3DVision-Part 2

Announcements

- Lab Assignment 2 Due: 11:59pm, 2nd May 2021
- Lab Assignment 3 will be released by this weekend.
- Tutorial on CLab Assignment 3 will happen next Wed.
- Survey about the course

https://forms.office.com/Pages/ResponsePage.aspx?id=XHJ941yrJEaa5fBTPkhkN4nAdRBV5JFGik5sof70GGpUM1lRUUtON0JIOFRCMkhMV0I5Q0NENzNWTi4u

Outline

- Review - Camera model, Camera Calibration (DLT algorithm)
- Image Measurement (vanishing point and vanishing line)
- Homography

A Projective Camera

The camera model for perspective projection is a linear map between homogeneous point coordinates

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} & \mathrm{P}\ (3 \times 4) & \end{bmatrix} \begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{pmatrix}$$

Image Point — Scene Point

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

- The camera centre is the null-vector of $\mathbf{P}$ => PC=0, where C is the camera centre.

e.g. if $\mathbf{P} = [\mathbf{I}|0]$ then the centre is $\mathbf{X} = (0, 0, 0, 1)^\top$.

- $\mathbf{P}$ has 11 degrees of freedom (essential parameters).
- $\mathbf{P}$ has rank 3.

Camera Calibration (Resectioning)

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

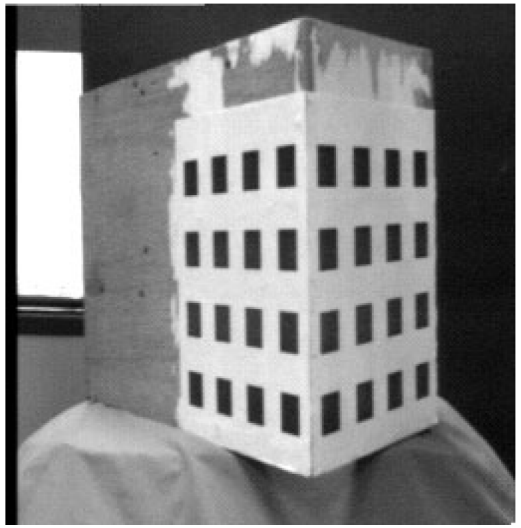
$\mathtt{P} = \mathtt{K}\left[\mathtt{R}\middle|\ \mathbf{t}\right]$ such that $\mathbf{x}_i = \mathtt{P}\mathbf{X}_i$.

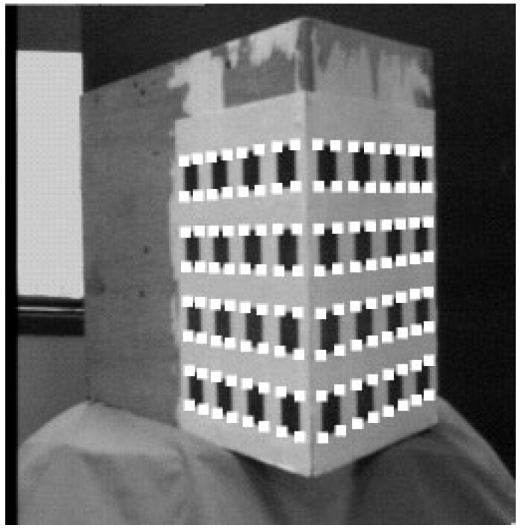
The algorithm for camera calibration has two parts:

(i) Compute the matrix $\mathtt{P}$ from a set of point correspondences.

(ii) Decompose $\mathtt{P}$ into $\mathtt{K}, \mathtt{R}$ and $\mathbf{t}$ via the $\mathtt{QR}$ decomposition.

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.

The DLT algorithm – camera resection

(Direct Linear Transformation (DLT))

Problem Statement:

Given n correspondences $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$, points in the image and the world.

Compute camera matrix $\mathbf{P}$ such that $\mathbf{x}'_i \approx \mathbf{P}\mathbf{X}_i$.

Each correspondence generates two equations

$$x_i = \frac{p_{11}\mathrm{X}_i + p_{12}\mathrm{Y}_i + p_{13}\mathrm{Z}_i + p_{14}}{p_{31}\mathrm{X}_i + p_{32}\mathrm{Y}_i + p_{33}\mathrm{Z}_i + p_{34}} \qquad y_i = \frac{p_{21}\mathrm{X}_i + p_{22}\mathrm{Y}_i + p_{23}\mathrm{Z}_i + p_{24}}{p_{31}\mathrm{X}_i + p_{32}\mathrm{Y}_i + p_{33}\mathrm{Z}_i + p_{34}}$$

Multiplying out gives equations linear in the matrix elements of $\mathbf{P}$

$$\begin{aligned} x_i(p_{31}\mathrm{X}_i + p_{32}\mathrm{Y}_i + p_{33}\mathrm{Z}_i + p_{34}) &= p_{11}\mathrm{X}_i + p_{12}\mathrm{Y}_i + p_{13}\mathrm{Z}_i + p_{14} \\ y_i(p_{31}\mathrm{X}_i + p_{32}\mathrm{Y}_i + p_{33}\mathrm{Z}_i + p_{34}) &= p_{21}\mathrm{X}_i + p_{22}\mathrm{Y}_i + p_{23}\mathrm{Z}_i + p_{24} \end{aligned}$$

These equations can be rearranged as

$$\begin{pmatrix} \mathrm{X} & \mathrm{Y} & \mathrm{Z} & 1 & 0 & 0 & 0 & 0 & -x\mathrm{X} & -x\mathrm{Y} & -x\mathrm{Z} & -x \\ 0 & 0 & 0 & 0 & \mathrm{X} & \mathrm{Y} & \mathrm{Z} & 1 & -y\mathrm{X} & -y\mathrm{Y} & -y\mathrm{Z} & -y \end{pmatrix} \mathbf{p} = \mathbf{0}$$

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.

Camera resection continued

Solving for $\mathbf{P}$

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

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

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

$$\min_{\mathbf{P}} \sum_i ((x_i, y_i) - \mathrm{dehom}(\mathbf{P}(\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i, 1)))^2$$

Direct Linear Transformation (DLT)

Objective

Given $n \geq 6$ 2D-to-3D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{X}_i\}$, determine the 3x4 projection matrix $\mathbf{P}$ such that $\mathbf{x}_i = \mathbf{P}\mathbf{X}_i$

Algorithm

(i) For each correspondence $\{\mathbf{x}_i \leftrightarrow \mathbf{X}_i\}$ compute $\mathbf{A}_i$. Usually only the two first rows are needed.

(ii) Assemble n 2x12 matrices $\mathbf{A}_i$ into a single $2n$x12 matrix $\mathbf{A}$

(iii) Compute the SVD of $\mathbf{A}$. The solution for $\mathbf{p}$ is the last column of $\mathbf{V}$

(iv) Normalize $\mathbf{p}$ to 1, and rearrange to obtain $\mathbf{P}$

Numerical issues

- Analysis of the potential numerical issues

These equations can be rearranged as

$$\begin{pmatrix} \mathrm{X} & \mathrm{Y} & \mathrm{Z} & 1 & 0 & 0 & 0 & 0 & -x\mathrm{X} & -x\mathrm{Y} & -x\mathrm{Z} & -x \\ 0 & 0 & 0 & 0 & \mathrm{X} & \mathrm{Y} & \mathrm{Z} & 1 & -y\mathrm{X} & -y\mathrm{Y} & -y\mathrm{Z} & -y \end{pmatrix} \mathbf{p} = \mathbf{0}$$

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.

- Magnitude issues in this matrix. Eg. X =1, Y=4, Z=3, x = 1000, y = 1000

$$\begin{pmatrix} 1 & 4 & 3 & 1 & 0 & 0 & 0 & 0 & -1000 & -4000 & -3000 & -1000 \\ 0 & 0 & 0 & 0 & 1 & 4 & 3 & 1 & -1000 & -4000 & -3000 & -1000 \end{pmatrix}$$

Normalized DLT algorithm

Objective

Given $n \geq 6$ 2D-to-3D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{X}_i\}$, determine the 3x4 projection matrix $\mathbf{P}$ such that $\mathbf{x}_i = \mathbf{P}\mathbf{X}_i$

Algorithm

(i) Normalize points: $\tilde{\mathbf{x}}_i = \mathbf{T}_{\text{norm}}\mathbf{x}_i, \tilde{\mathbf{X}}_i = \mathbf{S}_{\text{norm}}\mathbf{X}_i$

(ii) Apply the DLT algorithm to $\{\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{X}}_i\}$

(iii) Denormalize the recovered solution $\tilde{\mathbf{P}}$ using $\mathbf{P} = \mathbf{T}_{\text{norm}}^{-1}\tilde{\mathbf{P}}\mathbf{S}_{\text{norm}}$

- Example normalizations:

$$\mathbf{T}_{\text{norm}} = \begin{bmatrix} w+h & 0 & w/2 \\ 0 & w+h & h/2 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \qquad \mathbf{S}_{\text{norm}} = \begin{bmatrix} \mathbf{V}\,\text{diag}(\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1})\mathbf{V}^{-1} & -\mathbf{V}\,\text{diag}(\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1})\mathbf{V}^{-1}\boldsymbol{\mu}_{\mathbf{X}_i} \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{V}\,\text{diag}(\lambda_1, \lambda_2, \lambda_3)\mathbf{V}^{-1} = \text{eig}\left(\sum_i \left(\mathbf{X}_{i,\text{nonhom}} - \boldsymbol{\mu}_{\mathbf{X}_i}\right)\left(\mathbf{X}_{i,\text{nonhom}} - \boldsymbol{\mu}_{\mathbf{X}_i}\right)^T \right)$$

Projective Geometry

2.3 Projective transformations

In the view of geometry set forth by Felix Klein in his famous "Erlangen Program", [Klein-39], geometry is the study of properties invariant under groups of transformations. From this point of view, 2D projective geometry is the study of properties of the projective plane $\mathbb{P}^2$ that are invariant under a group of transformations known as *projectivities*.

A projectivity is an invertible mapping from points in $\mathbb{P}^2$ (that is homogeneous 3-vectors) to points in $\mathbb{P}^2$ that maps lines to lines. More precisely,

Definition 2.9. A *projectivity* is an invertible mapping h from $\mathbb{P}^2$ to itself such that three points $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ lie on the same line if and only if $h(\mathbf{x}_1), h(\mathbf{x}_2)$ and $h(\mathbf{x}_3)$ do.

Projectivities form a group since the inverse of a projectivity is also a projectivity, and so is the composition of two projectivities. A projectivity is also called a *collineation*

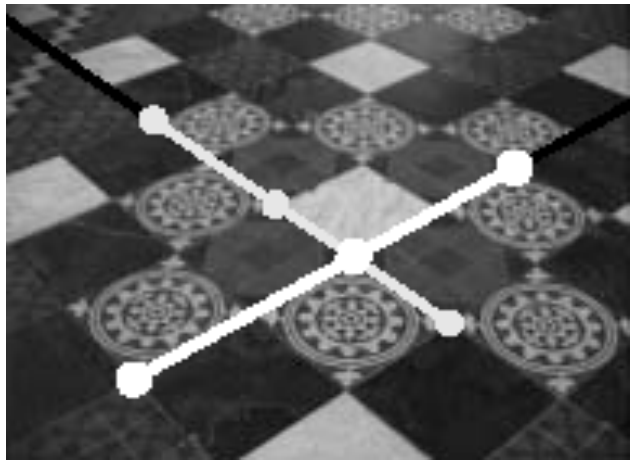
Multiple View geometry in computer vision.

Projective Transformation

Any mapping of the projective plane to itself that preserves lines is called a “projective transformation”, or “collineation”, or “homography”

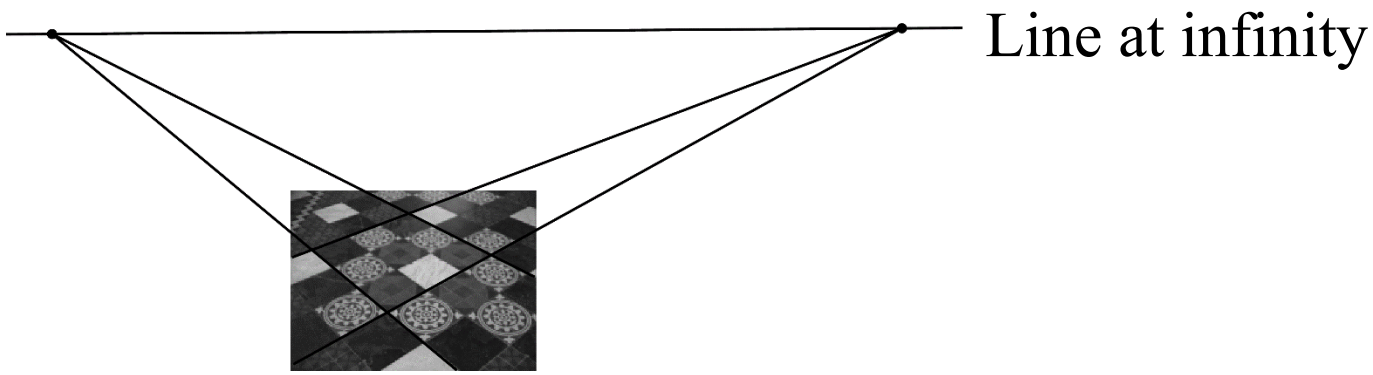
Mapping: A mapping from points in the plane to points in the plane.

Preserves lines: If three points lie on a line, then their corresponding points under the mapping lie on a line.



The Projective Plane $\mathcal{P}^2$

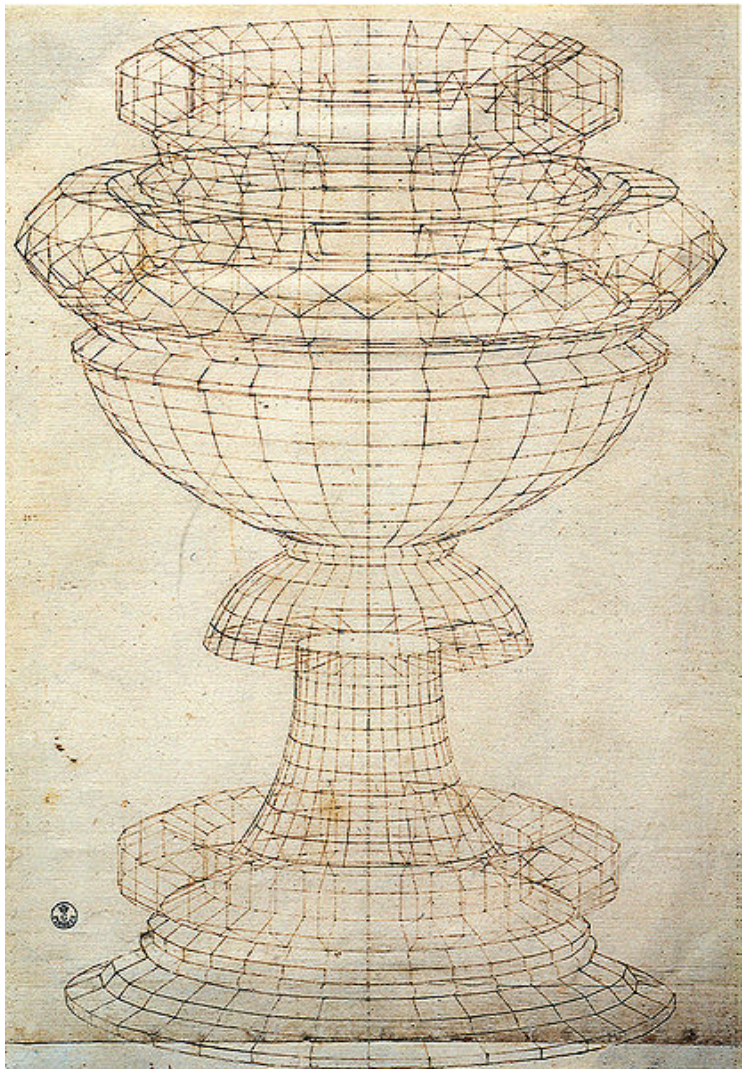
Extend the Euclidean Plane by adding a line at infinity.



Any two lines meet in exactly one point

Projective geometry—what's it good for?

- Uses of projective geometry
  - Drawing
  - Measurements
  - Mathematics for projection
  - Undistorting images
  - Camera pose estimation
  - **Object recognition**

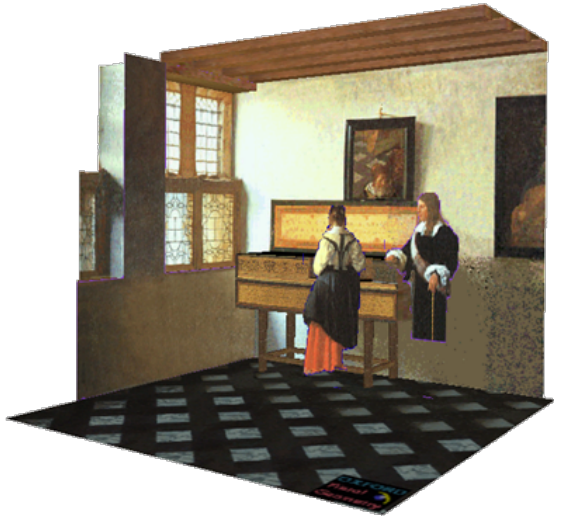


Paolo Uccello

Credits: Noah Snavely

Applications of projective geometry

Vermeer's *Music Lesson*



Reconstructions by Criminisi et al.

Credits: Noah Snavely

Making measurements in images

WARBY PARKER

Measure your pupillary distance (PD)

Your PD is the distance between your pupils. To measure it, follow the instructions below — once you submit your photo, our team of experts will determine your PD and email you once we've applied it to your order.

1

Wearing glasses?
Take 'em off before you get started.

2

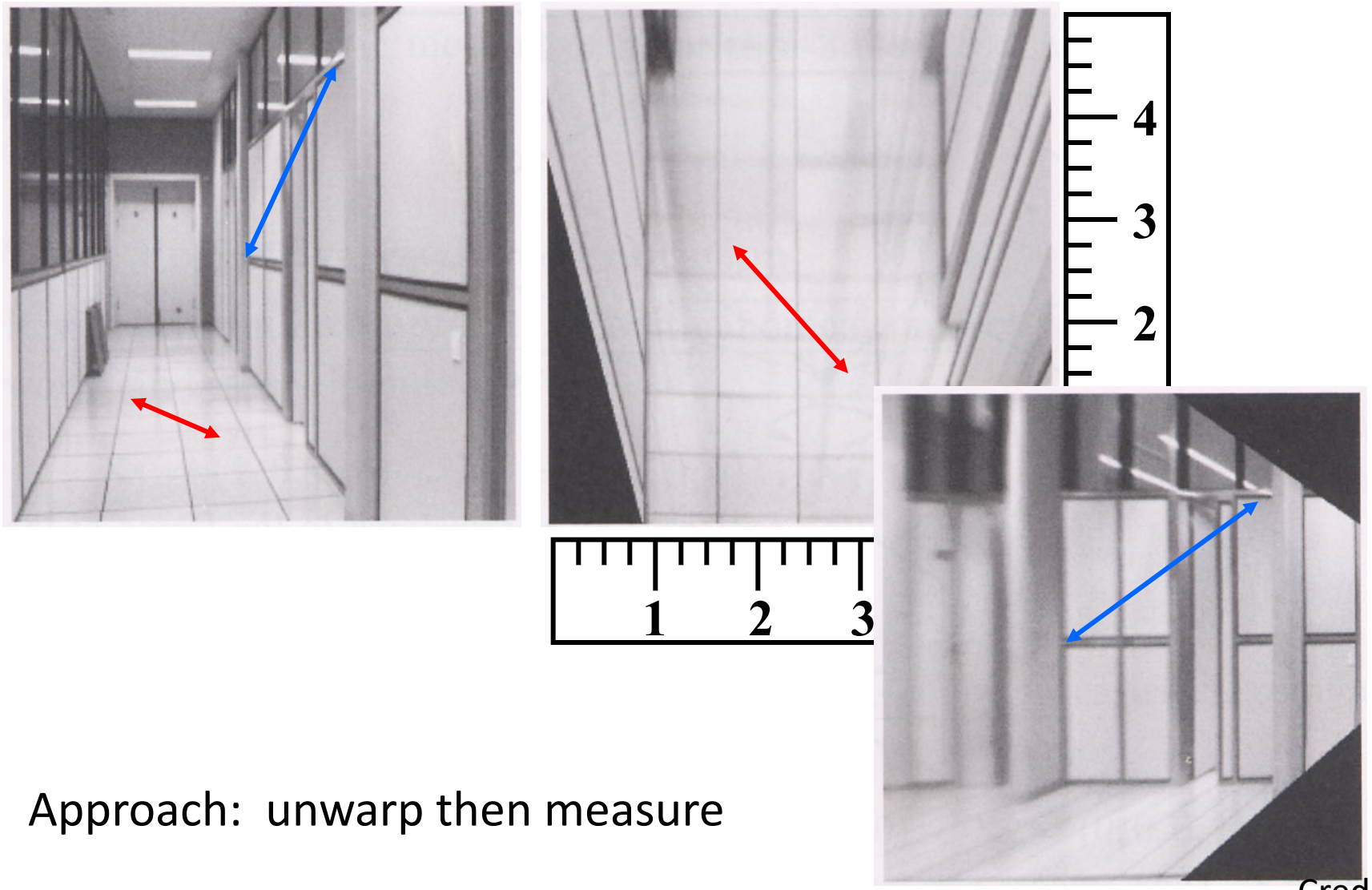
Hold up any card with a magnetic strip (we use this for scale).

3

Look straight ahead and snap a photo.

Credits: Noah Snavely

Measurements on planes

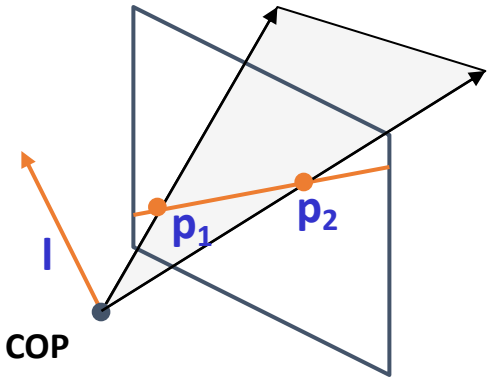


Approach: unwarp then measure

Credits: Noah Snavely

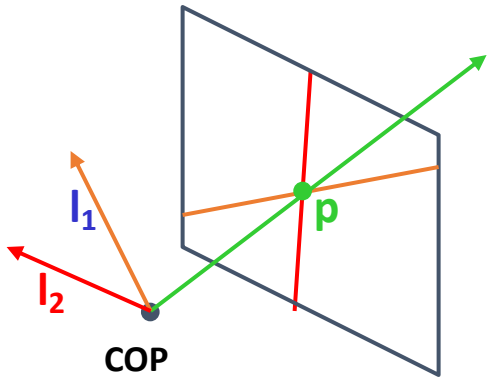
Point and line duality

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



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

- **l** is ⊥ 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*



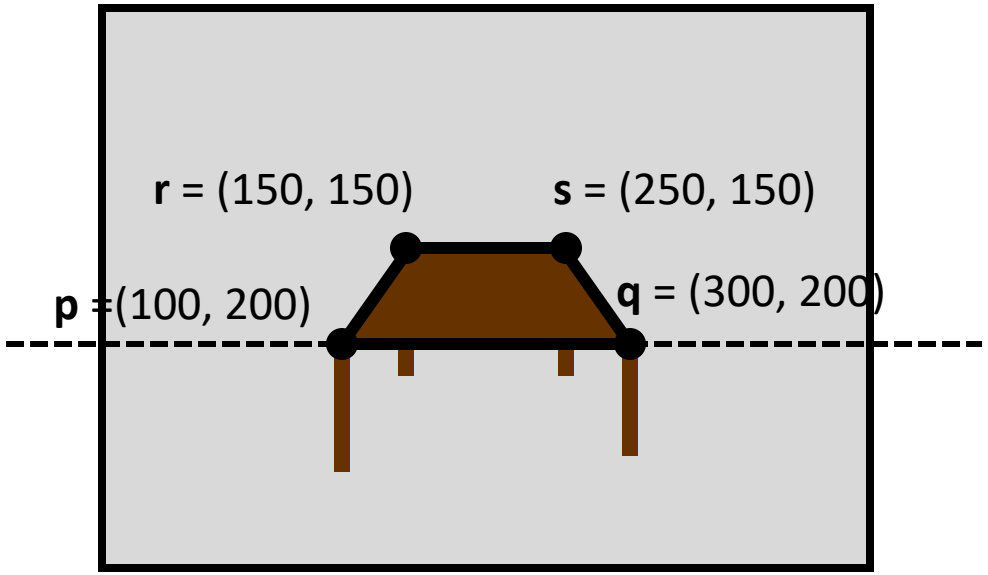
What is the intersection of two lines $\mathbf{l}_1$ and $\mathbf{l}_2$?

- **p** is ⊥ 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

Credits: Noah Snavely

Example

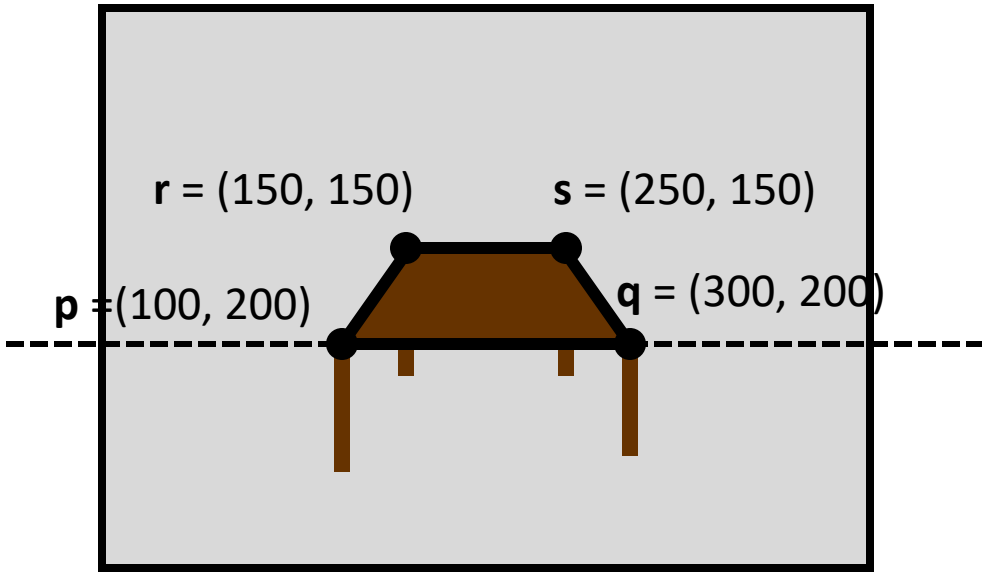


What is the line passing through points **p** and **q**?

$\mathbf{p} \times \mathbf{q}$

Credits: Noah Snavely

Example

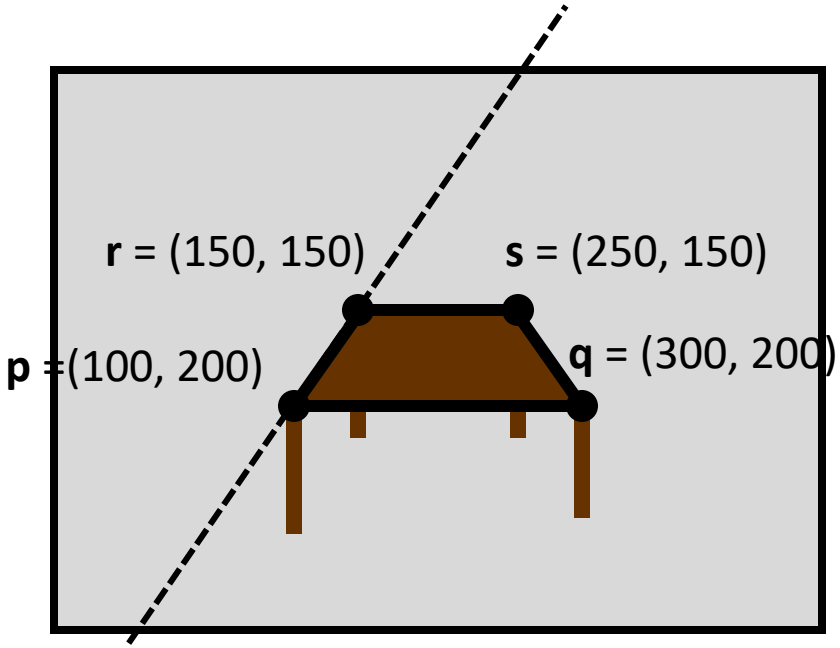


How do we interpret the line $\ell = \begin{bmatrix} 0 \\ 1 \\ -200 \end{bmatrix}$?

Answer, the set of points (*x*, *y*) such that $\ell \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$, i.e., $y - 200 = 0$

Credits: Noah Snavely

Example



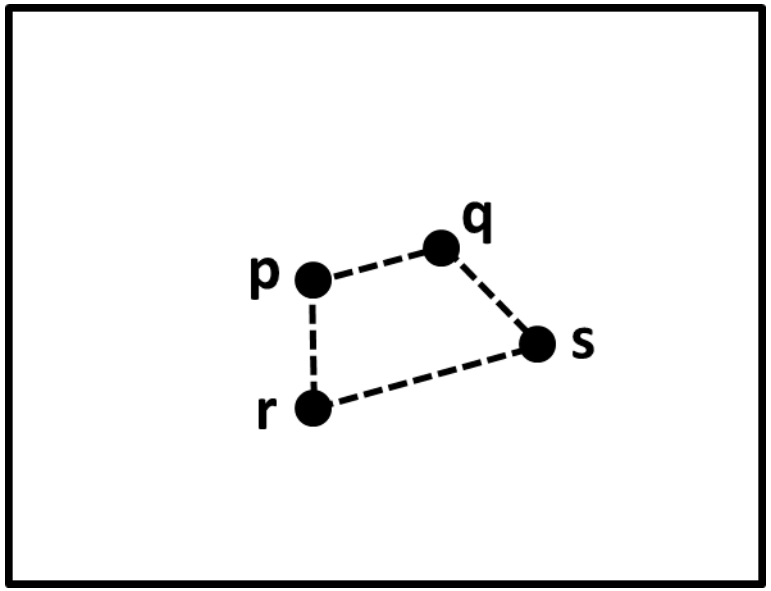
What is the line passing through points **p** and **r**?

$$\mathbf{p} \times \mathbf{r} = \begin{bmatrix} 100 \\ 200 \\ 1 \end{bmatrix} \times \begin{bmatrix} 150 \\ 150 \\ 1 \end{bmatrix} = \begin{bmatrix} 200 \cdot 1 - 150 \cdot 1 \\ 150 \cdot 1 - 100 \cdot 1 \\ 100 \cdot 150 - 150 \cdot 200 \end{bmatrix} = \begin{bmatrix} 50 \\ 50 \\ -15000 \end{bmatrix} \sim \begin{bmatrix} 1 \\ 1 \\ -300 \end{bmatrix}$$

i.e., all points (x, y) such that $x + y = 300$

Credits: Noah Snavely

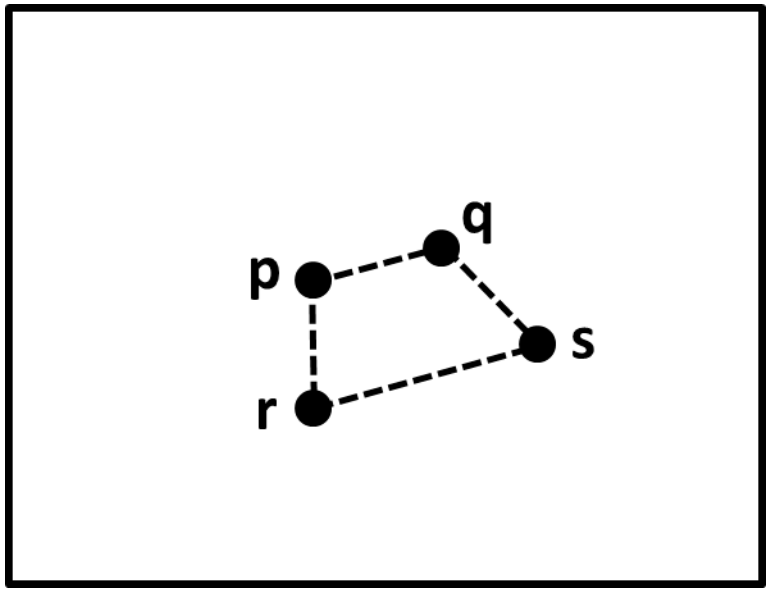
Question time



Consider the above image, with four points **p**, **q**, **r**, **s**, labeled (assume these are 2D homogeneous points). What is a simple expression for the point of intersection between the lines **pr** and **qs**?

Credits: Noah Snavely

Question time



Consider the above image, with four points **p**, **q**, **r**, **s**, labeled (assume these are 2D homogeneous points). What is a simple expression for the point of intersection between the lines **pr** and **qs**?

Answer: (**p** x **r**) x (**q** x **s**)

Credits: Noah Snavely

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
    - Three points define a plane, three planes define a point

Credits: Noah Snavely

3D to 2D: perspective projection

Projection:

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

Credits: Noah Snavely

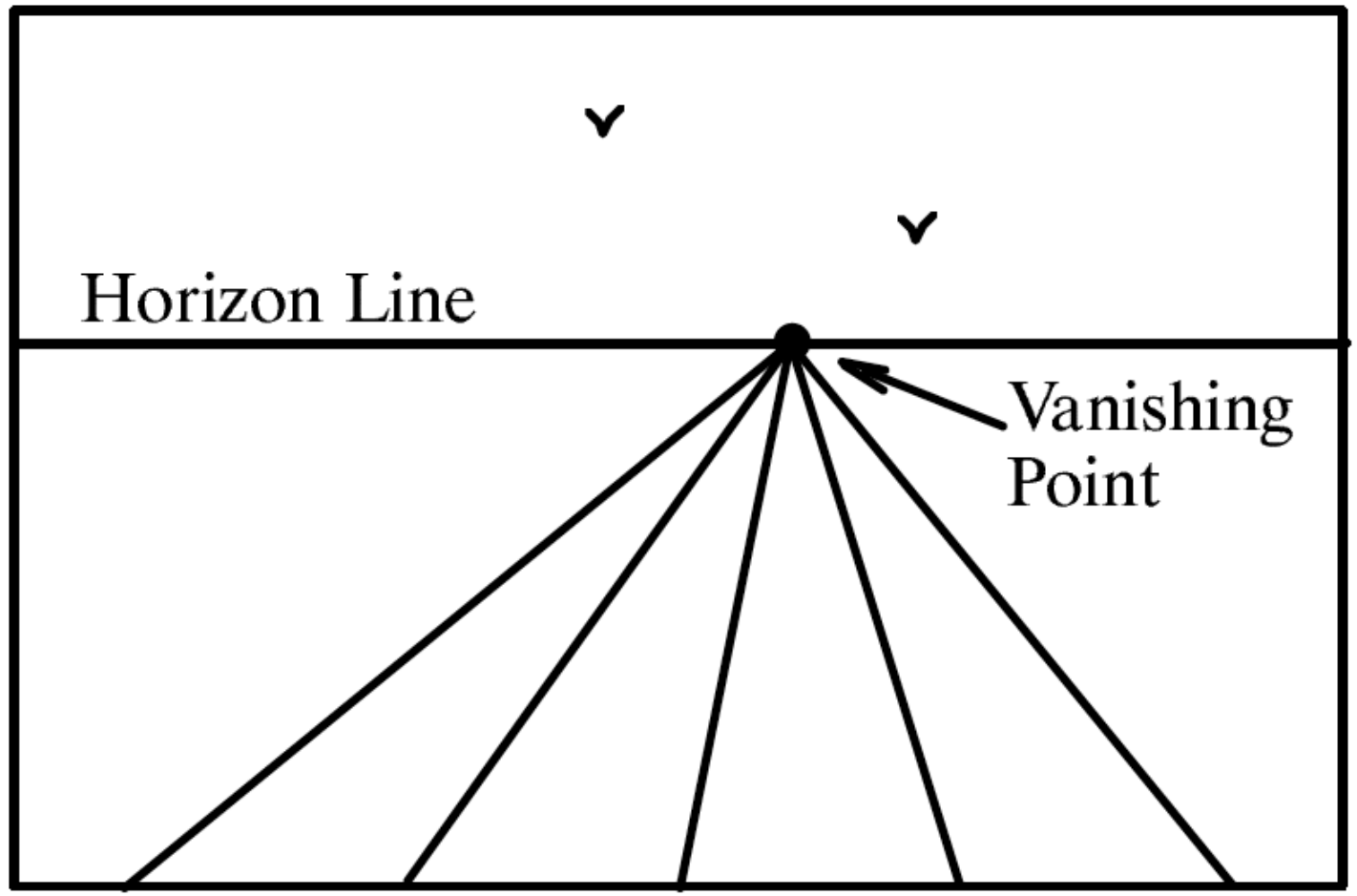
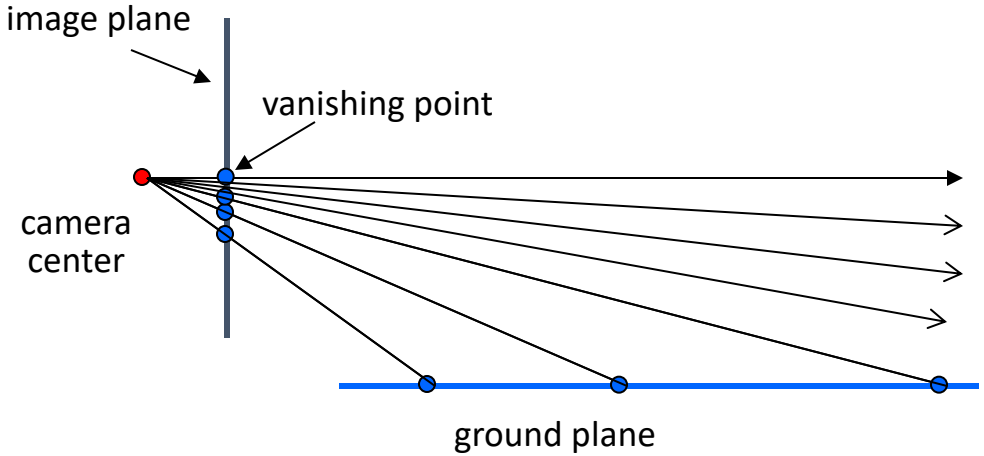


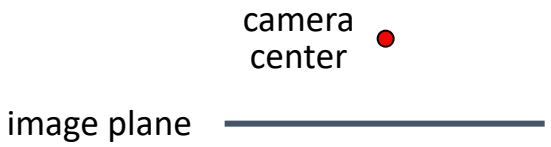
Figure 23.4
A perspective view of a set of parallel lines in the plane. All of the lines converge to a single vanishing point.

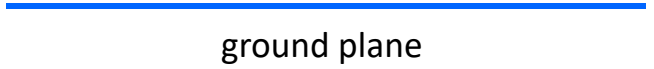
Credits: Noah Snavely

Vanishing points (1D)



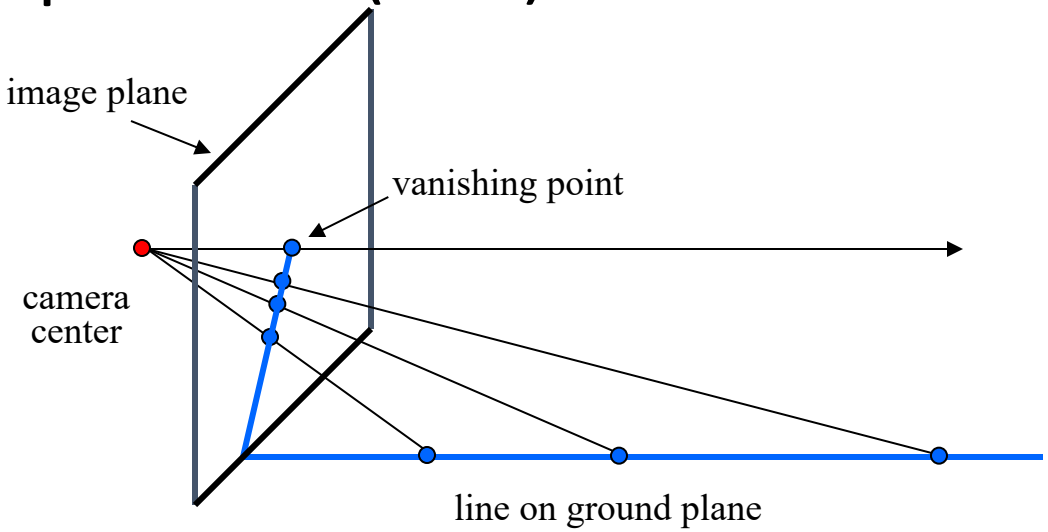
- Vanishing point
  - projection of a point at infinity
  - can often (but not always) project to a finite point in the image





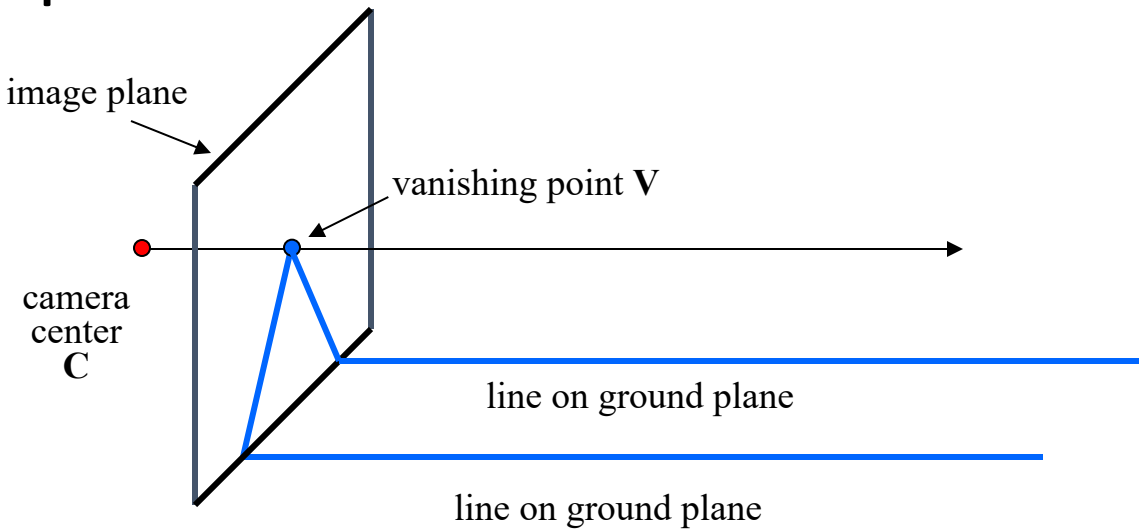
Credits: Noah Snavely

Vanishing points (2D)



Credits: Noah Snavely

Vanishing points



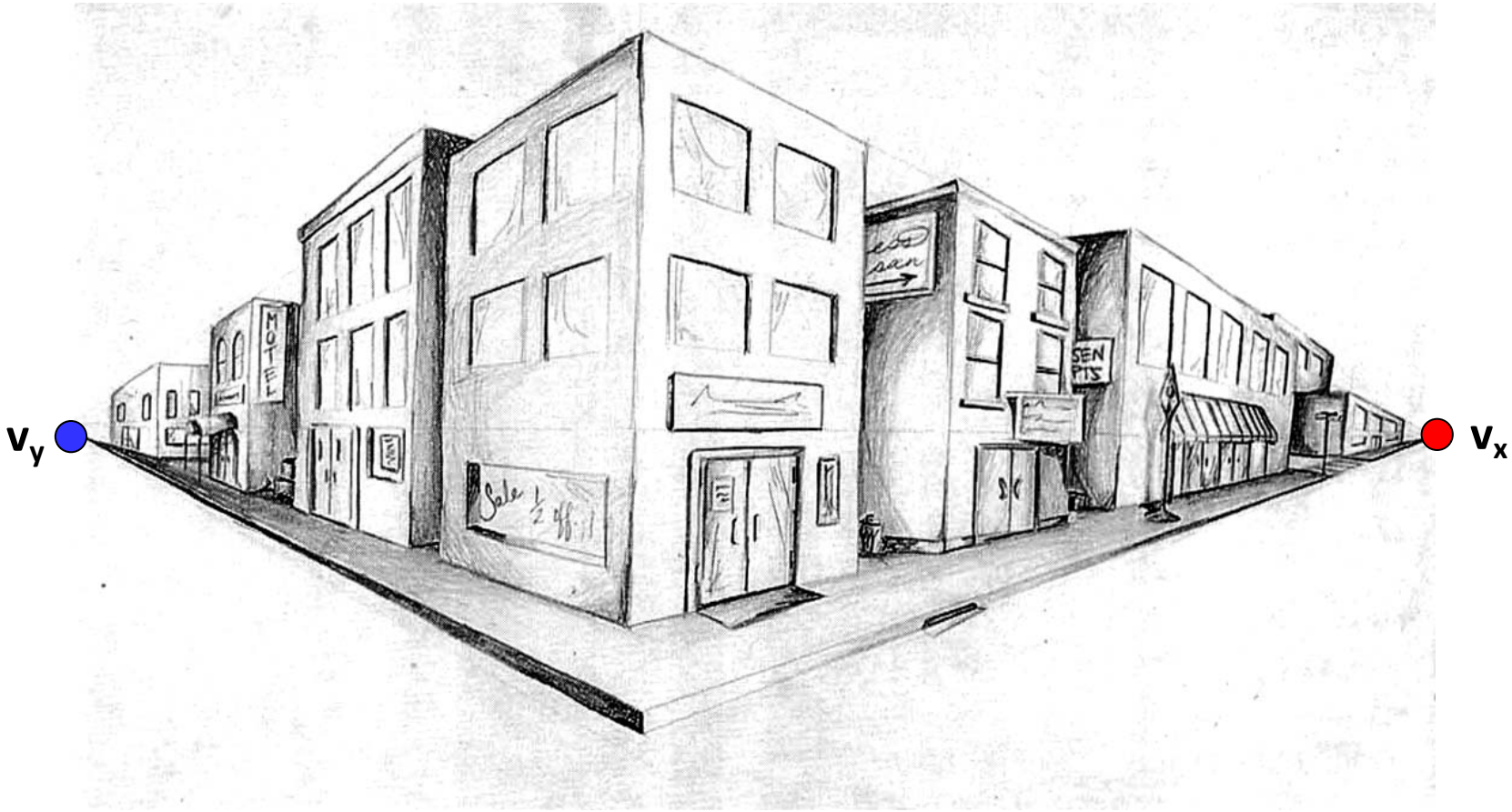
- Properties
  - Any two parallel lines (in 3D) have the same vanishing point **v**
  - The ray from **C** through **v** is parallel to the lines
  - An image may have more than one vanishing point
    - in fact, every image point is a potential vanishing point

Credits: Noah Snavely

One-point perspective

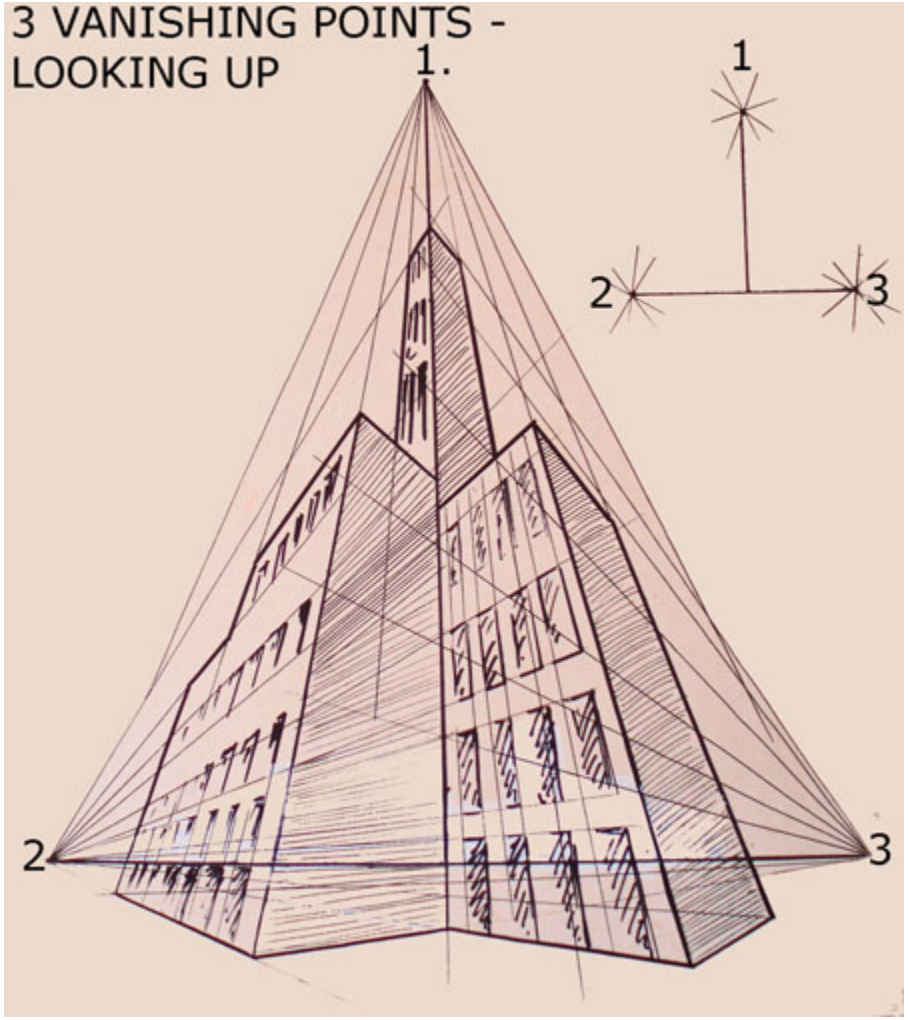
Credits: Noah Snavely

Two-point perspective



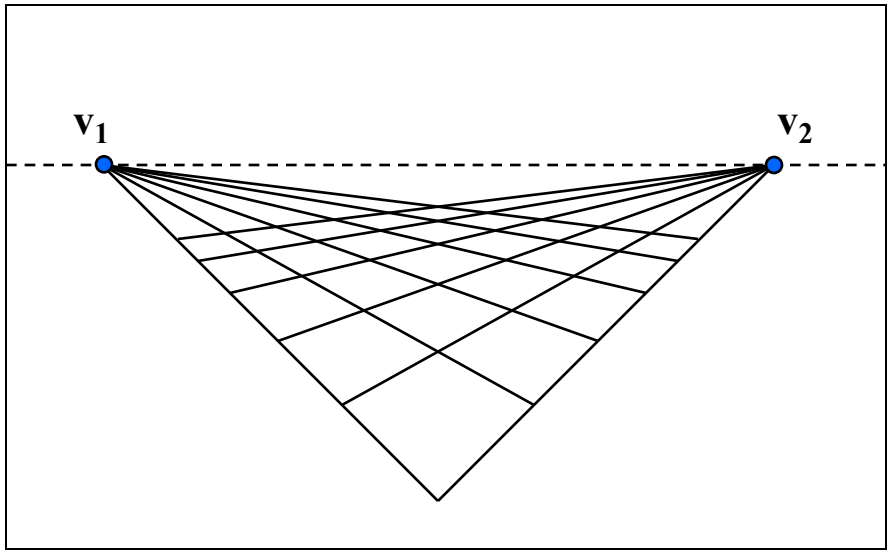
Credits: Noah Snavely

Three-point perspective



Credits: Noah Snavely

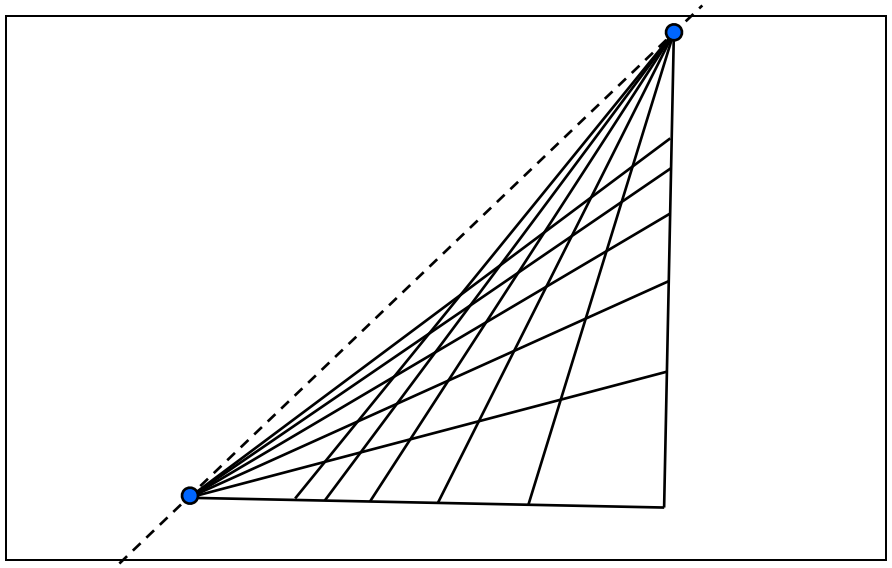
Vanishing lines



- 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

Credits: Noah Snavely

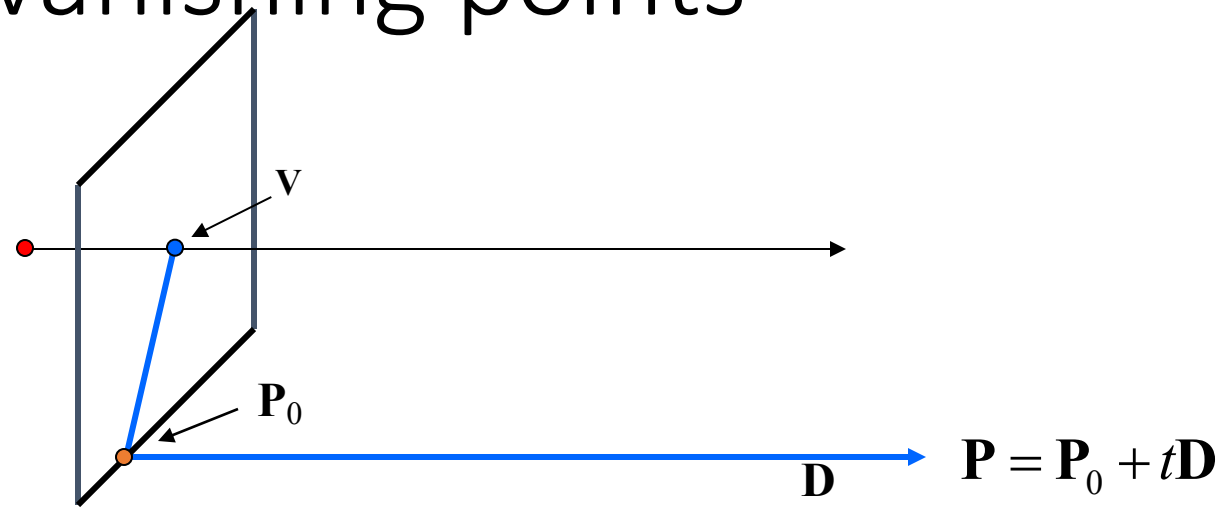
Vanishing lines



- 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

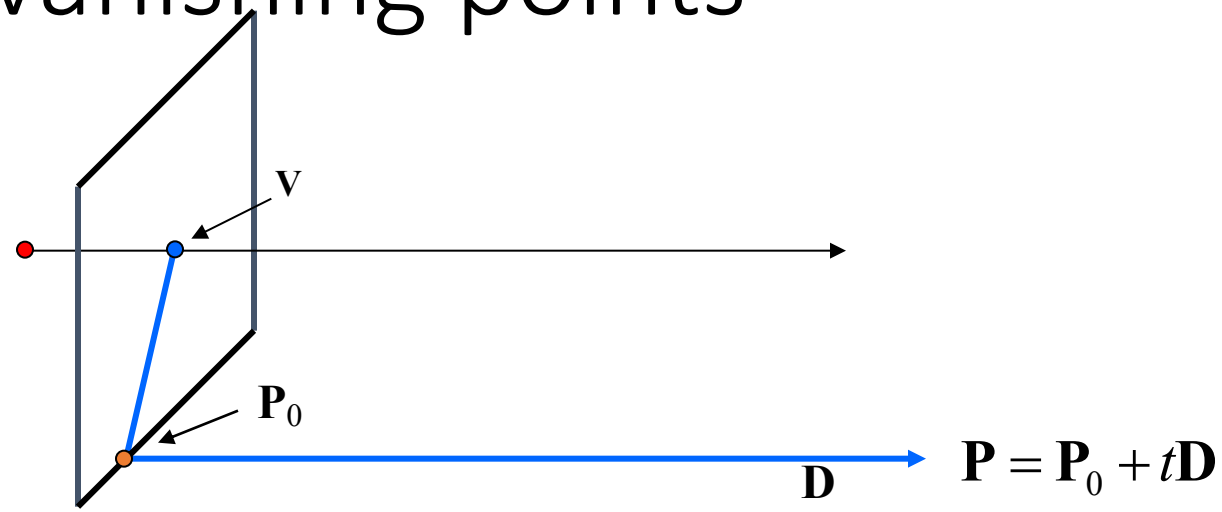
Credits: Noah Snavely

Computing vanishing points



Credits: Noah Snavely

Computing vanishing points



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

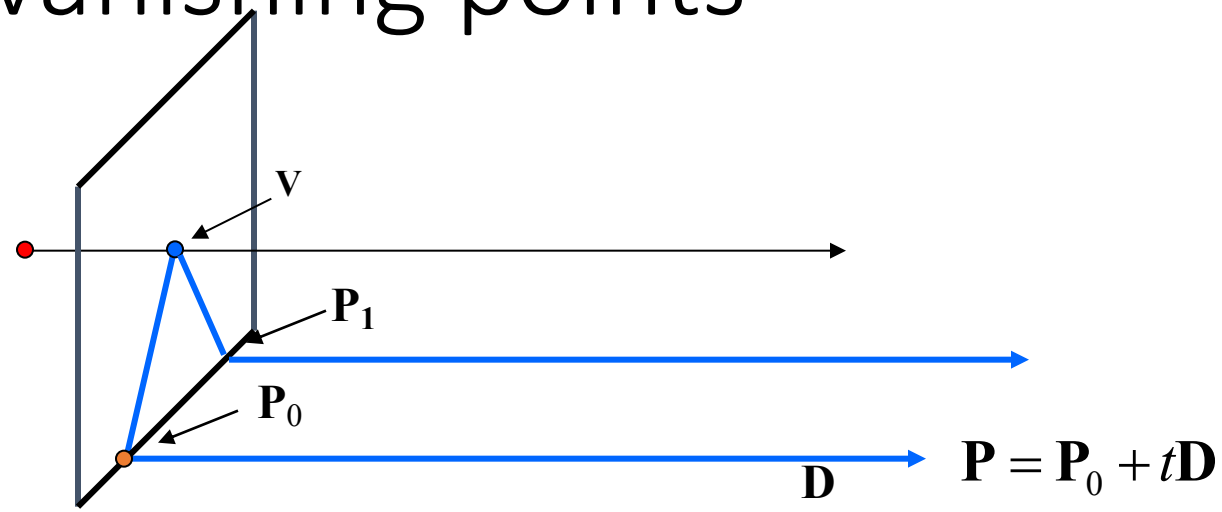
- Properties

$$\mathbf{v} = \Pi\mathbf{P}_\infty$$

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

Credits: Noah Snavely

Computing vanishing points



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

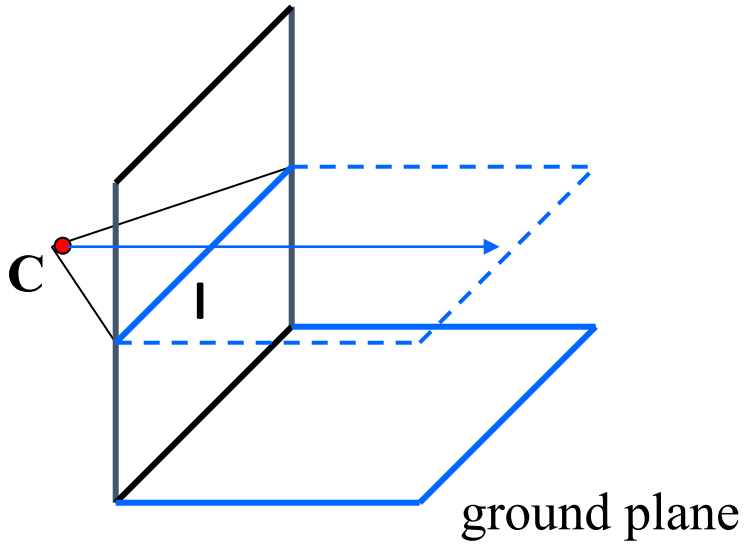
- Properties

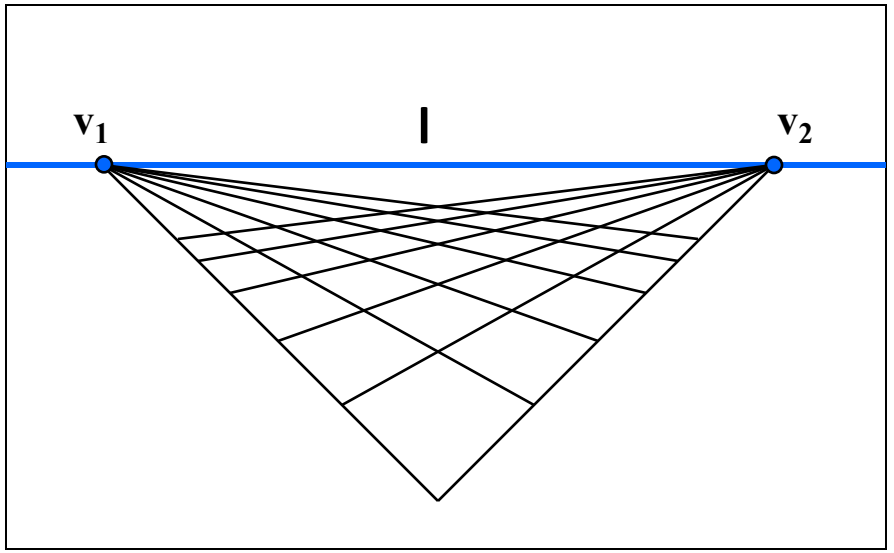
$$\mathbf{v} = \Pi \mathbf{P}_\infty$$

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

Credits: Noah Snavely

Computing vanishing lines



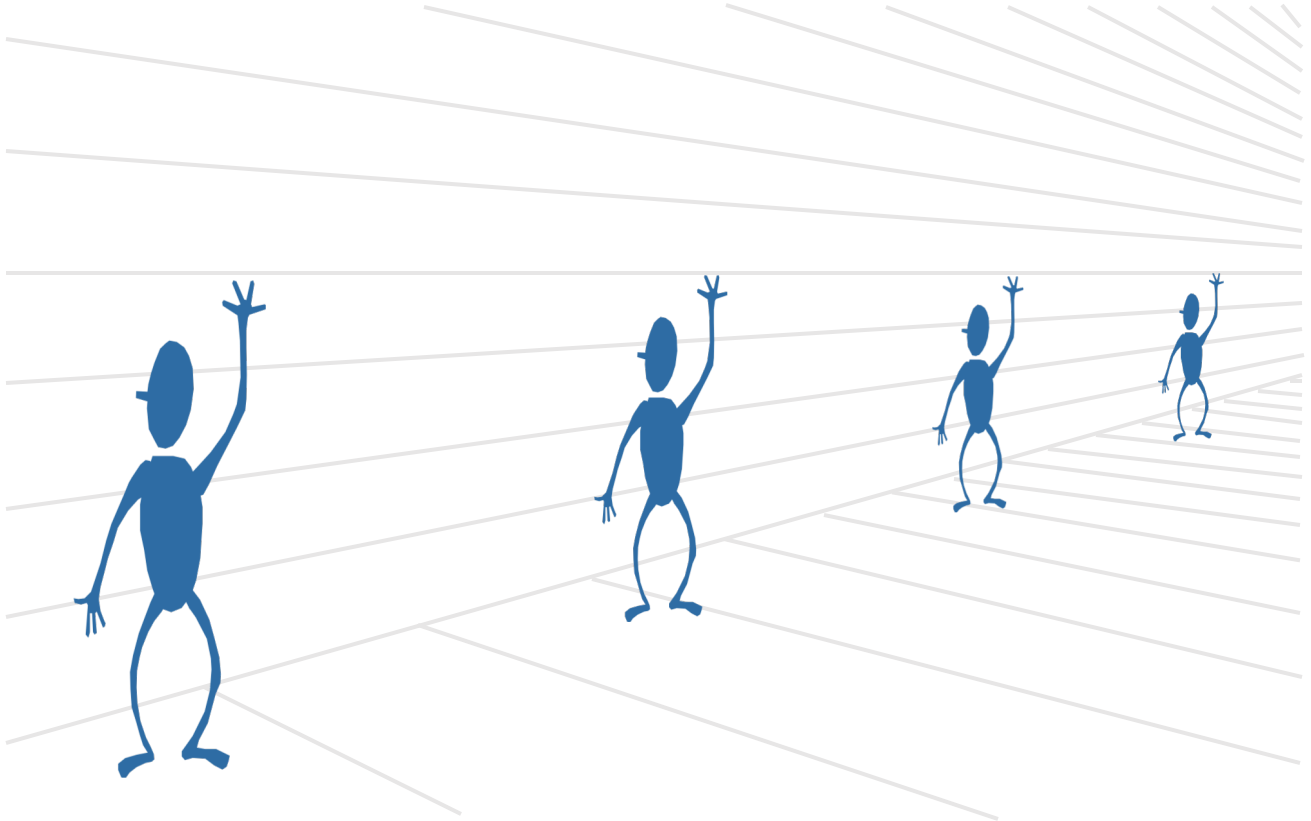


- Properties
  - **l** is intersection of horizontal plane through **C** with image plane
  - Compute **l** from two sets of parallel lines on ground plane
  - All points at same height as **C** project to **l**
    - points higher than C project above l
  - Provides way of comparing height of objects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

Credits: Noah Snavely

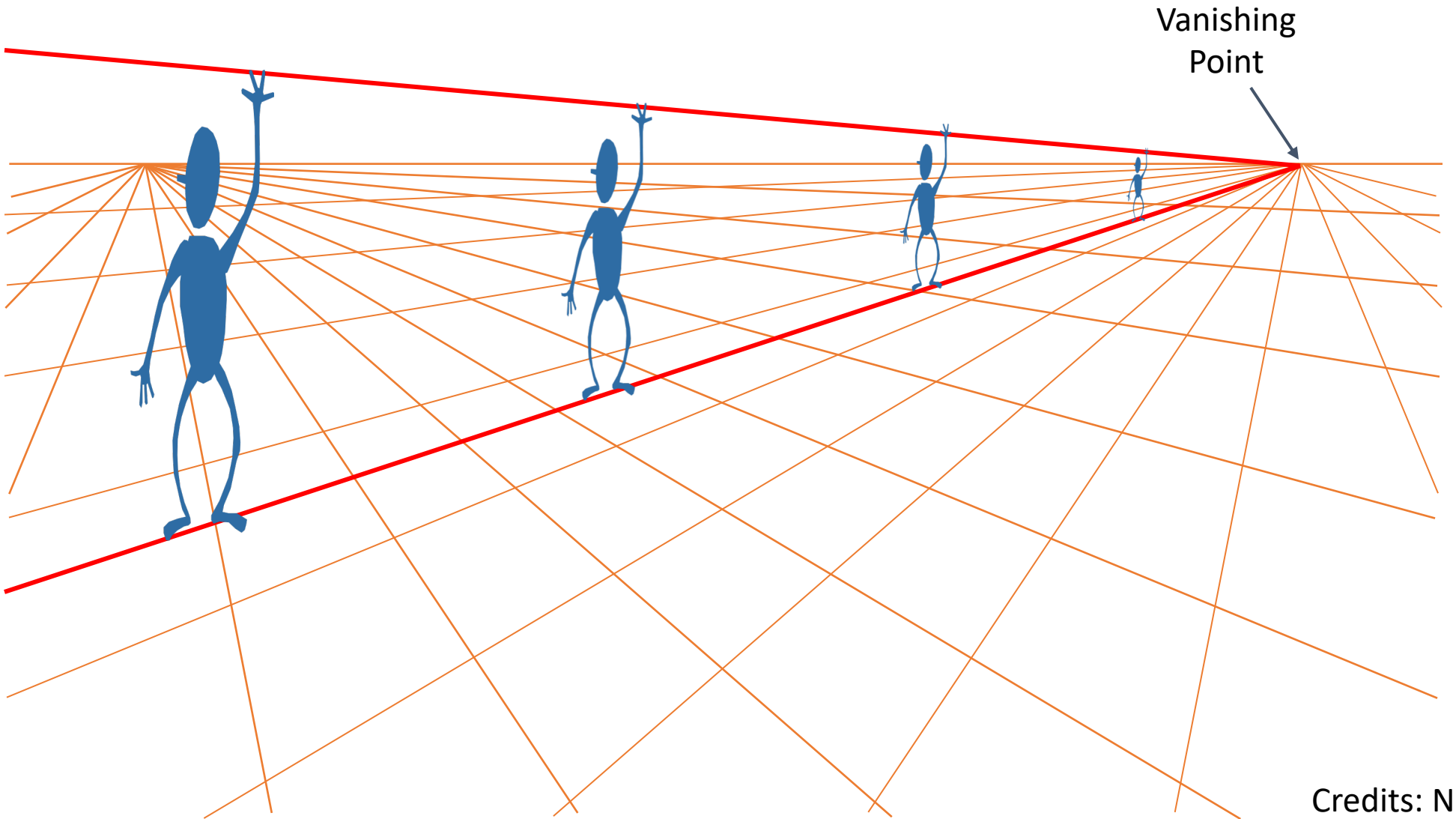
Credits: Noah Snavely

Perspective cues



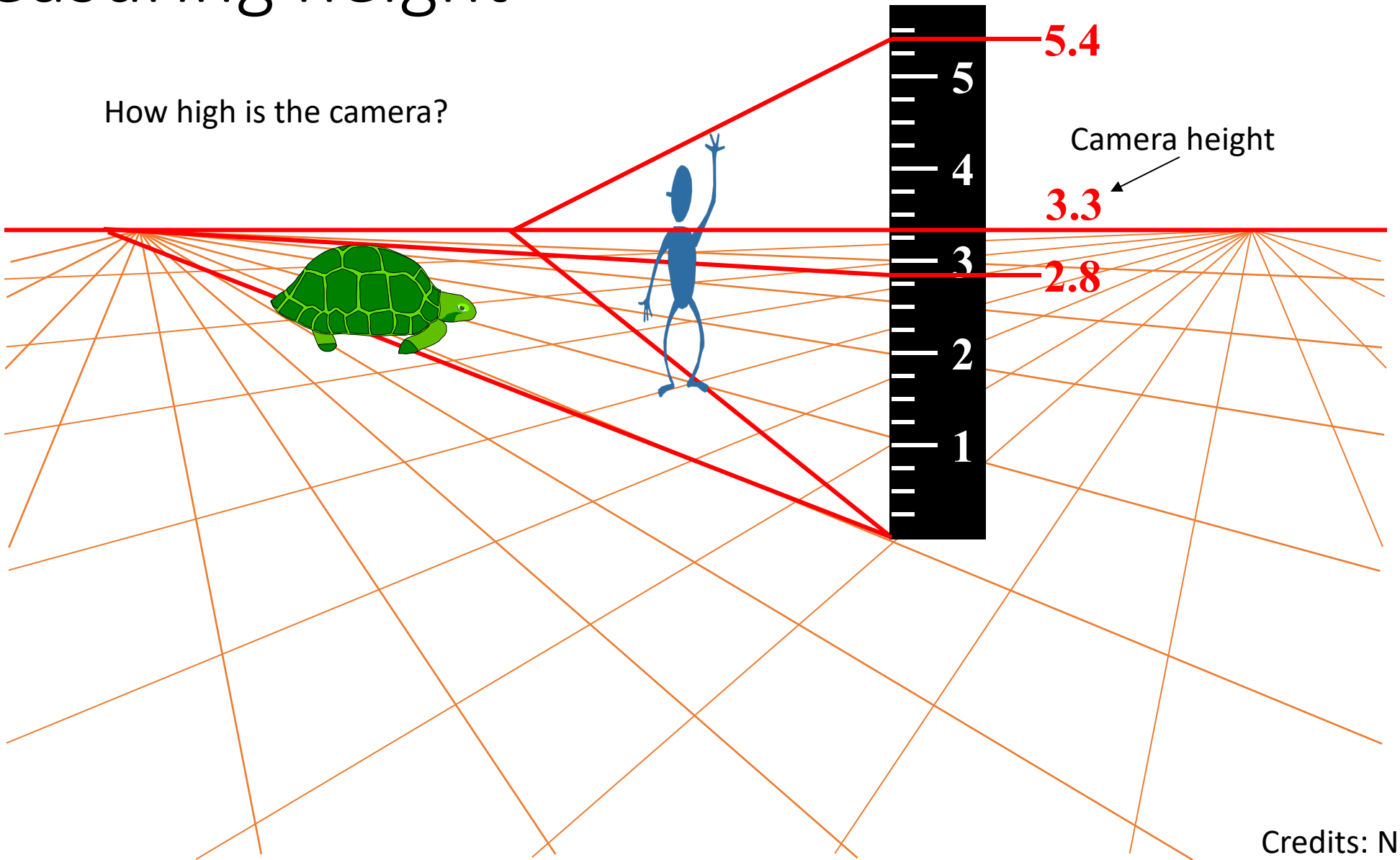
Credits: Noah Snavely

Comparing heights



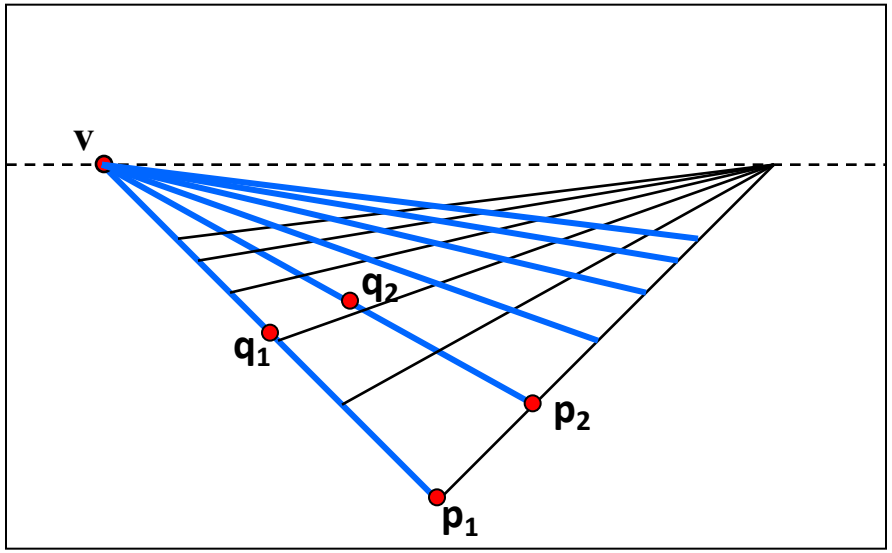
Credits: Noah Snavely

Measuring height



Credits: Noah Snavely

Computing vanishing points (from lines)



Intersect p_1q_1 with p_2q_2

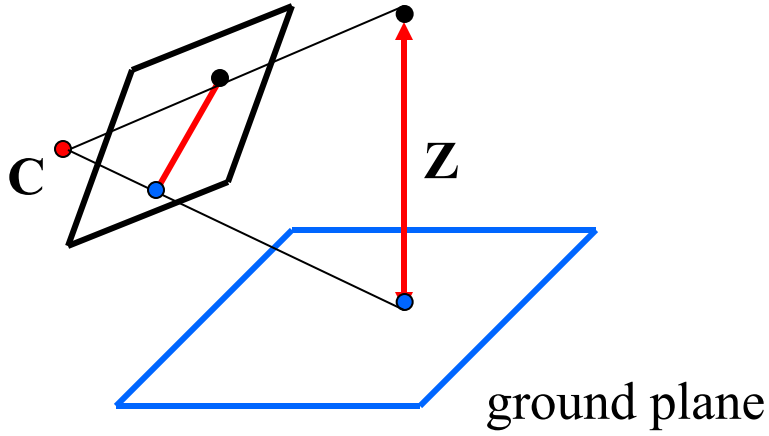
$$v = (p_1 \times q_1) \times (p_2 \times q_2)$$

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

Credits: Noah Snavely

Measuring height without a ruler



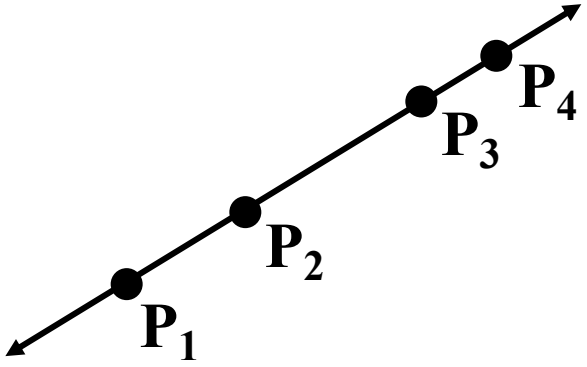
Compute Z from image measurements

- 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



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

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

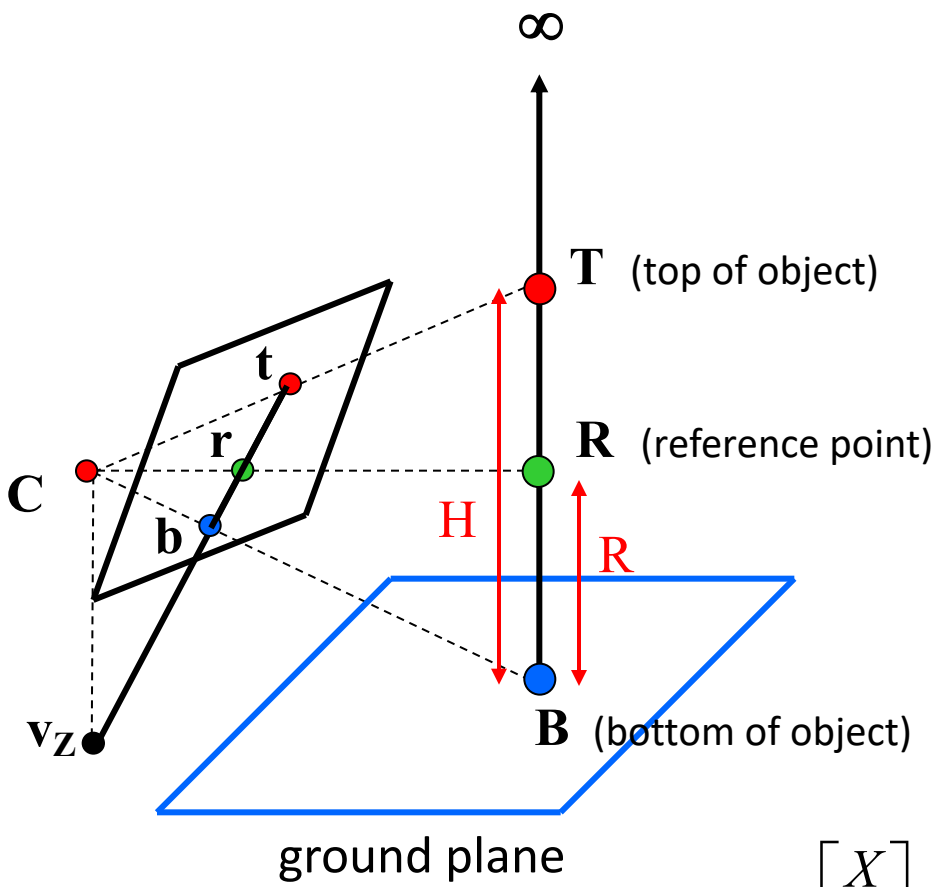
Can permute the point ordering

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

This is the fundamental invariant of projective geometry

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

Measuring height



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

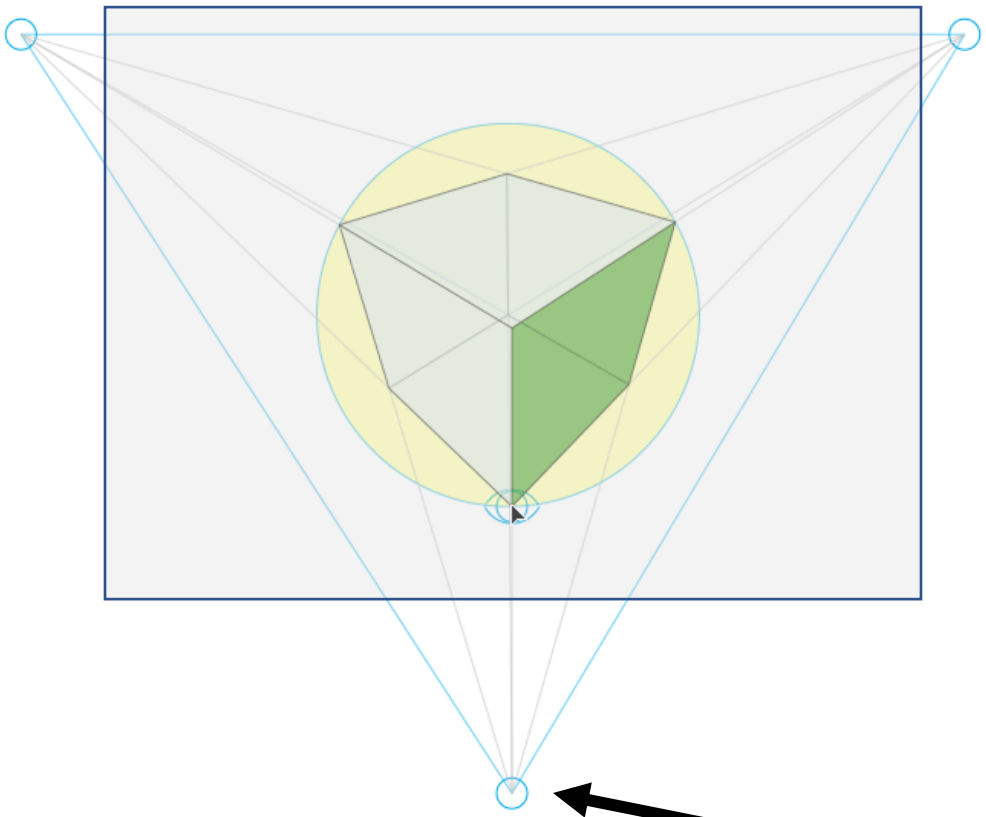
scene cross ratio

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

image cross ratio

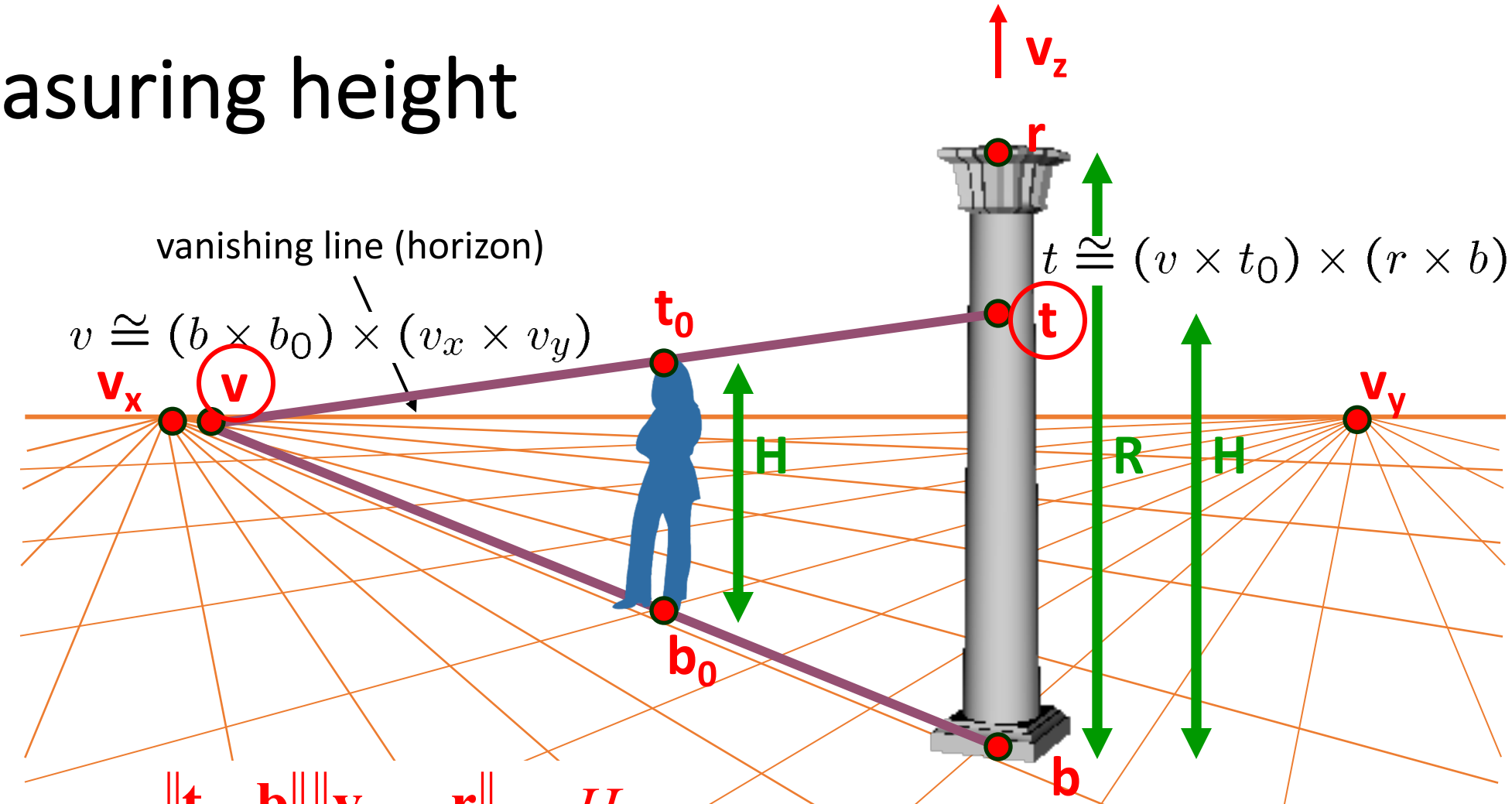
scene points represented as $\mathbf{P}=\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$ image points as $\mathbf{p}=\begin{bmatrix}x\\y\\1\end{bmatrix}$

Finding the vertical (z) vanishing point



Find intersection of projections of vertical lines

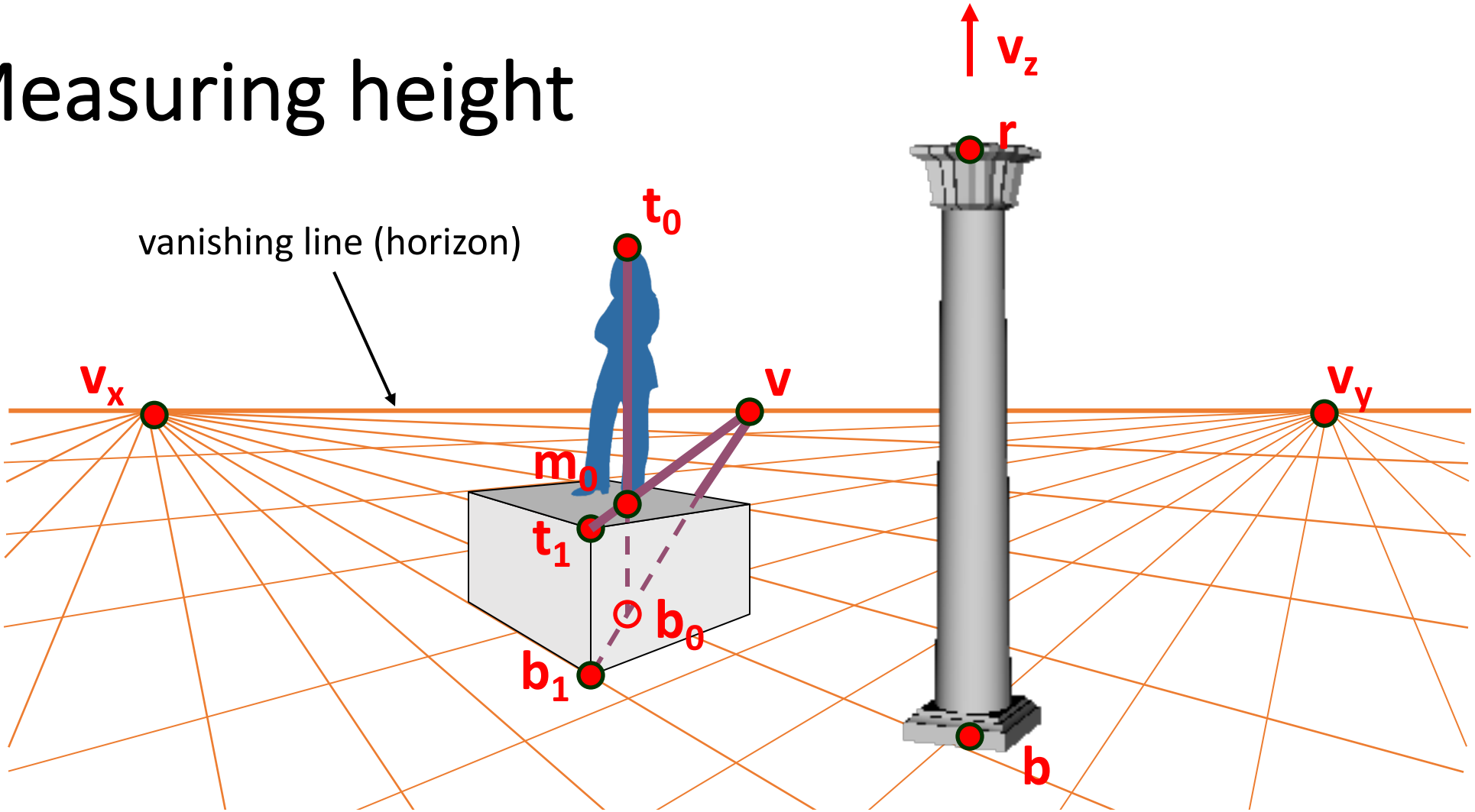
Measuring height



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

image cross ratio

Measuring height



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

- Here the person 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

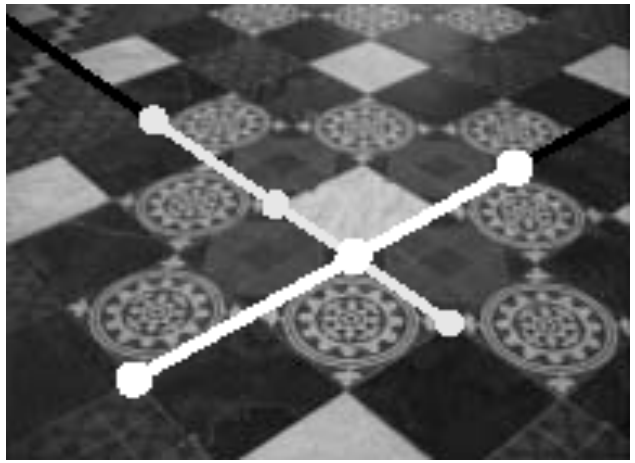
Homography

Projective Transformation

Any mapping of the projective plane to itself that preserves lines is called a “projective transformation”, or “collineation”, or “homography”

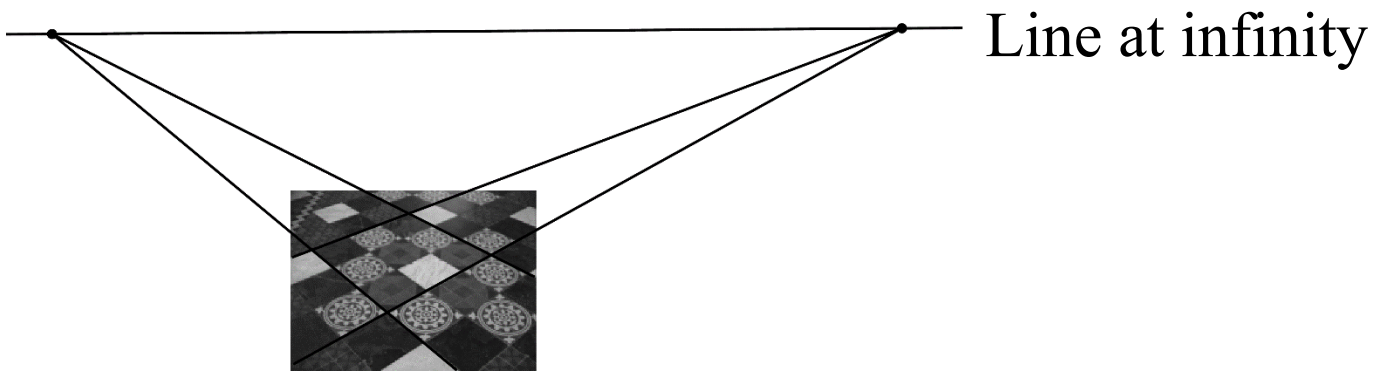
Mapping: A mapping from points in the plane to points in the plane.

Preserves lines: If three points lie on a line, then their corresponding points under the mapping lie on a line.



The Projective Plane $\mathcal{P}^2$

Extend the Euclidean Plane by adding a line at infinity.



Any two lines meet in exactly one point

Homogeneous coordinates

- A point (x, y) on a plane is represented by a 3-vector $(x, y, 1)^\top$.
- All multiples of $(x, y, 1)^\top$ represent the same point. Thus

  $$(x, y, 1)^\top = (2x, 2y, 2)^\top = ... = (kx, ky, k)^\top$$

  for any non-zero k.
- An arbitrary 3-vector $(x, y, w)^\top$ represents the point $(x/w, y/w)$
- Points with $w = 0$ represent points on the "plane at infinity"

Projective space, $\mathcal{P}^2$ and $\mathcal{P}^3$

- The set of all (equivalence classes of) homogeneous $(n+1)$-vectors is known as "projective n-space", $\mathcal{P}^n$.
- Points with last coordinate equal to zero are called "points at infinity".
- Thus, $\mathcal{P}^2 = R^2 \cup \{\text{line at infinity}\}$
- $\mathcal{P}^3 = R^3 \cup \{\text{plane at infinity}\}$

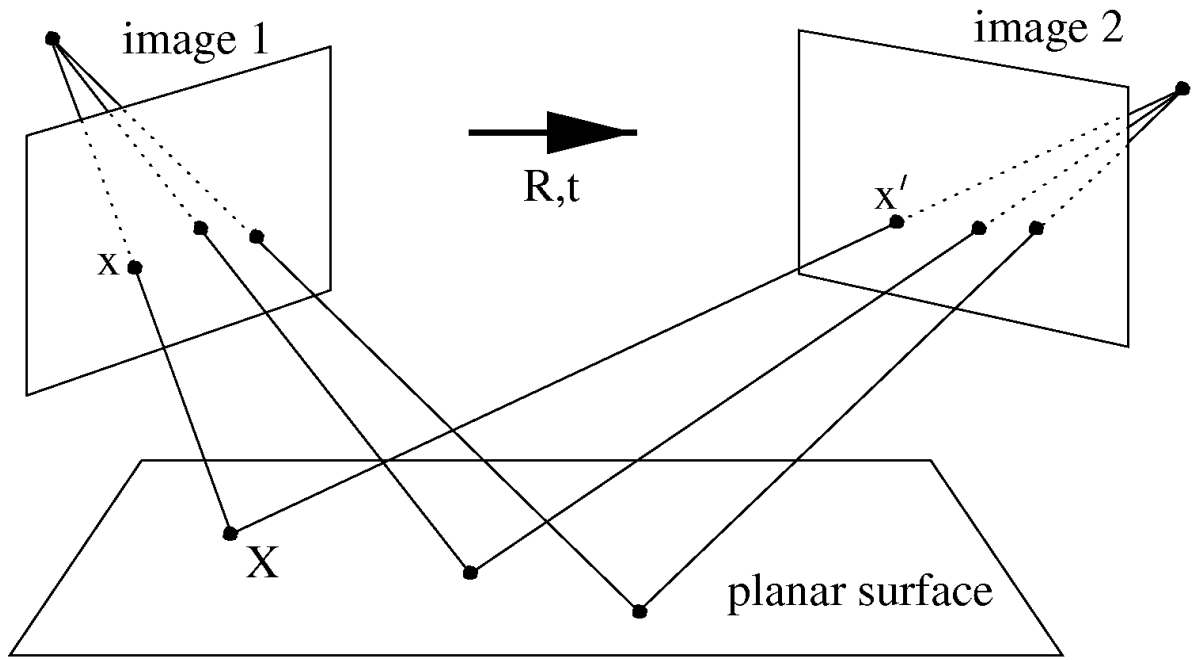
Homography between planes

Definition *Any point-to-point mapping from* $\mathcal{P}^2$ *to* $\mathcal{P}^2$ *that takes lines to lines is called a homography.*

Other names : collineation, projectivity, projective transform.

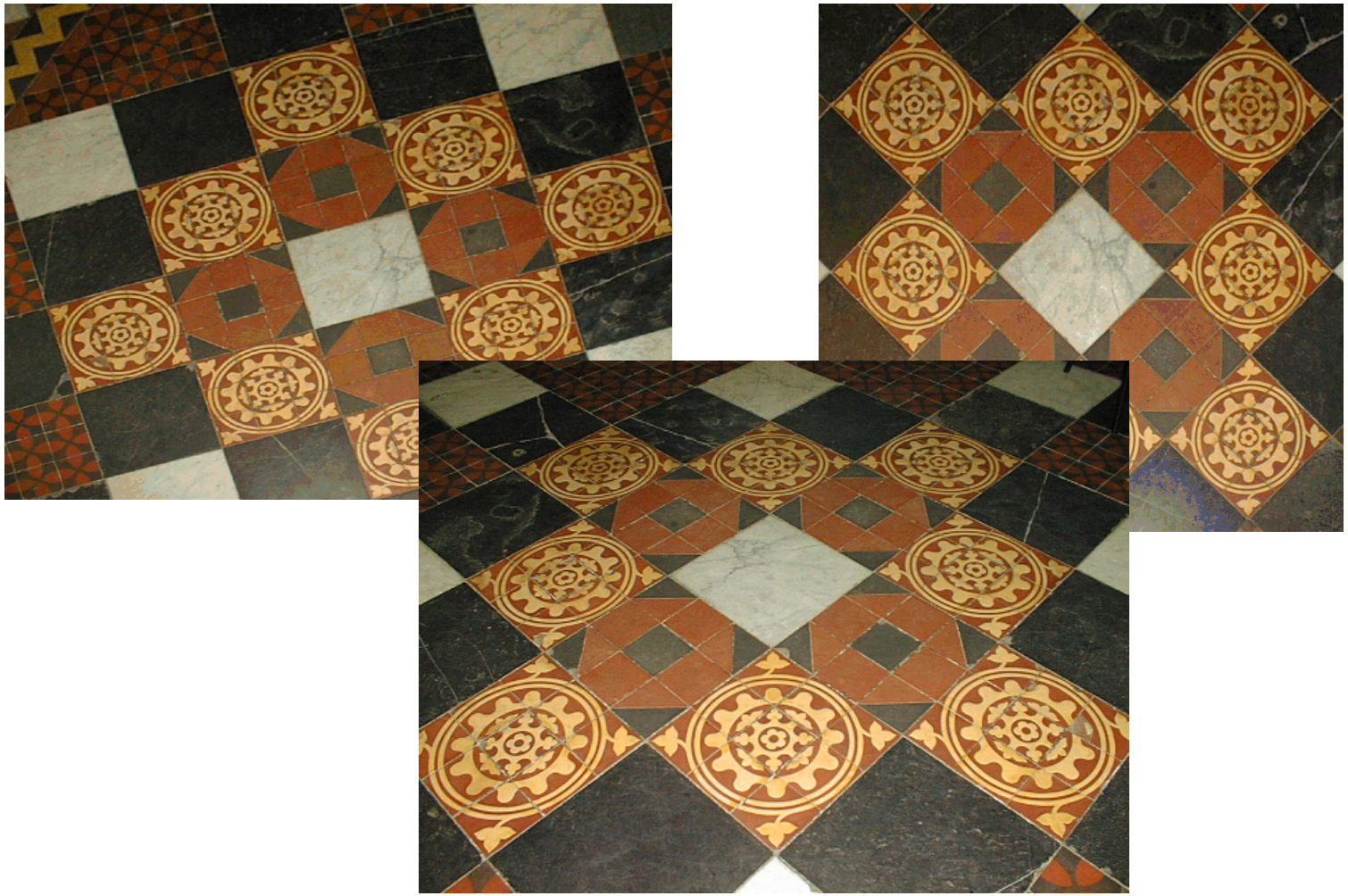
How do 2D homographies arise

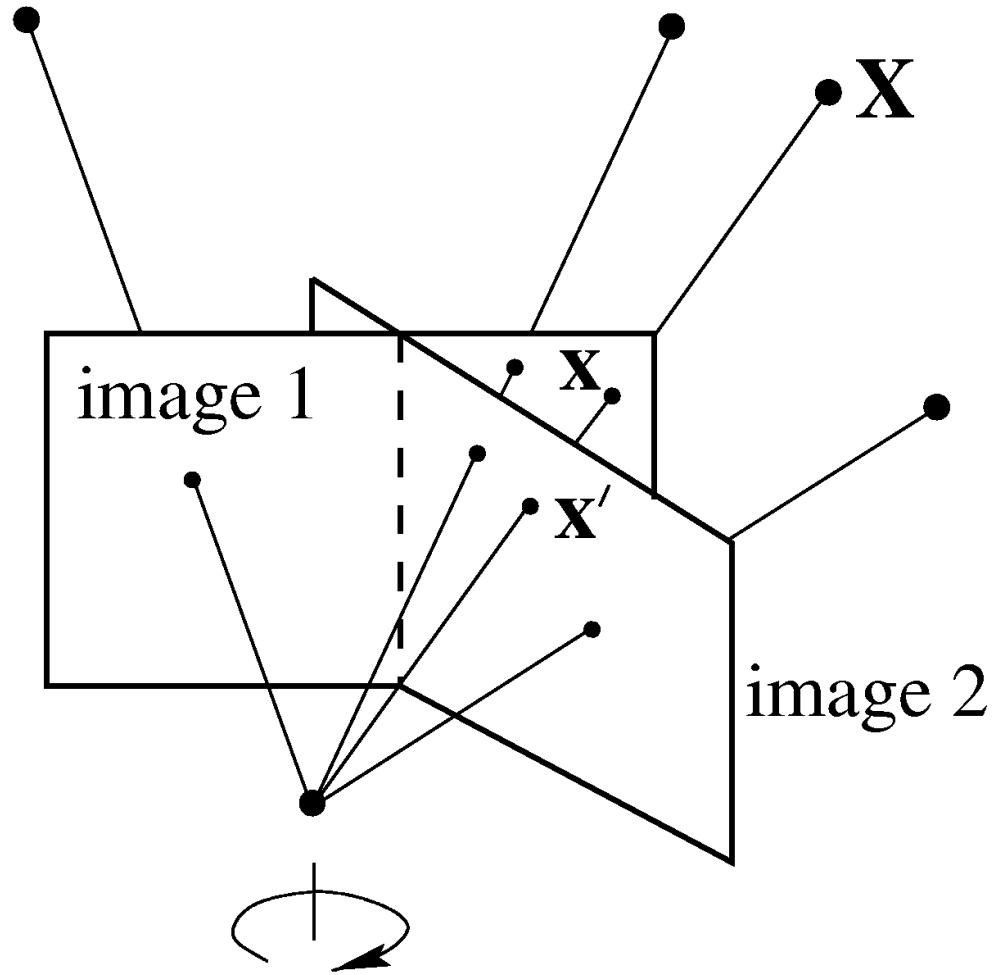
- Between a plane in the world and its image with a perspective camera.
- Between two images of a plane.
- Between two images of the world taken with a rotating (but not moving) camera.



Two images of a plane are related by a homography

Images of floor, related by homographies





Images taken with a rotating camera.

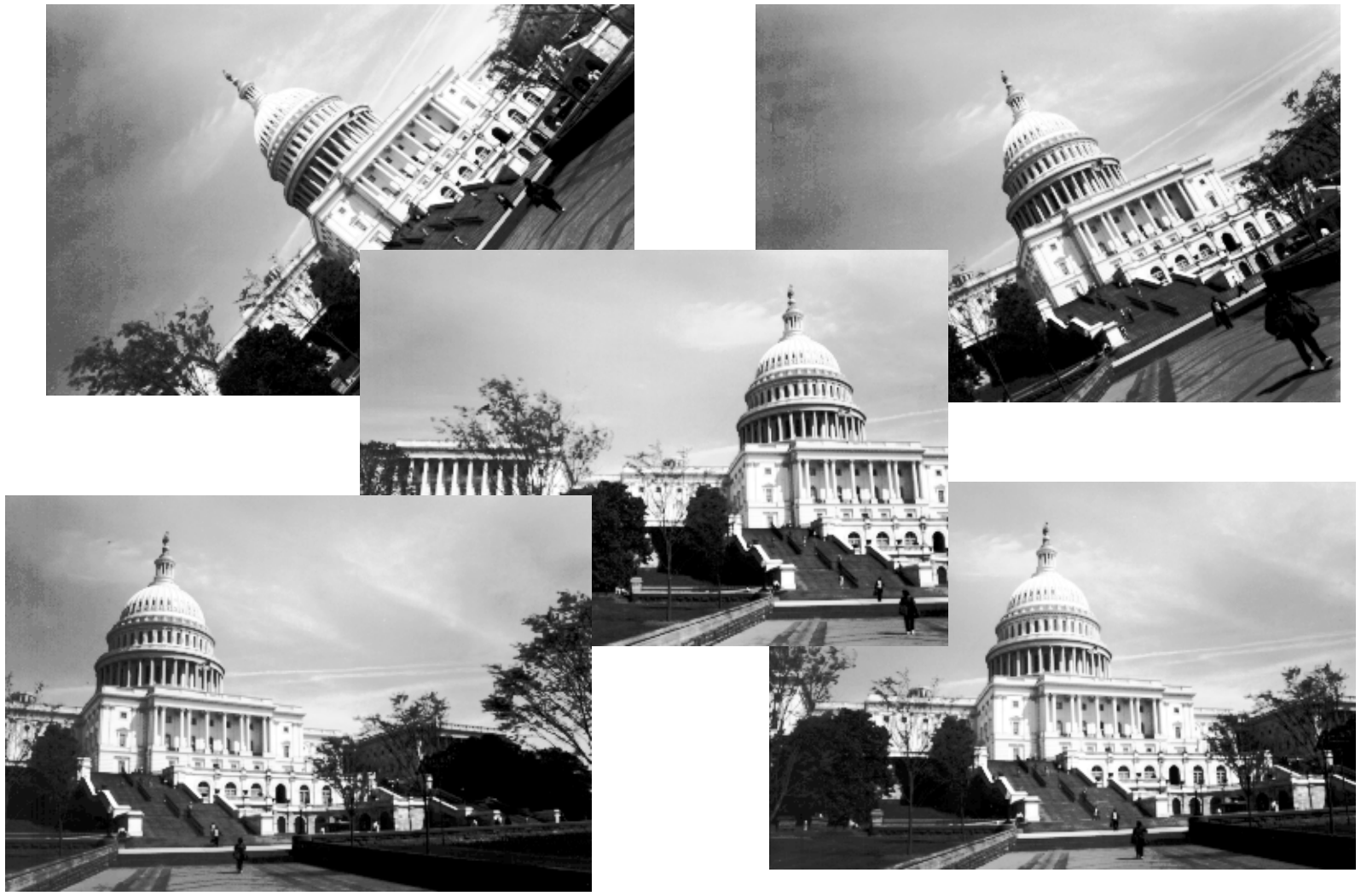
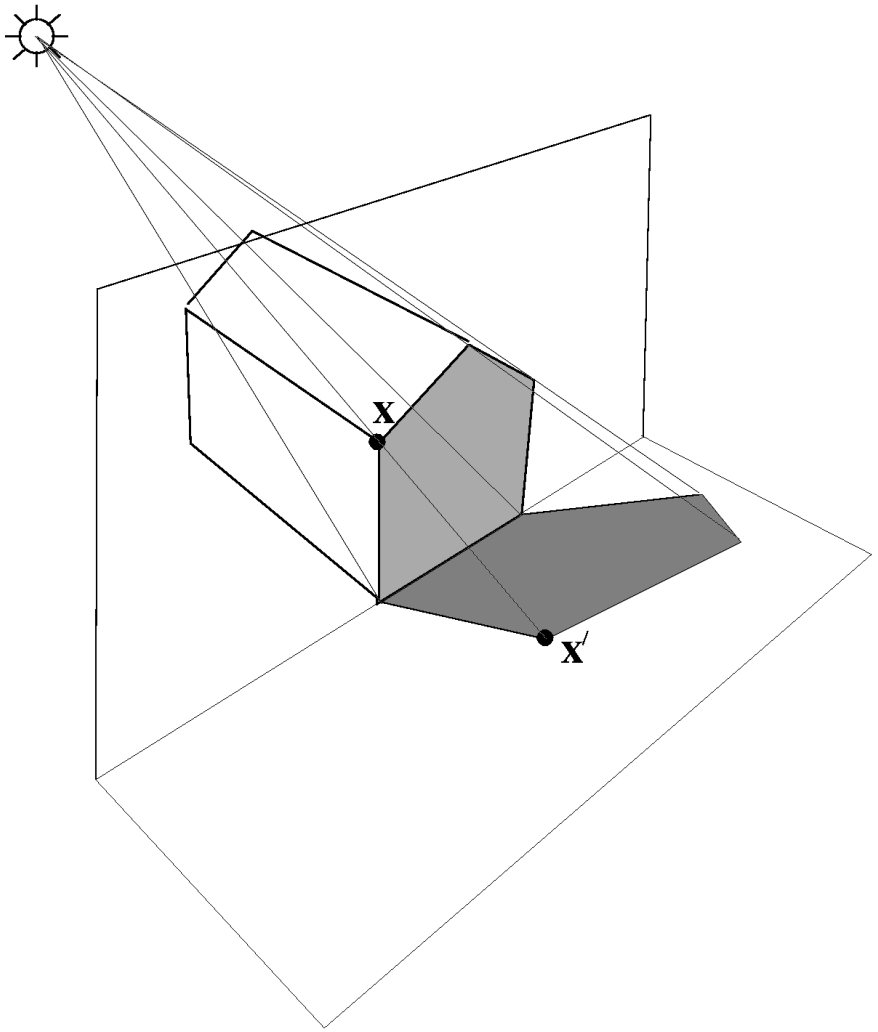


Image taken from the same location

Mosaicking by homographic warping



Images of a planar object and its shadow.

Projective Transformations

A projective transformation is a mapping

$$(x, y) \mapsto (x', y')$$

where

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Homography in non-homogeneous coordinates

- In non-homogeneous coordinates, homography is written as follows:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

- In homogeneous coordinates we write:

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

Homogeneous coordinates

- A point (x, y) on a plane is represented by a 3-vector $(x, y, 1)^\top$.
- All multiples of $(x, y, 1)^\top$ represent the same point. Thus

  $$(x, y, 1)^\top = (2x, 2y, 2)^\top = ... = (kx, ky, k)^\top$$

  for any non-zero k.
- An arbitrary 3-vector $(x, y, w)^\top$ represents the point $(x/w, y/w)$
- Points with $w = 0$ represent points on the "plane at infinity"

Rational Numbers (fractions)

- A rational number is represented by a pair of integers, (p/q).
- Two rational numbers are the same if the pairs differ by a constant multiple. Thus

$$(p/q) \approx (2p/2q) \approx (kp/kq)$$

- Finite rational numbers are (p/q) with $q \neq 0$.
- The fraction (p/q) with $q = 0$ represents ∞.

Algebraic formulation of a homography

- Points in the plane are represented by homogeneous coordinates $\mathbf{x} = (x, y, w)^\top$
- Homography is represented by a 3×3 matrix $\mathtt{H}$

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{23} & h_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

- Thus, homography is just a linear transformation on homogeneous coordinates.

Computation of 2D homography

- Homography H has 8 degrees of freedom (9 entries, but scale irrelevant).
- Each point provides two constraints on H.
- Thus 4 point matches are required to compute H.
- With more than 8 point matches, least-squares techniques are used.
- Computation of homographies is reliable and easy.

Homographies

(ii) We need a transformation:

$$\mathbf{x}' \approx \mathbf{H}\mathbf{x} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

(iii) In non-homogeneous coordinates:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{13}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{13}}$$

(iv) They are transformations of P^2 (the projective plane).

The DLT algorithm – computing a homography

Problem Statement:

Given n correspondences $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$, points in two images.

Compute homography $\mathbf{H}$ such that $\mathbf{x}'_i \approx \mathbf{H}\mathbf{x}_i$.

Each correspondence generates two equations

$$x'_i = \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \qquad y'_i = \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}}$$

Multiplying out gives equations linear in the matrix elements of $\mathbf{H}$

$$\begin{aligned} x'_i(h_{31}x_i + h_{32}y_i + h_{33}) &= h_{11}x_i + h_{12}y_i + h_{13} \\ y'_i(h_{31}x_i + h_{32}y_i + h_{33}) &= h_{21}x_i + h_{22}y_i + h_{23} \end{aligned}$$

These equations can be rearranged as

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y & -x' \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y & -y' \end{pmatrix} \mathbf{h} = \mathbf{0}$$

with $\mathbf{h} = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^\top$ a 9-vector.

Computing homography continued ...

Solving for $\mathbf{H}$

(i) Concatenate the equations from ($n \geq 4$) correspondences to generate $2n$ simultaneous equations, which can be written: $\mathbf{Ah} = \mathbf{0}$, where $\mathbf{A}$ is a $2n \times 12$ matrix.

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

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

$$\min_{\mathbf{H}} \sum_i \left((x_i', y_i') - \mathrm{dehom}(\mathbf{H}(x_i, y_i, 1))\right)^2$$

where

(i) hom is the homogenizing operation $(x, y) \mapsto (x, y, 1)$.

(ii) dehom is the dehomogenizing operation $(x, y, w) \mapsto (x/w, y/w)$.

DLT algorithm

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{x_i \leftrightarrow x_i'\}$, determine the 2D homography matrix H such that $x_i' = Hx_i$

Algorithm

(i) For each correspondence $x_i \leftrightarrow x_i'$ compute A_i. Usually only two first rows needed.

(ii) Assemble n 2x9 matrices A_i into a single $2n$x9 matrix A

(iii) Obtain SVD of A. Solution for h is last column of V

(iv) Determine H from h

Importance of normalization

$$\begin{bmatrix} 0 & 0 & 0 & -x_i & -y_i & -1 & y'_i x_i & y'_i y_i & y'_i \\ x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \end{bmatrix} \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix} = 0$$

Orders of magnitude!

Check Page 110, Multiple view geometry in computer Vision.

Normalized DLT algorithm

Objective

Given n≥4 2D to 2D point correspondences $\{x_i \leftrightarrow x_i'\}$, determine the 2D homography matrix H such that $x_i' = Hx_i$

Algorithm

(i) Normalize points $\tilde{x}_i = T_{norm} x_i, \tilde{x}_i' = T_{norm}' x_i'$

(ii) Apply DLT algorithm to $\tilde{x}_i \leftrightarrow \tilde{x}_i'$,

(iii) Denormalize solution $H = T_{norm}'^{-1} \tilde{H} T_{norm}$

$$T_{norm} = \begin{bmatrix} w+h & 0 & w/2 \\ 0 & w+h & h/2 \\ 0 & 0 & 1 \end{bmatrix}^{-1}$$

Readings

- Chapter 8.6 Vanishing points, vanishing lines
- Chapter 4.1 Homography
- Chapter 4.4 Transformation invariance and normalisation

  In Multiple View Geometry in Computer Vision.