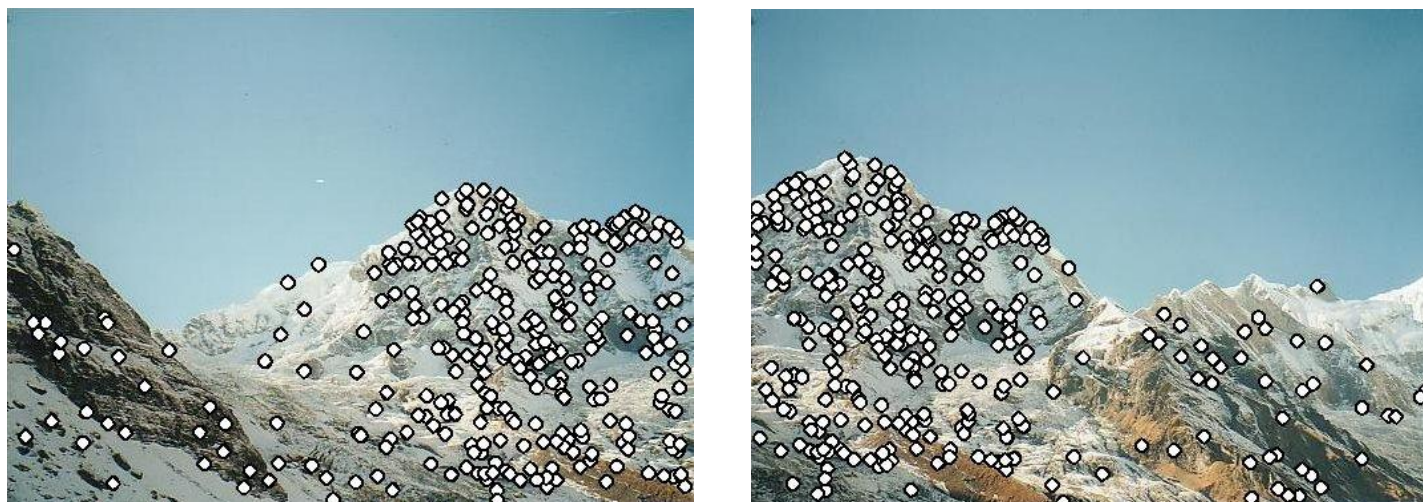


Last Two Lectures



Panoramic Image Stitching



Feature Detection and Matching

Today

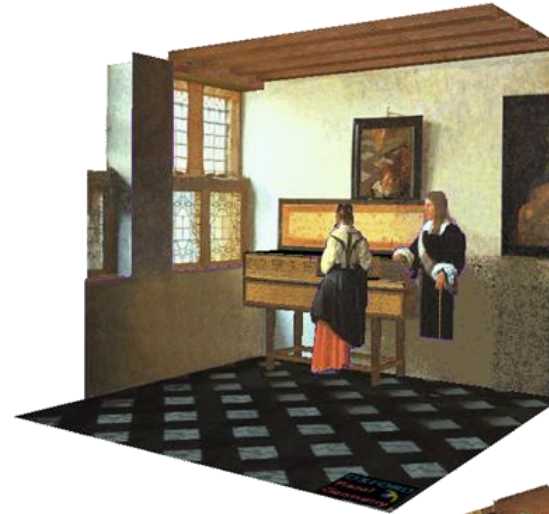
More on Mosaic

Projective Geometry

Single View Modeling

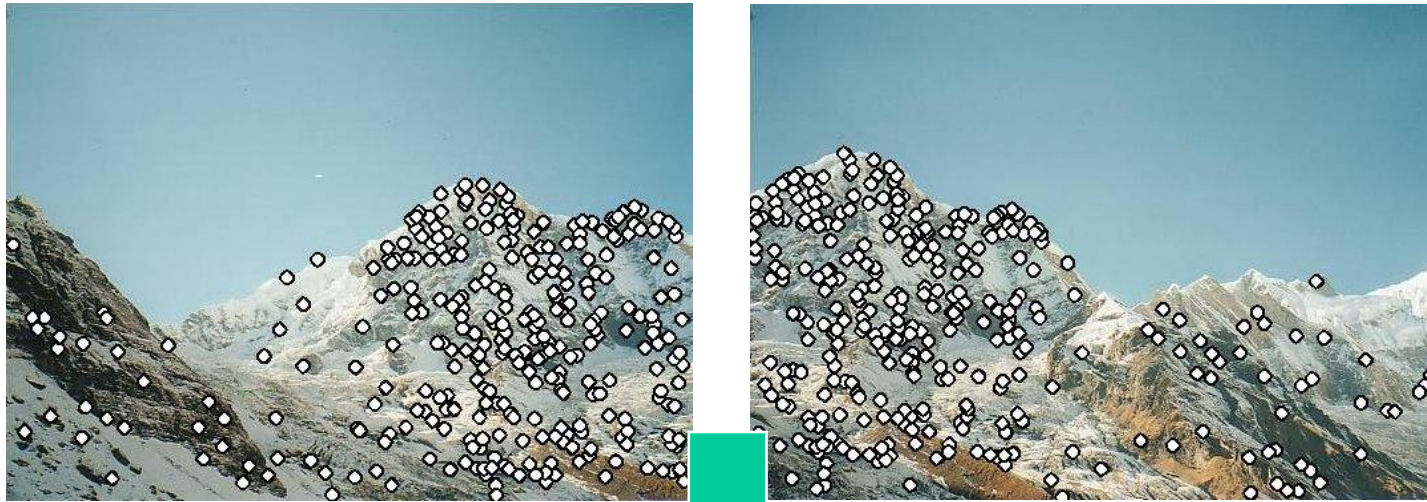


Vermeer's *Music Lesson*



Reconstructions by Criminisi et al.

Image Alignment



Feature Detection and Matching

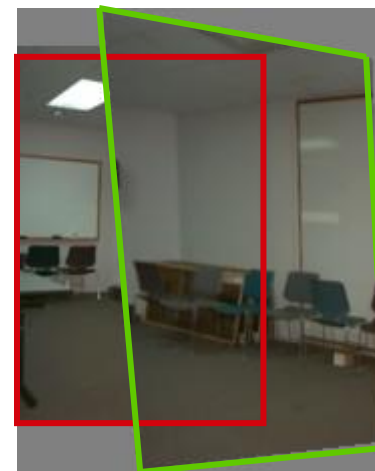


Cylinder:
Translation
2 DoF

Plane:
Homography
8 DoF

Plane perspective mosaics

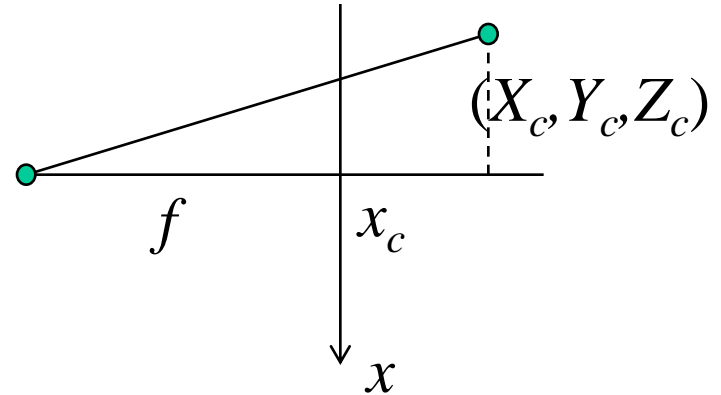
- 8-parameter generalization of affine motion
 - works for pure rotation or planar surfaces
- Limitations:
 - local minima
 - slow convergence



Revisit Homography

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

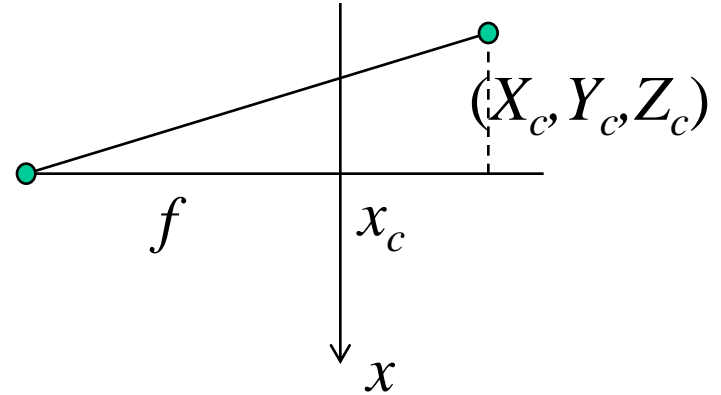
$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$



$$\mathbf{K}\mathbf{R}\mathbf{K}^{-1}\mathbf{x}_1 \sim \mathbf{x}_2$$

Estimate f from H?

$$\begin{pmatrix} x_1 - x_c \\ y_1 - y_c \\ 1 \end{pmatrix} \sim \begin{bmatrix} f_1 & 0 & 0 \\ 0 & f_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$



$$\begin{pmatrix} x_2 - x_c \\ y_2 - y_c \\ 1 \end{pmatrix} \sim \begin{bmatrix} f_2 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

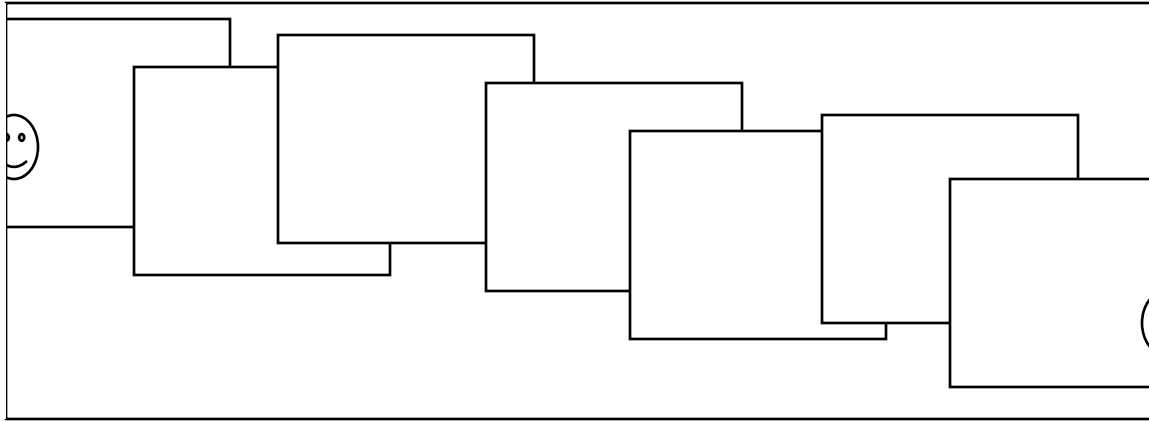
$$\mathbf{R} \sim \mathbf{K}_2^{-1} \mathbf{H} \mathbf{K}_1$$

$$= \begin{bmatrix} a & b & c/f_1 \\ d & e & g/f_1 \\ h^* f_2 & i^* f_2 & j^* \frac{f_2}{f_1} \end{bmatrix}$$

$$\underbrace{(\mathbf{K}_2 \mathbf{R} \mathbf{K}_1^{-1})}_{\mathbf{H}} \mathbf{x}_1 \sim \mathbf{x}_2$$

$$f_1 = ?, f_2 = ?$$

The drifting problem



- Error accumulation
 - small errors accumulate over time

Bundle Adjustment

Associate each image i with \mathbf{K}_i \mathbf{R}_i

Each image i has features \mathbf{p}_{ij}

Trying to minimize total matching residuals

$$E(\text{all } f_i \text{ and } \mathbf{R}_i) = \sum_{(i,m)} \sum_j \left\| \mathbf{p}_{ij} - \mathbf{K}_i \mathbf{R}_i \mathbf{R}_m^{-1} \mathbf{K}_m^{-1} \mathbf{p}_{mj} \right\|^2$$

Rotations

- How do we represent rotation matrices?

1. Axis / angle (\mathbf{n}, θ)

$$\mathbf{R} = \mathbf{I} + \sin\theta [\mathbf{n}]_{\times} + (1 - \cos\theta) [\mathbf{n}]_{\times}^2$$

(Rodriguez Formula), with

$[\mathbf{n}]_{\times}$ be the cross product matrix.

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_{\times} \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Incremental rotation update

1. Small angle approximation

$$\Delta \mathbf{R} = \mathbf{I} + \sin\theta [\mathbf{n}]_{\times} + (1 - \cos\theta) [\mathbf{n}]_{\times}^2$$

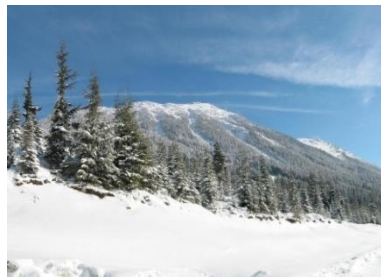
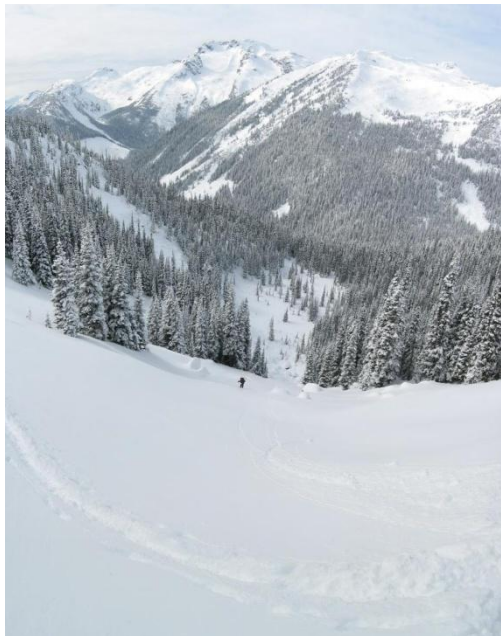
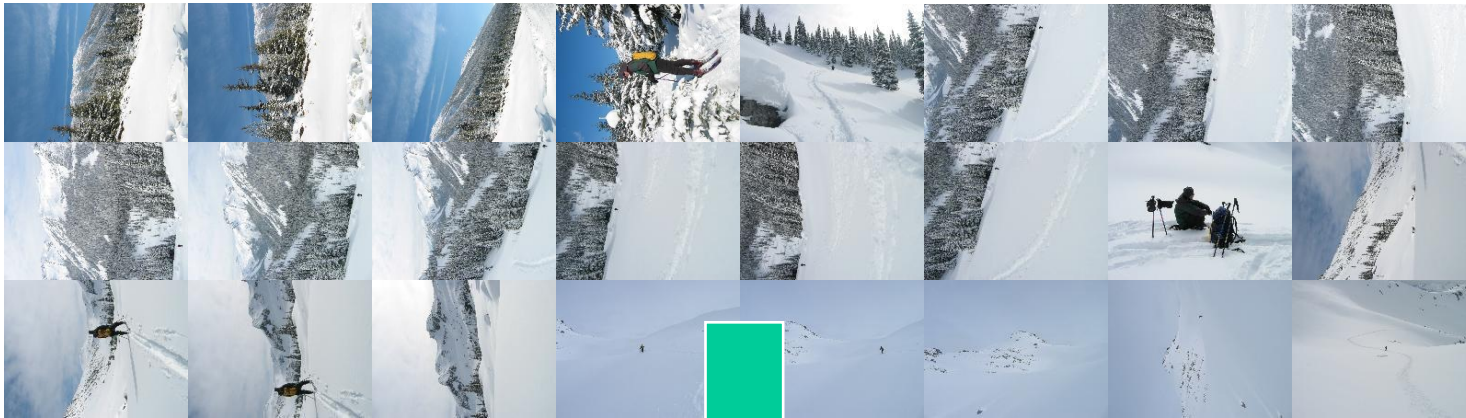
$$\approx \mathbf{I} + \theta [\mathbf{n}]_{\times} = \mathbf{I} + [\boldsymbol{\omega}]_{\times}$$

linear in $\boldsymbol{\omega} = \theta \mathbf{n}$

2. Update original \mathbf{R} matrix

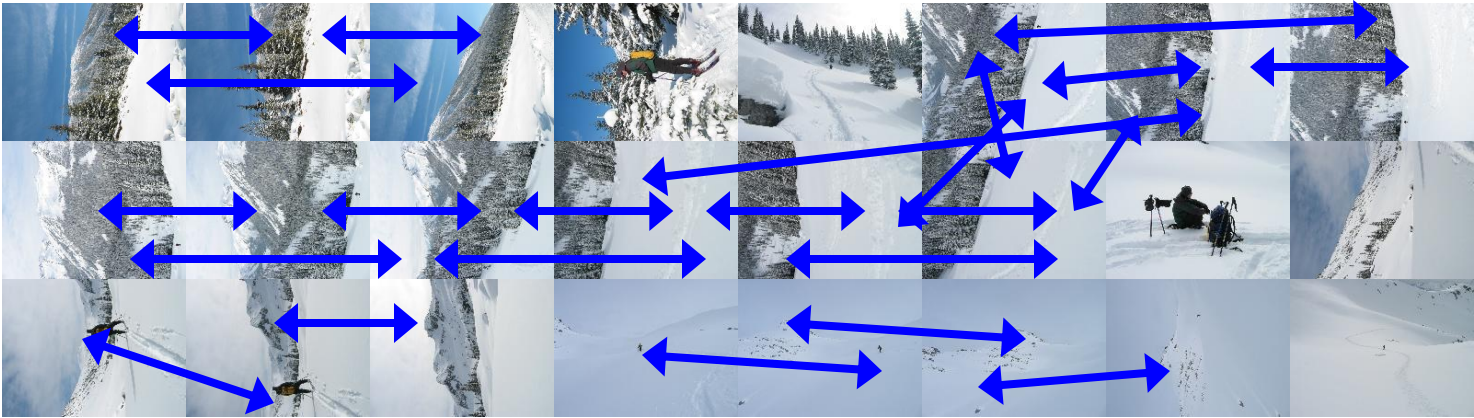
$$\mathbf{R} \leftarrow \mathbf{R} \Delta \mathbf{R}$$

Recognizing Panoramas

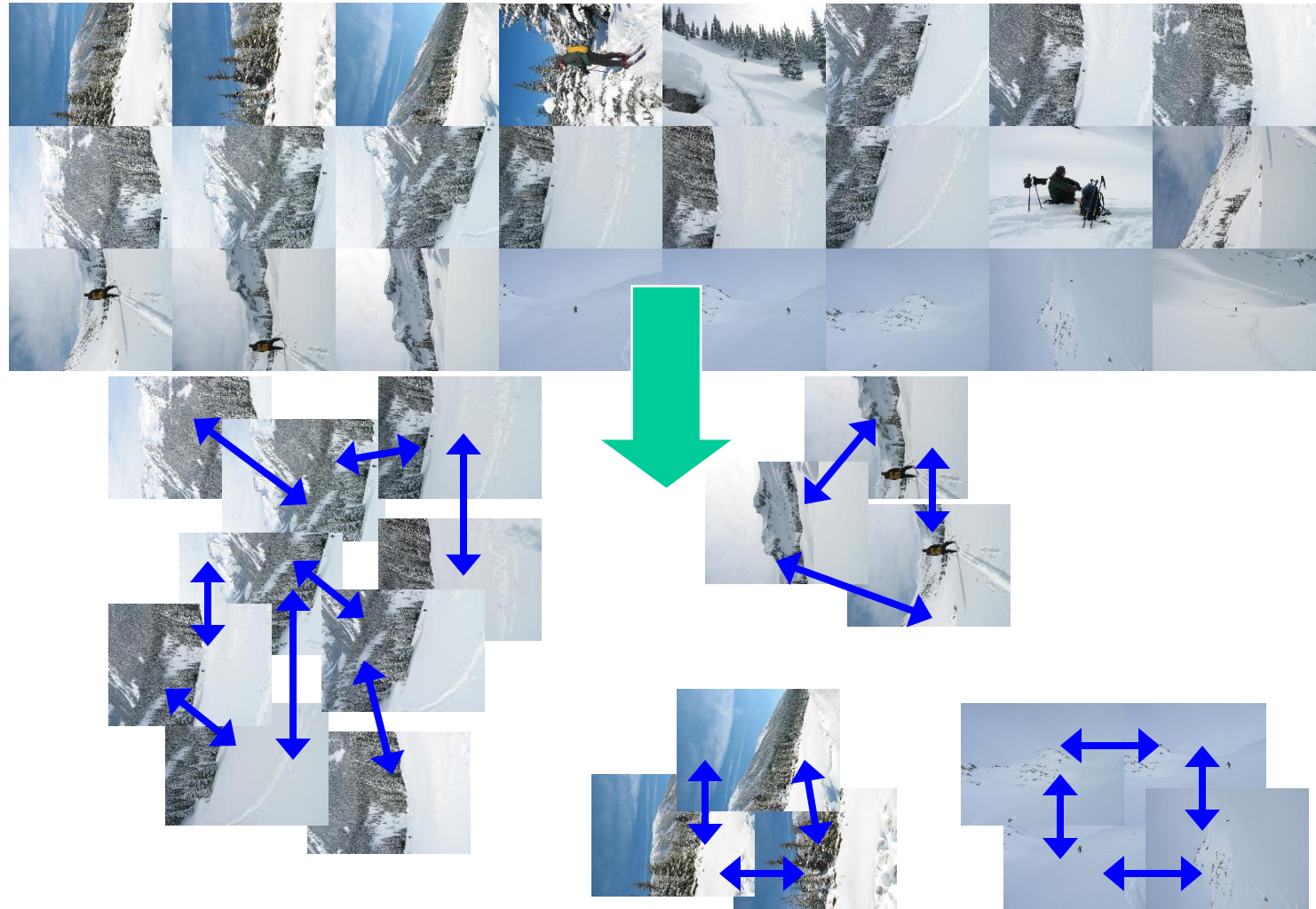


[Brown & Lowe,
ICCV'03]

Finding the panoramas



Finding the panoramas



Algorithm overview

Algorithm: Panoramic Recognition

Input: n unordered images

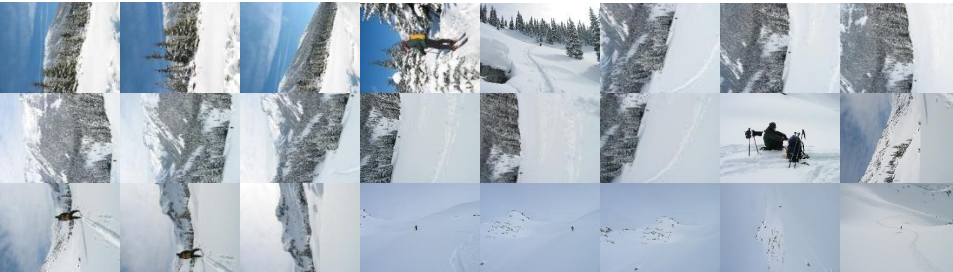


Algorithm overview

Algorithm: Panoramic Recognition

Input: n unordered images

- I. Extract SIFT features from all n images

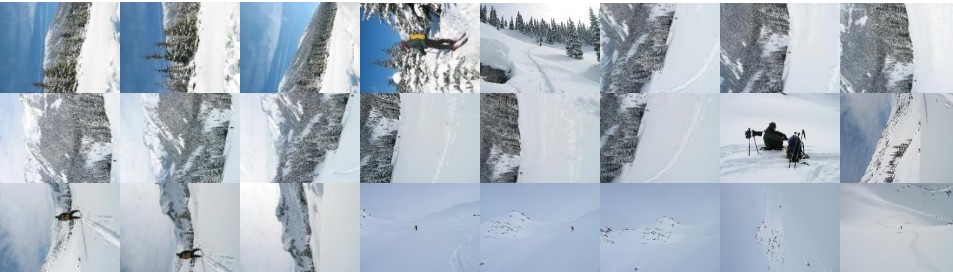


Algorithm overview

Algorithm: Panoramic Recognition

Input: n unordered images

- I. Extract SIFT features from all n images
- II. Find k nearest-neighbours for each feature using a k-d tree



Algorithm overview

Algorithm: Panoramic Recognition

Input: n unordered images

- I. Extract SIFT features from all n images
- II. Find k nearest-neighbours for each feature using a k-d tree
- III. For each image:
 - (i) Select m candidate matching images (with the maximum number of feature matches to this image)



Algorithm overview

Algorithm: Panoramic Recognition

Input: n unordered images

- I. Extract SIFT features from all n images
- II. Find k nearest-neighbours for each feature using a k-d tree
- III. For each image:
 - (i) Select m candidate matching images (with the maximum number of feature matches to this image)
 - (ii) Find geometrically consistent feature matches using RANSAC to solve for the homography between pairs of images

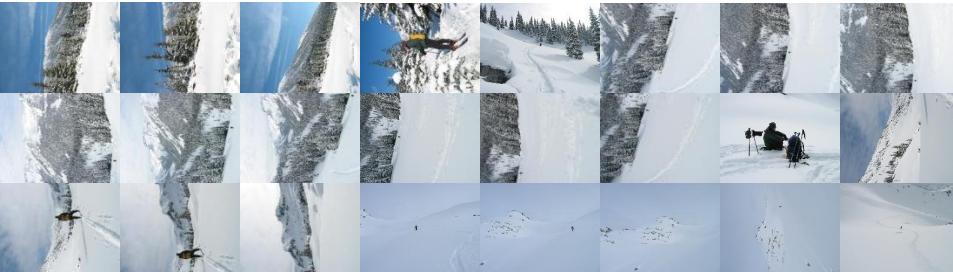


Algorithm overview

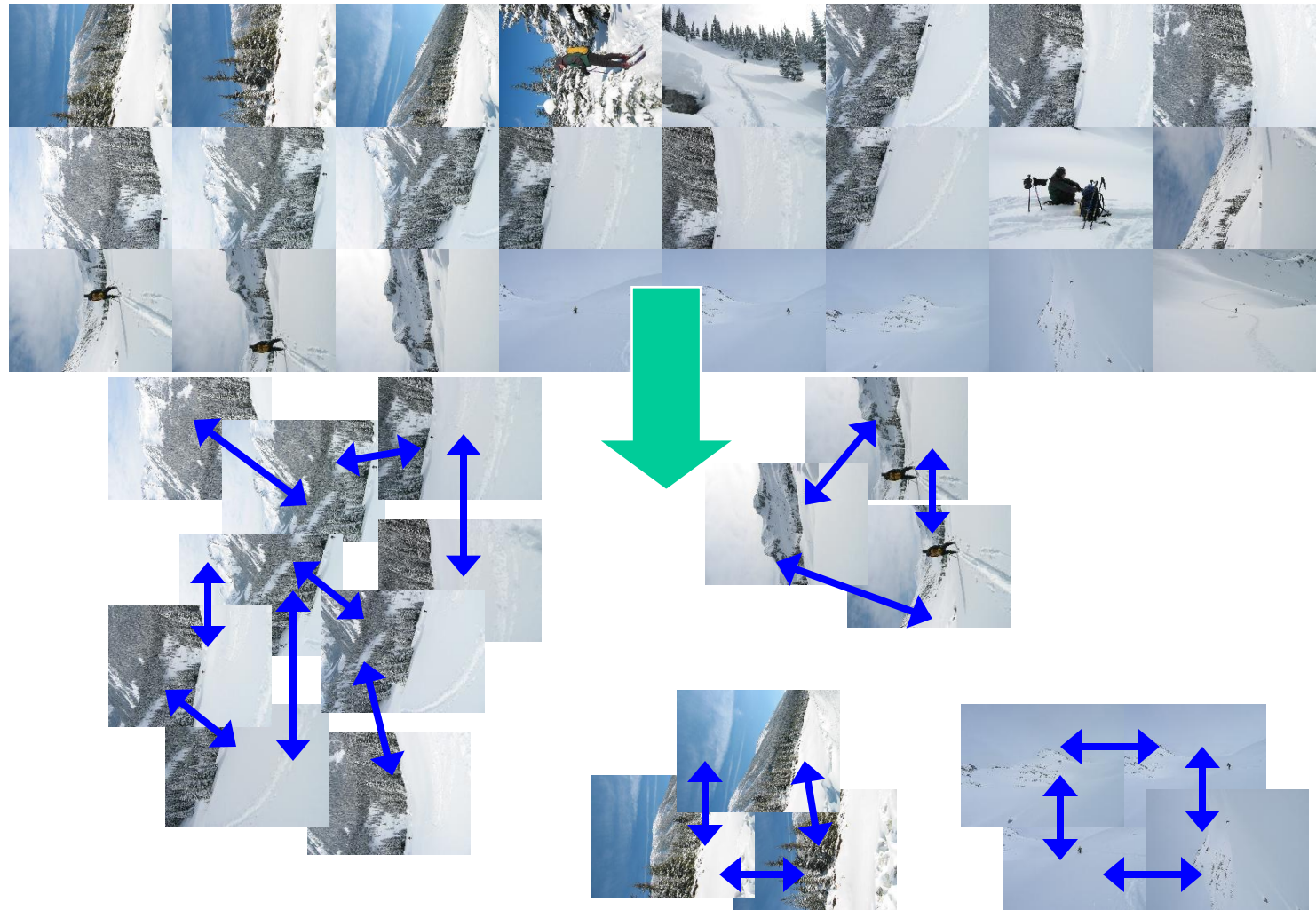
Algorithm: Panoramic Recognition

Input: n unordered images

- I. Extract SIFT features from all n images
- II. Find k nearest-neighbours for each feature using a k-d tree
- III. For each image:
 - (i) Select m candidate matching images (with the maximum number of feature matches to this image)
 - (ii) Find geometrically consistent feature matches using RANSAC to solve for the homography between pairs of images
 - (iii) Verify image matches using probabilistic model
- IV. Find connected components of image matches



Finding the panoramas



Finding the panoramas



Algorithm overview

Algorithm: Panoramic Recognition

Input: n unordered images

- I. Extract SIFT features from all n images
- II. Find k nearest-neighbours for each feature using a k-d tree
- III. For each image:
 - (i) Select m candidate matching images (with the maximum number of feature matches to this image)
 - (ii) Find geometrically consistent feature matches using RANSAC to solve for the homography between pairs of images
 - (iii) Verify image matches using probabilistic model
- IV. Find connected components of image matches

Algorithm overview

Algorithm: Panoramic Recognition

Input: n unordered images

- I. Extract SIFT features from all n images
- II. Find k nearest-neighbours for each feature using a k-d tree
- III. For each image:
 - (i) Select m candidate matching images (with the maximum number of feature matches to this image)
 - (ii) Find geometrically consistent feature matches using RANSAC to solve for the homography between pairs of images
 - (iii) Verify image matches using probabilistic model
- IV. Find connected components of image matches
- V. For each connected component:
 - (i) Perform bundle adjustment to solve for the rotation $\theta_1, \theta_2, \theta_3$ and focal length f of all cameras

Algorithm overview

Algorithm: Panoramic Recognition

Input: n unordered images

- I. Extract SIFT features from all n images
- II. Find k nearest-neighbours for each feature using a k-d tree
- III. For each image:
 - (i) Select m candidate matching images (with the maximum number of feature matches to this image)
 - (ii) Find geometrically consistent feature matches using RANSAC to solve for the homography between pairs of images
 - (iii) Verify image matches using probabilistic model
- IV. Find connected components of image matches
- V. For each connected component:
 - (i) Perform bundle adjustment to solve for the rotation $\theta_1, \theta_2, \theta_3$ and focal length f of all cameras
 - (ii) Render panorama using multi-band blending

Output: Panoramic image(s)

Get you own copy!

The screenshot shows the Amazon.com website interface in Microsoft Internet Explorer. The browser's address bar displays the URL: <http://www.amazon.com/exec/obidos/tg/de>. The page features the Amazon logo and navigation links such as "Shop in Musical Instruments", "Matt's : Store | Account | Wish List | Cart | Help", and various category buttons like "WELCOME", "MATT'S STORE", "BOOKS", "APPAREL & ACCESSORIES", "ELECTRONICS", "TOYS & GAMES", "KITCHEN & HOUSEWARES", "MAGAZINE SUBSCRIPTIONS", and "SOFTWARE". A promotional banner for T-Mobile offers a \$250 discount on a Motorola V600. The main product listing is for "Microsoft Picture It! Digital Image Pro 10.0". The product image shows a woman's face. The price is listed as \$78.99, with a list price of \$89.99 and a savings of \$11.00 (12%). The availability is "Usually ships within 24 hours". A "READY TO BUY?" section includes an "Add to Shopping Cart" button and a "Sign in" link. A "MORE BUYING CHOICES" section lists alternative products like "J&R Music and Computer World" and "Office Depot".

Amazon.com: Software: Microsoft Picture It! Digital Image Pro 10.0 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Address <http://www.amazon.com/exec/obidos/tg/de>

Shop in Musical Instruments (Beta-What is this?) amazon.com. Matt's : Store | Account | Wish List | Cart | Help

WELCOME MATT'S STORE BOOKS APPAREL & ACCESSORIES ELECTRONICS TOYS & GAMES KITCHEN & HOUSEWARES MAGAZINE SUBSCRIPTIONS SOFTWARE SEE MORE STORES

BROWSE BRANDS & PRODUCTS TOP SELLERS NEW & FUTURE RELEASES CHILDREN'S SOFTWARE GAMES SOFTWARE DOWNLOADS TODAY'S DEALS OUTLET

T-Mobile authorized dealer Save \$250 on the powerful Motorola V600 from T-Mobile Shop now

SEARCH Software GO!

Microsoft Picture It! Digital Image Pro 10.0 Other products by Microsoft

List Price: \$89.99 Price: \$78.99 You Save: \$11.00 (12%) Availability: Usually ships within 24 hours

Want it delivered Thursday, August 19? Order it in the next 22 hours and 45 minutes, and choose One-Day Shipping at checkout.

See details.

This item ships for FREE with Super Saver Shipping. See details.

8 used & new from \$69.99

Based on customer purchases, this is the #24 Early Adopter Product in Software.

READY TO BUY? Amazon.com Price: \$78.99 Availability: Usually ships in 24 hours Add to Shopping Cart or Sign in to turn on 1-Click ordering.

MORE BUYING CHOICES J&R Music and Computer World Price: \$79.88 Availability: Usually ships in 1-2 business days Add to Cart Office Depot Price: \$88.95 Availability: Usually ships in 1-2 business days Add to Cart

[Brown & Lowe, ICCV 2003]

[Brown, Szeliski, Winder, CVPR'05]

How well does this work?

Test on 100s of examples...

How well does this work?

Test on 100s of examples...

...still too many failures (5-10%)
for consumer application

Matching Mistakes: False Positive



Matching Mistakes: False Positive



Matching Mistakes: False Negative

- Moving objects: large areas of disagreement



Matching Mistakes

- Accidental alignment
 - repeated / similar regions
- Failed alignments
 - moving objects / parallax
 - low overlap
 - “feature-less” regions
- No 100% reliable algorithm?



How can we fix these?

- Tune the feature detector
- Tune the feature matcher (cost metric)
- Tune the RANSAC stage (motion model)
- Tune the verification stage
- Use “higher-level” knowledge
 - e.g., typical camera motions
- → Sounds like a big “learning” problem
 - Need a large training/test data set (panoramas)

on to 3D...

Enough of images!

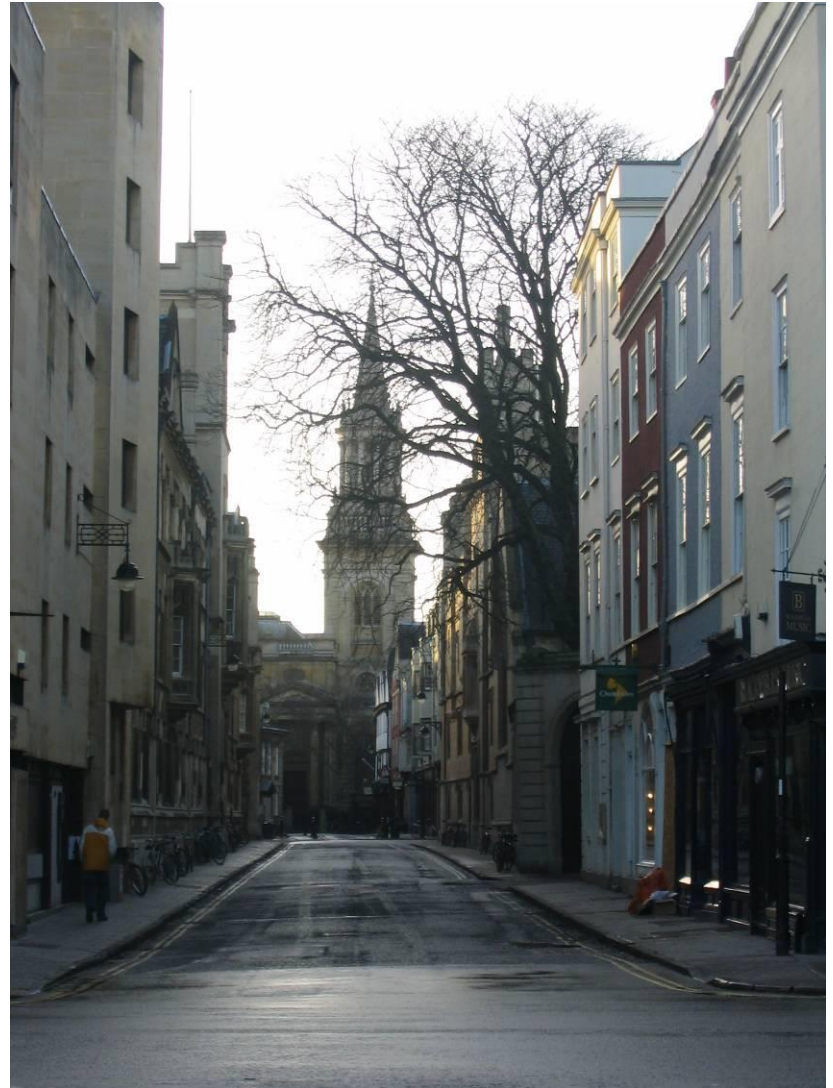
We want more
from the image

We want real 3D
scene

walk-throughs:

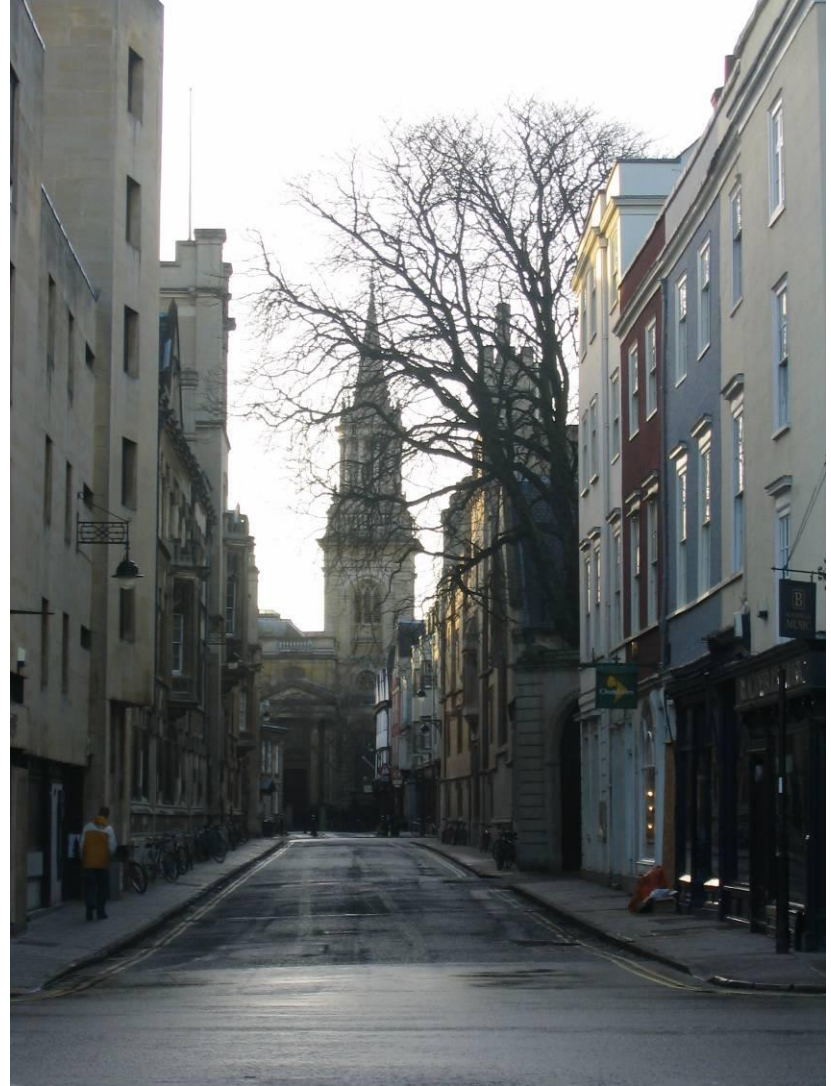
Camera rotation

Camera
translation



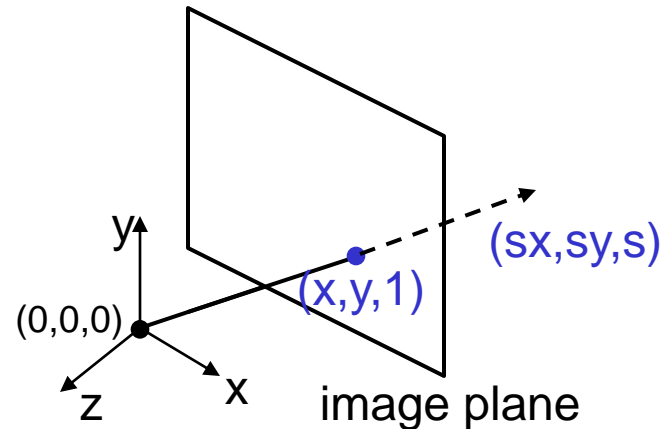
So, what can we do here?

- Model the scene as a set of planes!



The projective plane

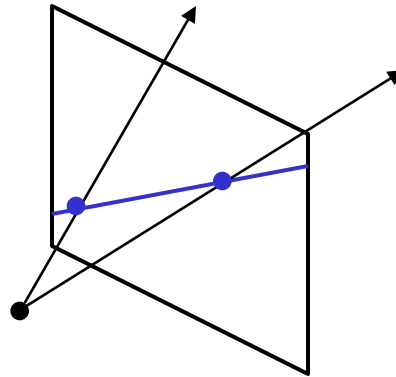
- Why do we need homogeneous coordinates?
 - represent points at infinity, homographies, perspective projection, multi-view relationships
- What is the geometric intuition?
 - a point in the image is a *ray* in projective space



- 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: $ax + by + cz = 0$

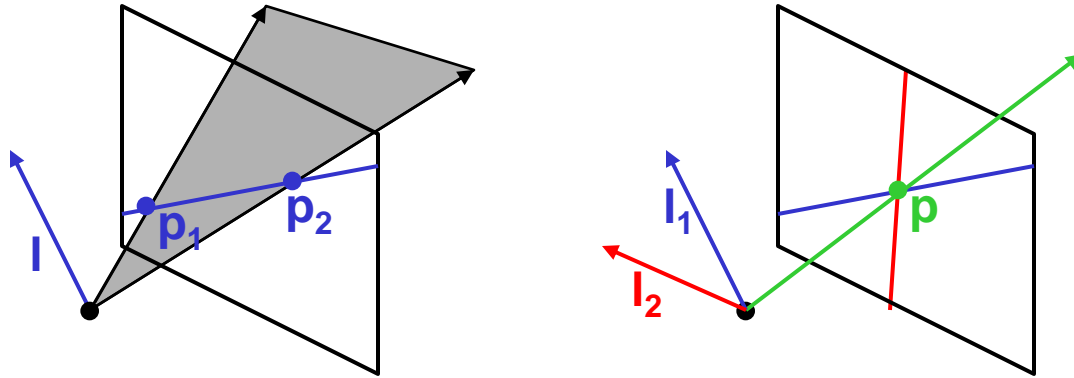
in vector notation :

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

- A line is also represented as a homogeneous 3-vector \mathbf{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

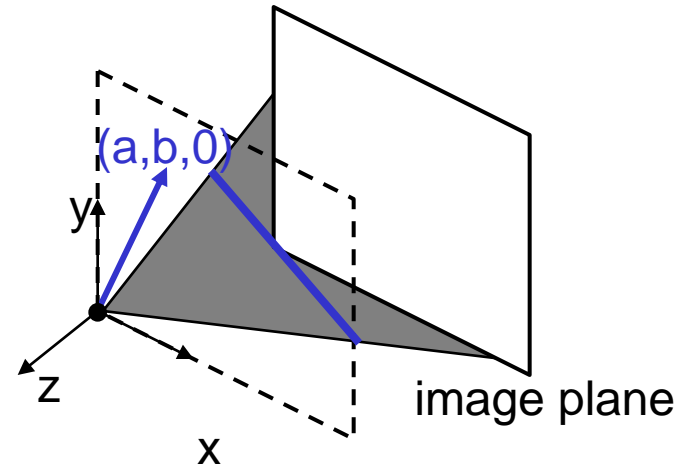
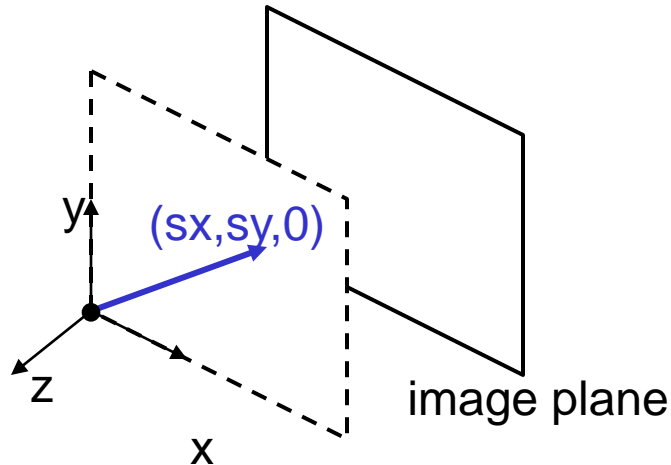
What is the intersection of two lines \mathbf{l}_1 and \mathbf{l}_2 ?

- \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

- 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 postmultiplication 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 4D: $\mathbf{N} \mathbf{P} = 0$
 - Projective transformations
 - Represented by 4x4 matrices \mathbf{T} : $\mathbf{P}' = \mathbf{T} \mathbf{P}$, $\mathbf{N}' = \mathbf{N} \mathbf{T}^{-1}$

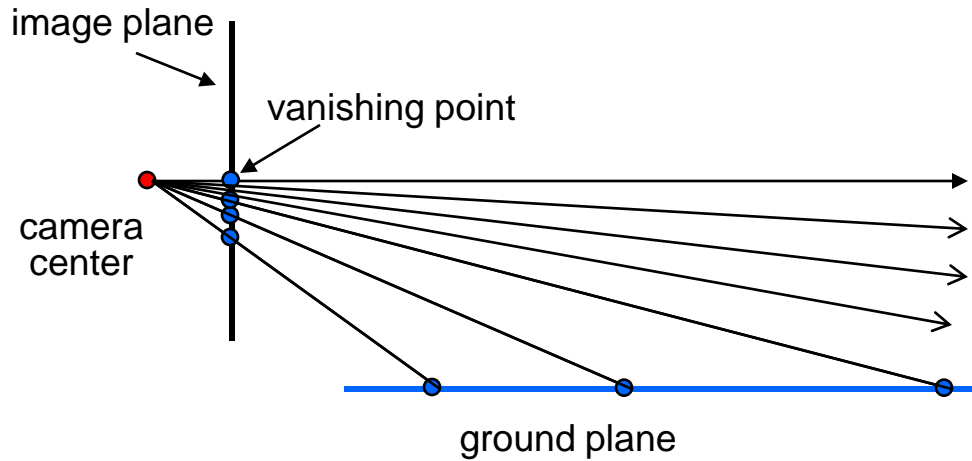
3D to 2D: “perspective” projection

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

What is *not* preserved under perspective projection?

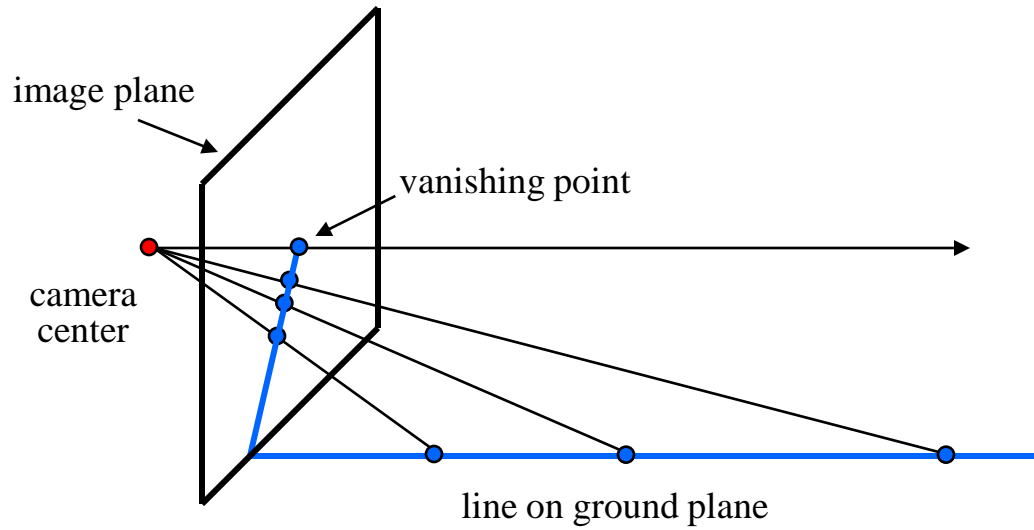
What IS preserved?

Vanishing points

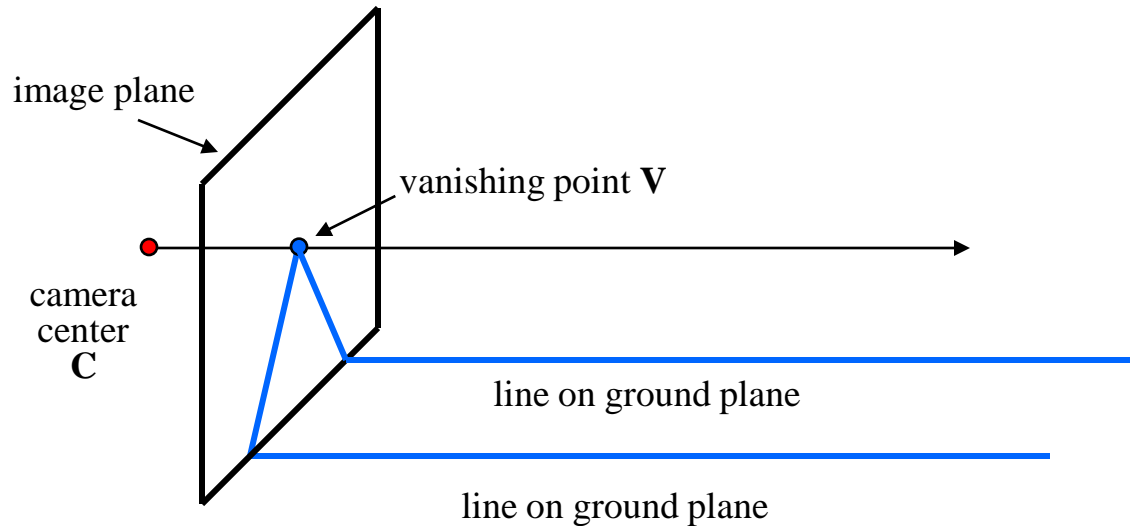


- Vanishing point
 - projection of a point at infinity

Vanishing points (2D)

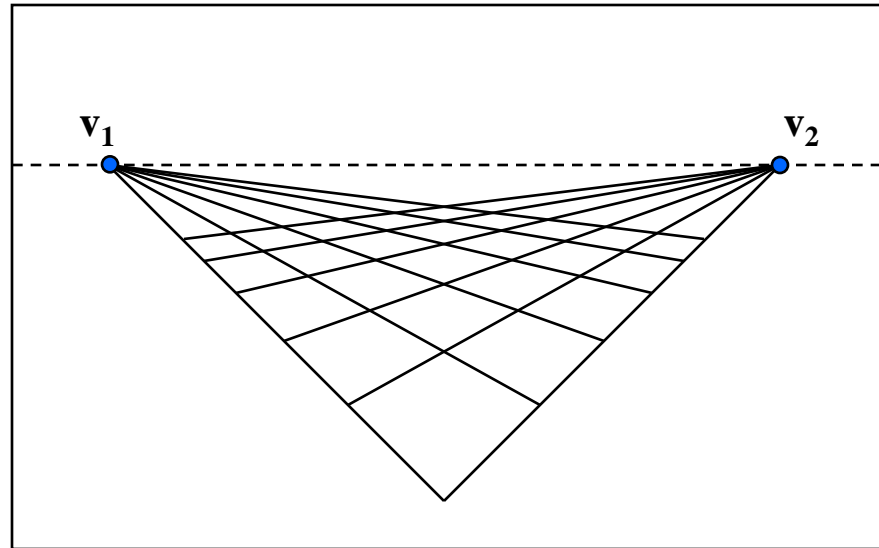


Vanishing points



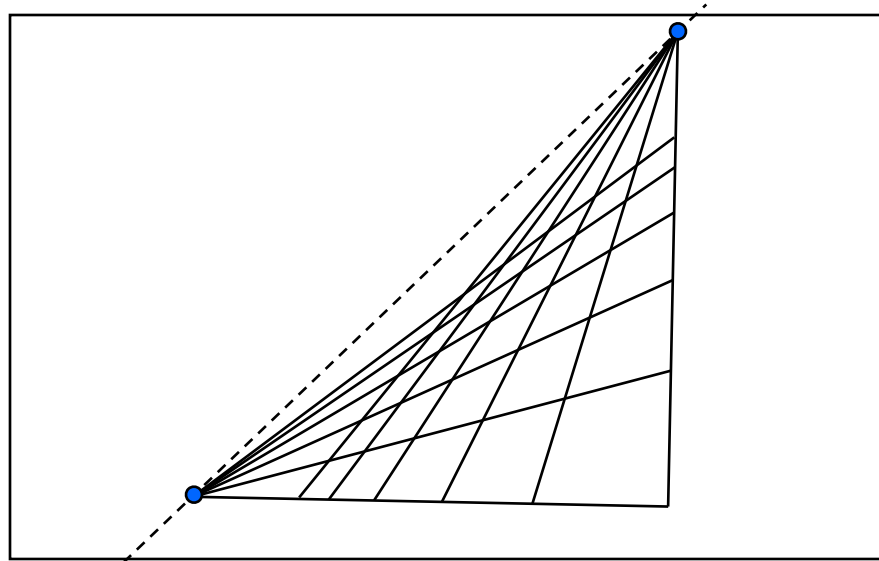
- Properties
 - Any two parallel lines have the same vanishing point v
 - The ray from C through v is parallel to the lines
 - An image may have more than one vanishing point
 - in fact every pixel is a potential vanishing point

Vanishing lines



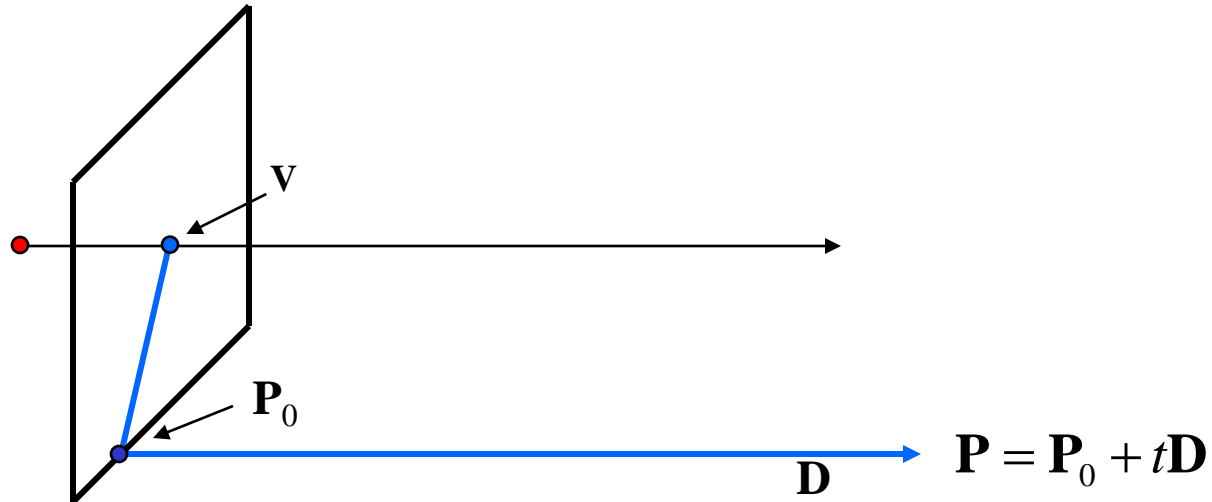
- Multiple Vanishing Points
 - Any set of parallel lines on the plane define a vanishing point
 - The union of all of these vanishing points is the *horizon line*
 - also called *vanishing line*
 - Note that different planes define different vanishing lines

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 define different vanishing lines

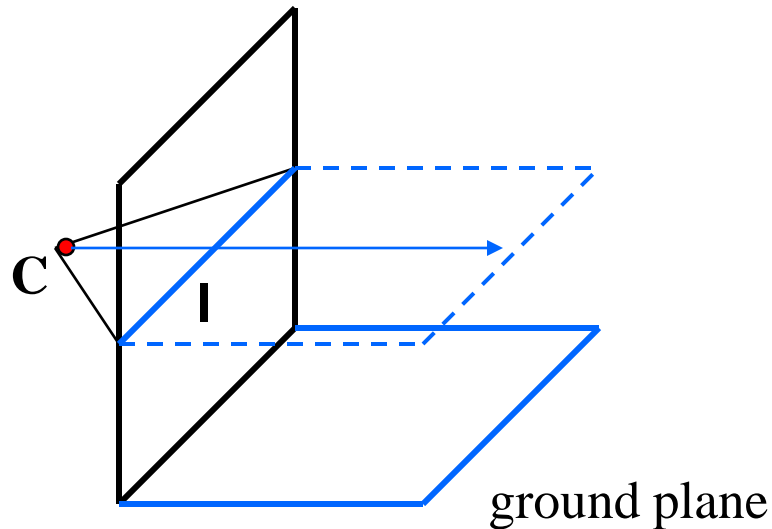
Computing vanishing points



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

- Properties $\mathbf{v} = \mathbf{I}\mathbf{P}_\infty$
 - \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 vanishing lines

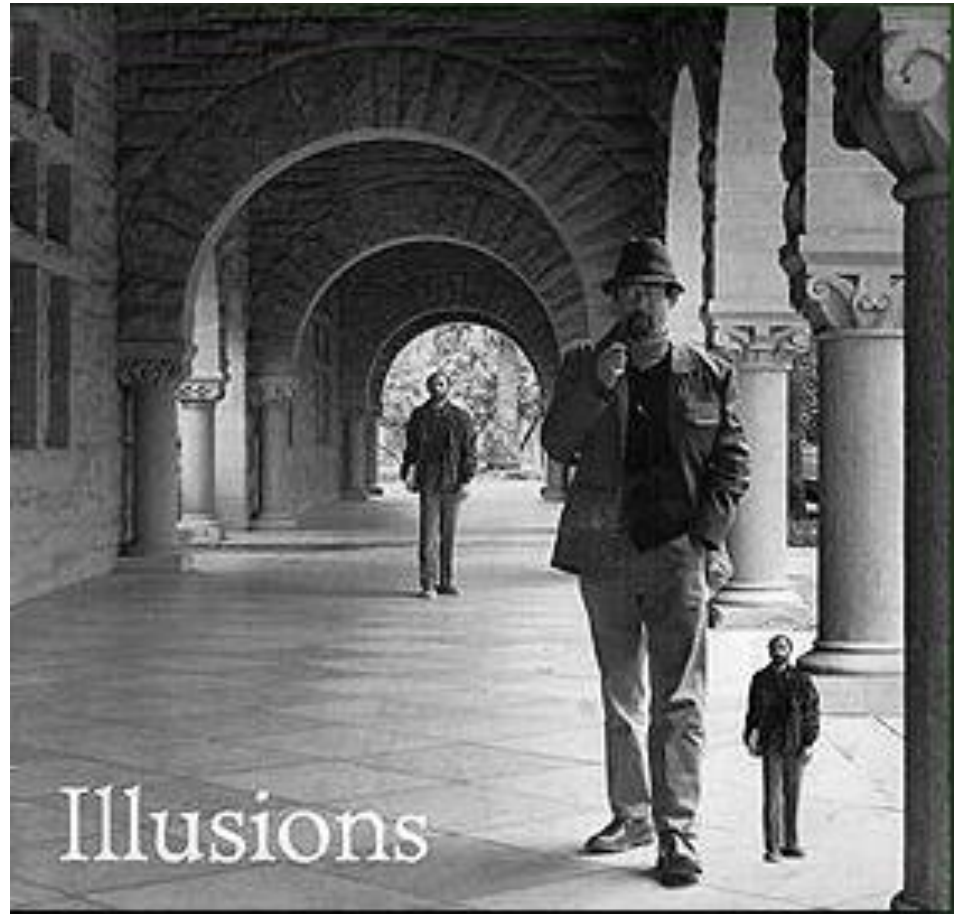
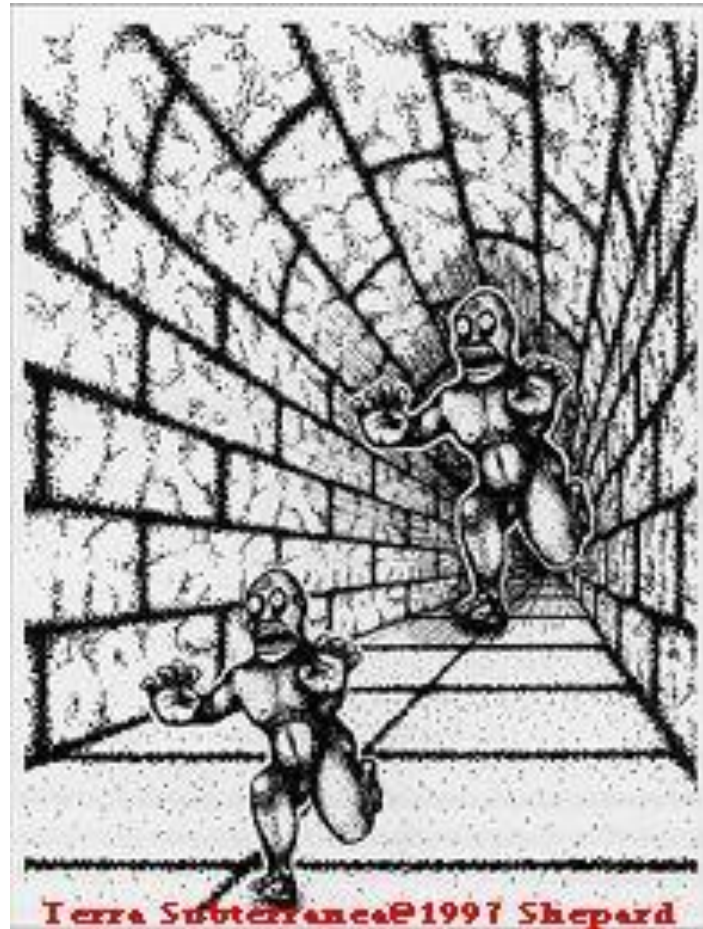


- **Properties**

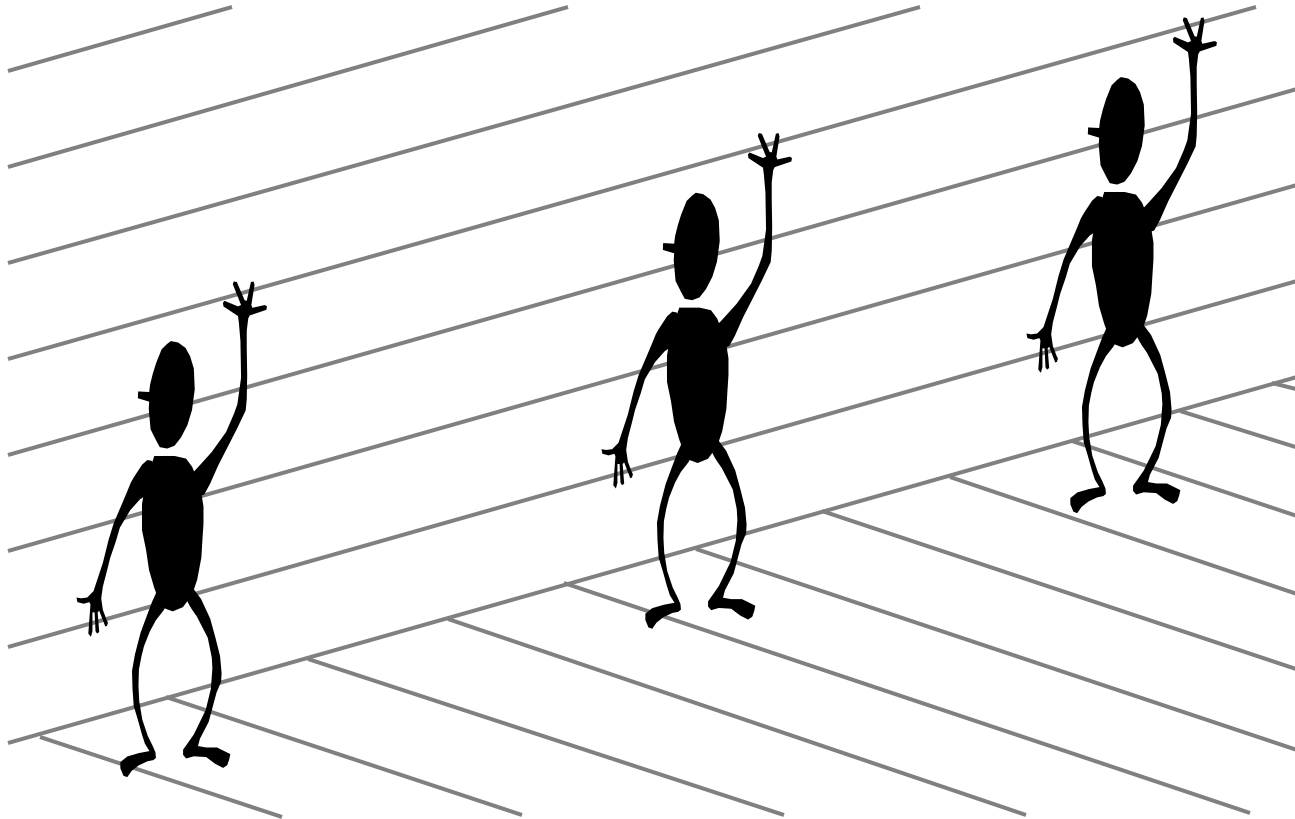
- **I** is intersection of horizontal plane through **C** with image plane
- Compute **I** from two sets of parallel lines on ground plane
- All points at same height as **C** project to **I**
 - points higher than **C** project above **I**
- Provides way of comparing height of objects in the scene



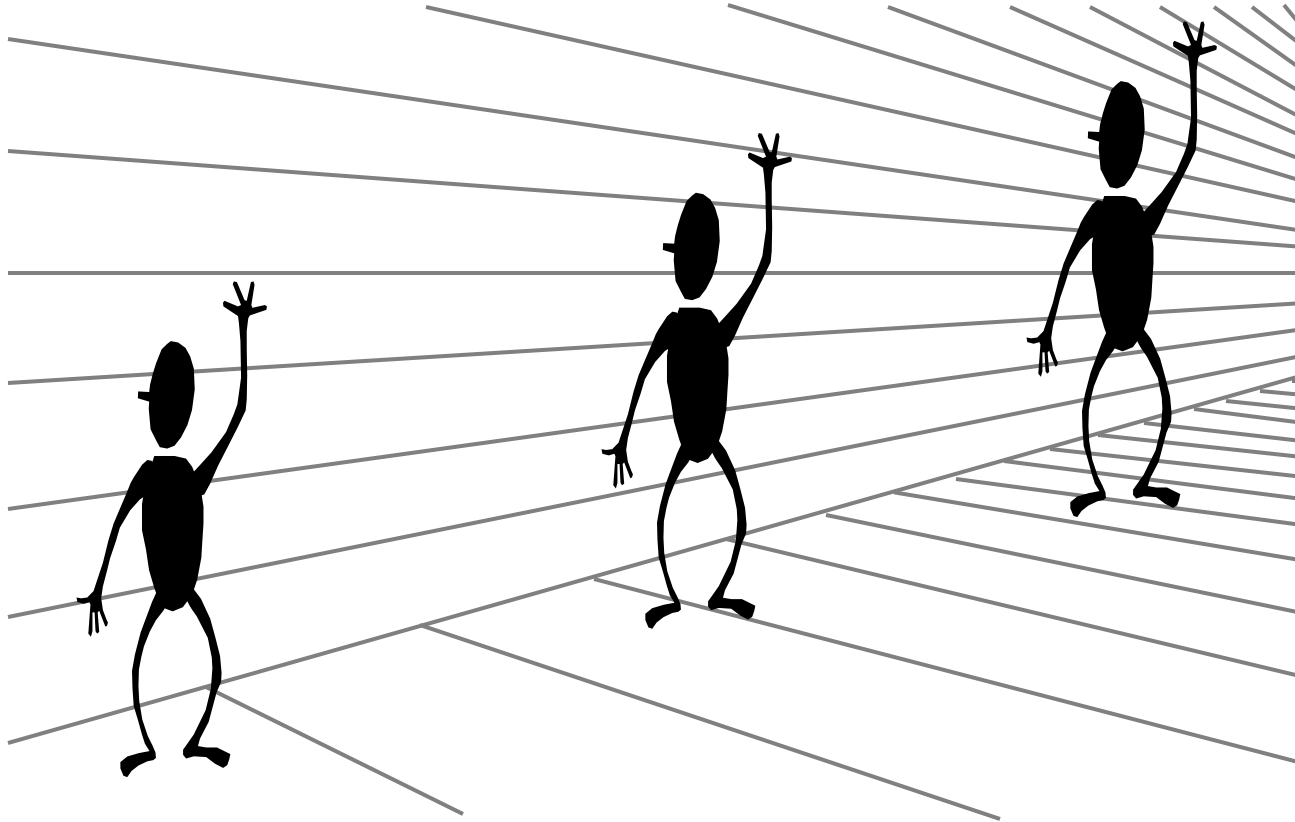
Fun with vanishing points



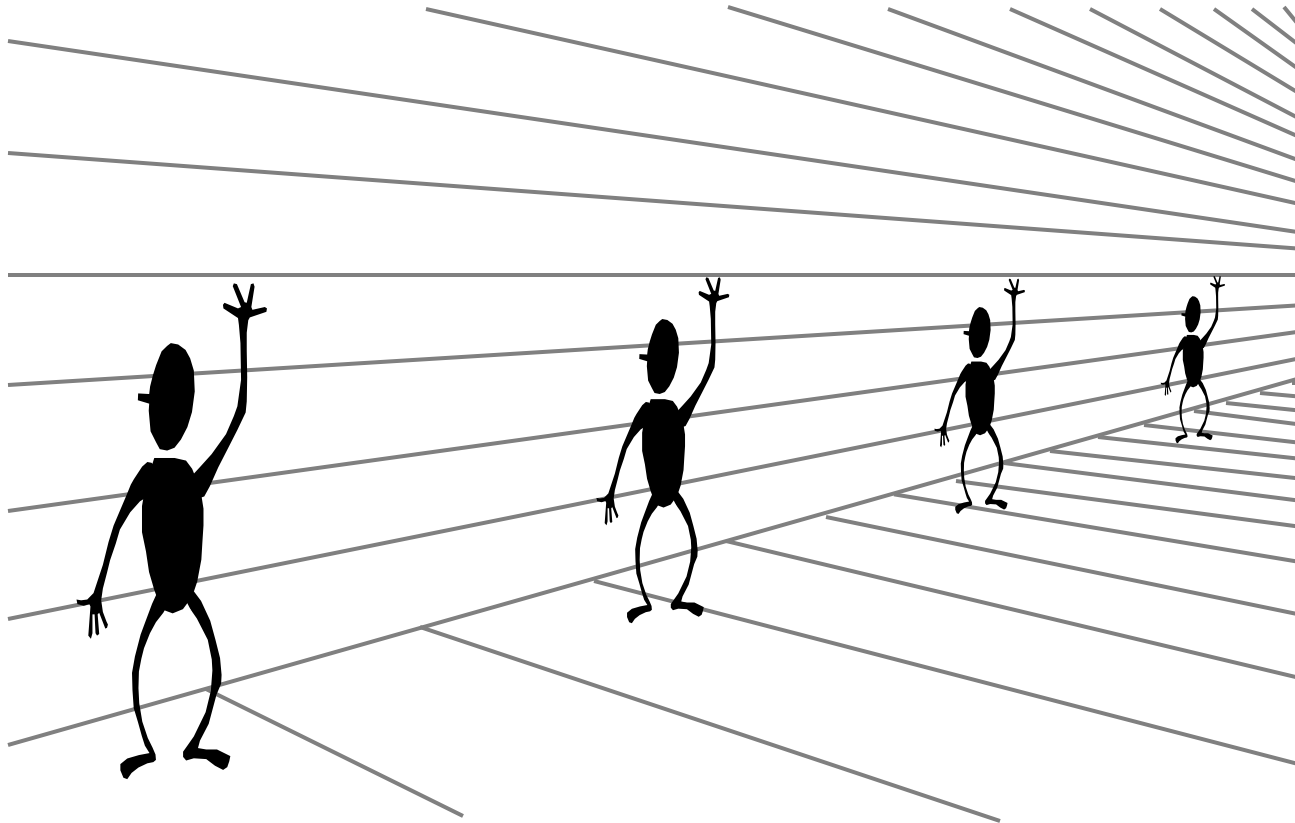
Perspective cues



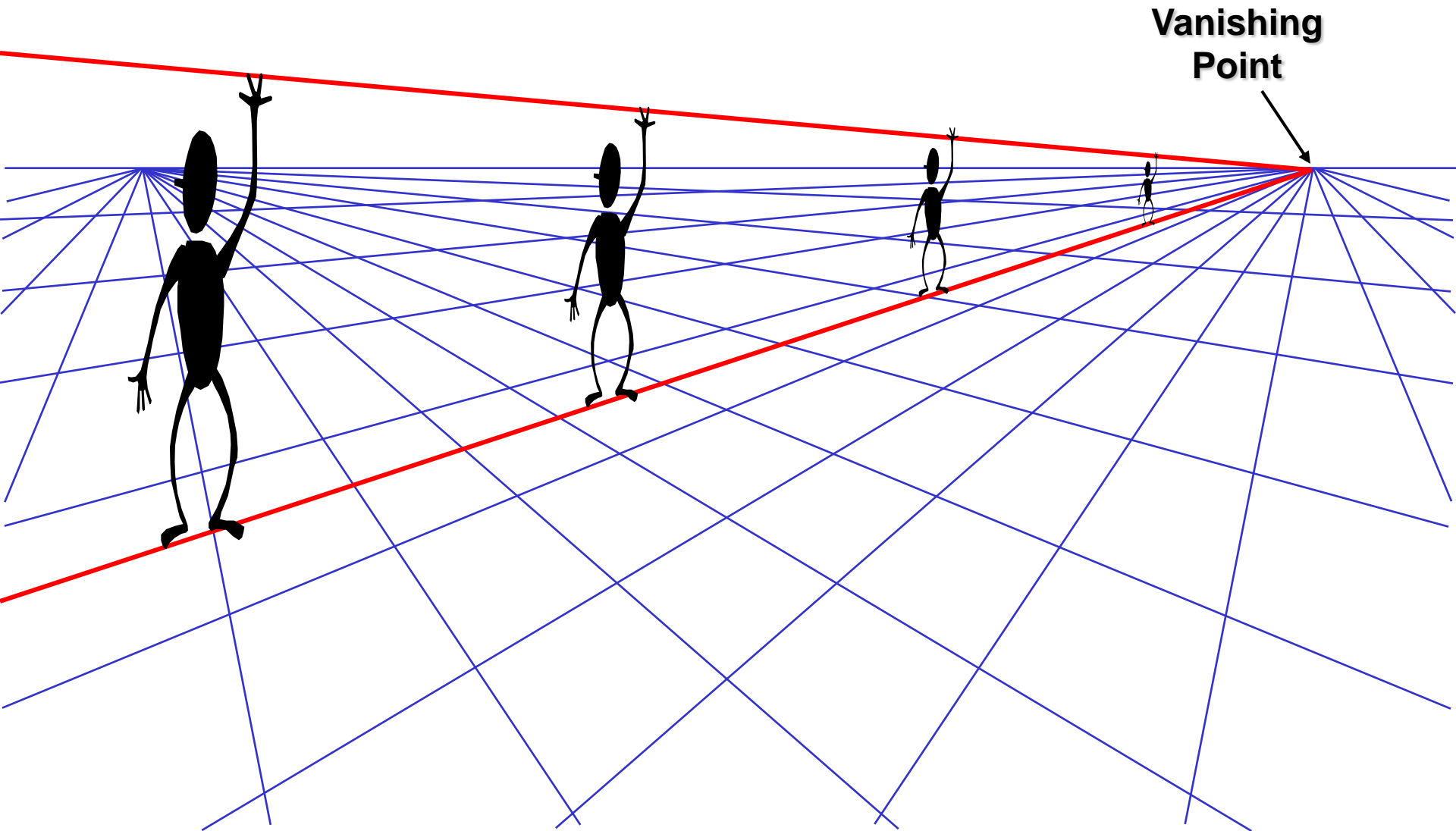
Perspective cues



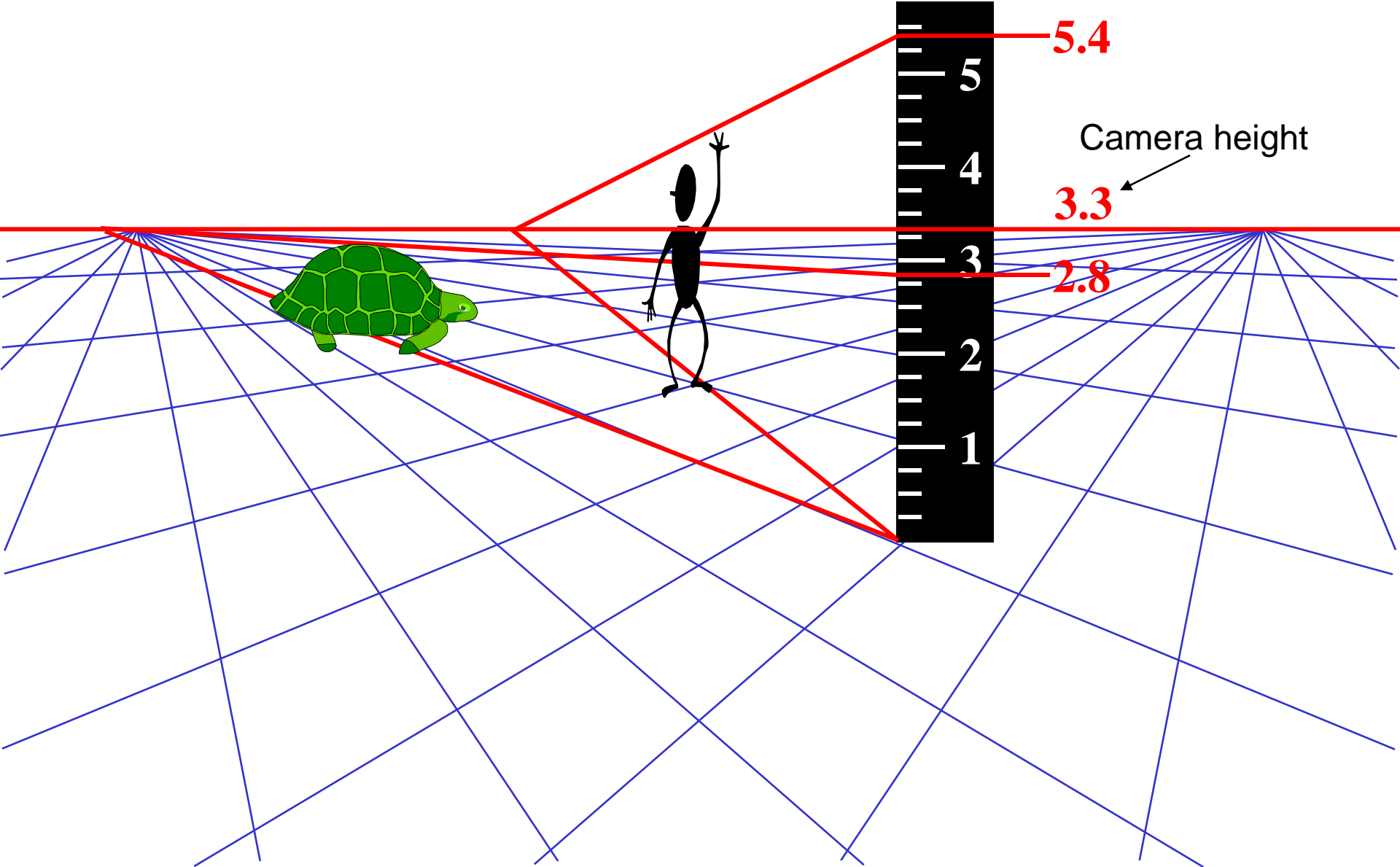
Perspective cues



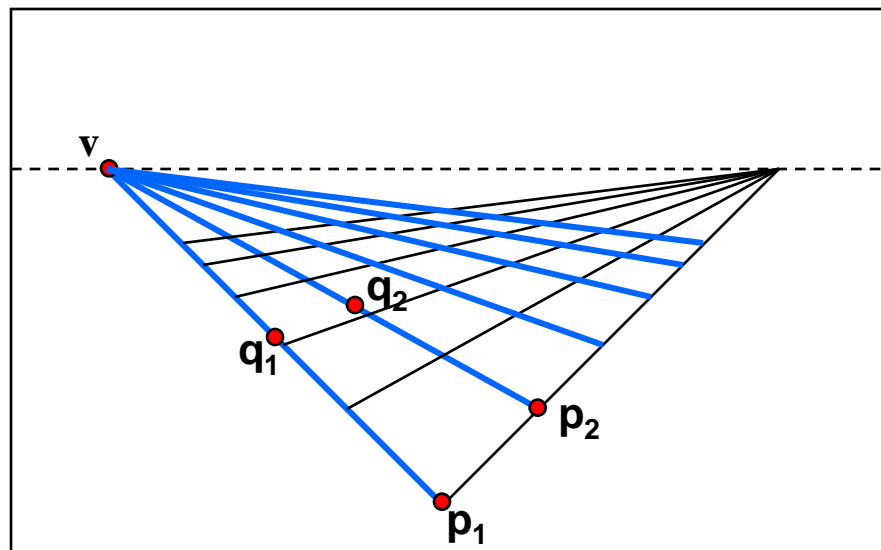
Comparing heights



Measuring height



Computing vanishing points (from lines)

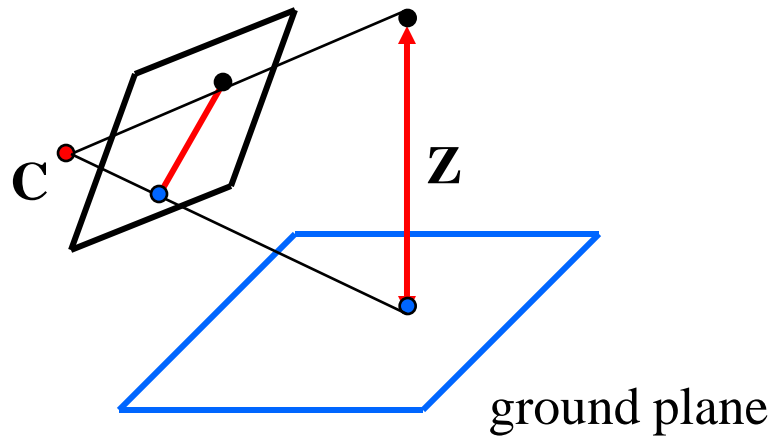


- Intersect p_1q_1 with p_2q_2
$$v = (p_1 \times q_1) \times (p_2 \times q_2)$$

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>

Measuring height without a ruler



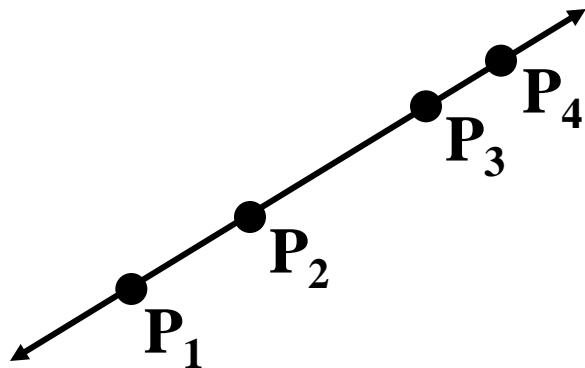
Compute Z from image measurements

- 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



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

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

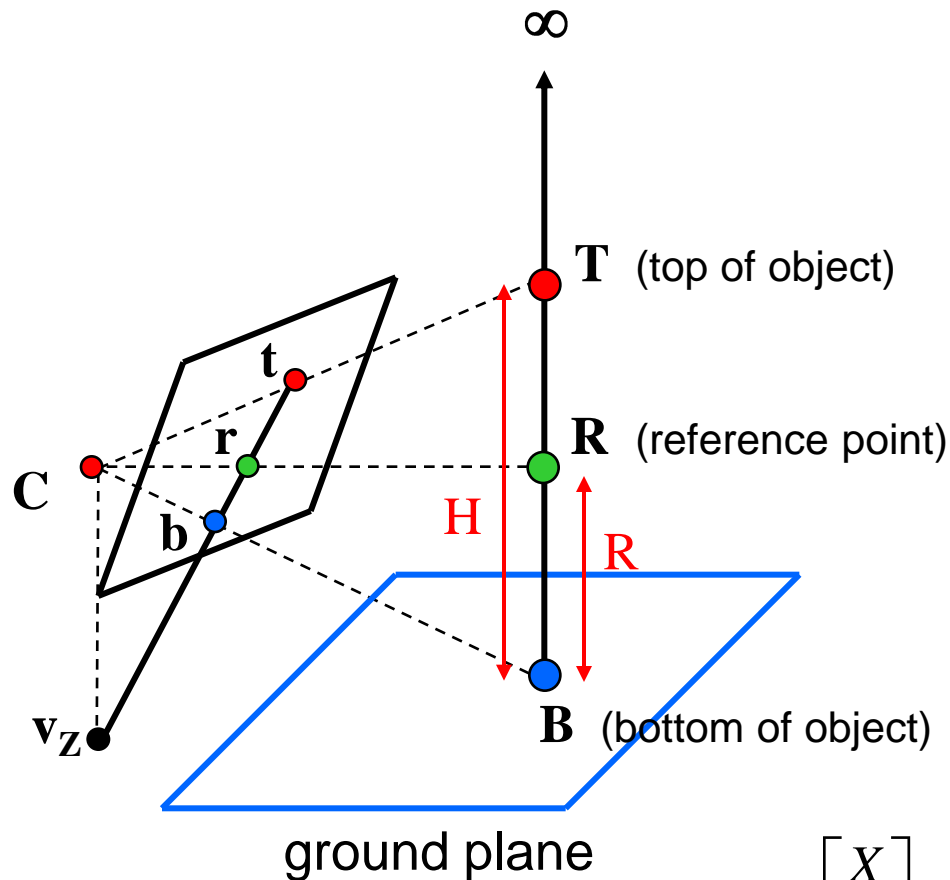
Can permute the point ordering

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

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

This is the fundamental invariant of projective geometry

Measuring height



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

image points as $\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

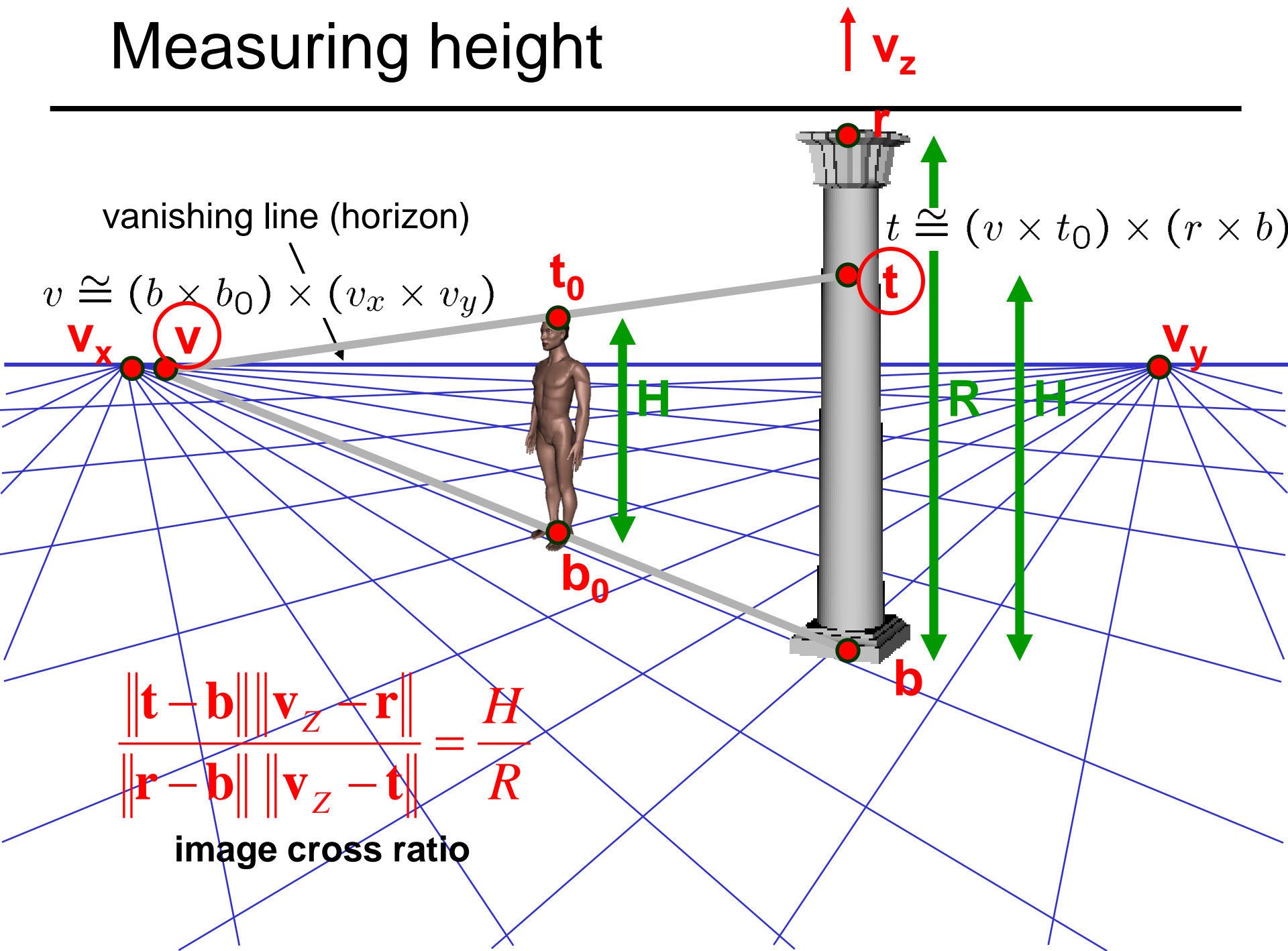
$$\frac{\|\mathbf{T} - \mathbf{B}\| \|\infty - \mathbf{R}\|}{\|\mathbf{R} - \mathbf{B}\| \|\infty - \mathbf{T}\|} = \frac{H}{R}$$

scene cross ratio

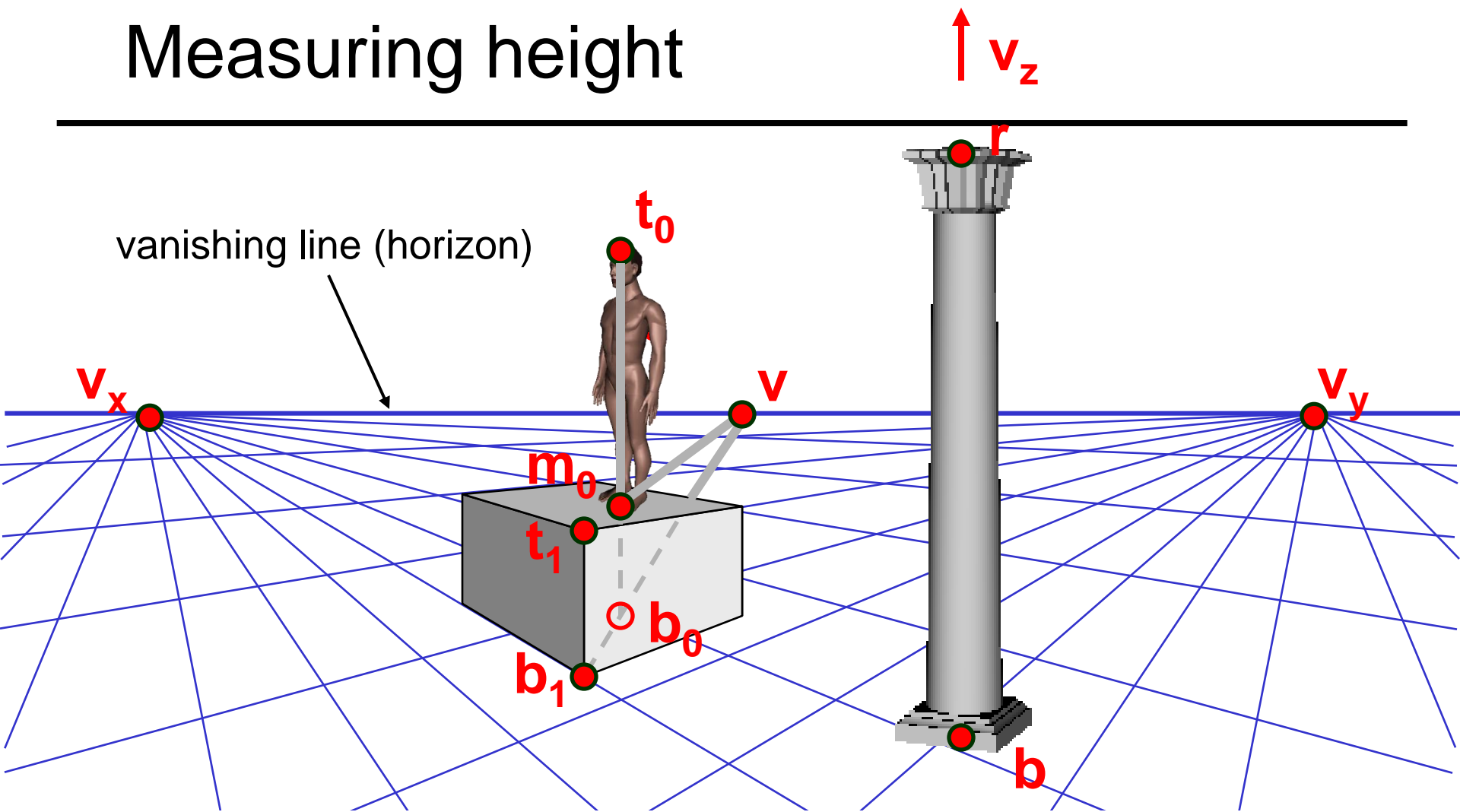
$$\frac{\|\mathbf{t} - \mathbf{b}\| \|\mathbf{v}_Z - \mathbf{r}\|}{\|\mathbf{r} - \mathbf{b}\| \|\mathbf{v}_Z - \mathbf{t}\|} = \frac{H}{R}$$

image cross ratio

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

- Okay, we know how to compute height (Z coords)
 - how can we compute X, Y?

Camera calibration

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

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

- 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

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ - & - & - & - \end{bmatrix} = \begin{bmatrix} \boldsymbol{\pi}_1 & \boldsymbol{\pi}_2 & \boldsymbol{\pi}_3 & \boldsymbol{\pi}_4 \\ - & - & - & - \end{bmatrix}$$

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

$$\mathbf{\Pi} = \begin{bmatrix} \mathbf{v}_X & \mathbf{v}_Y & \mathbf{v}_Z & \mathbf{o} \\ - & - & - & - \end{bmatrix}$$

Not So Fast! We only know \mathbf{v} 's up to a scale factor

$$\mathbf{\Pi} = \begin{bmatrix} a\mathbf{v}_X & b\mathbf{v}_Y & c\mathbf{v}_Z & \mathbf{o} \\ - & - & - & - \end{bmatrix}$$

- Can fully specify by providing 3 reference points

3D Modeling from a photograph



<https://research.microsoft.com/vision/cambridge/3d/3dart.htm>