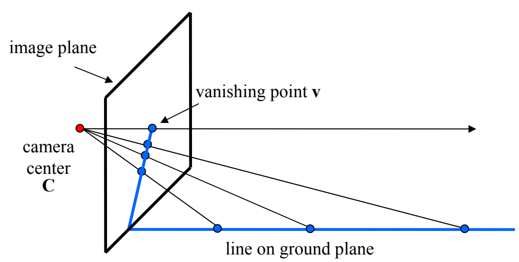
straight lines

intersections

points

conics

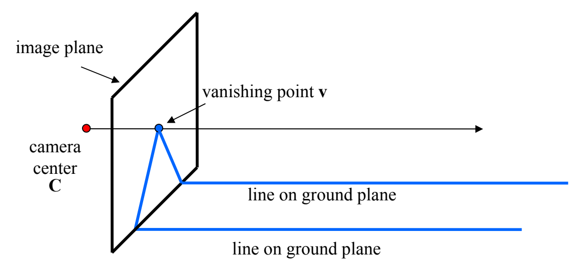
Vanishing points

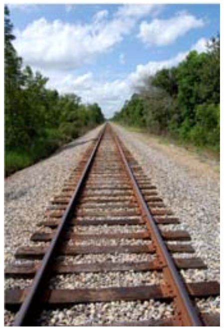


Vanishing point

- projection of a point at infinity

Vanishing points

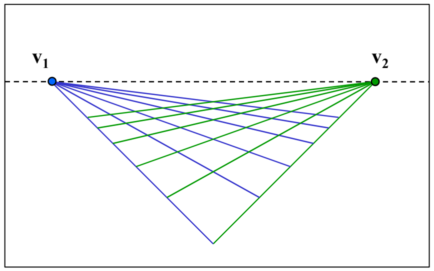




Properties

- Any two parallel lines 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 pixel is a potential vanishing point

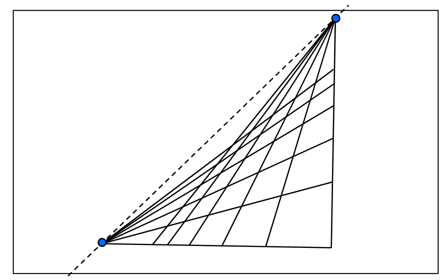
Vanishing lines



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*

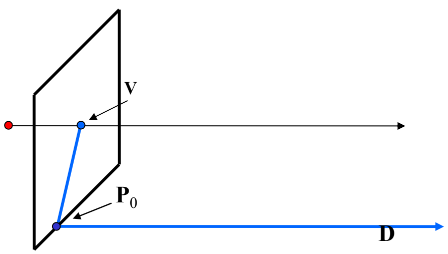
Vanishing lines



Multiple Vanishing Points

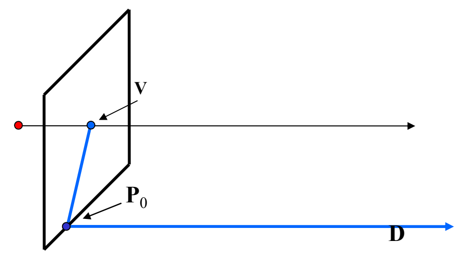
- Different planes define different vanishing lines

Computing vanishing points



$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \frac{1}{t} = \lim_{t\to\infty} \begin{bmatrix} \frac{1}{t}P_x + D_x \\ \frac{1}{t}P_y + D_y \\ \frac{1}{t}P_z + D_z \\ 1/t \end{bmatrix} = \begin{bmatrix} D_x \\ D_y \\ D_z \\ 0 \end{bmatrix}$$

Computing vanishing points

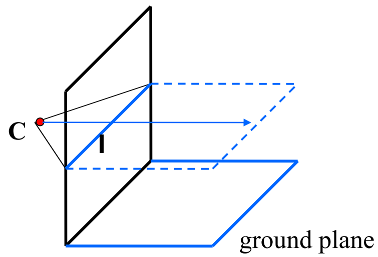


$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_X/t + D_X \\ P_Y/t + D_Y \\ P_Z/t + D_Z \\ 1/t \end{bmatrix} \qquad t \to \infty \qquad \mathbf{P}_\infty \cong \begin{bmatrix} D_X \\ D_Y \\ D_Z \\ 0 \end{bmatrix}$$

Properties $\mathbf{v} = \Pi \mathbf{P}_\infty$ ($\prod$ is camera projection matrix)

- $\mathbf{P}_\infty$ is a point at *infinity*, **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 $\mathbf{P}_\infty$

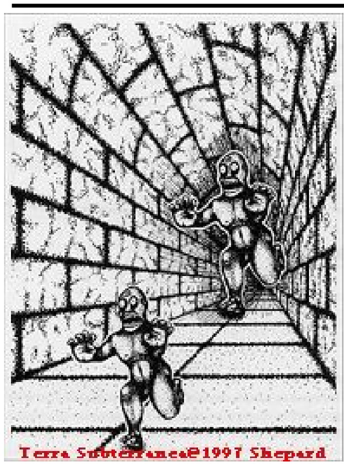
Computing the horizon

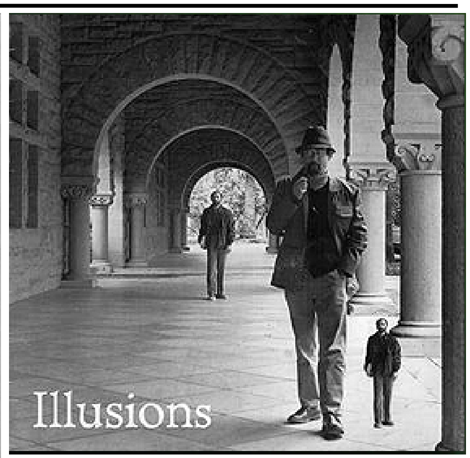


Properties

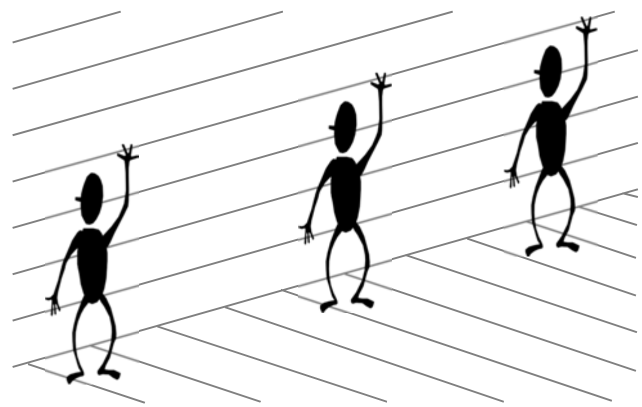
- **l** is intersection of horizontal plane through **C** with image plane
- Compute **l** from two sets of parallel lines on ground plane
- All points at same height as **C** project to **l**
  - points higher than **C** project above **l**
- Provides way of comparing height of objects in the scene

Fun with vanishing points

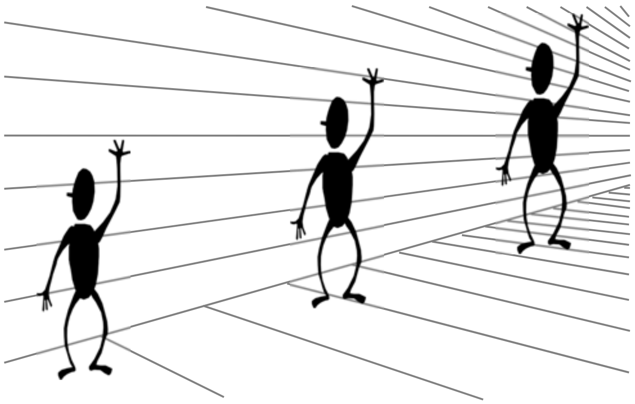




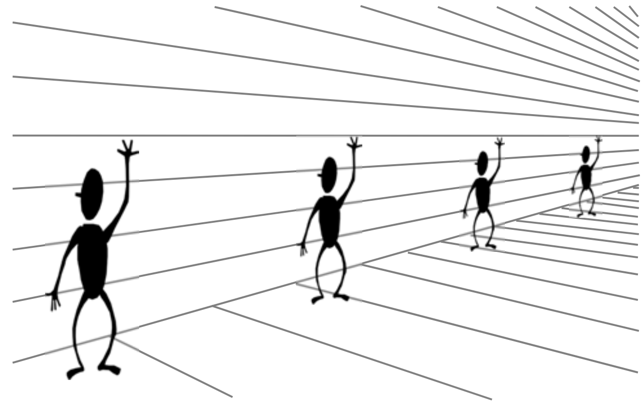
Perspective cues



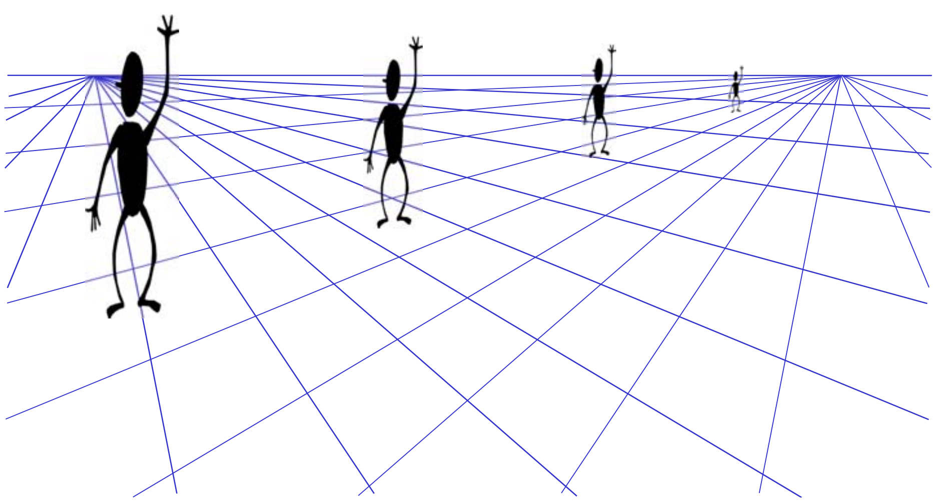
Perspective cues



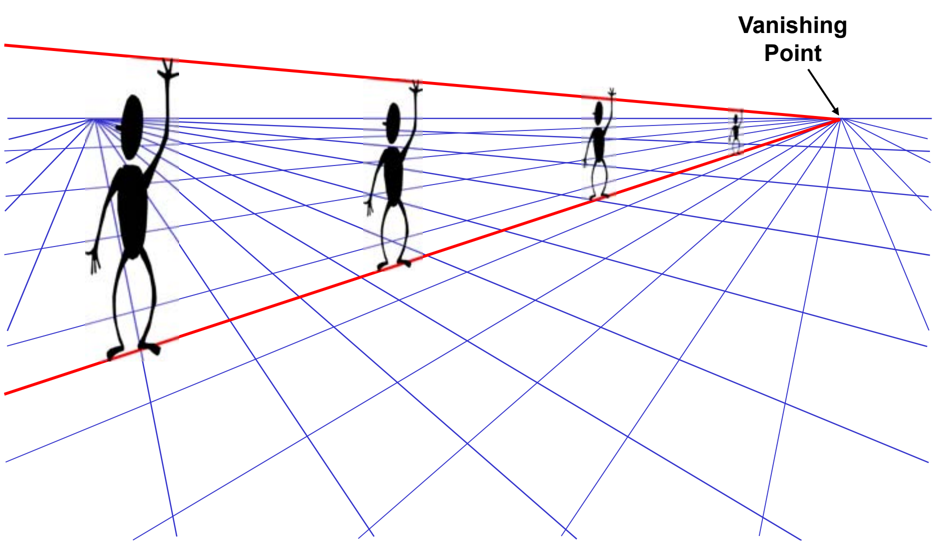
Perspective cues



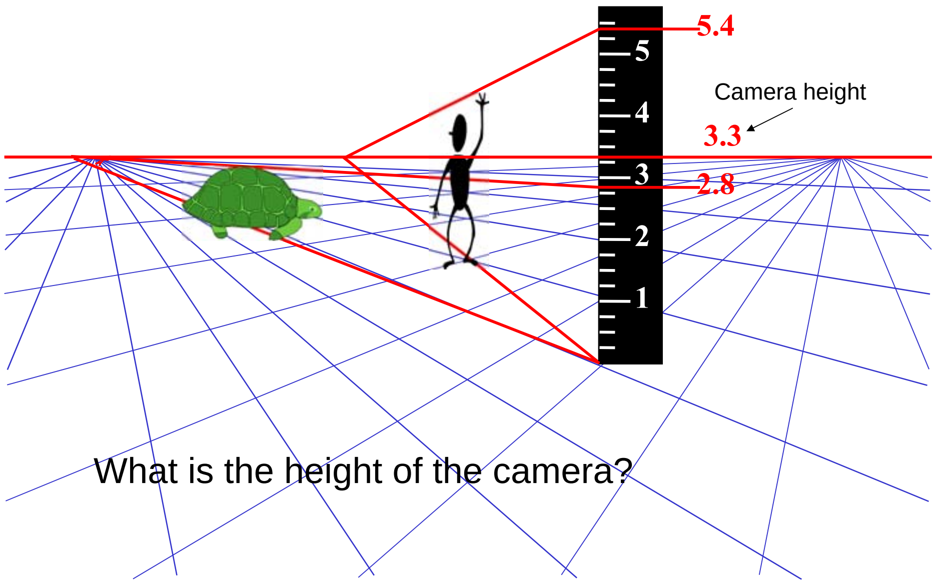
Are these guys the same height?



Comparing heights

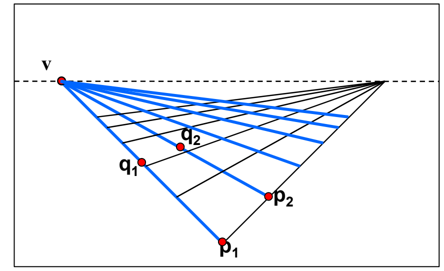


Measuring height



What is the height of the camera?

Computing vanishing points (from lines)



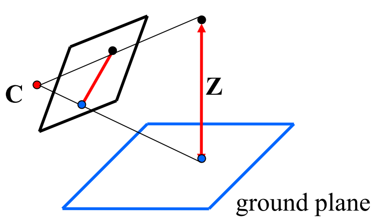
Intersect p_1q_1 with p_2q_2

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

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

Measuring height without a ruler



Compute Z from image measurements

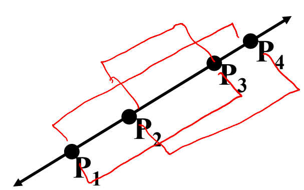
- Need more than vanishing points to do this

The cross ratio

A Projective Invariant

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

The cross-ratio of 4 collinear points



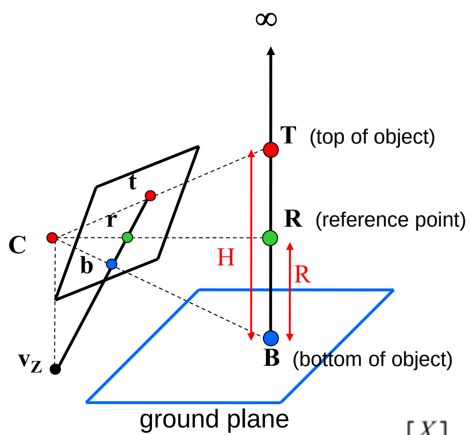
$$\frac{\|\mathbf{P}_3 - \mathbf{P}_1\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_3 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_1\|} \qquad \mathbf{P}_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Can permute the point ordering $\quad \frac{\|\mathbf{P}_1 - \mathbf{P}_3\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_1 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_3\|}$

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

This is the fundamental invariant of projective geometry

Measuring height



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

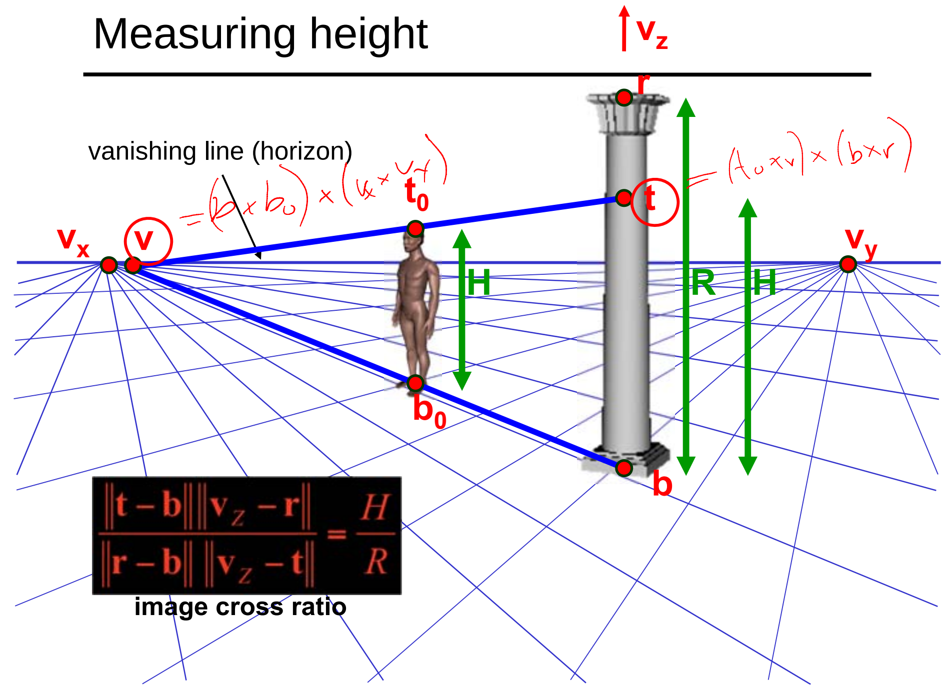
scene cross ratio

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

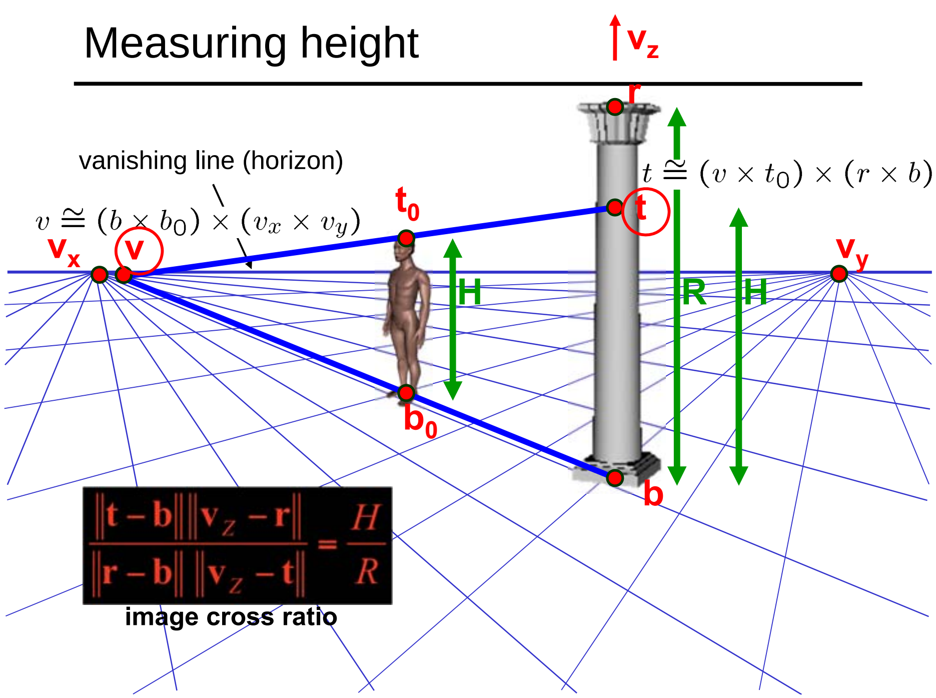
image cross ratio

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

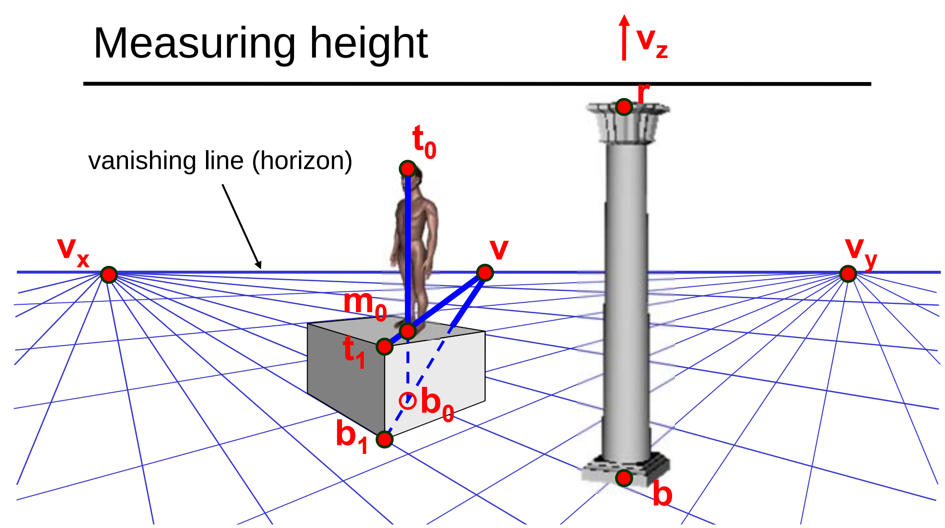
Measuring height



Measuring height



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

Computing (X,Y,Z) coordinates

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, principle point, pixel size)
  - extrinsics (rotation angles, translation)
  - radial distortion

Vanishing points and projection matrix

$$\mathbf{\Pi} = \begin{array}{c} \begin{bmatrix} \boxed{\begin{matrix} * \\ * \\ * \end{matrix}} & \boxed{\begin{matrix} * \\ * \\ * \end{matrix}} & \boxed{\begin{matrix} * \\ * \\ * \end{matrix}} & \boxed{\begin{matrix} * \\ * \\ * \end{matrix}} \end{bmatrix} \\ \begin{matrix} \boldsymbol{\pi}_1 & \boldsymbol{\pi}_2 & \boldsymbol{\pi}_3 & \boldsymbol{\pi}_4 \end{matrix} \end{array} = [\boldsymbol{\pi}_1 \ \boldsymbol{\pi}_2 \ \boldsymbol{\pi}_3 \ \boldsymbol{\pi}_4]$$

- $\boldsymbol{\pi}_1 = \mathbf{\Pi}[1\ 0\ 0\ 0]^T$ = $\mathbf{v}_x$ (X vanishing point)
- similarly, $\boldsymbol{\pi}_2 = \mathbf{v}_Y$, $\boldsymbol{\pi}_3 = \mathbf{v}_Z$
- $\boldsymbol{\pi}_4 = \mathbf{\Pi}[0\ 0\ 0\ 1]^T$ = projection of world origin

$$\mathbf{\Pi} = [\mathbf{v}_X \quad \mathbf{v}_Y \quad \mathbf{v}_Z \quad o]^T$$

Not So Fast! We only know **v**'s and **o** up to a scale factor

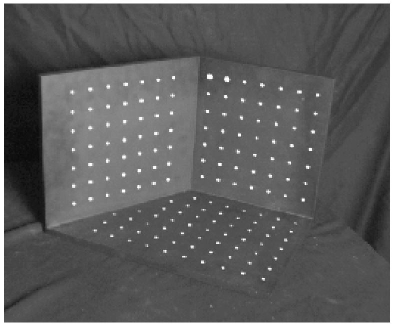
$$\mathbf{\Pi} = [a\mathbf{v}_X \quad b\mathbf{v}_Y \quad c\mathbf{v}_Z \quad do]^T$$

- Need more info to solve for these scale parameters (we won't cover this today)

Calibration using a reference object

Place a known object in the scene

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



Issues

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

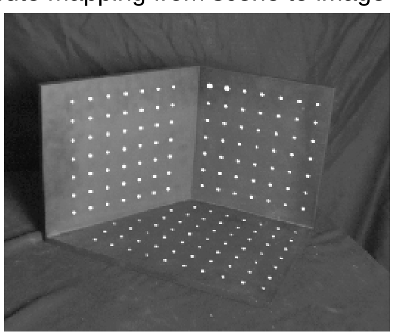
Chromaglyphs

Courtesy of Bruce Culbertson, HP Labs
http://www.hpl.hp.com/personal/Bruce_Culbertson/ibr98/chromagl.htm

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

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

$$\begin{aligned} u_i &= \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}} \\ v_i &= \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}} \end{aligned}$$

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

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

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

Direct linear calibration

Advantage:

- Very simple to formulate and solve

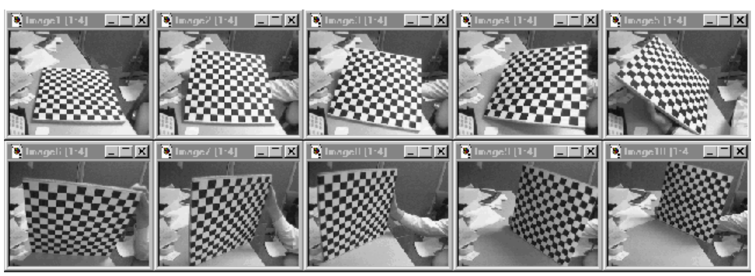
Disadvantages:

- Doesn't tell you the camera parameters
- Doesn't model radial distortion
- Hard to impose constraints (e.g., known focal length)
- Doesn't minimize the right 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
  – e.g., variants of Newton's method (e.g., Levenberg Marquart)

Alternative: multi-plane calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
  – OpenCV library: http://opencv.org/
  – Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  – GML Toolkit http://graphics.cs.msu.ru/en/node/909