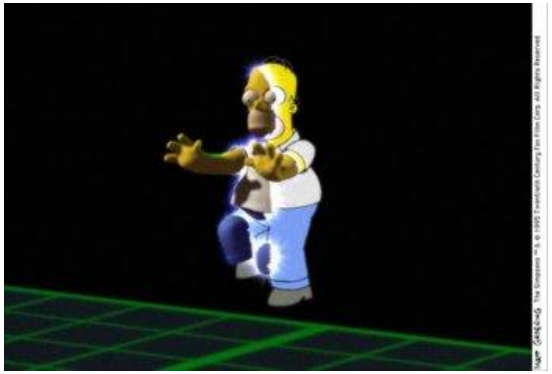
Computational Photography

Week 8

Instructor: Lou Kratz

Single View Modeling

Breaking out of 2D



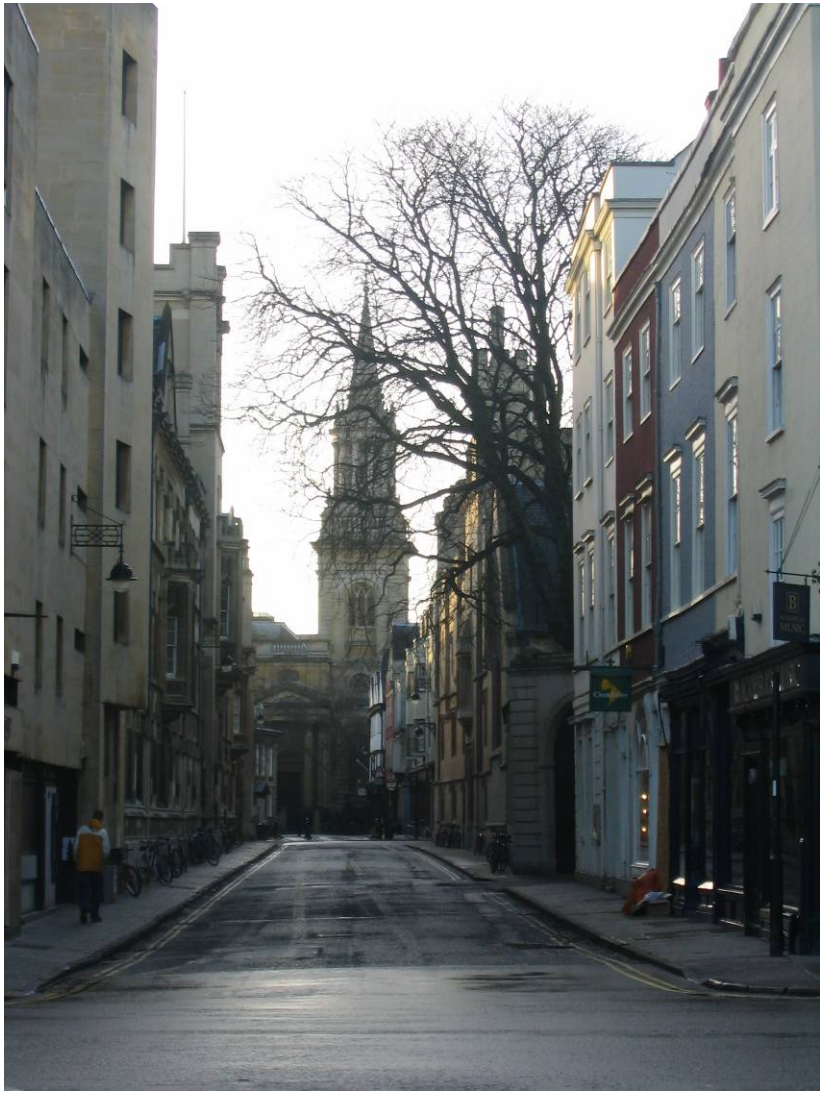
on to 3D…

Enough of 2D images!

We want real 3D scene walk-throughs:

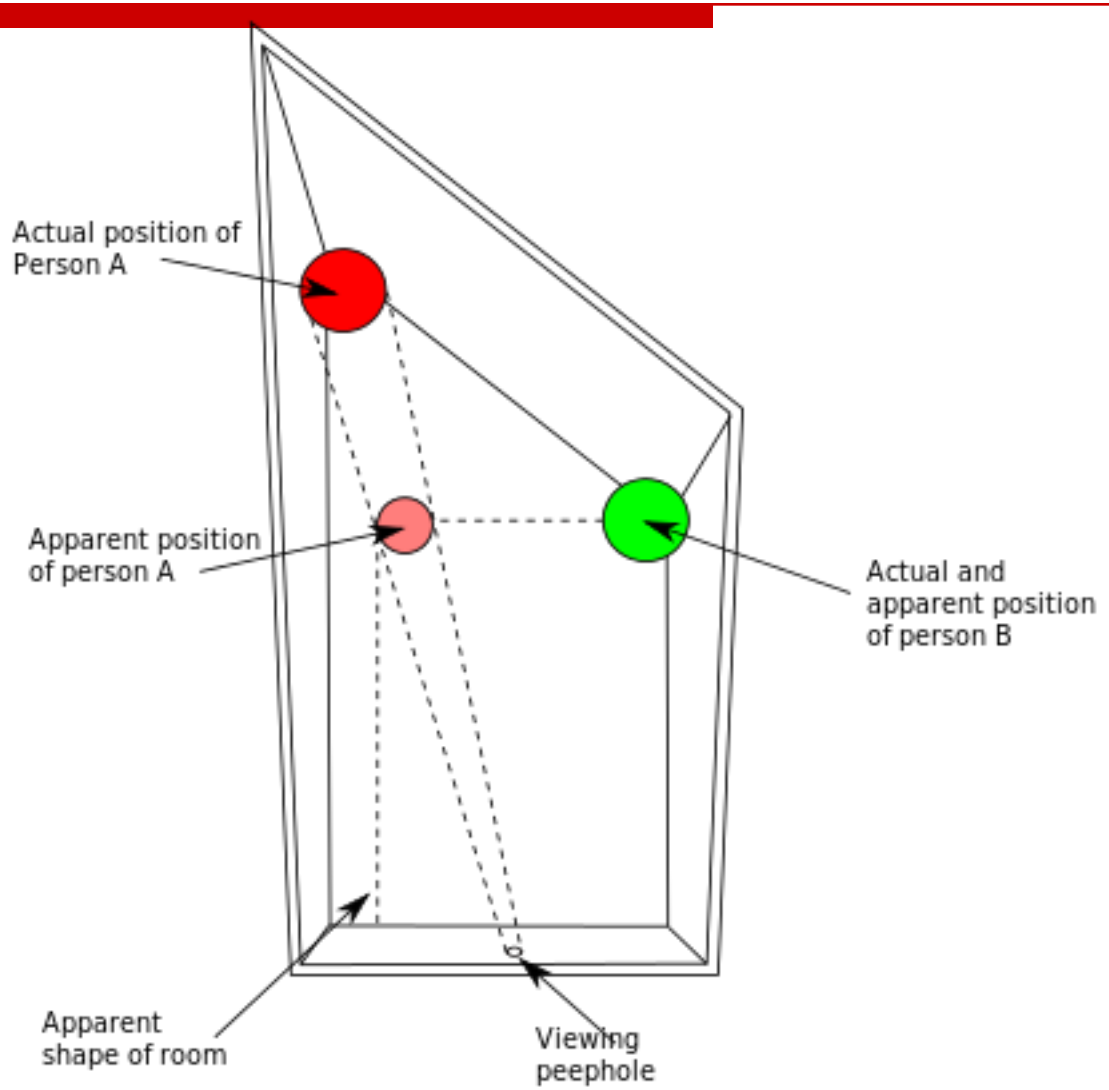
- Camera rotation
- Camera translation

Can we do it from a single photograph?



Some Preliminaries: Projective Geometry

Ames Room


Actual position of
Person A
Apparent position
of person A
Actual and
apparent position
of person B
Apparent
shape of room
Viewing
peephole

- https://commons.wikimedia.org/w/index.php?title=File%3AAmes_room.ogv

Homogeneous Coordinates

- Cartesian to Homogeneous Coordinates:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene coordinates

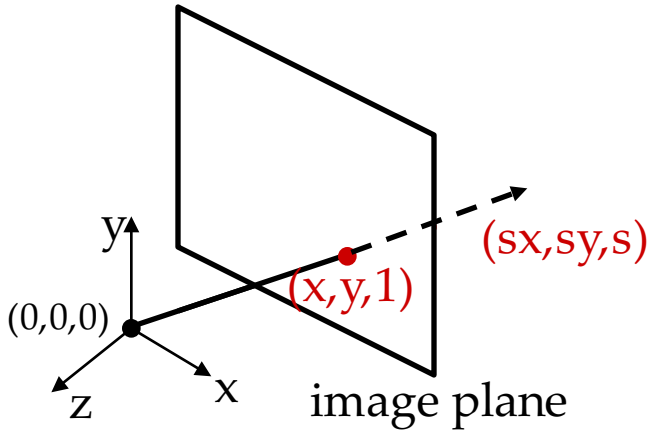
- Homogeneous Cartesian to Coordinates:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

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)

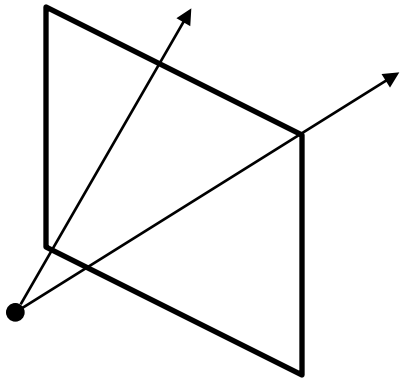
Homogeneous Pinhole Camera

- https://www.youtube.com/watch?v=fVJeJMWZcq8

Scale Invariance

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

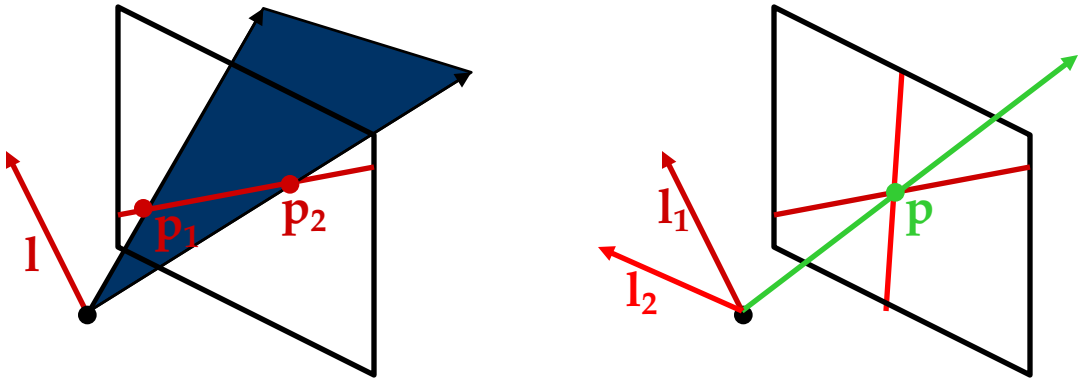
$$\text{in vector notation}: \quad 0 = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\mathbf{l} \quad \mathbf{p}$$

  - 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 perpendicular to every point (ray) $\mathbf{p}$ on the line: $\mathbf{l}\,\mathbf{p}=0$



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 $\mathbf{p}_2$ $\quad \mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2$
- $\mathbf{l}$ is the plane normal

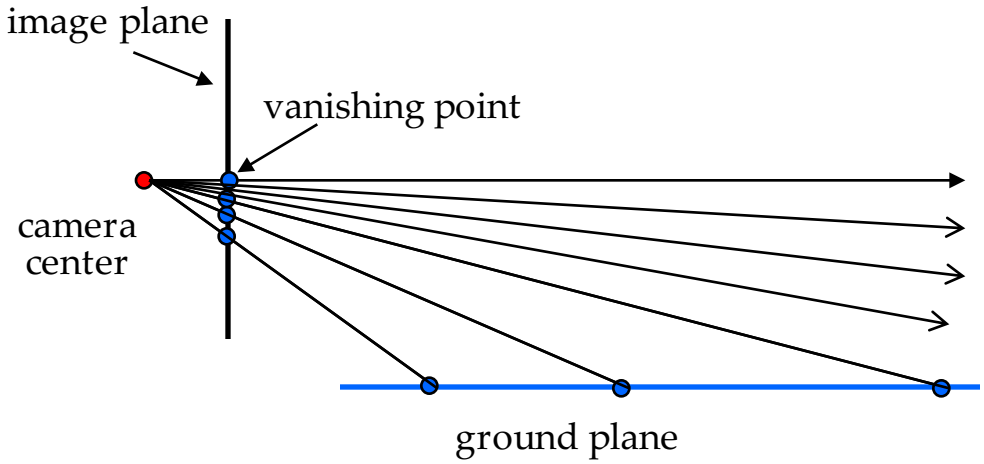
What is the intersection of two lines $\mathbf{l}_1$ and $\mathbf{l}_2$?

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

Points and lines are *dual* in projective space

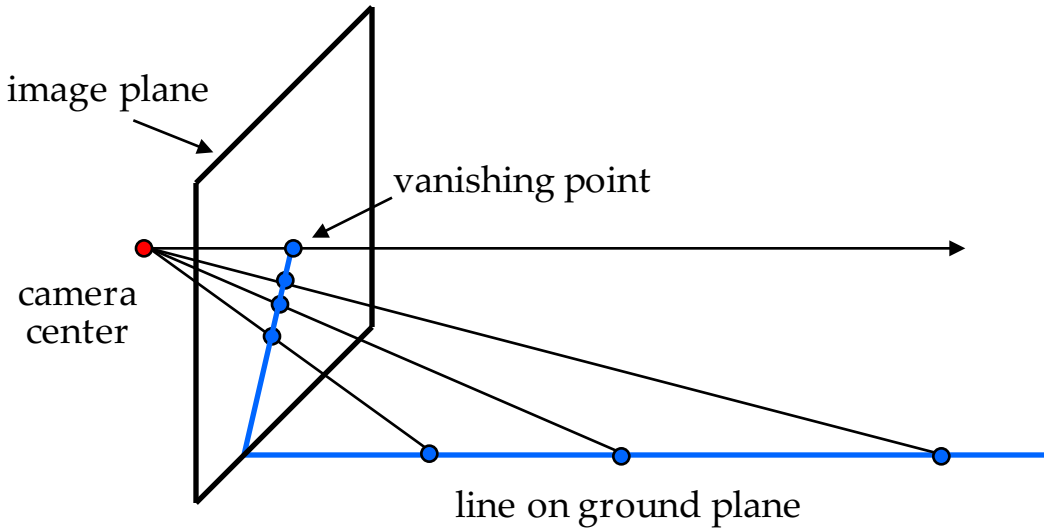
- given any formula, can switch the meanings of points and lines to get another formula

Vanishing Points

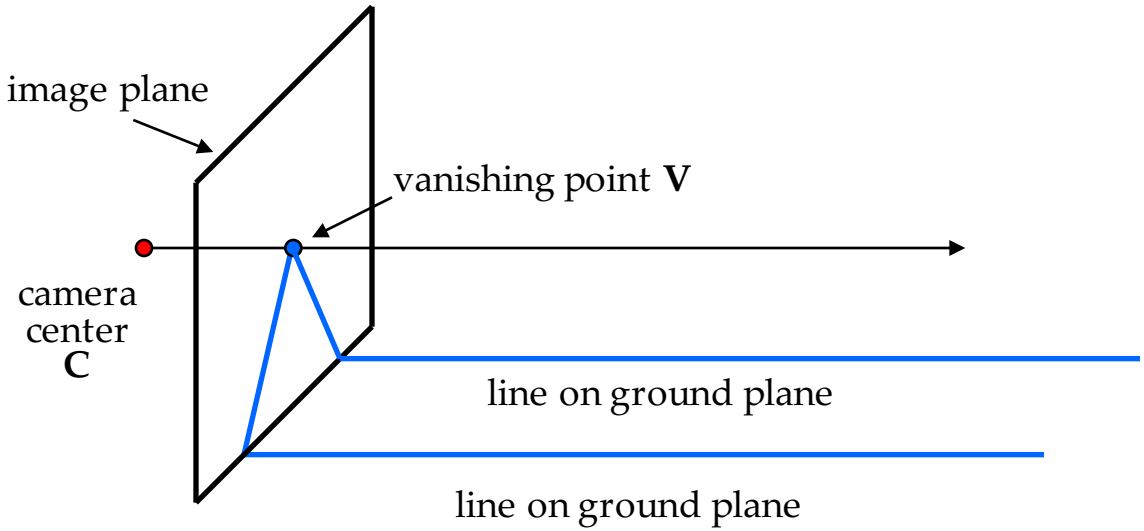


- Vanishing point
  - projection of a point at infinity

Vanishing Points (2D)

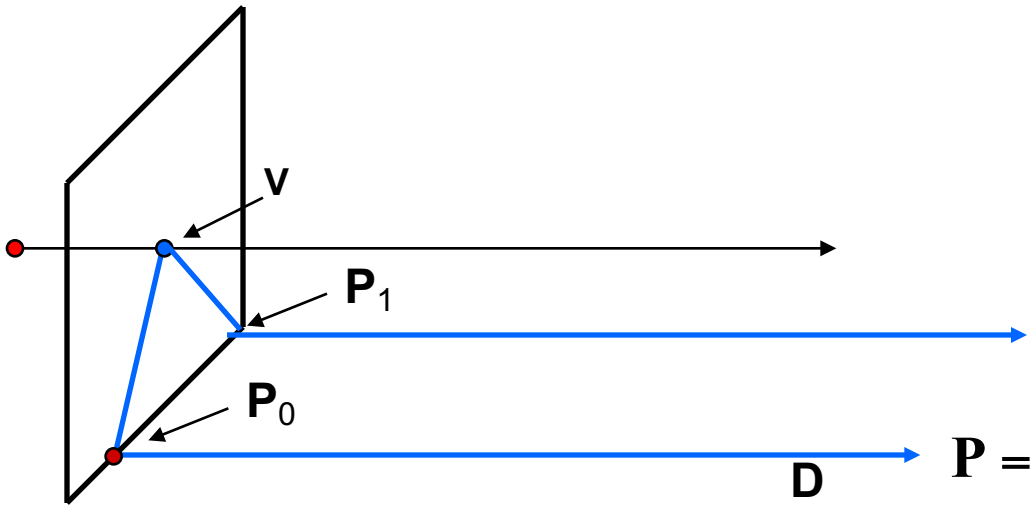


Vanishing Points



- Properties
  - Any two parallel lines on the same plane 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

Computing Vanishing Points

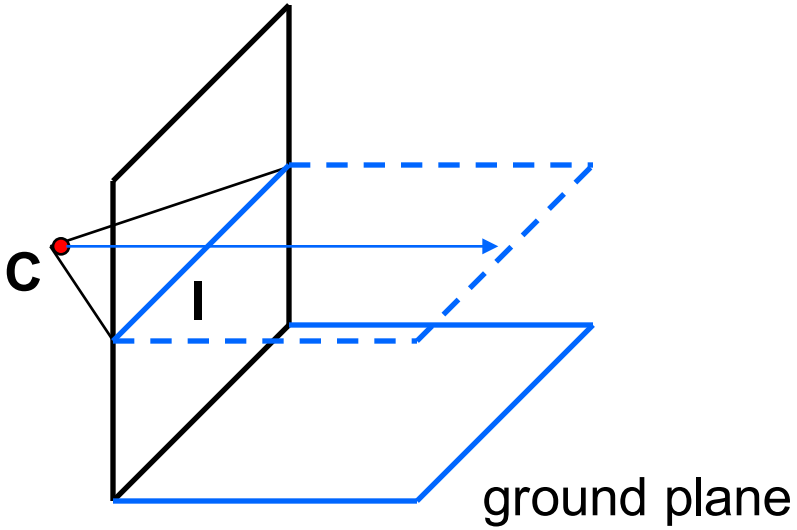


$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_X/t + D_X \\ P_Y/t + D_Y \\ P_Z/t + D_Z \\ 1/t \end{bmatrix} \qquad t \to \infty \qquad \mathbf{P}_\infty \cong \begin{bmatrix} D_X \\ D_Y \\ D_Z \\ 0 \end{bmatrix}$$

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

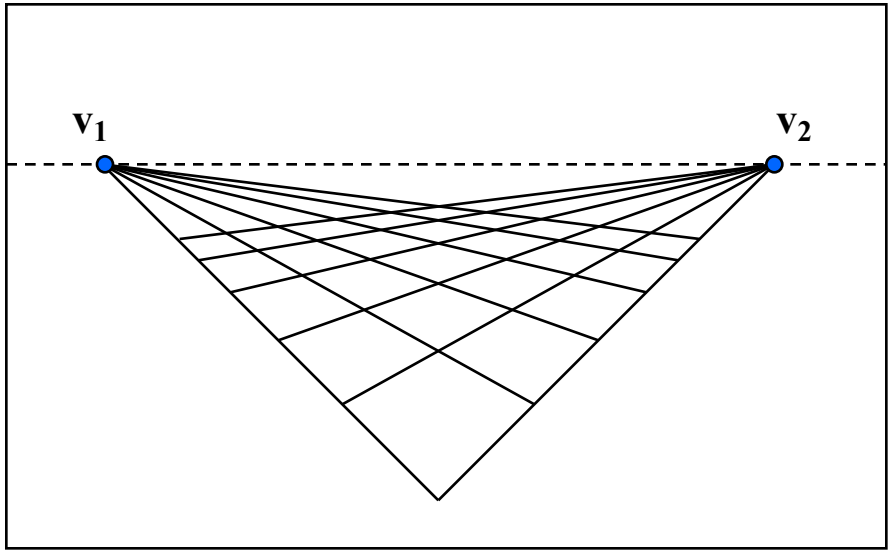
- $\mathbf{P}_{\text{inf}}$ is a point at *infinity*, $\mathbf{v}$ is its projection
- They depend only on line *direction*
- Parallel lines $\mathbf{P}_0 + t\mathbf{D}$, $\mathbf{P}_1 + t\mathbf{D}$ intersect at $\mathbf{P}_{\text{inf}}$

Computing Vanishing Lines



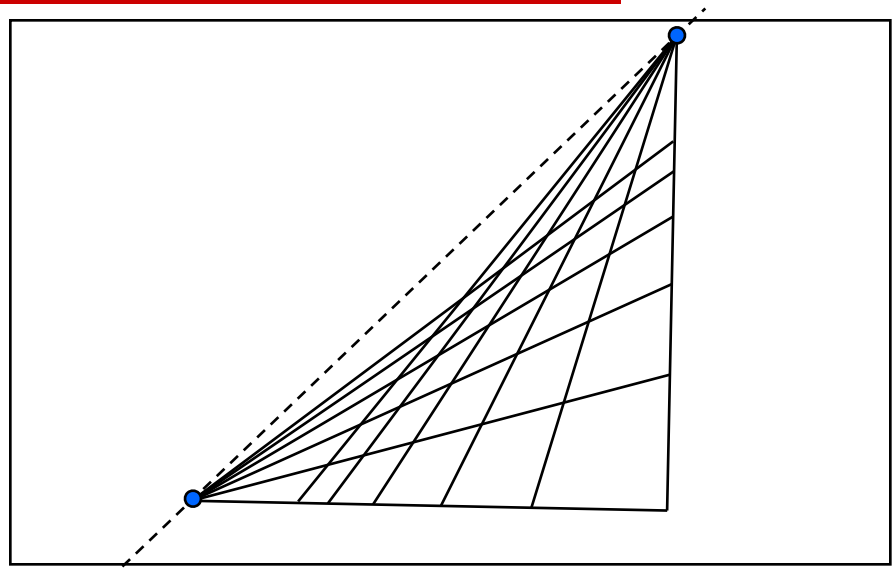
- **l** is the intersection of horizontal plane through **C** with image plane
- Compute **l** from two sets of parallel lines on ground plane
- All points at same height as **C** project to **l**
  - points higher than **C** project above **l**
- Provides way of comparing height of objects in the scene

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

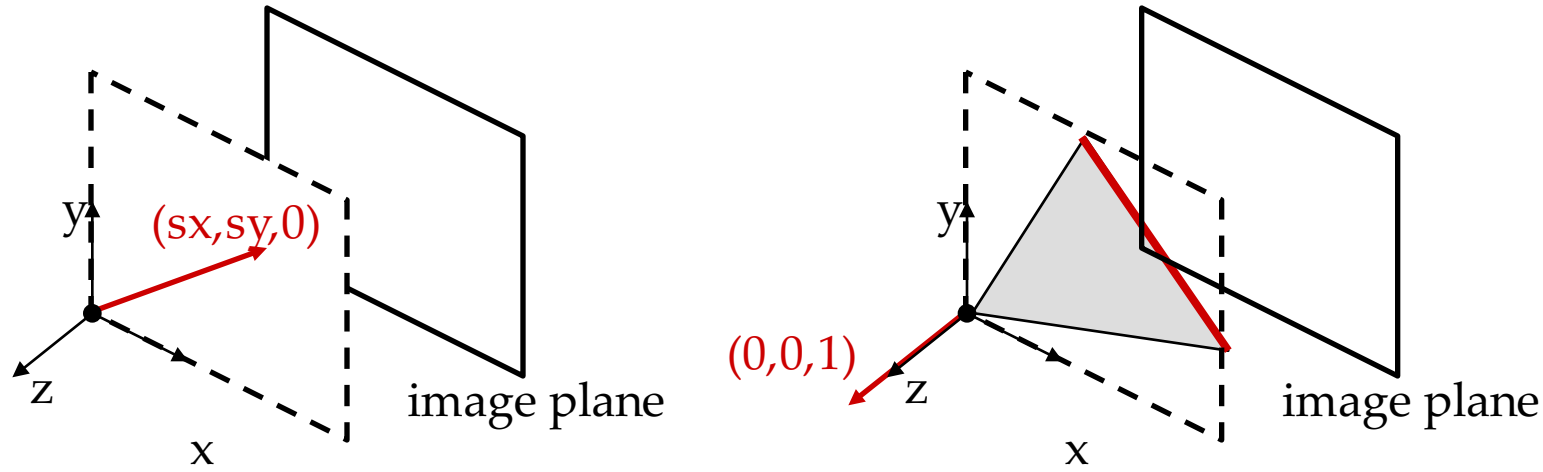
Vanishing Points

Image by Q-T. Luong (a vision researcher & photographer)

Vanishing Lines

Image by Q-T. Luong (a vision researcher & photographer)

Ideal Points and Lines



- Ideal point ("point at infinity")
  - **p** (x, y, 0) – parallel to image plane
  - Infinitely large coordinates (divide by zero!)
- Ideal line
  - **l** (0,0,1) – parallel to image plane

Homographies

- https://www.youtube.com/watch?v=fVJeJMWZcq8

Camera Rotation with Homography

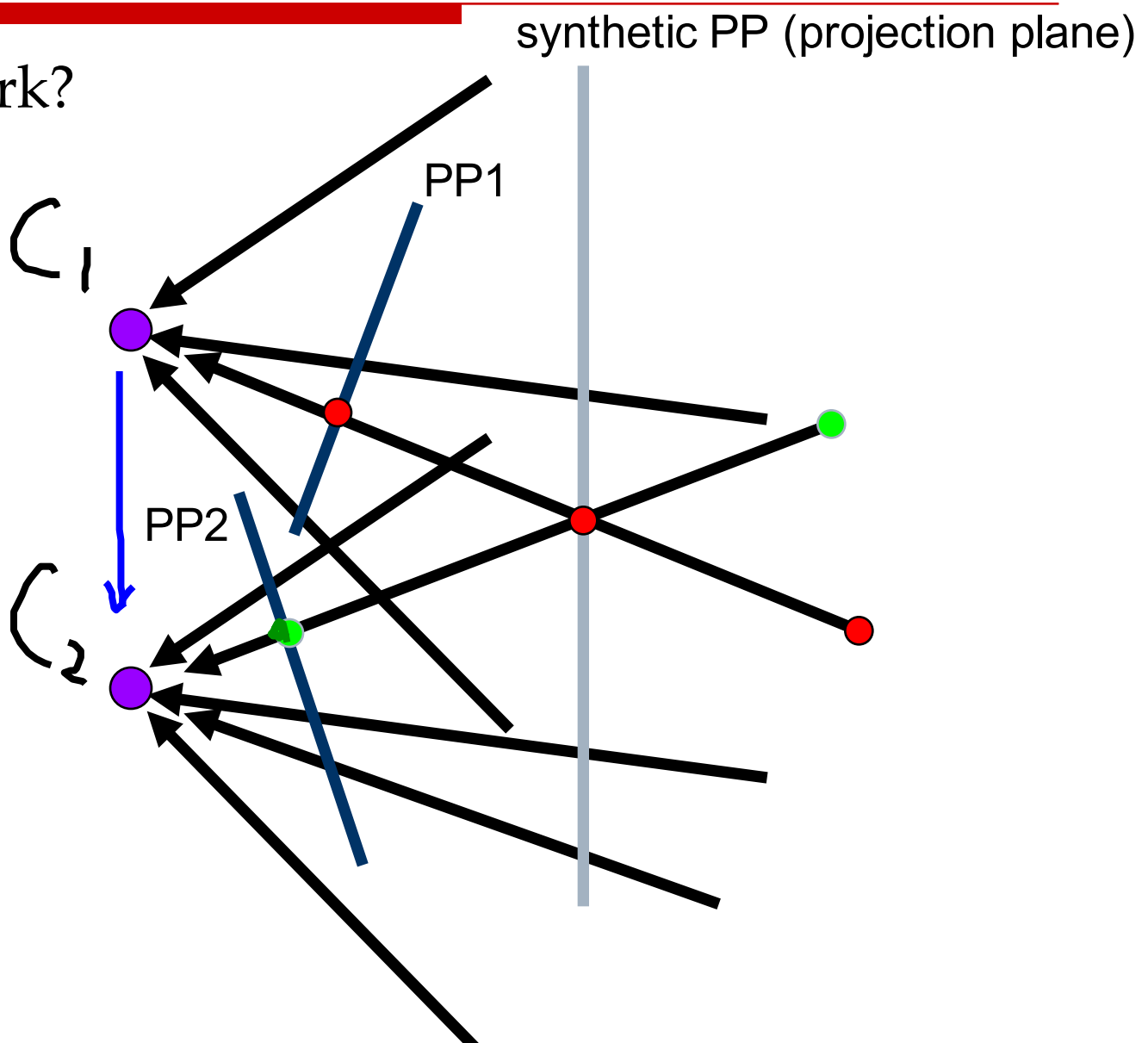
Original image

St.Petersburg
photo by A. Tikhonov

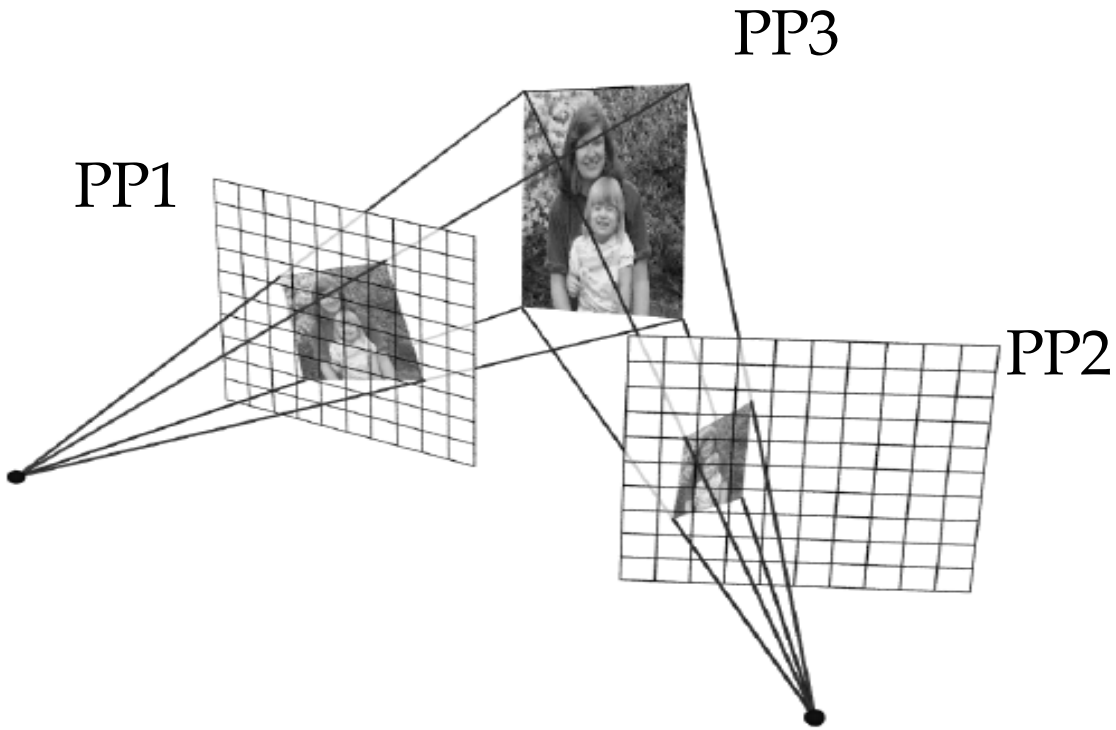
Virtual camera rotations

Camera Translation

- Does it work?



Yes, with planar scene (or far away)



- PP3 is a projection plane of both centers of projection, so we are OK!

Homographies: Can rotate camera!

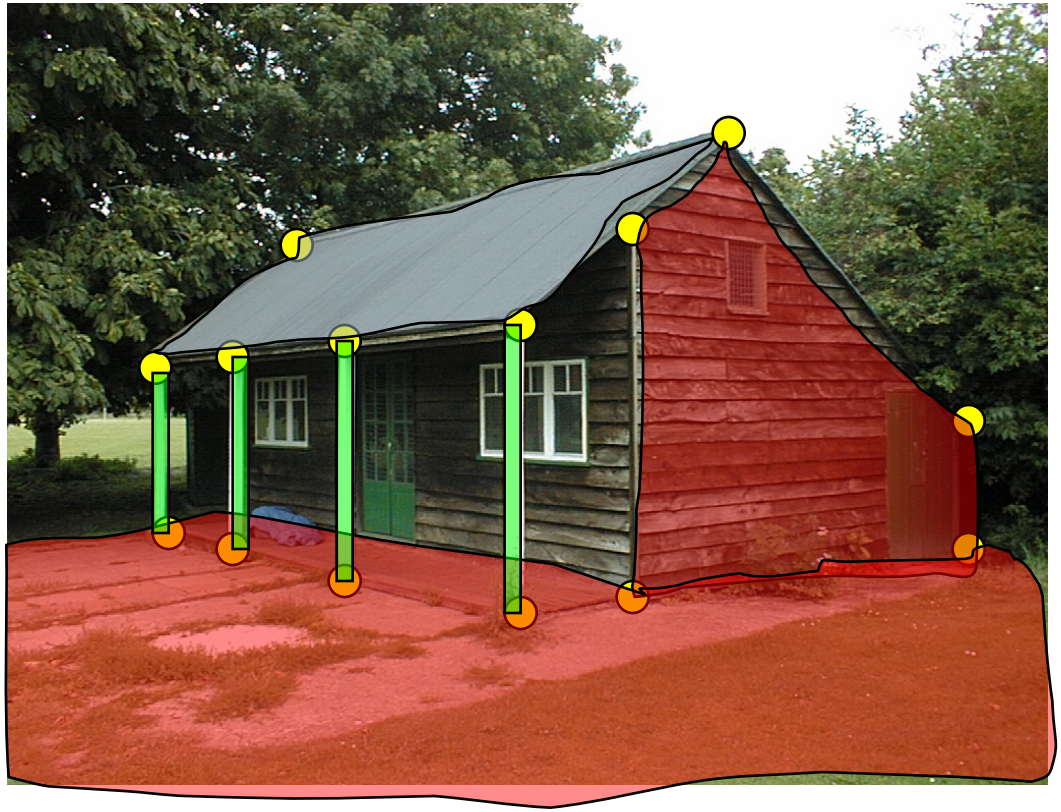
So,

- Rotation around the same view point (aka panning) can be modeled with homography transform
- Translation (zooming/moving) can be modeled with homography if the scene is **planar**

So, what can we do here?

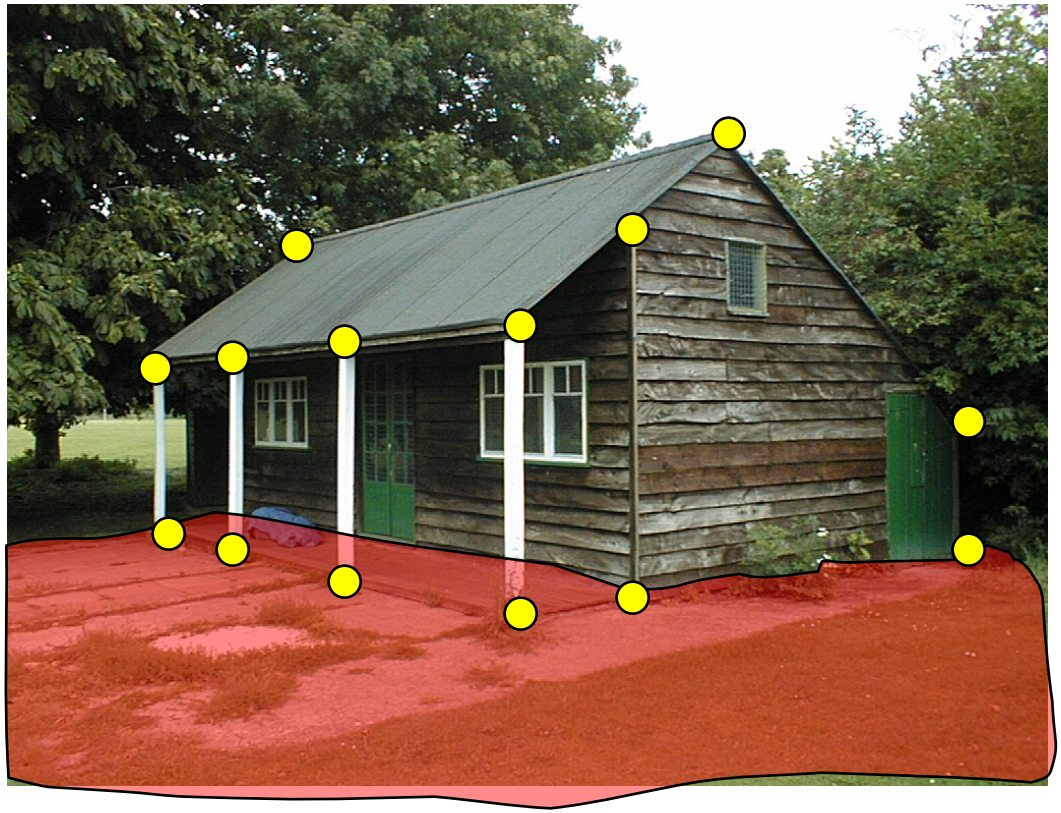
- Model the scene as a set of planes!

How can we model this scene?



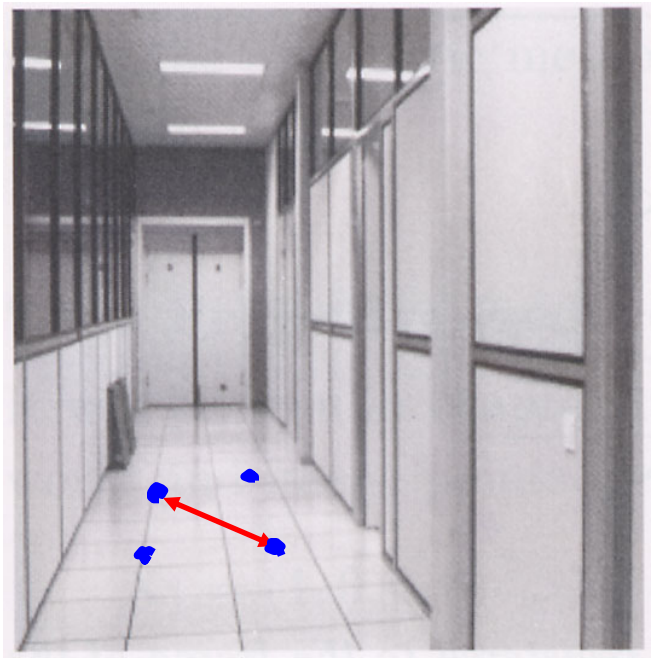
1. Find world coordinates (X,Y,Z) for a few points
2. Connect the points with planes to model geometry
   - Texture map the planes

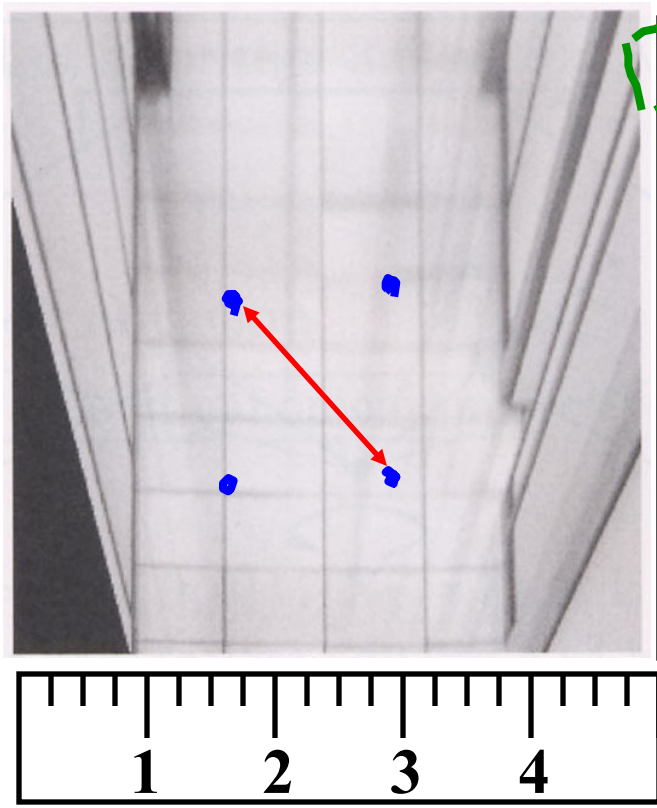
Finding World Coordinates (X,Y,Z)



1. Define the ground plane (Z=0)
2. Compute points (X,Y,0) on that plane
3. Compute the *heights* Z of all other points

Measurements on Planes

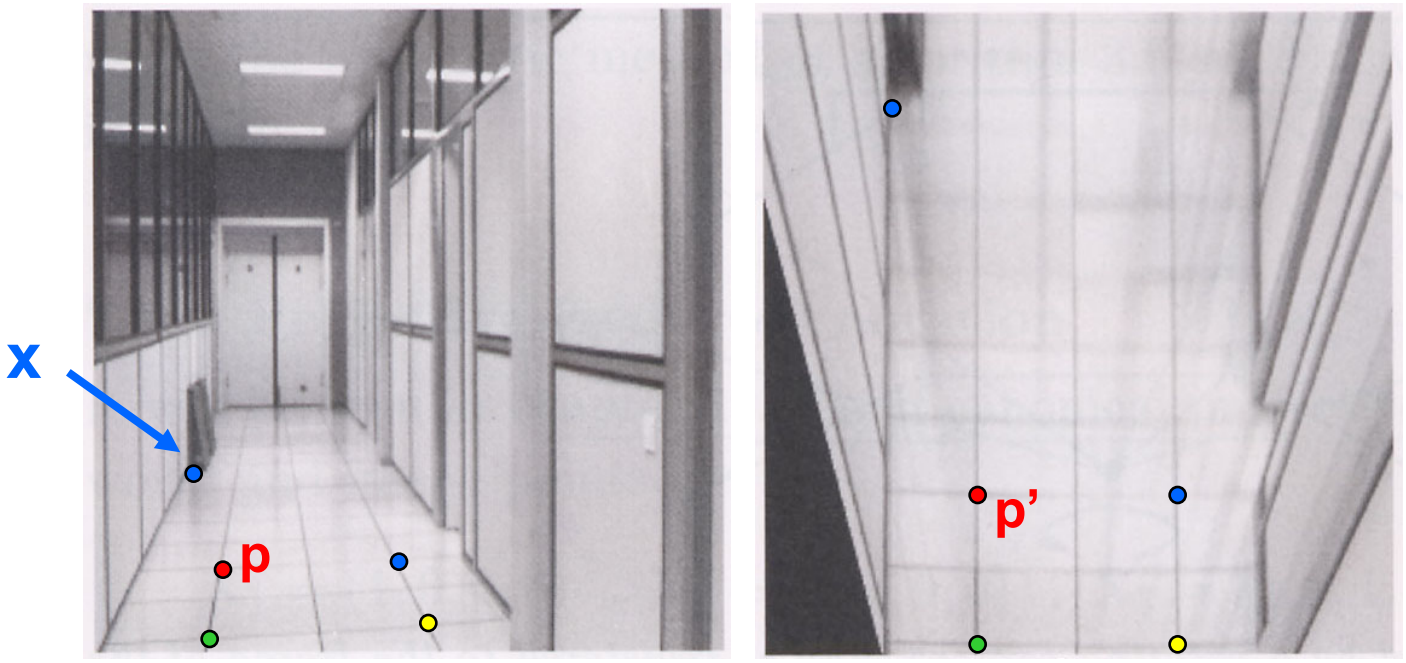




Approach: unwarp, then measure

What kind of warp is this?

Unwarp Ground Plane



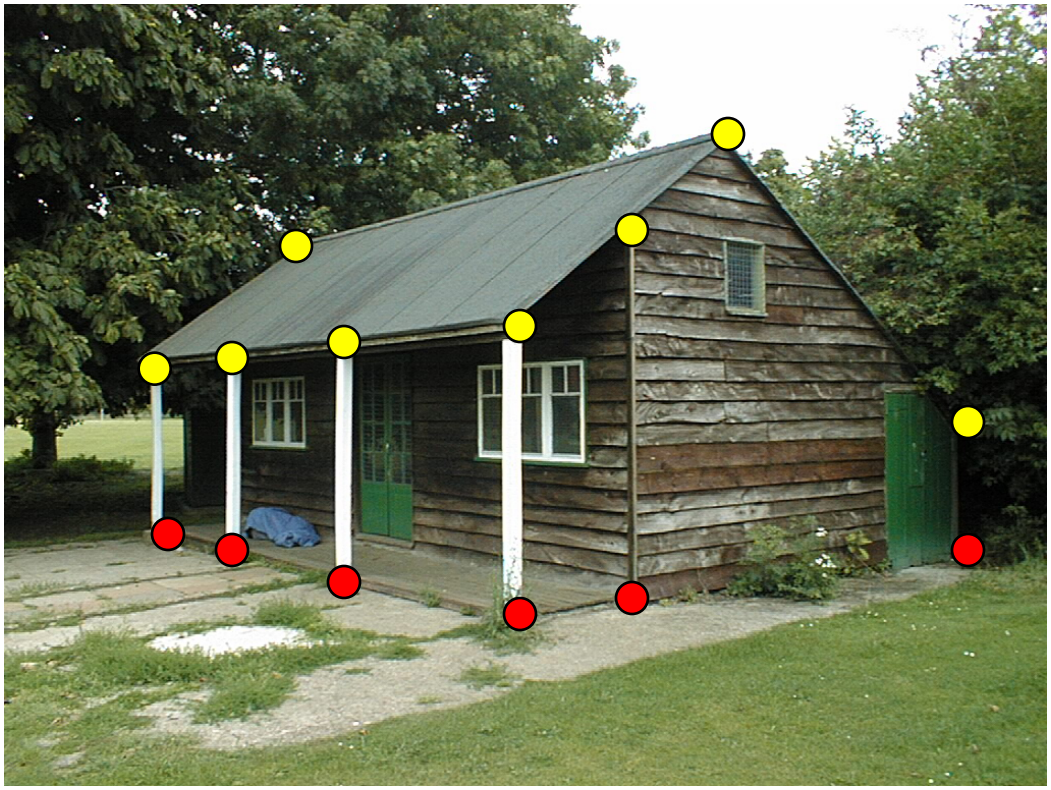
Called **Homography**

Need 4 reference points with world coordinates

$p = (x,y)$

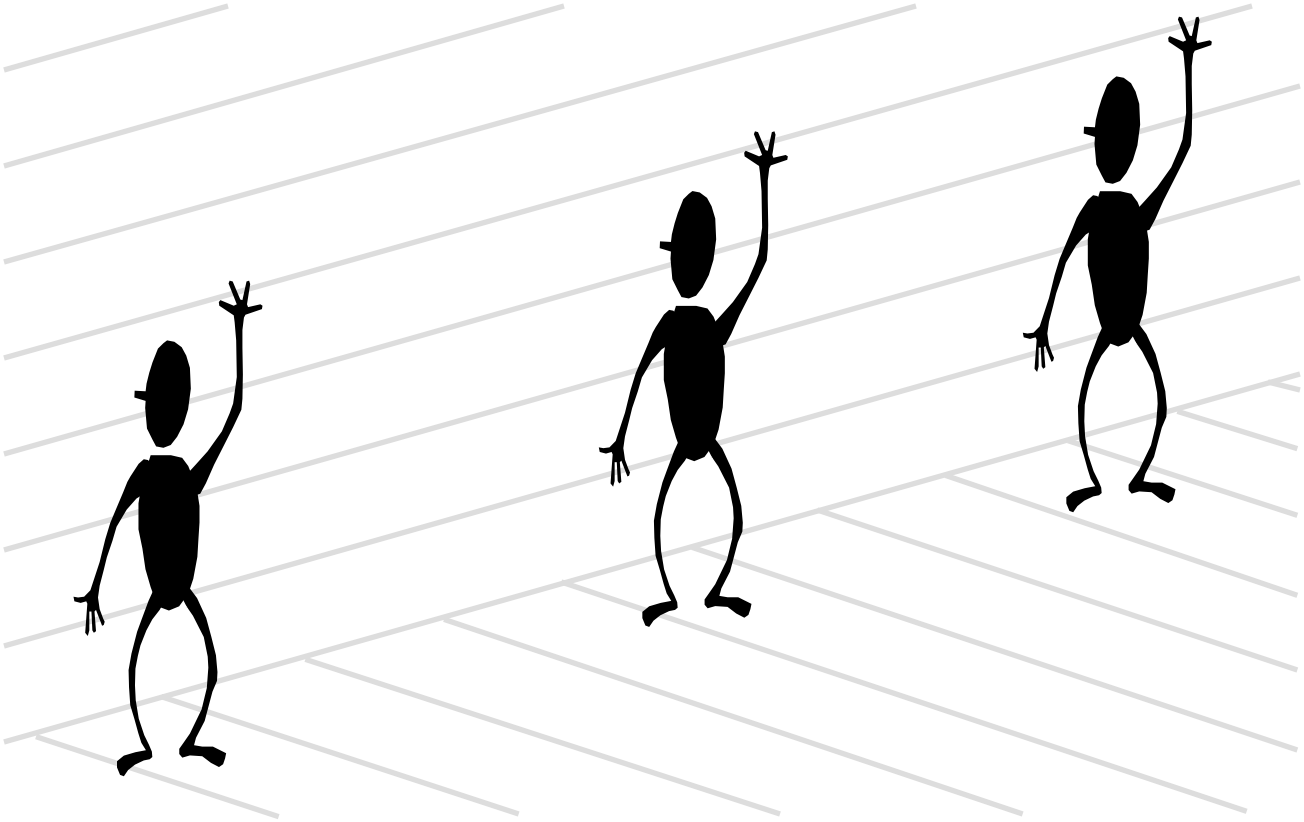
$p' = (X,Y,0)$

Finding World Coordinates (X,Y,Z)

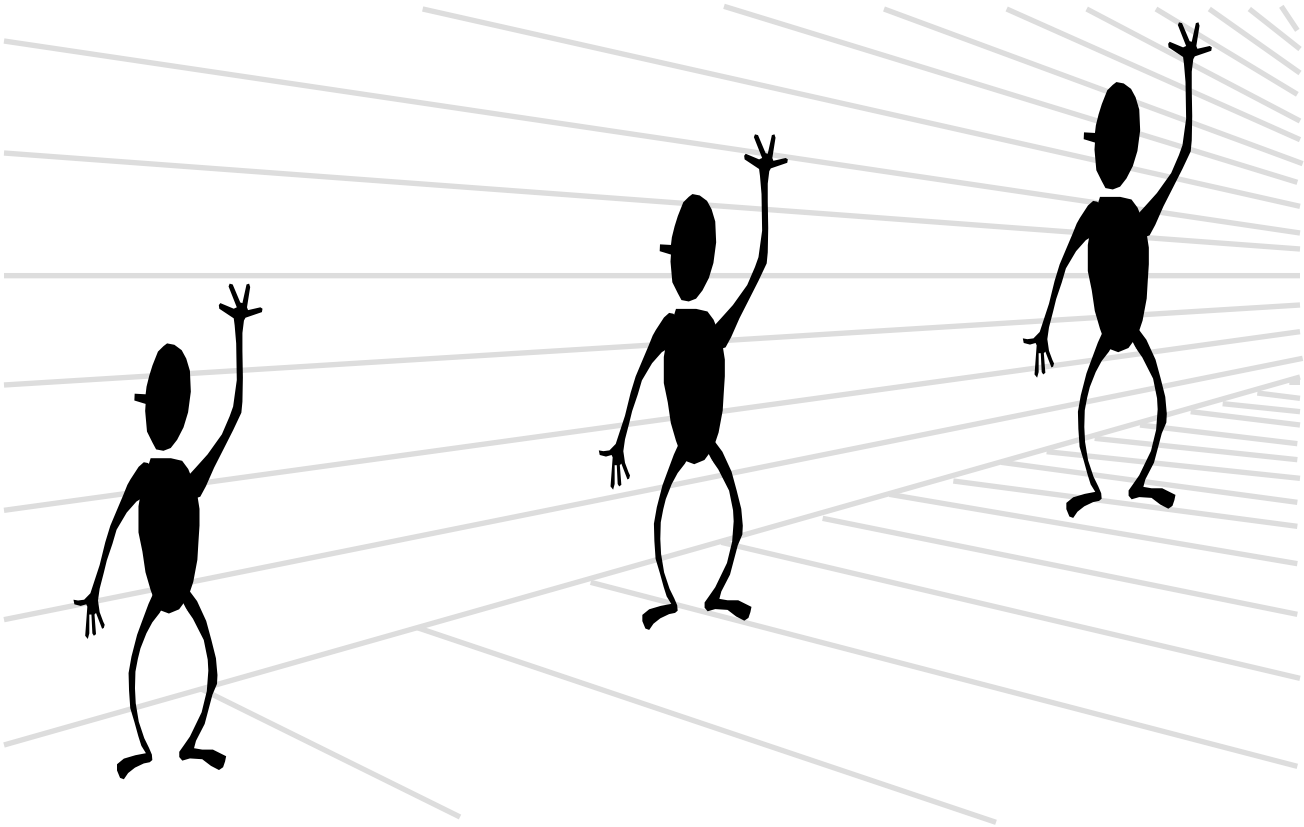


1. Define the ground plane (Z=0)
2. Compute points (X,Y,0) on that plane
3. Compute the *heights* Z of all other points

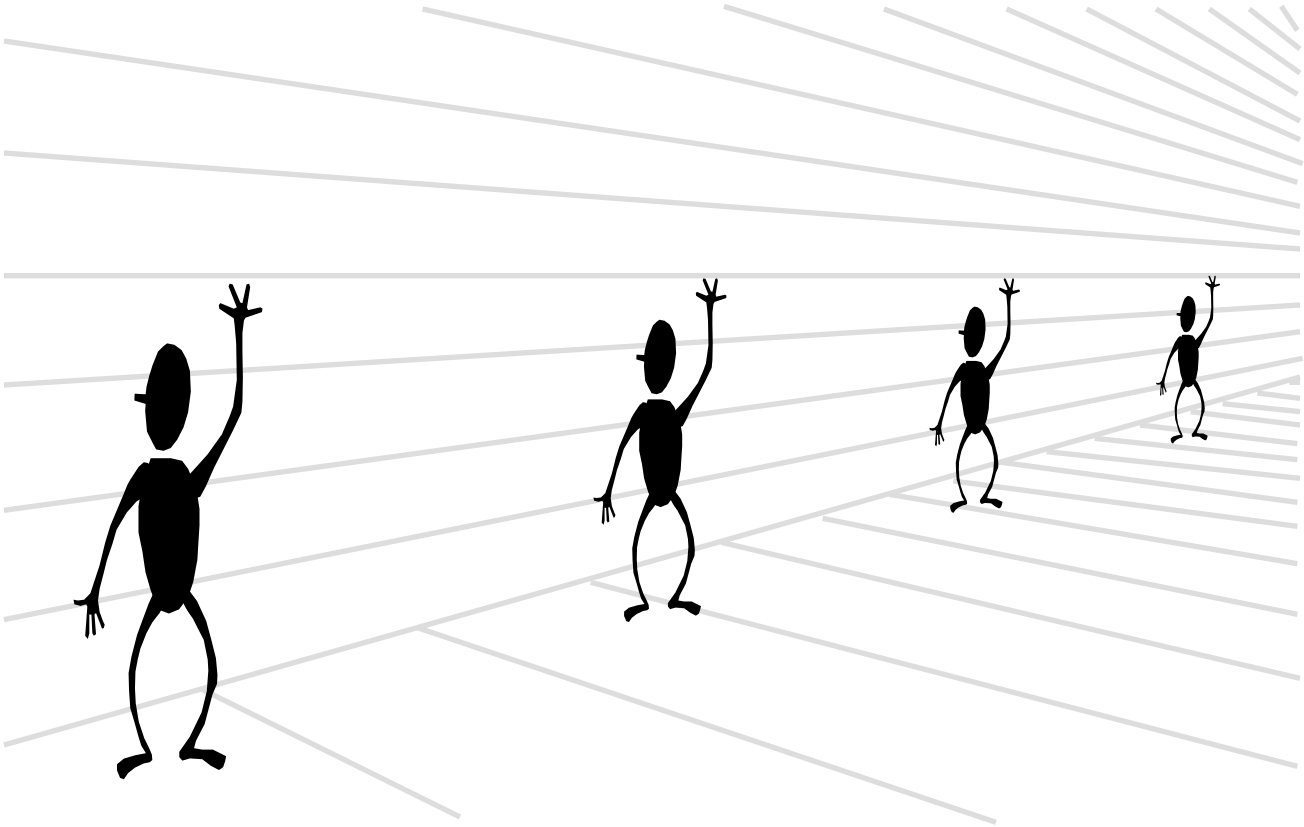
Comparing Heights



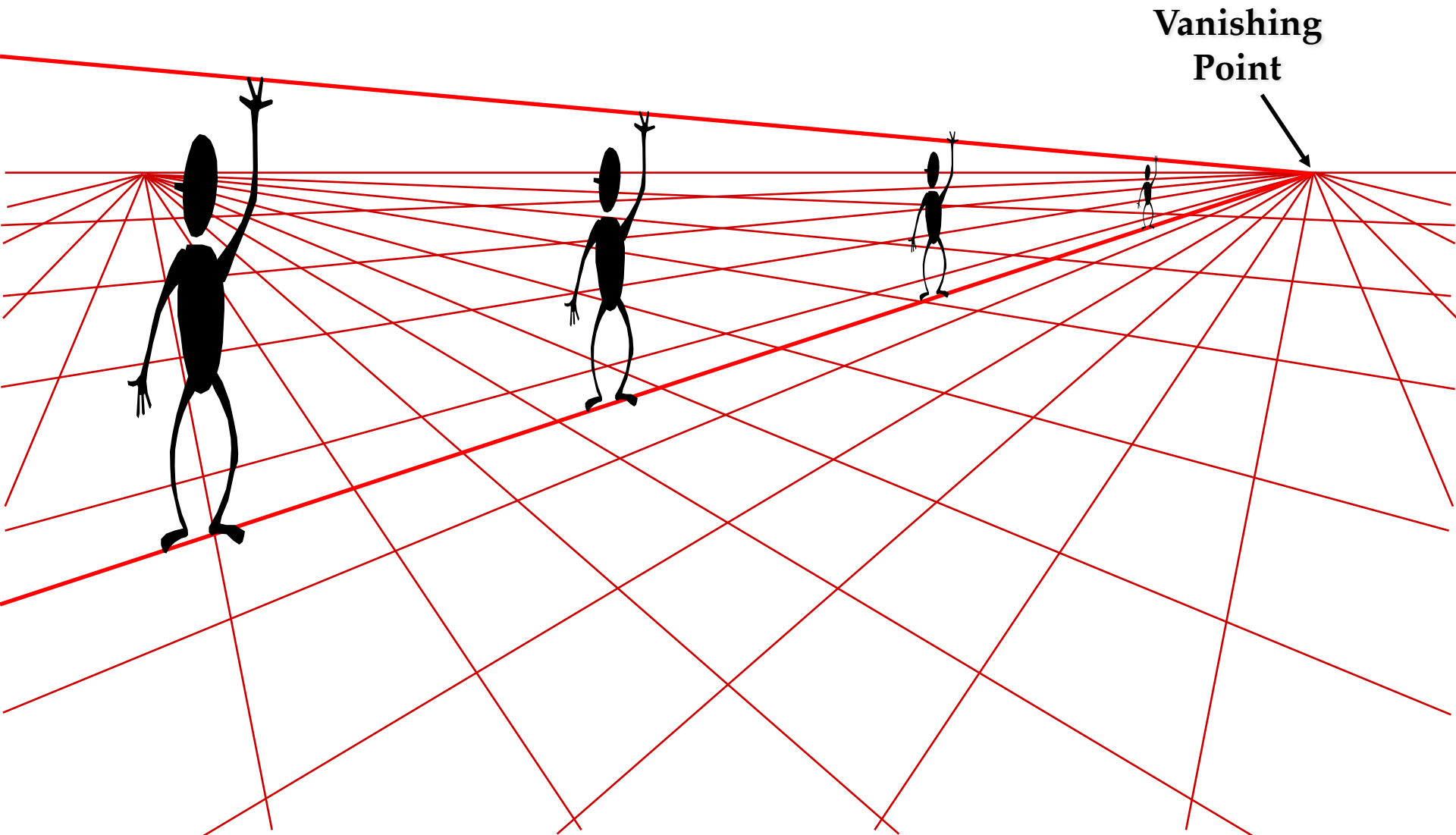
Perspective Cues



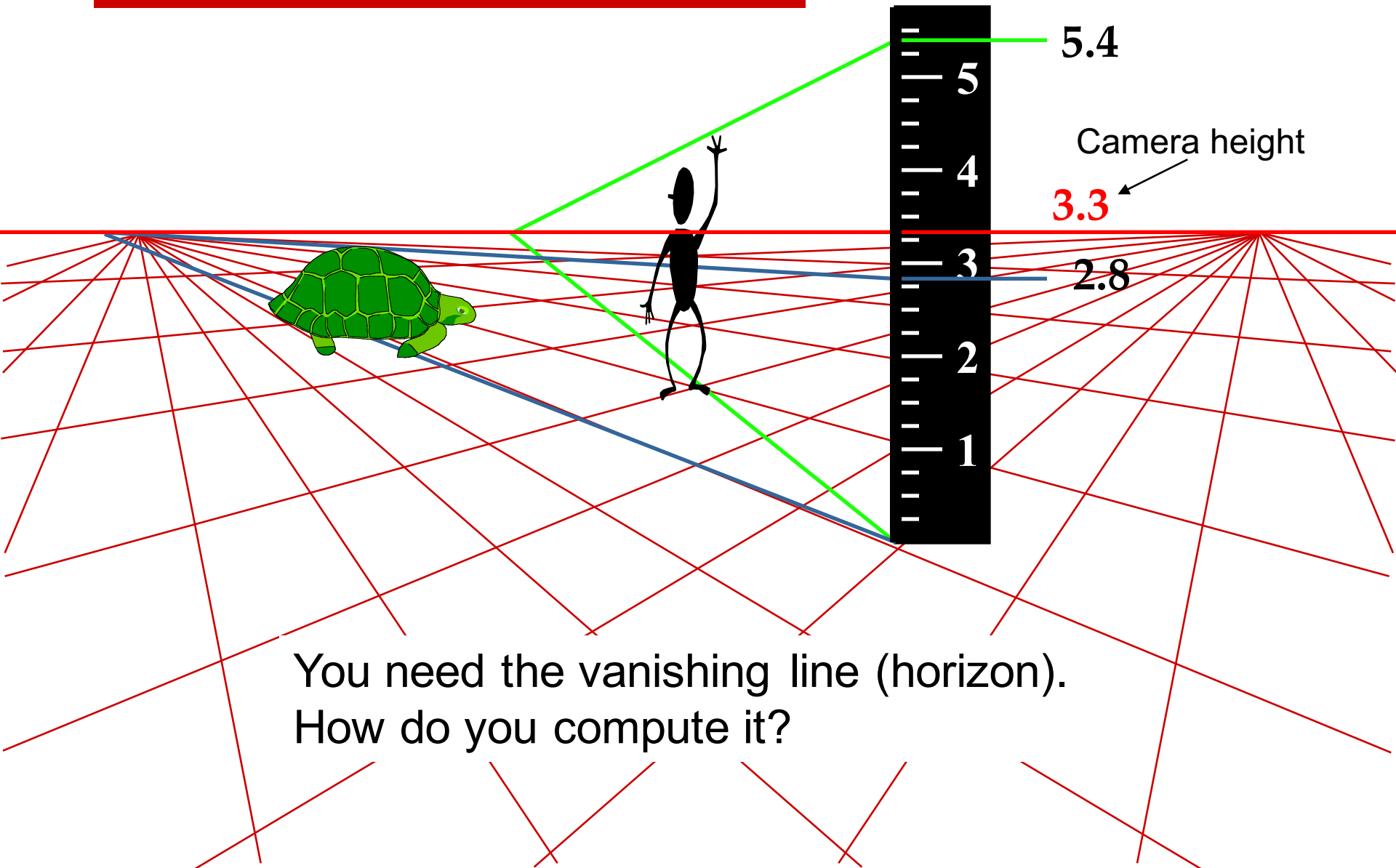
Perspective Cues



Comparing Heights



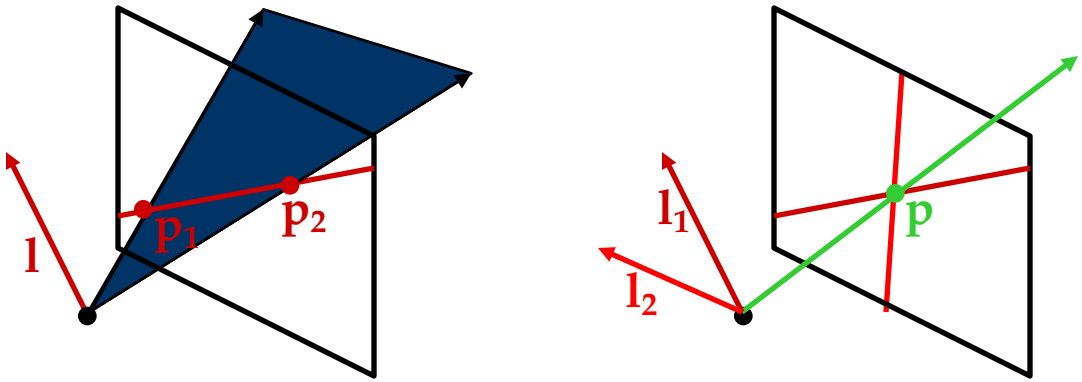
Measuring Height



You need the vanishing line (horizon).
How do you compute it?

Recall: Point and Line Duality

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



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

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

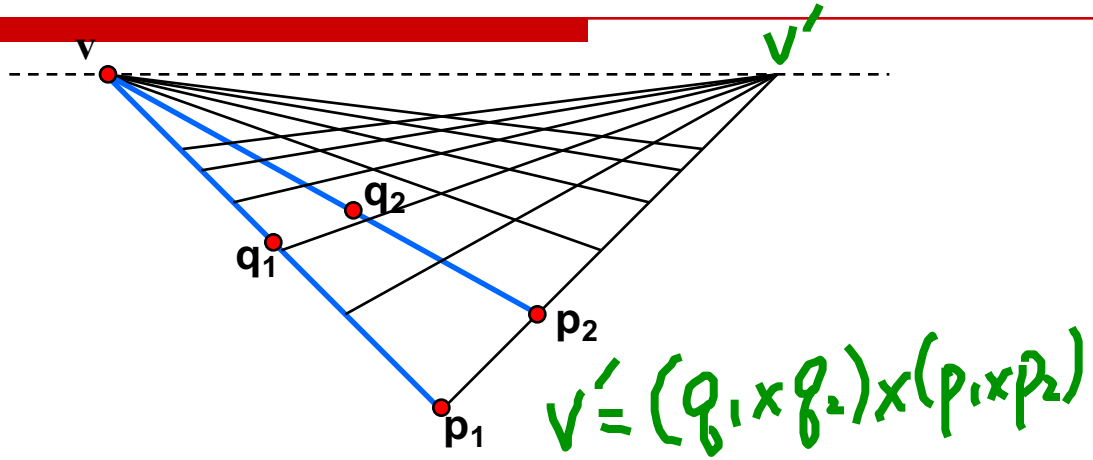
What is the intersection of two lines $\mathbf{l}_1$ and $\mathbf{l}_2$?

- $\mathbf{p}$ is to $\mathbf{l}_1$ and $\mathbf{l}_2$ $\mathbf{p} = \mathbf{l}_1 \quad \mathbf{l}_2$

Points and lines are *dual* in projective space

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

Computing Vanishing Points (from Lines)

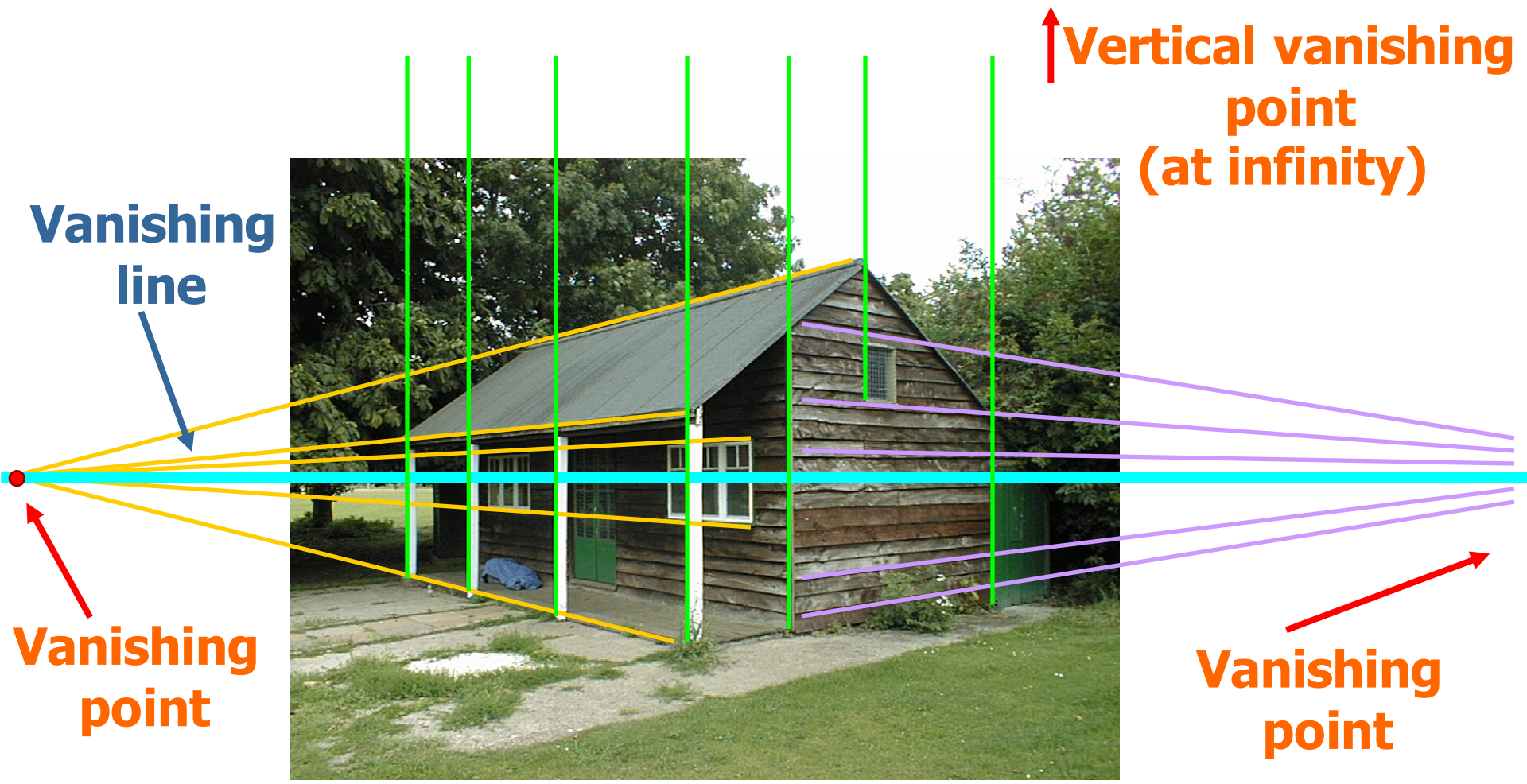


- Intersect p_1q_1 with p_2q_2

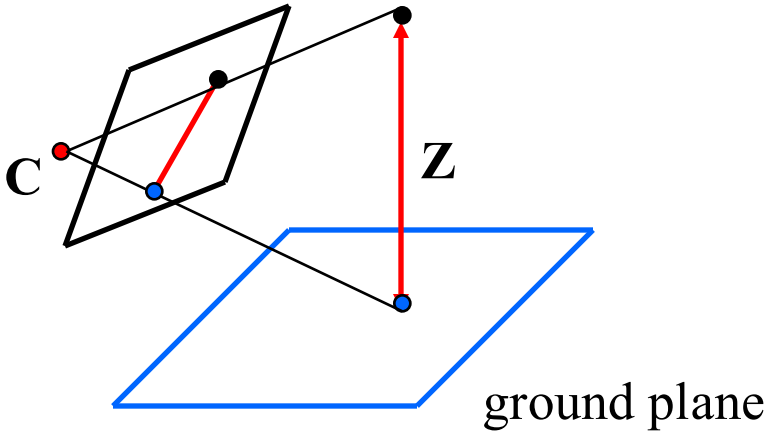
$$v = (p_1 \times q_1) \times (p_2 \times q_2)$$

- Compute another and then join the two
- 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

Criminisi '99

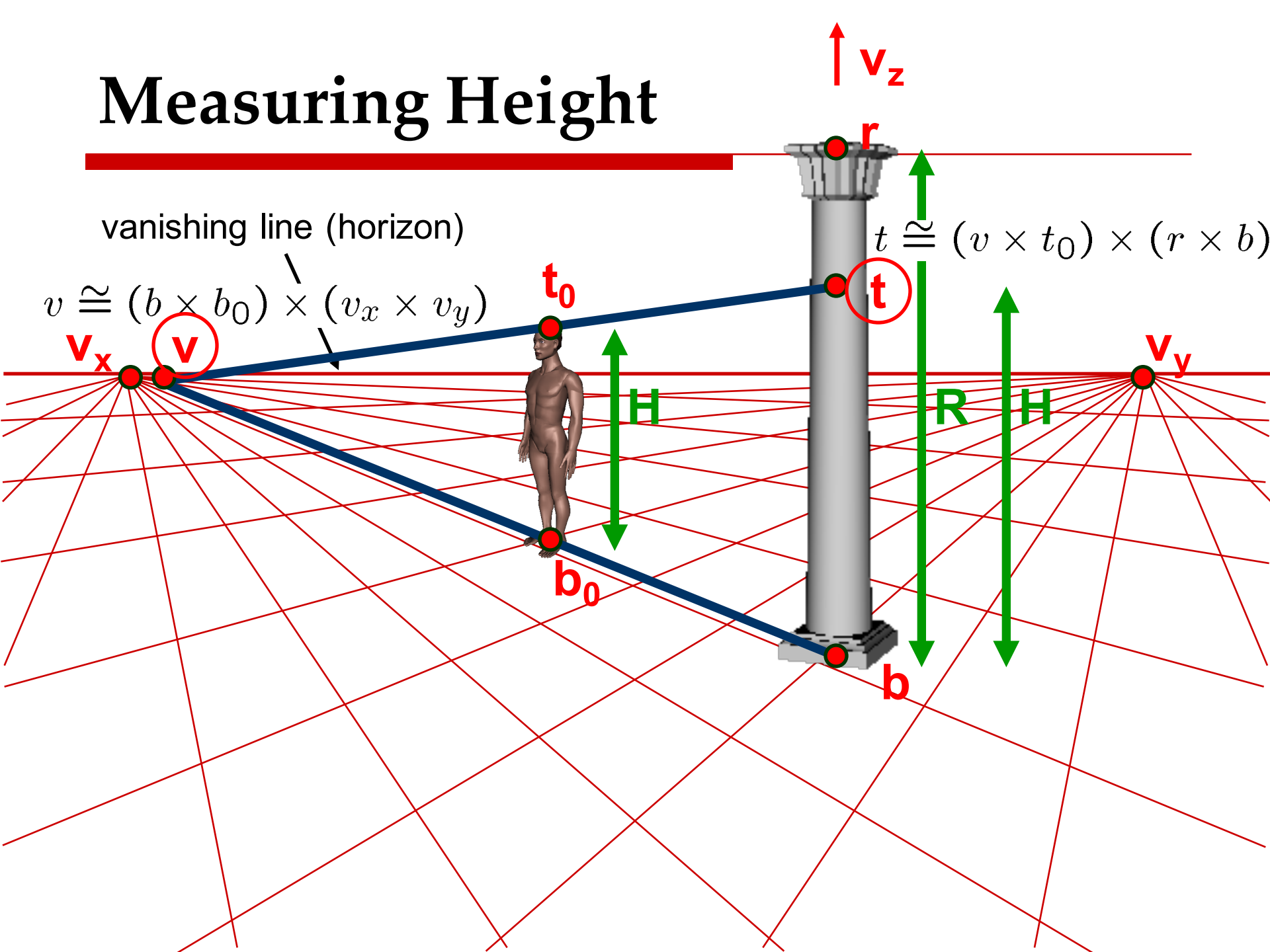


Measuring Height without a Ruler

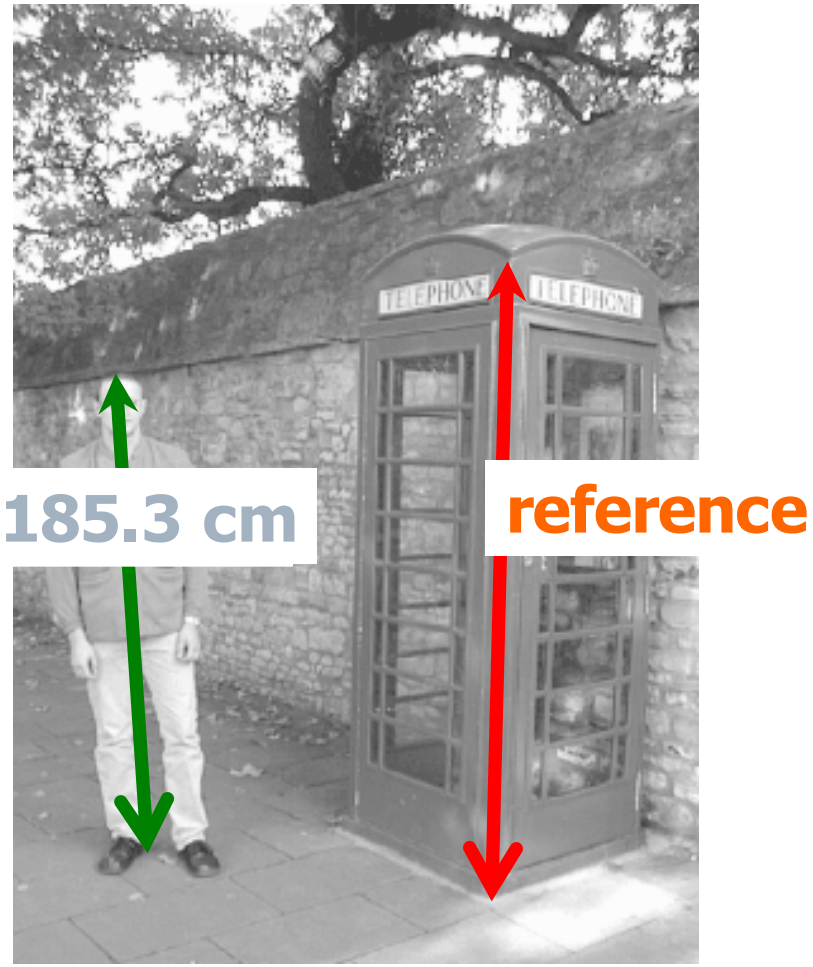


- Compute Z from image measurements
  - Need more than vanishing points to do this

Measuring Height



Measuring Heights of People

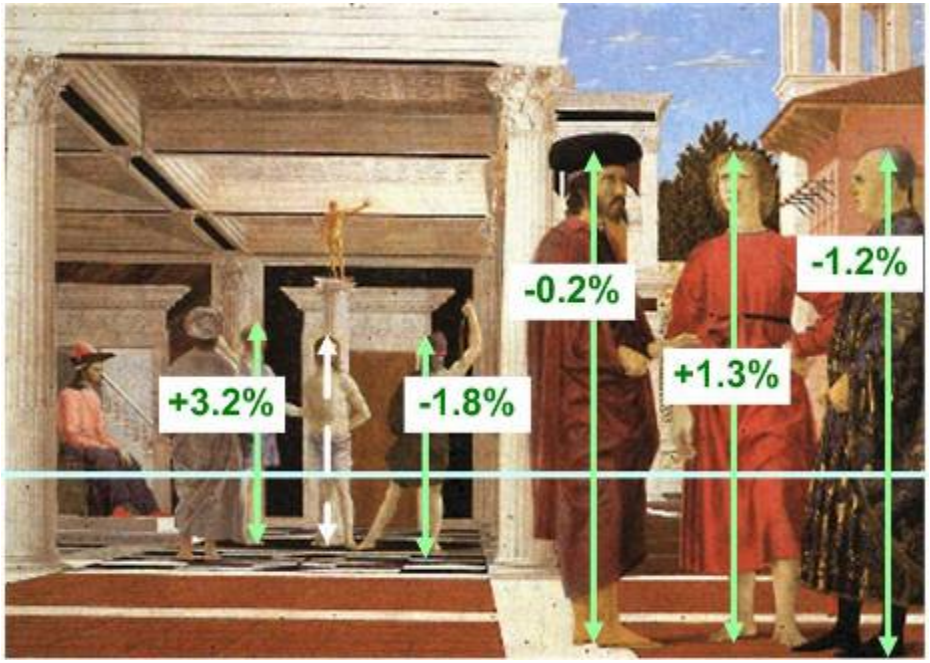


Forensic Science: Measuring Heights of Suspects

Assessing Geometric Accuracy

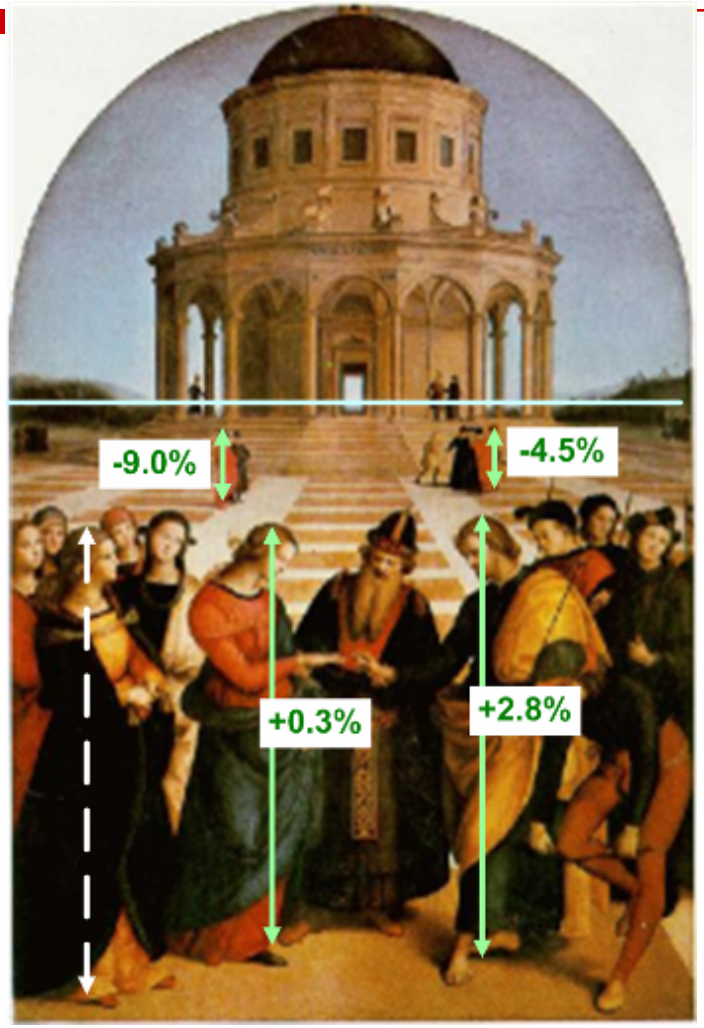
Are the heights of the 2 groups of people consistent with each other?

Flagellation,
Piero della Francesca



Estimated relative heights

Assessing Geometric Accuracy



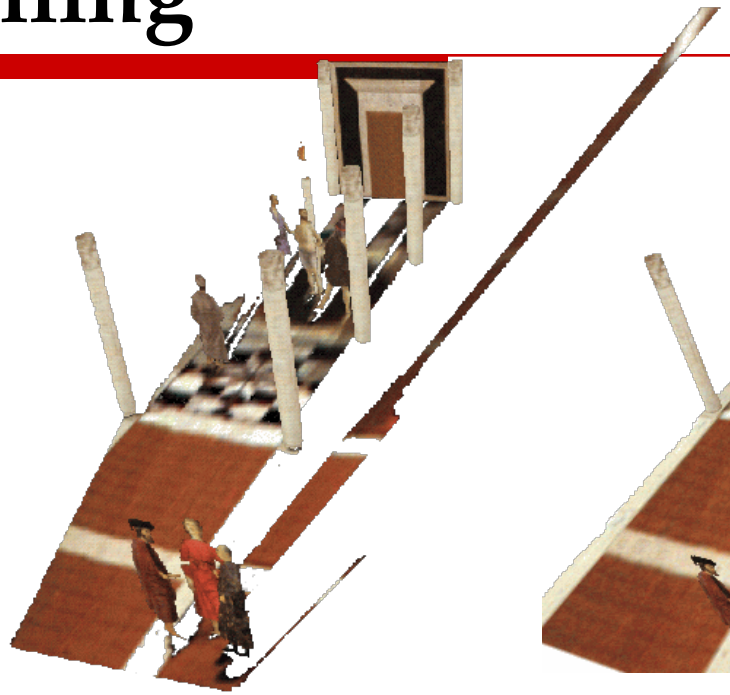
Estimated relative heights

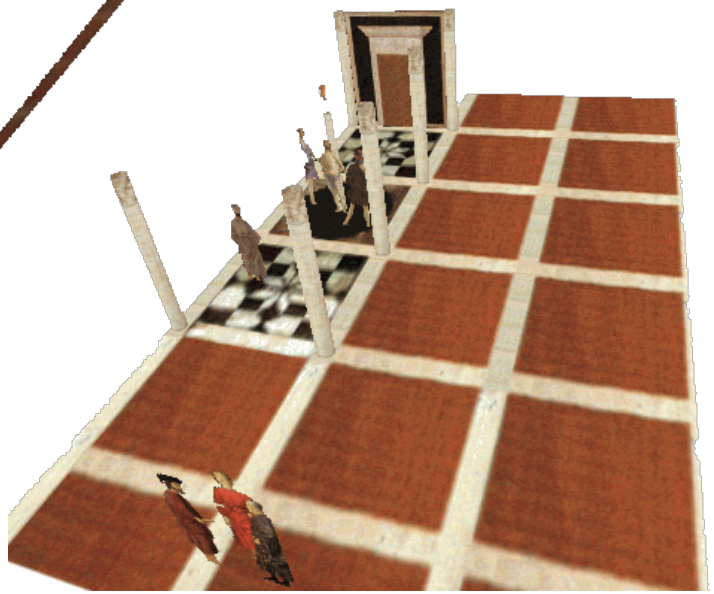
Criminisi et al., ICCV 99

- Complete approach
  - Load in an image
  - Click on lines parallel to X axis
    - repeat for Y, Z axes
  - Compute vanishing points
  - Specify 3D and 2D positions of 4 points on reference plane
  - Compute homography H
  - Specify a reference height
  - Compute 3D positions of several points
  - Create a 3D model from these points
  - Extract texture maps
    - Cut out objects
    - Fill in holes
  - Output a VRML model

Interactive Silhouette Cut-out

Occlusion Filling

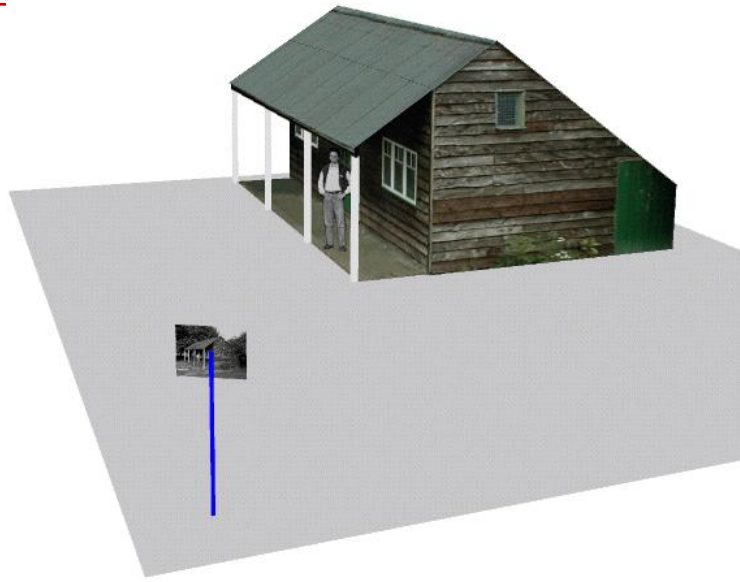




- Geometric filling by exploiting:
  - symmetries
  - repeated regular patterns
- Texture synthesis
  - repeated stochastic patterns

Complete 3D Reconstruction

Single View algorithms



Single image

- Planar measurements
- Height measurements
- Automatic vanishing point/line computation
- Interactive segmentation
- Occlusion filling
- Object placement in 3D model

3D model

Reconstruction from Single Photographs


The Virtual Museum
A. Criminisi @ Microsoft, 2002

"Tour into the Picture" (SIGGRAPH '97)

- Create a 3D "theatre stage" of five billboards

- Specify foreground objects through bounding polygons

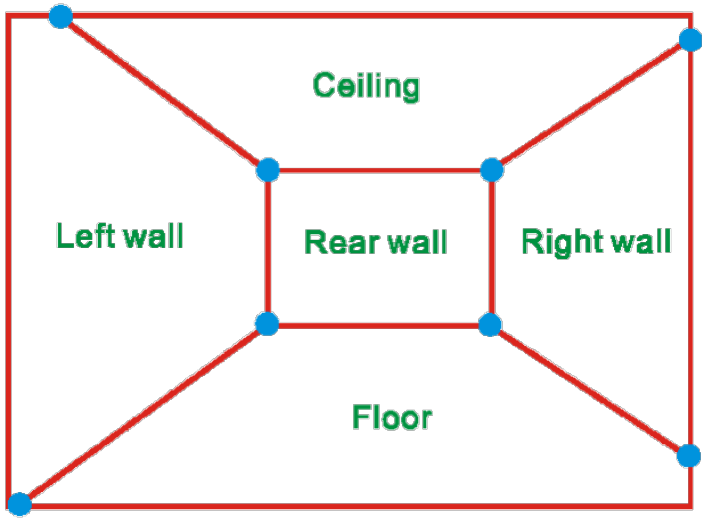
- Use camera transformations to navigate through the scene

The Idea

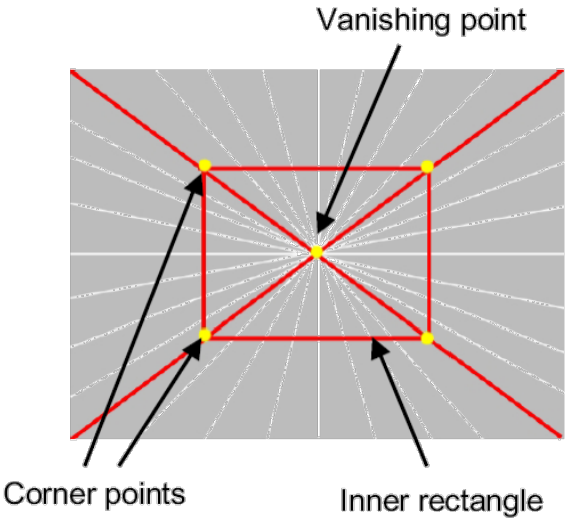
- Many scenes (especially paintings), can be represented as an axis-aligned box volume (i.e. a stage)
- Key assumptions:
  - All walls of volume are orthogonal
  - Camera view plane is parallel to back of volume
  - Camera up is normal to volume bottom

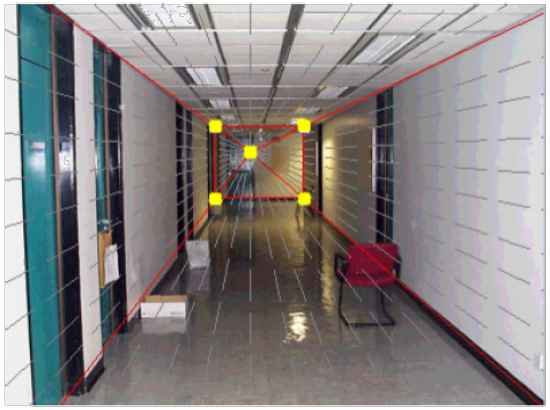
- How many vanishing points does the box have?
  - Three, but two at infinity
  - Single-point perspective

- **Can use a single vanishing point to fit the box to the particular scene!**



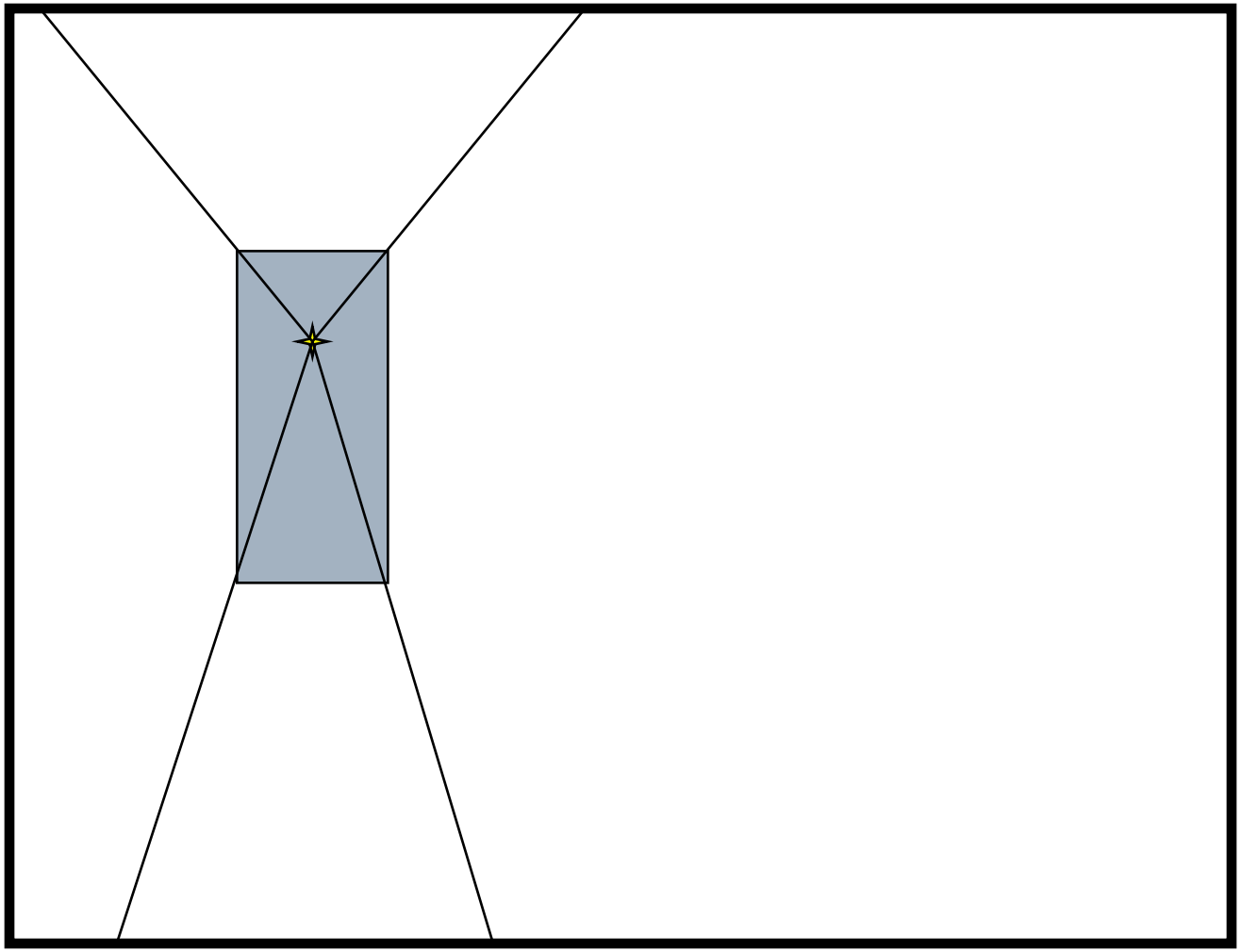
Fitting the Box Volume





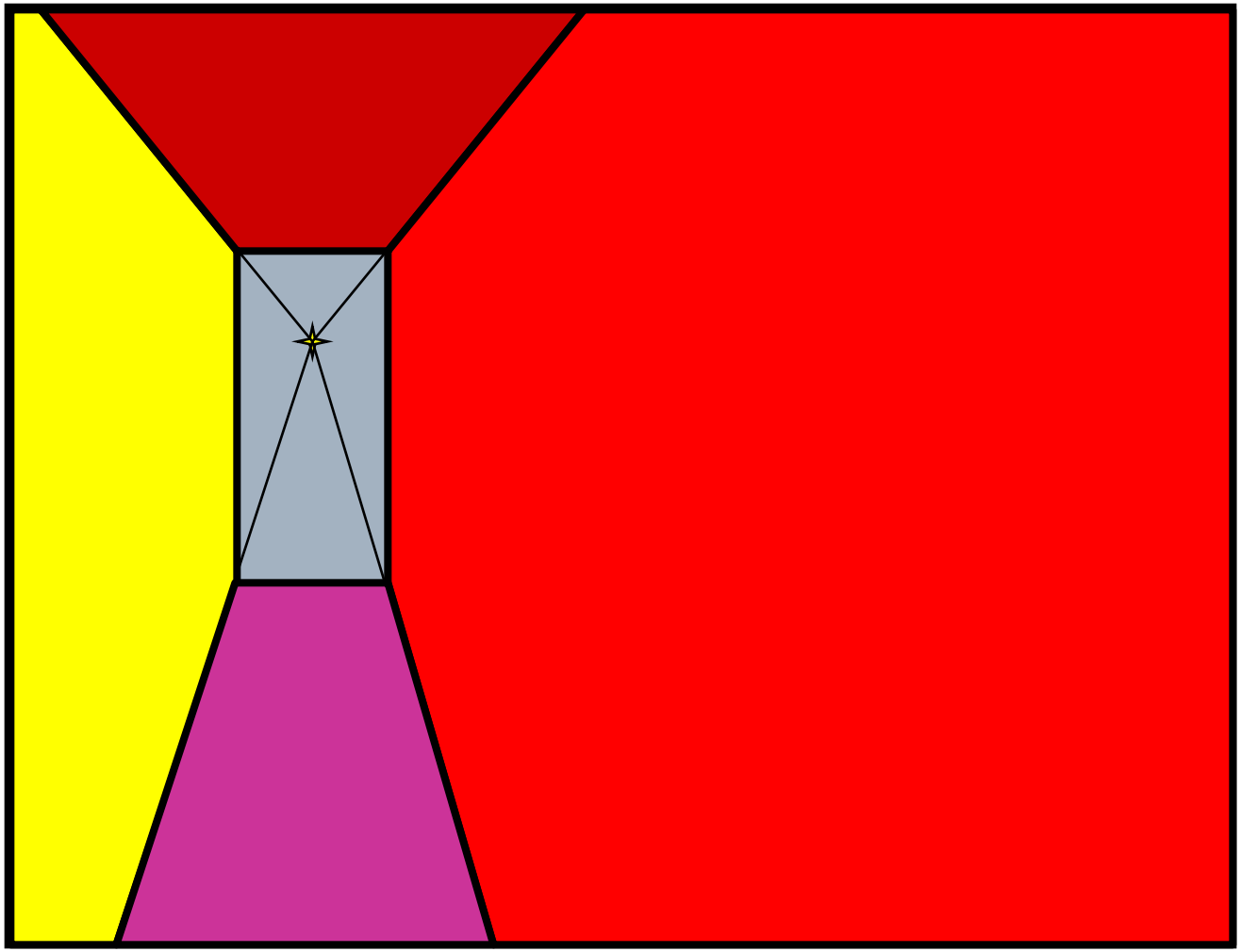
- User controls the inner box and the vanishing point placement
- Q: What's the significance of the vanishing point location?
- A: It's at eye level: ray from COP to VP is perpendicular to image plane.

Example of user input: vanishing point and back face of view volume are defined

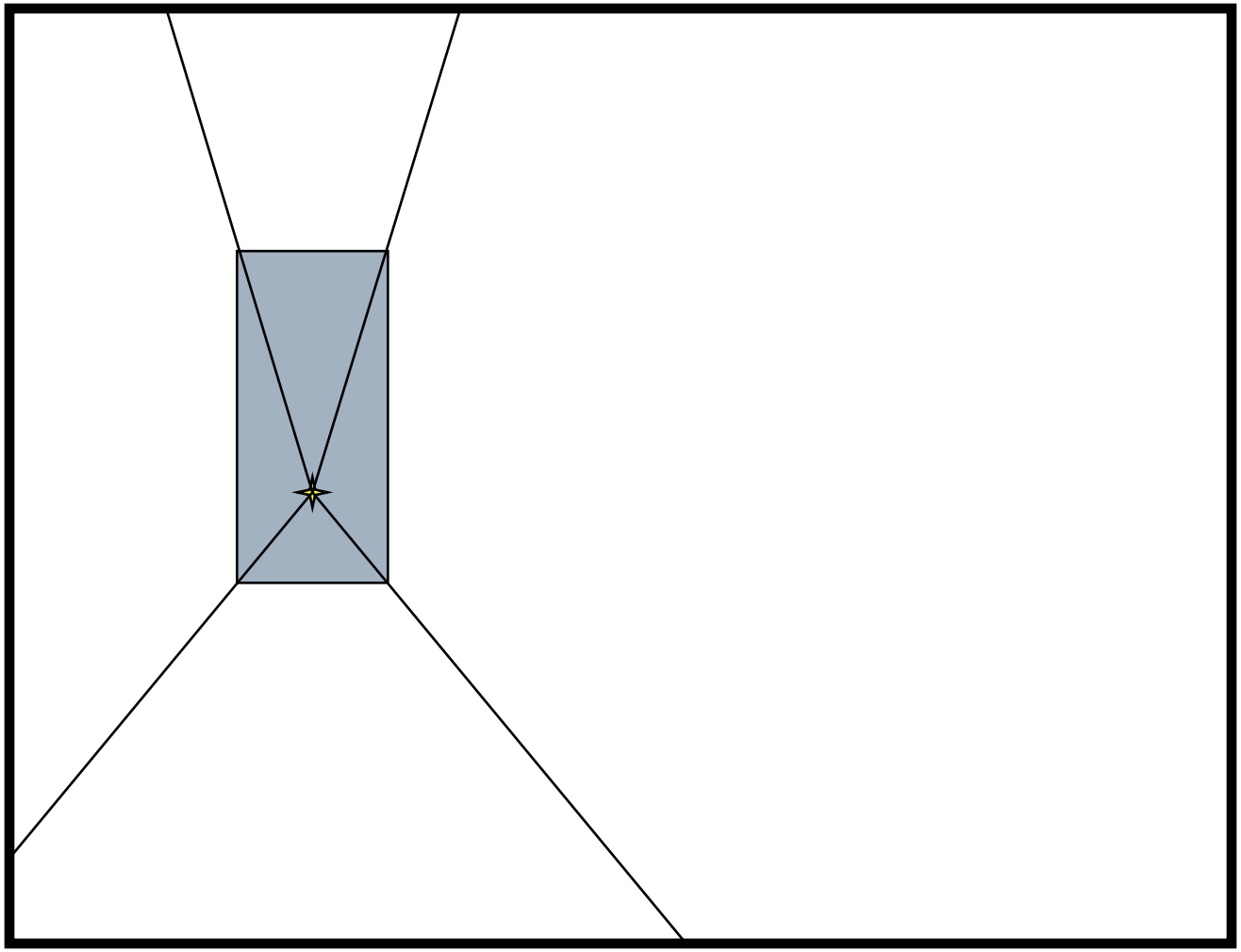


High Camera

Example of user input: vanishing point and back face of view volume are defined

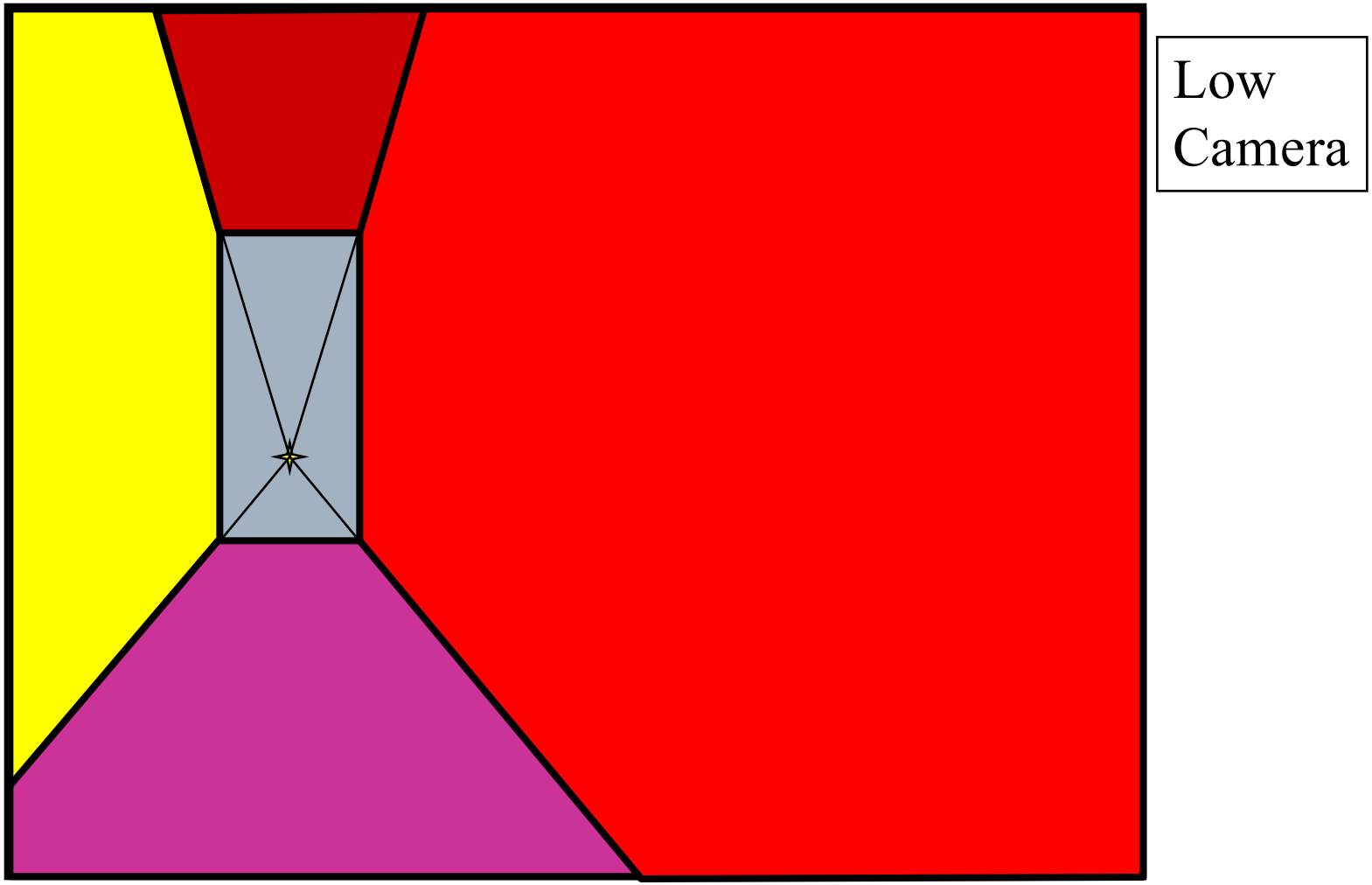


Example of user input: vanishing point and back face of view volume are defined

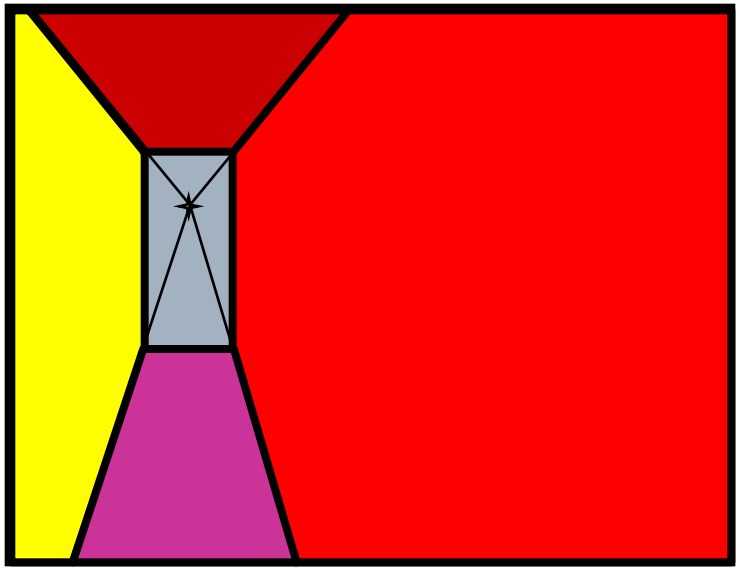


Low Camera

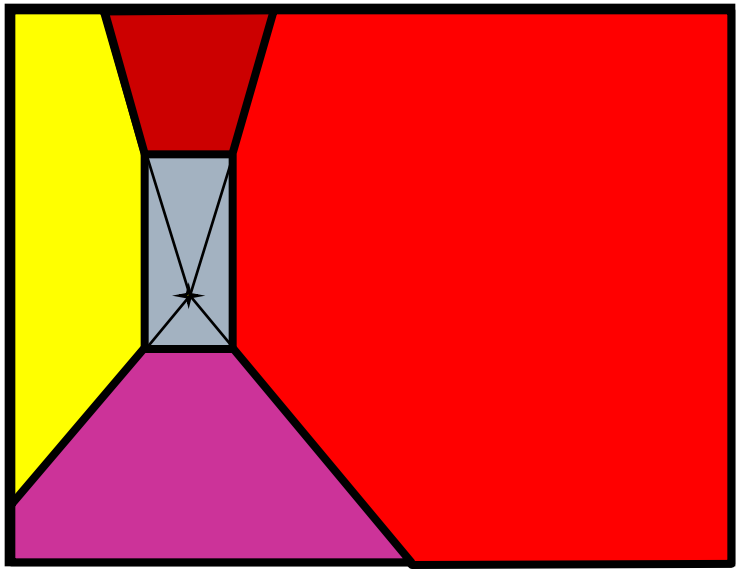
Example of user input: vanishing point and back face of view volume are defined



Comparison of how image is subdivided based on two different camera positions. You should see how moving the vanishing point corresponds to moving the eyepoint in the 3D world.

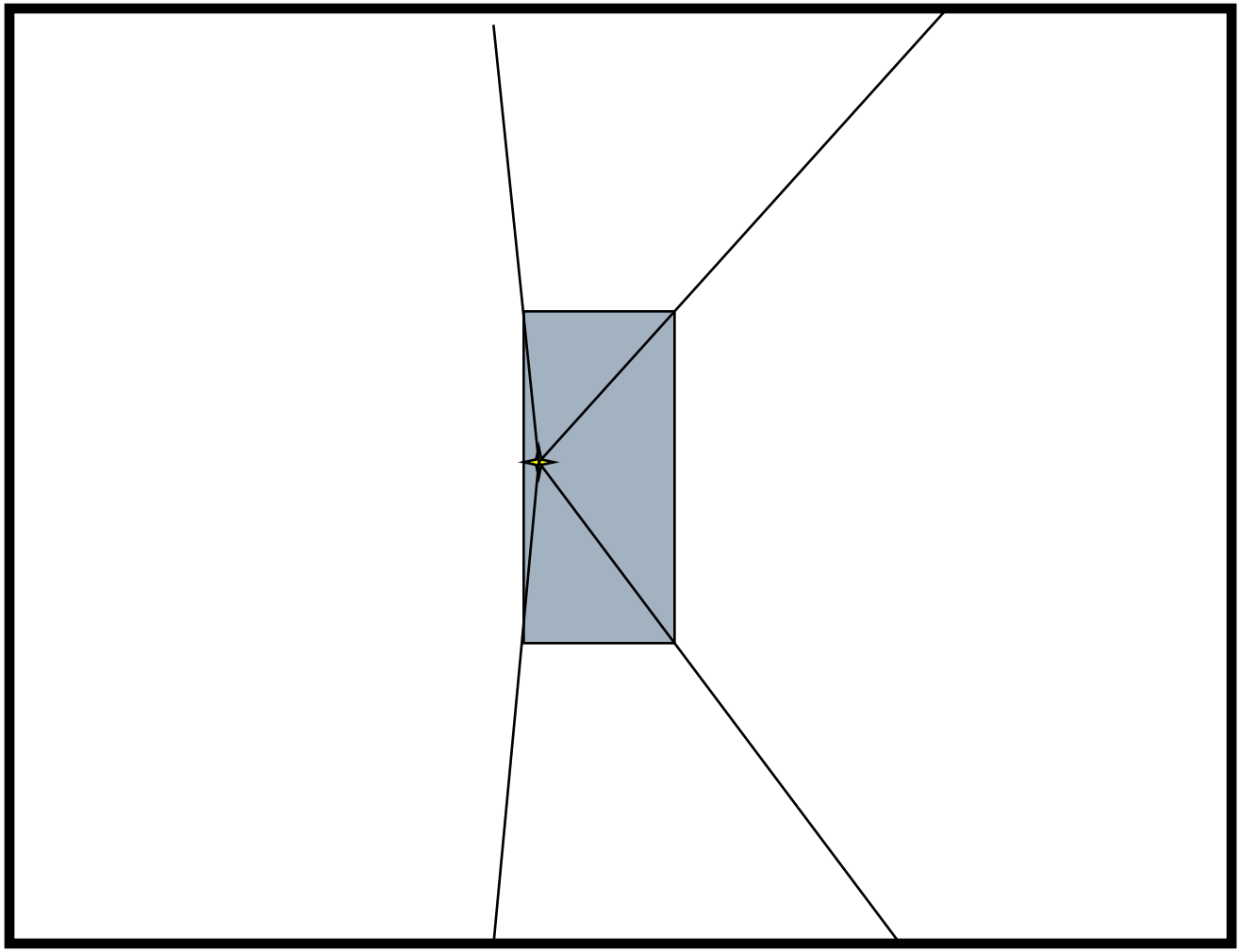


High Camera

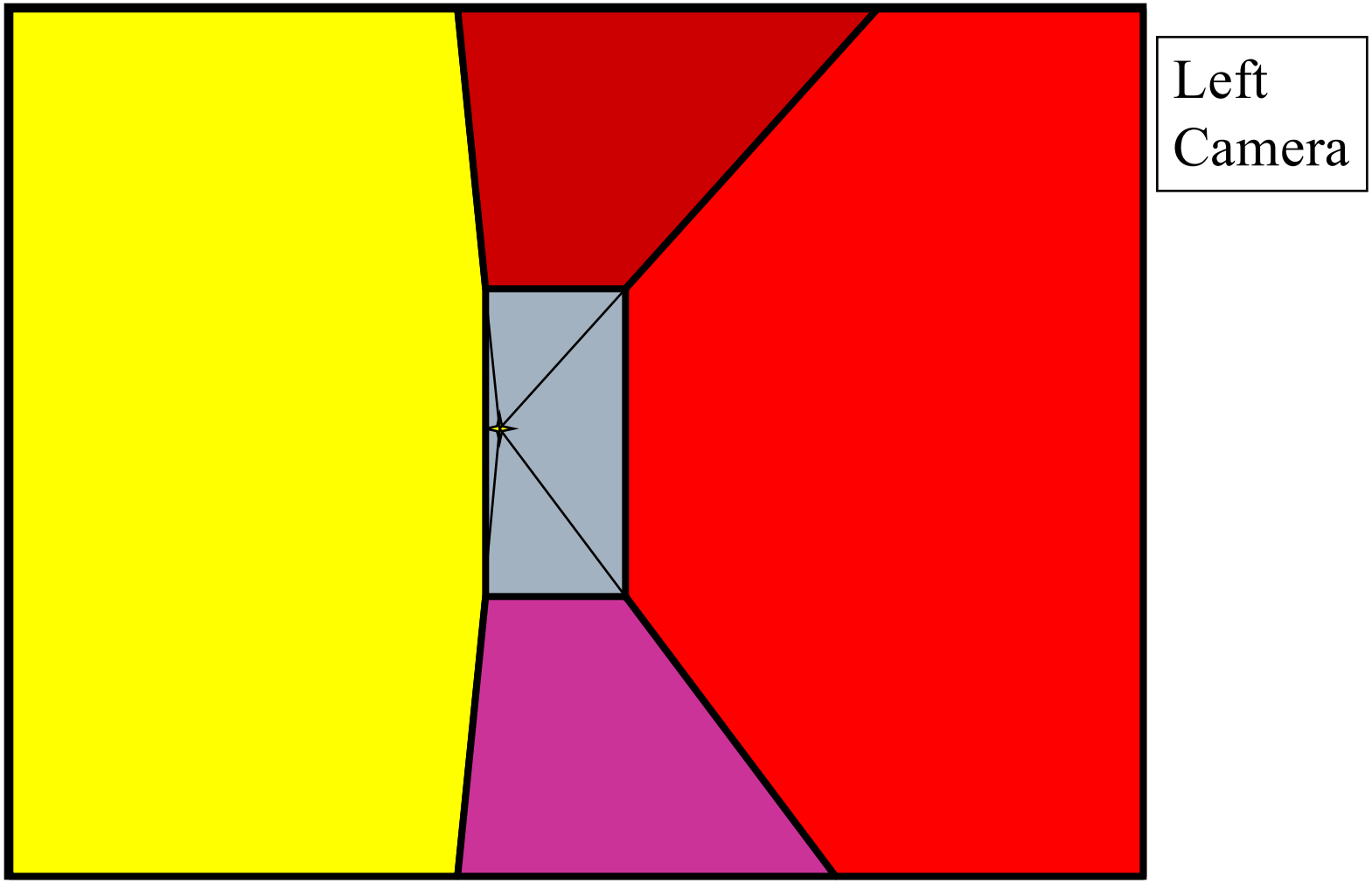


Low Camera

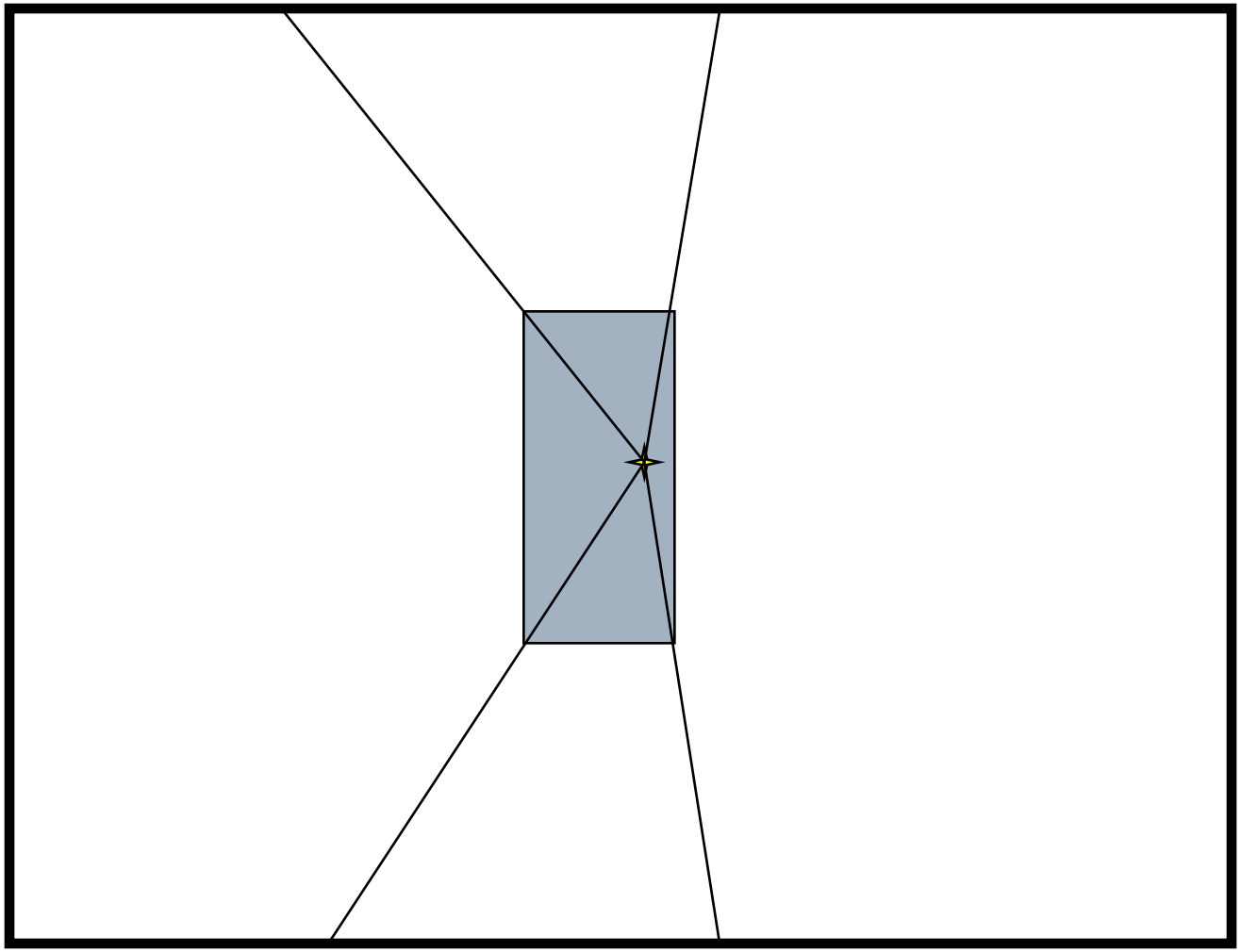
Another example of user input: vanishing point and back face of view volume are defined



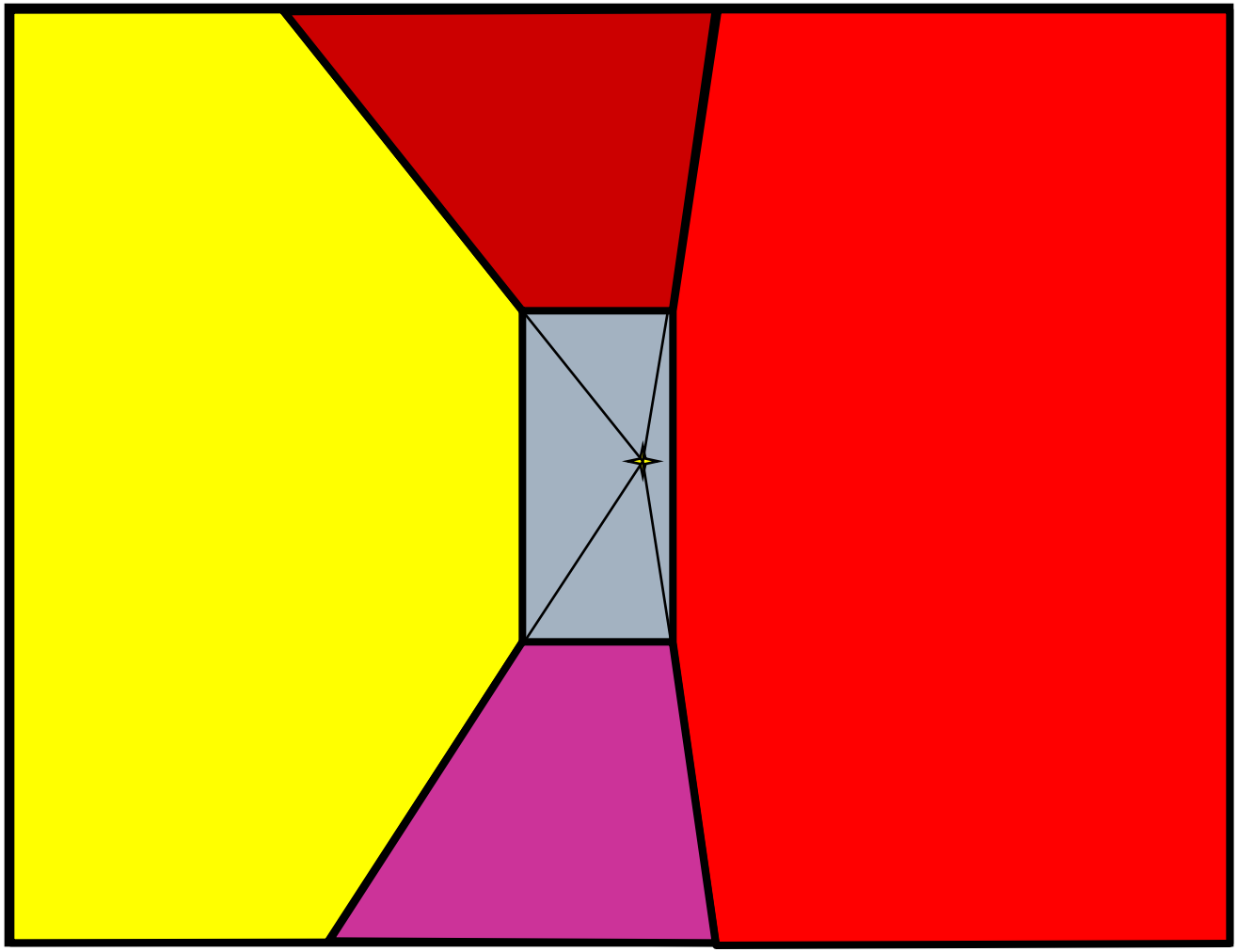
Another example of user input: vanishing point and back face of view volume are defined



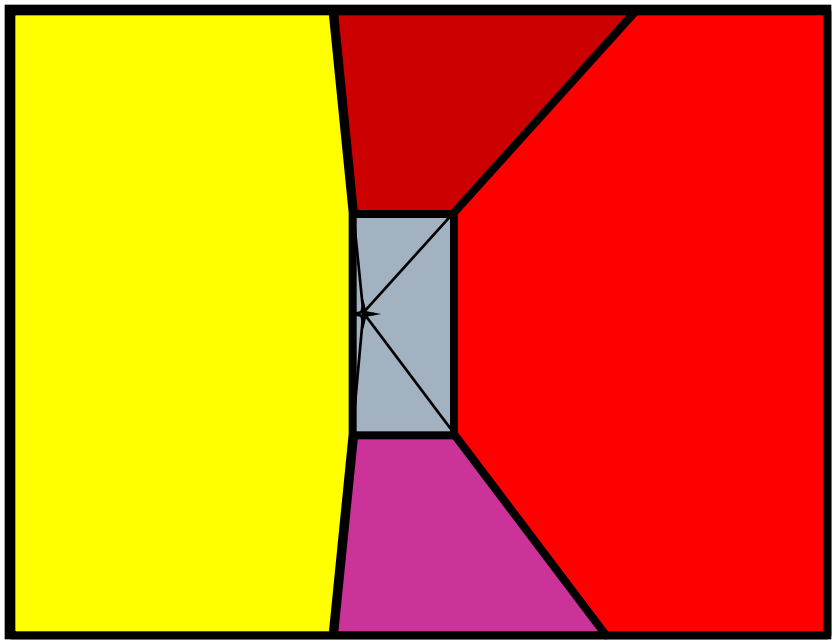
Another example of user input: vanishing point and back face of view volume are defined



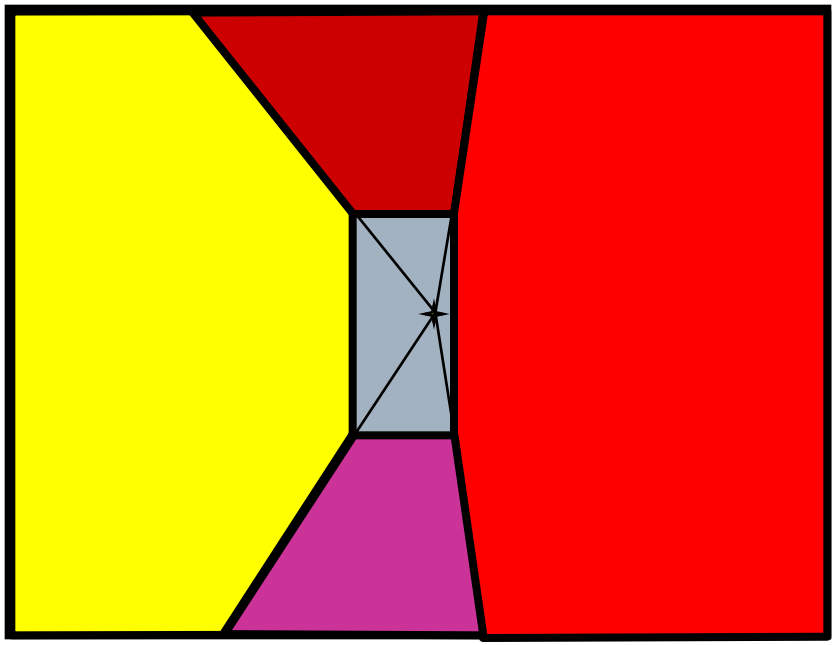
Another example of user input: vanishing point and back face of view volume are defined



Comparison of two camera placements – left and right. Corresponding subdivisions match view you would see if you looked down a hallway.



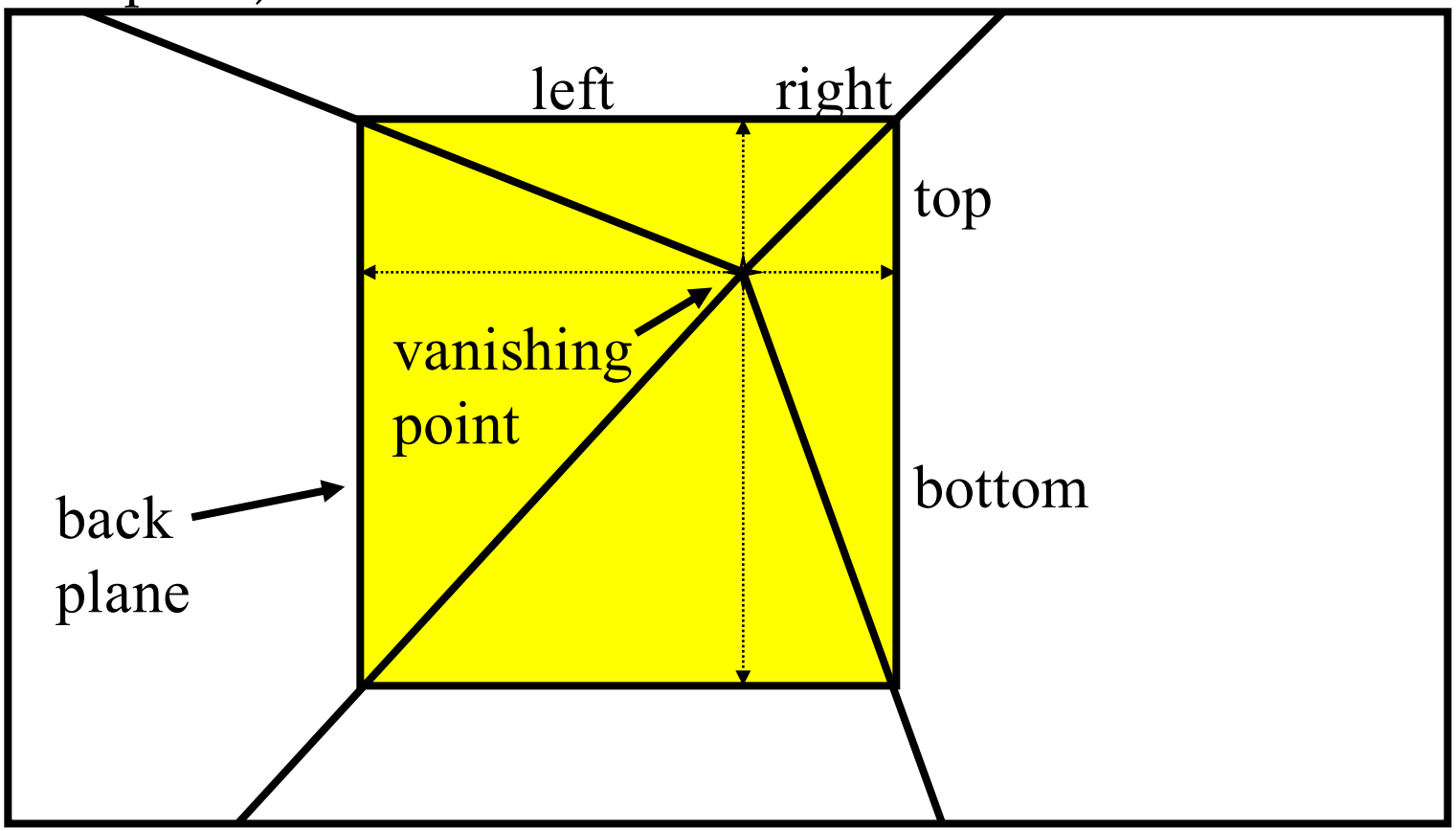
Left Camera



Right Camera

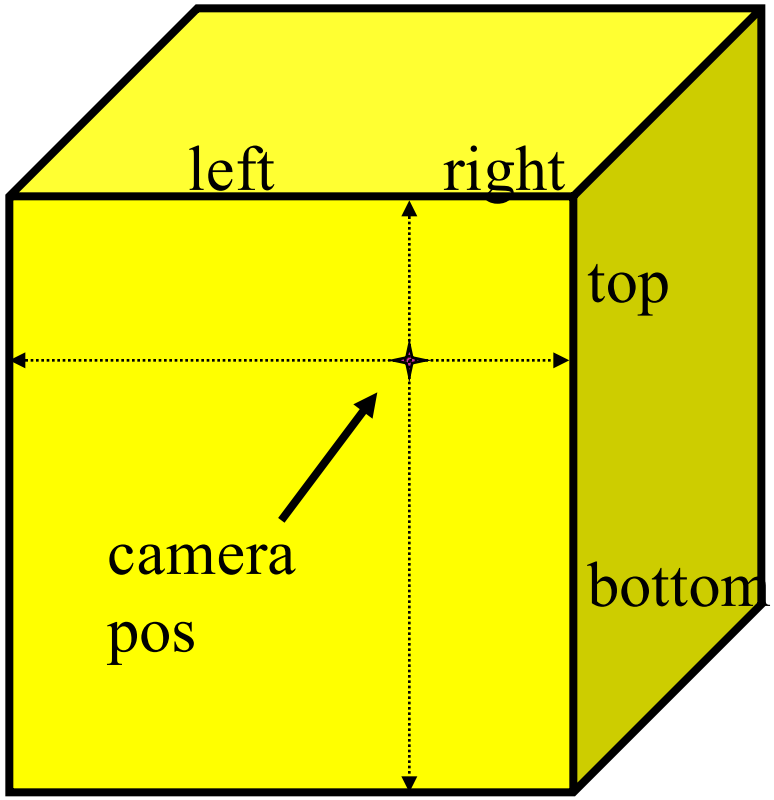
2D to 3D Conversion

- First, we can get ratios
  - All five points can be computed from manually provided three points (lefttop, right bottom, vanishing point)

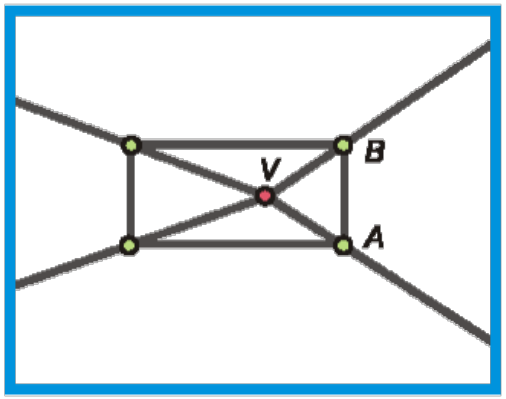


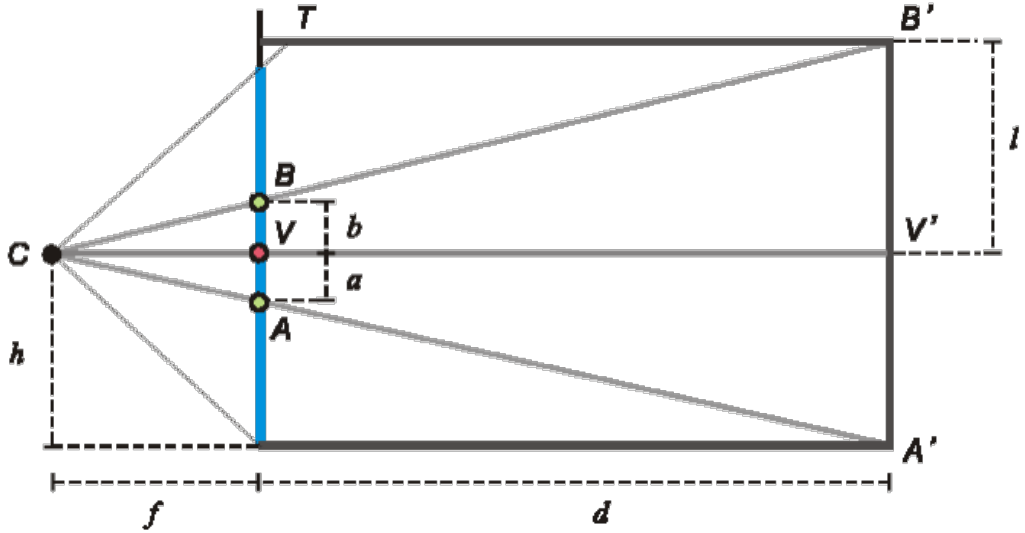
2D to 3D Conversion

- Size of user-defined back plane must equal size of camera plane (orthogonal sides)
- Use top versus side ratio to determine relative height and width dimensions of box
- Left/right and top/bot ratios determine part of 3D camera placement



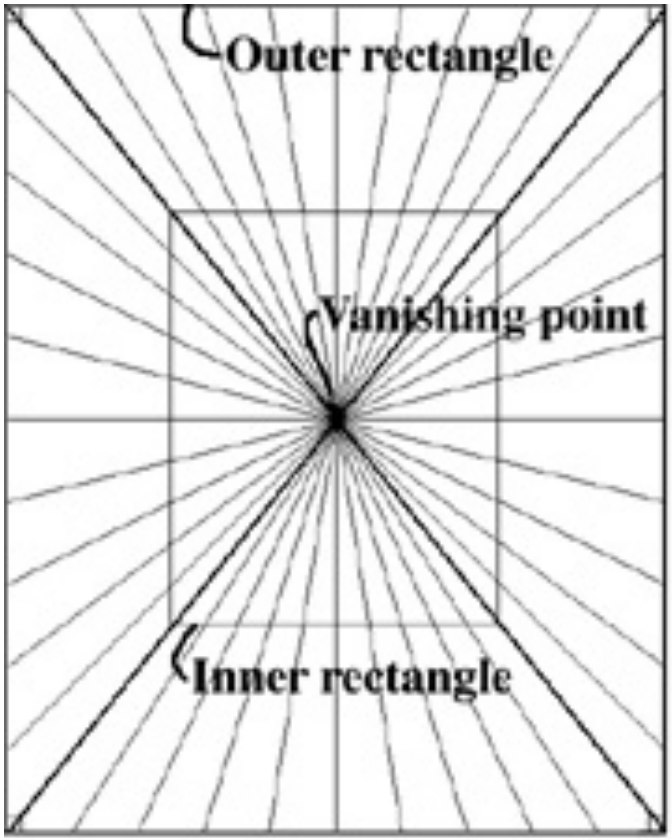
Depth of the Box



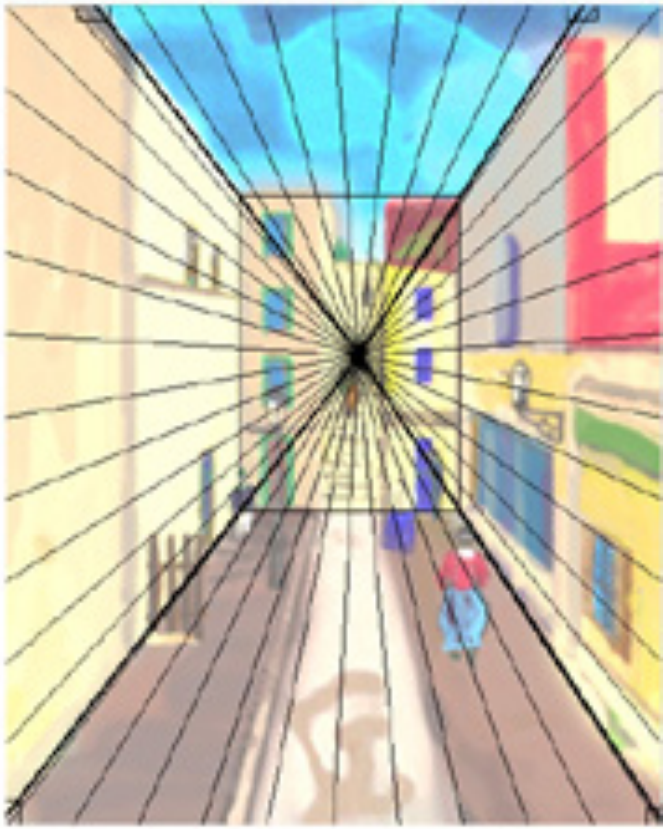


- Can compute by similar triangles (CVA vs. CV′A′)
- Need to know focal length f (or FOV)

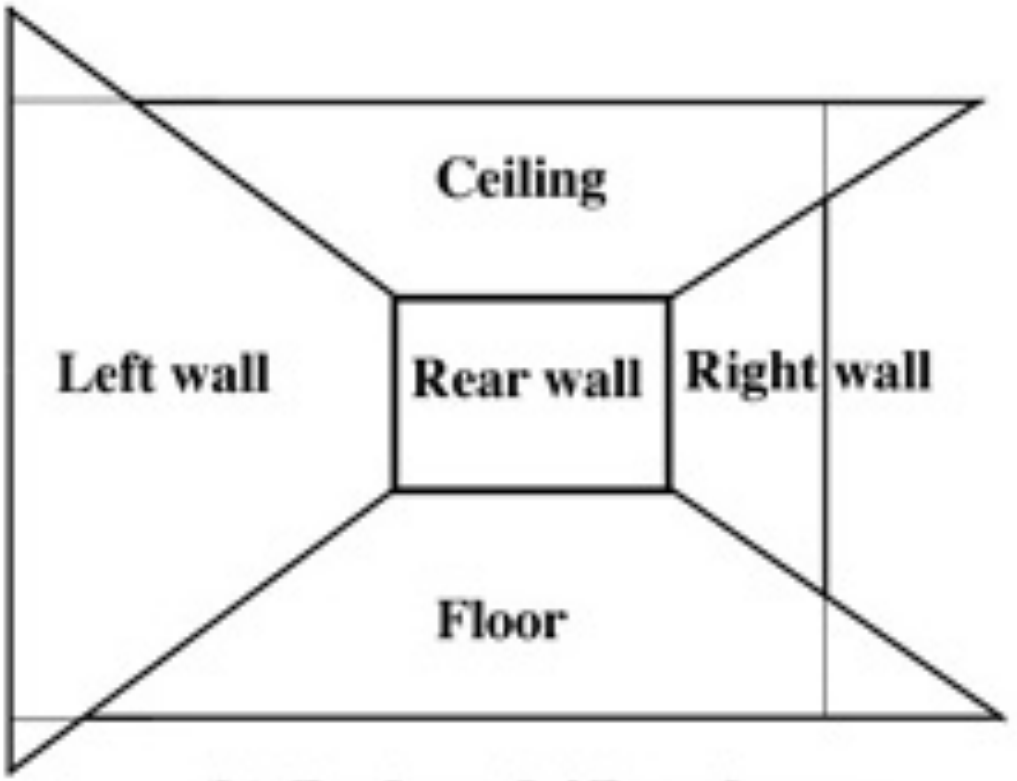
- Note: can compute position on any object on the ground
  - Simple unprojection
  - What about things off the ground?



(a) Initial state

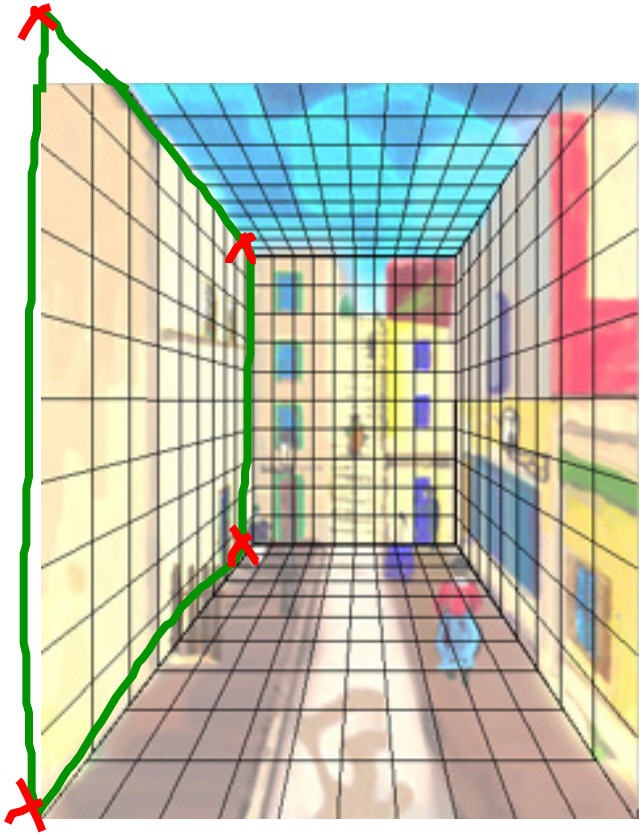


(b) Specification result

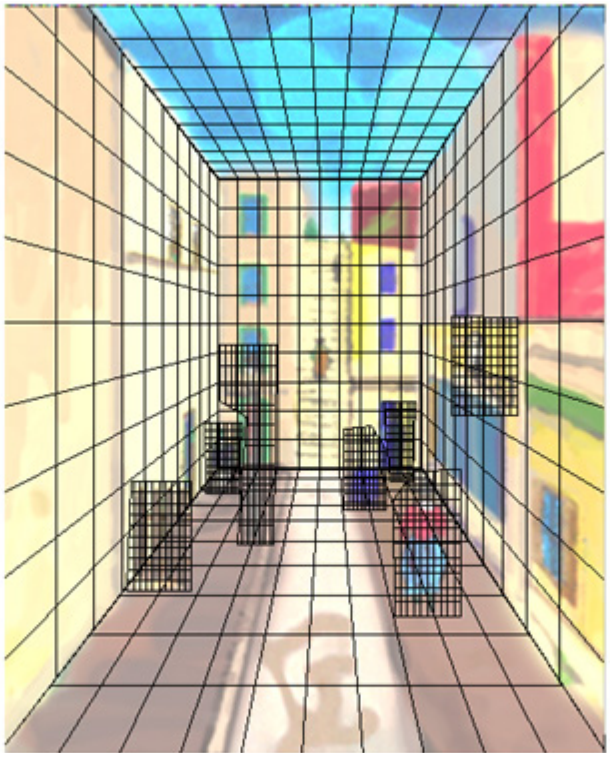


(b) Deduced 2D polygon

Pay attention to the corners coming out of the image!

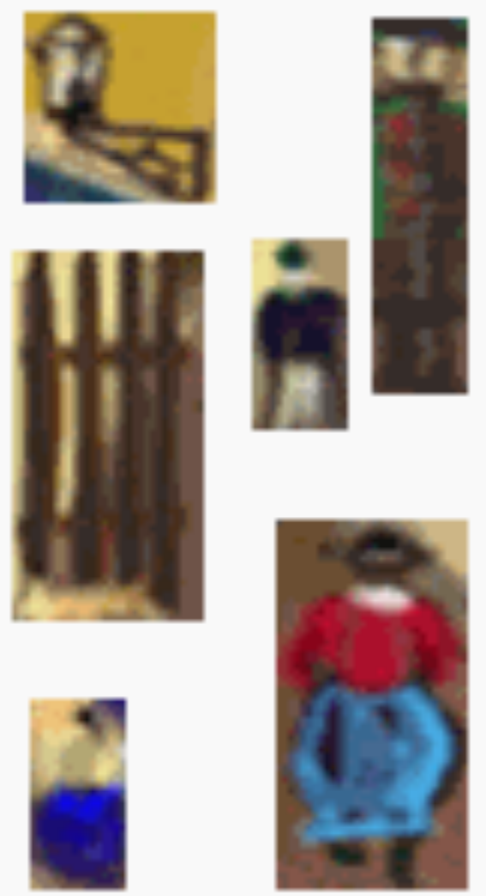


Background Model



Foreground with meshes

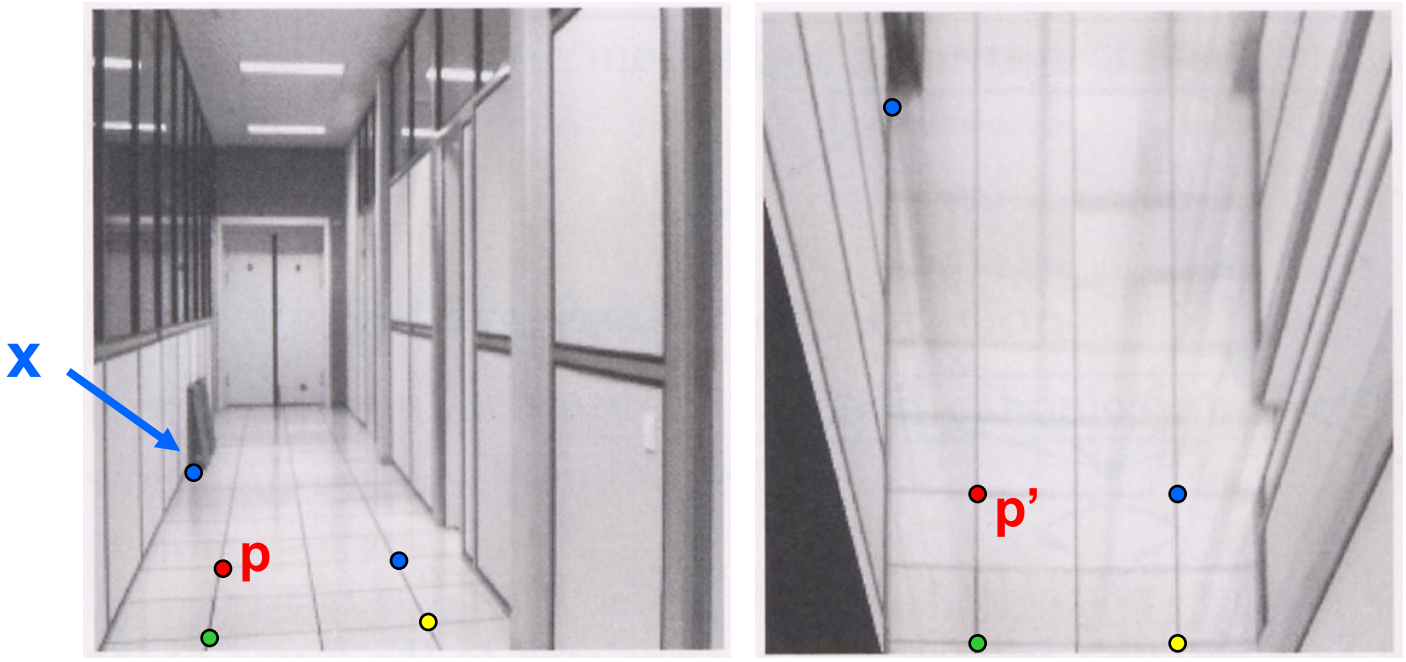
Background



Foreground

We need to texture map each wall!

Unwarp Ground Plane

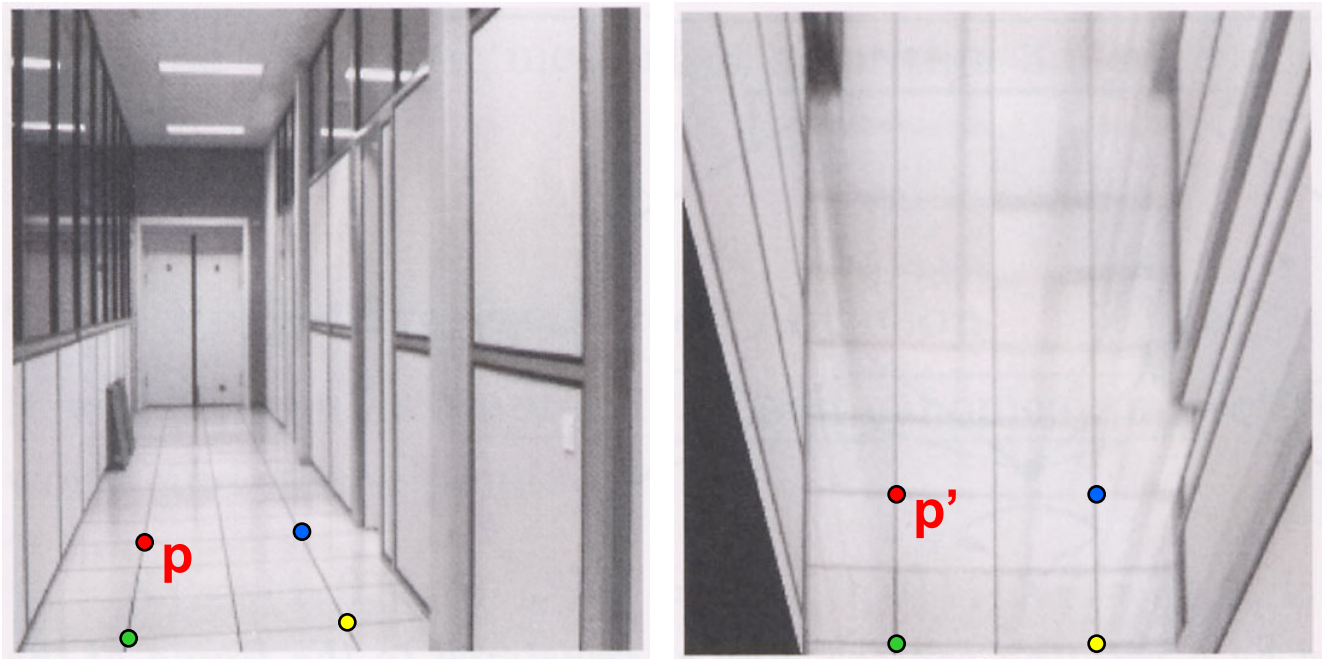


Called **Homography**

Need 4 reference points

Don’t forget to inverse warp!!

Homography



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**?

Solving for Homographies

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

$$\begin{aligned} x_i' &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y_i' &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

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

Solving for Homographies

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{A}_{2n \times 9} \qquad \mathbf{h}_{9} \qquad \mathbf{0}_{2n}$$

- Total least squares
  - Since **h** is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$ $\|\mathbf{A}\hat{\mathbf{h}}\|^2$
  - Minimize $\|\mathbf{A}\hat{\mathbf{h}}\|^2 = (\mathbf{A}\hat{\mathbf{h}})^T\mathbf{A}\hat{\mathbf{h}} = \hat{\mathbf{h}}^T\mathbf{A}^T\mathbf{A}\hat{\mathbf{h}}$
  - Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
  - Works with 4 or more points (more points more accurate)

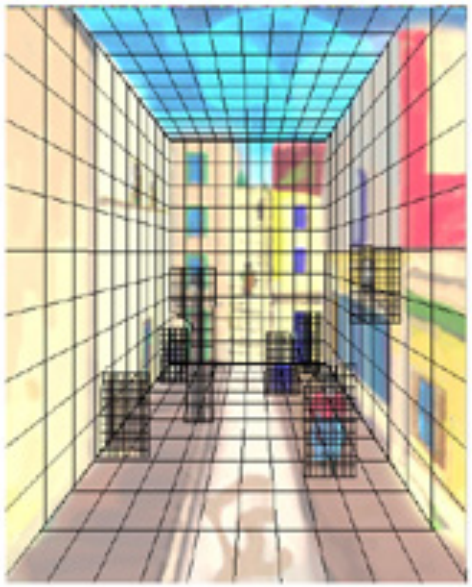
Texture map the walls

- Give the correspondences
- Solve for the homography
- Inverse warp using the inverse of the estimated homography (or compute the inverse from the beginning)
- Don’t forget to bilinearly interpolate

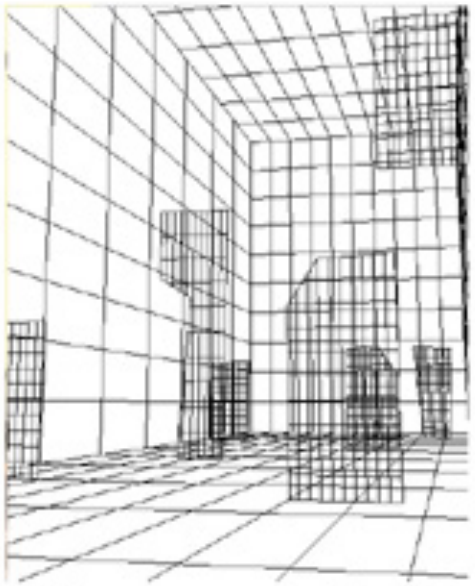
(a) Input image

(b) Background

(c) Foreground mask



(f) Foreground model



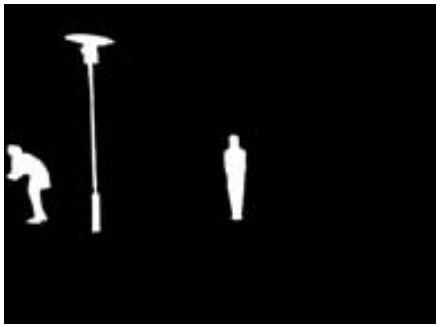
(g) Camera positioning

(h) Rendered image

Foreground Objects

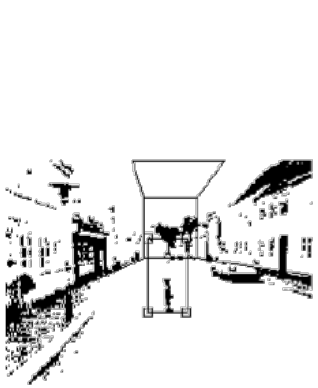
- Use separate billboard for each

- For this to work, three separate images used:
  - Original image.
  - Mask to isolate desired foreground images.
  - Background with objects removed

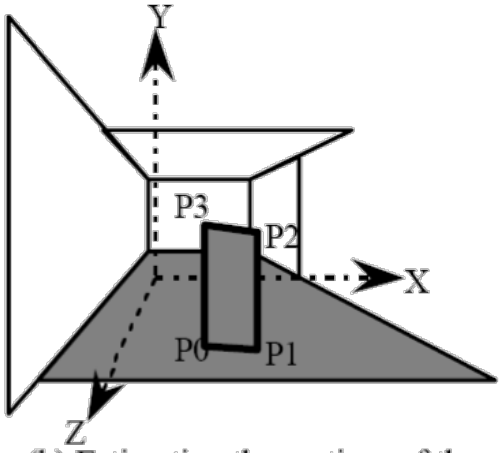


Foreground Objects

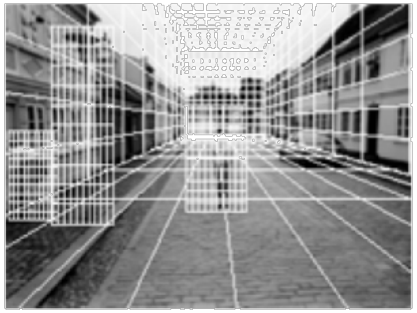
- Add vertical rectangles for each foreground object
- Can compute 3D coordinates P0, P1 since they are on known plane.
- P2, P3 can be computed as before (similar triangles)



(a) Specifying of a foreground object



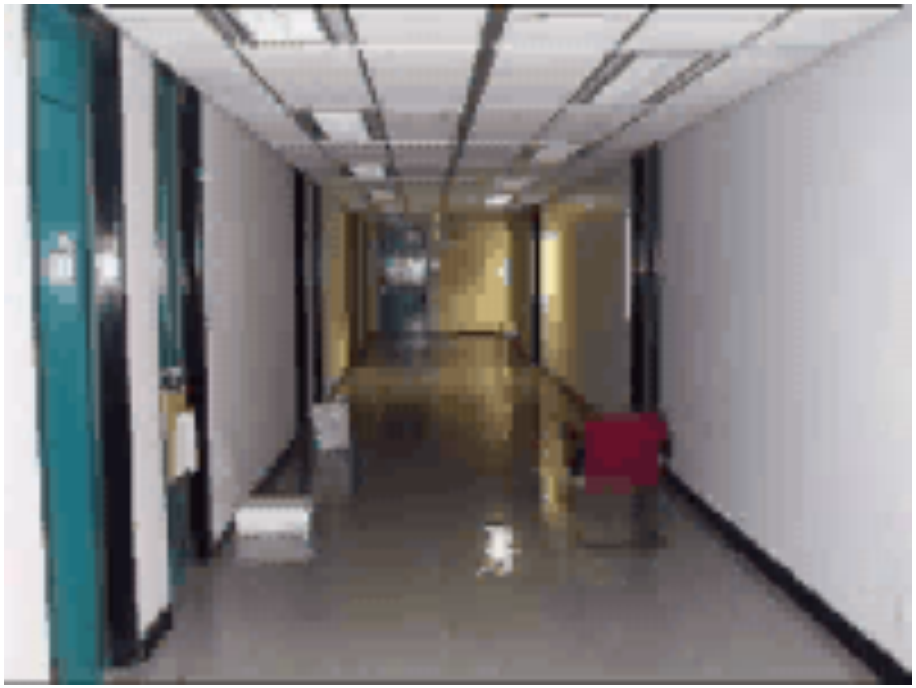
(b) Estimating the vertices of the foreground object model



(c) Three foreground object models

Project 4

Tour into a painting!



Nelson Chu

Automatic Photo Popup

Derek Hoiem
Alexei A. Efros
Martial Hebert

Carnegie Mellon University

The World Behind the Image

Our Goals

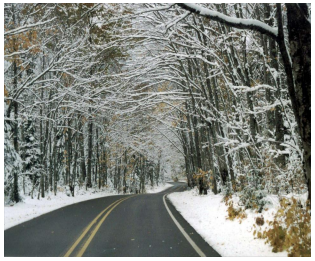
- Simple, piecewise planar models
- Outdoor scenes
- Doesn't need to work all the time (~35%)

Pop-up Book

Our Approach: Learning

- Learn structure of the world and appearance-based models of geometry









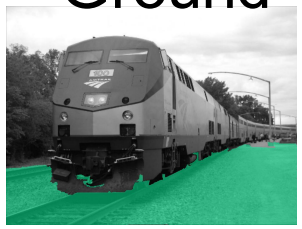



Overview

Input | Geometric Labels | Cut'n'Fold | 3D Model

Image

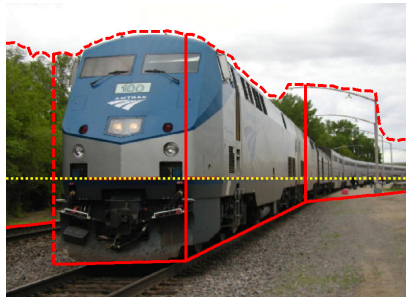
Learned Models



Ground

Vertical

Sky

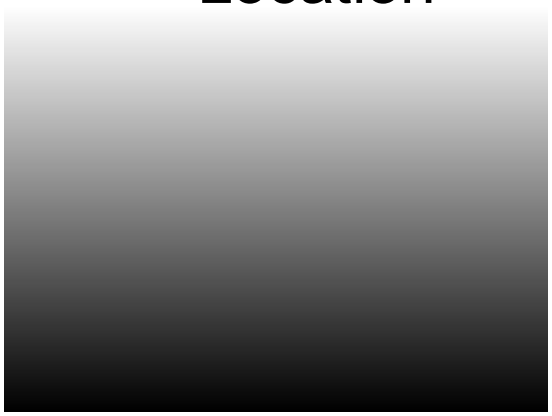


Geometric Cues

Color

Texture

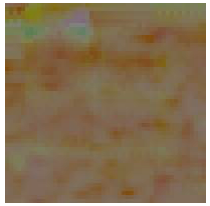
Location



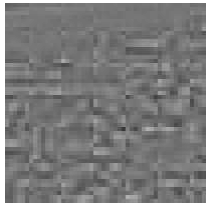
Perspective

Need Spatial Support

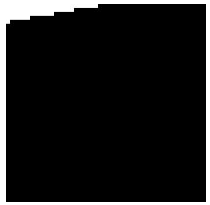
50x50 Patch



Color

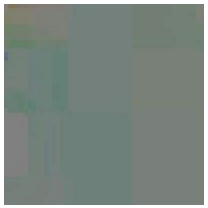


Texture



Perspective

50x50 Patch



Color

Texture

Perspective

Robust Spatial Support

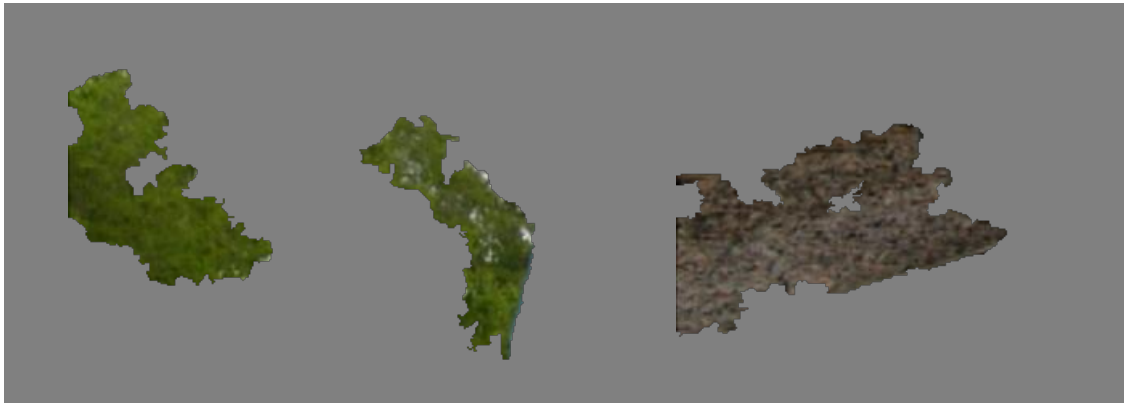
RGB Pixels

Superpixels

[Felzenszwalb and Huttenlocher 2004]

- Safe oversegmentation of image
- Better but not still not enough spatial support

Grouping Superpixels

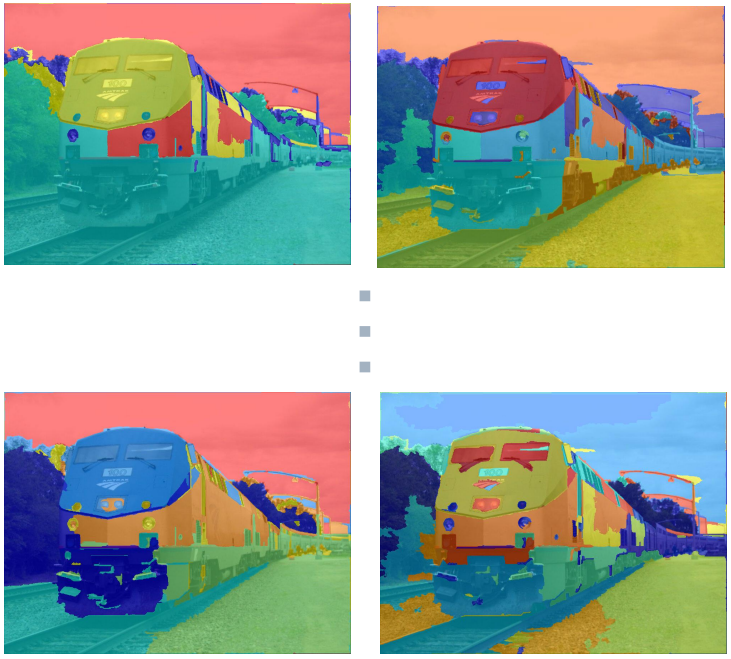


- Simple color, texture, location cues
- Which superpixels come from the same surface?
- Learn from training images
  - Boosted kernel density estimation

Multiple Segmentations

Superpixels

Multiple Segmentations



- Group superpixels likely to be from the same surface into segments
- Single segmentation unreliable
- Create multiple segmentations

Learning Appearance-based Geometry

- All geometric cues available
- Does this segment correspond to a single surface? (*homogeneity likelihood*)
- If so, what is the geometry of that surface? (*label likelihood*)

Learn from training images

Homogeneity Likelihood

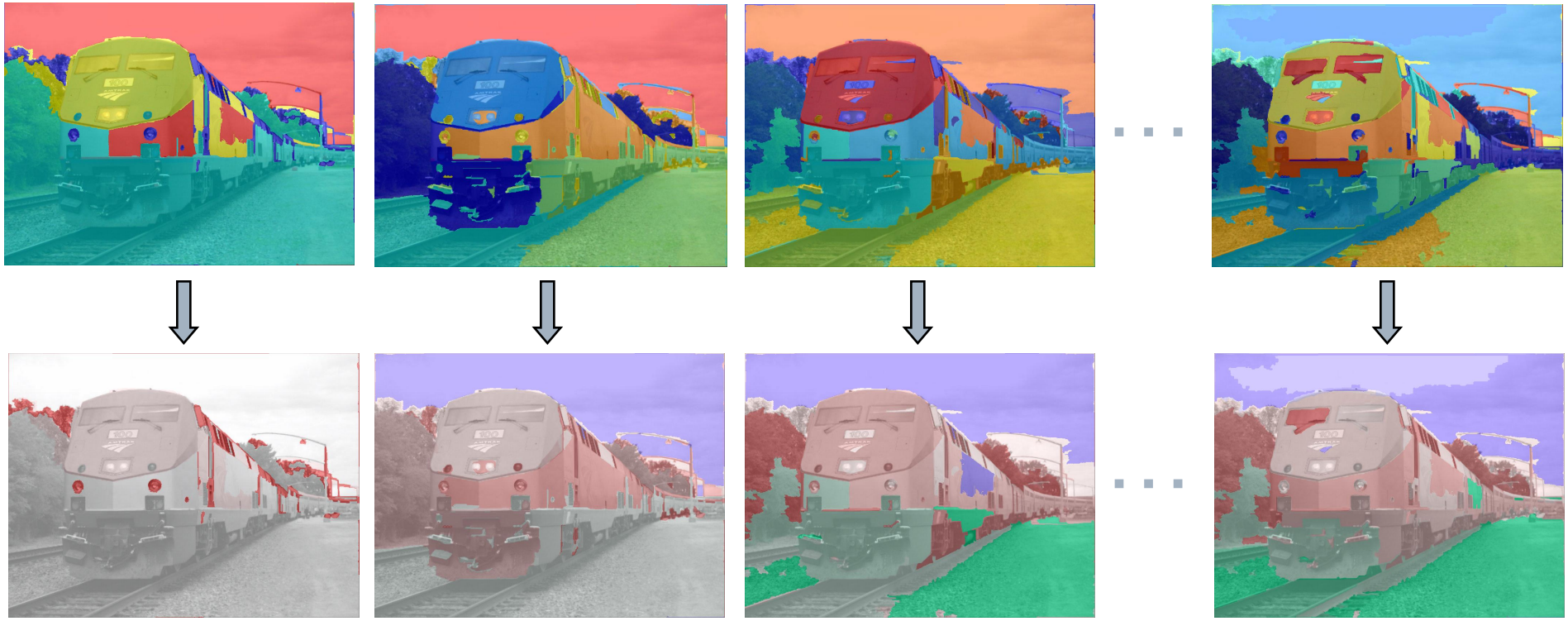
$$P(\mathbf{h}_{ji}|\mathbf{x})$$

Label Likelihood

$$P(y_j = v|\mathbf{x}, \mathbf{h}_{ji})$$

- Prepare training images
  - Create multiple segmentations of training images
  - Get segment labels from ground truth – ground, vertical, sky, or "mixed"
- Density estimation by boosted decision trees
  - 8 nodes per tree
  - Logistic regression version of Adaboost
- See [Collins and Schapire and Singer 2002]

Labeling Segments



For each segment:

- Get $P(y_j = v|\mathbf{x}, \mathbf{h}_{ji})P(\mathbf{h}_{ji}|\mathbf{x})$

Image Labeling

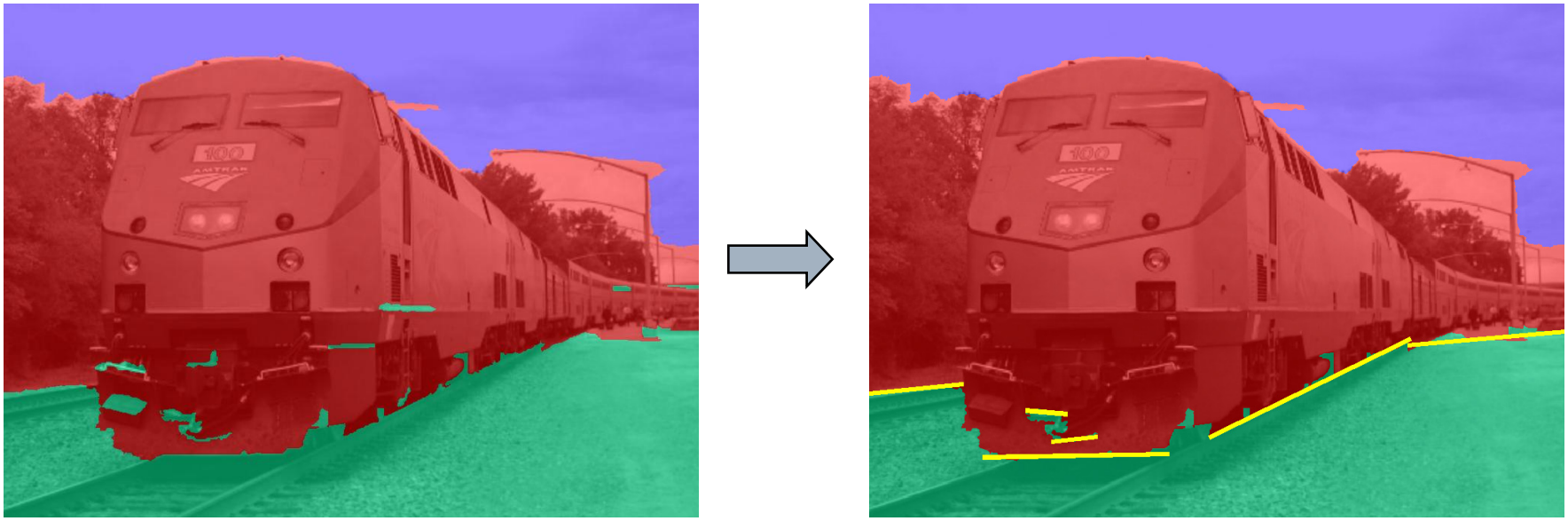
Labeled Segmentations

$$C(y_i = v|\mathbf{x}) = \sum_{j}^{n_h} P(y_j = v|\mathbf{x}, \mathbf{h}_{ji}) P(\mathbf{h}_{ji}|\mathbf{x})$$

Labeled Pixels

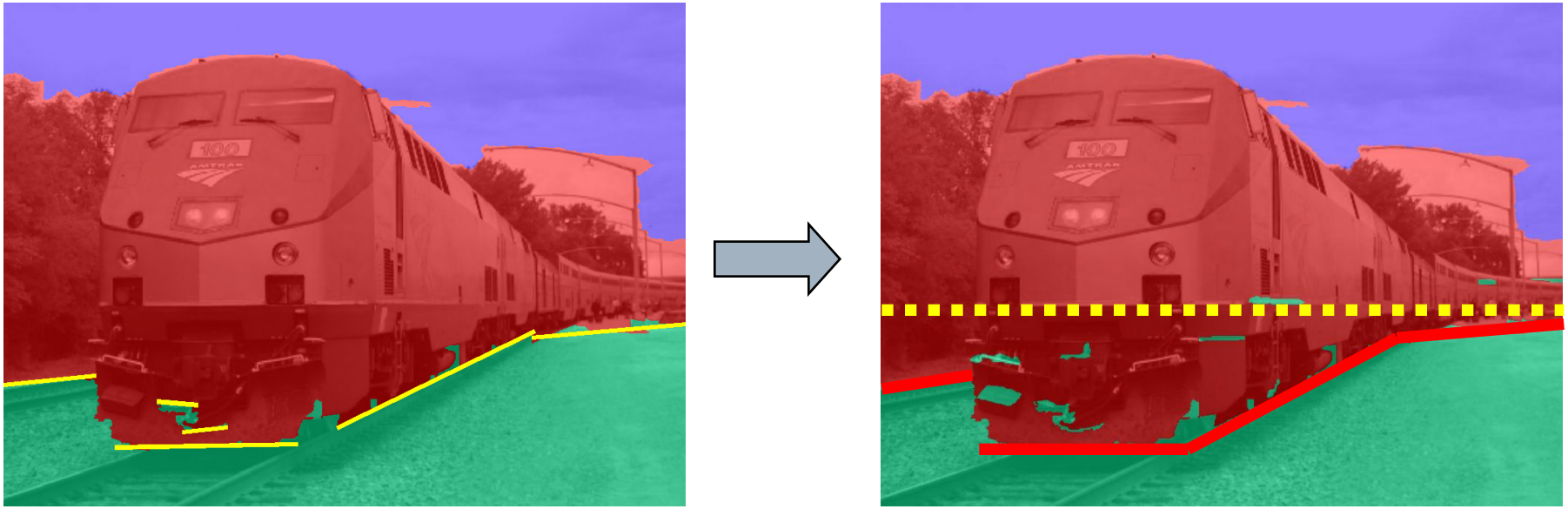
Learned from training images

Cutting and Folding



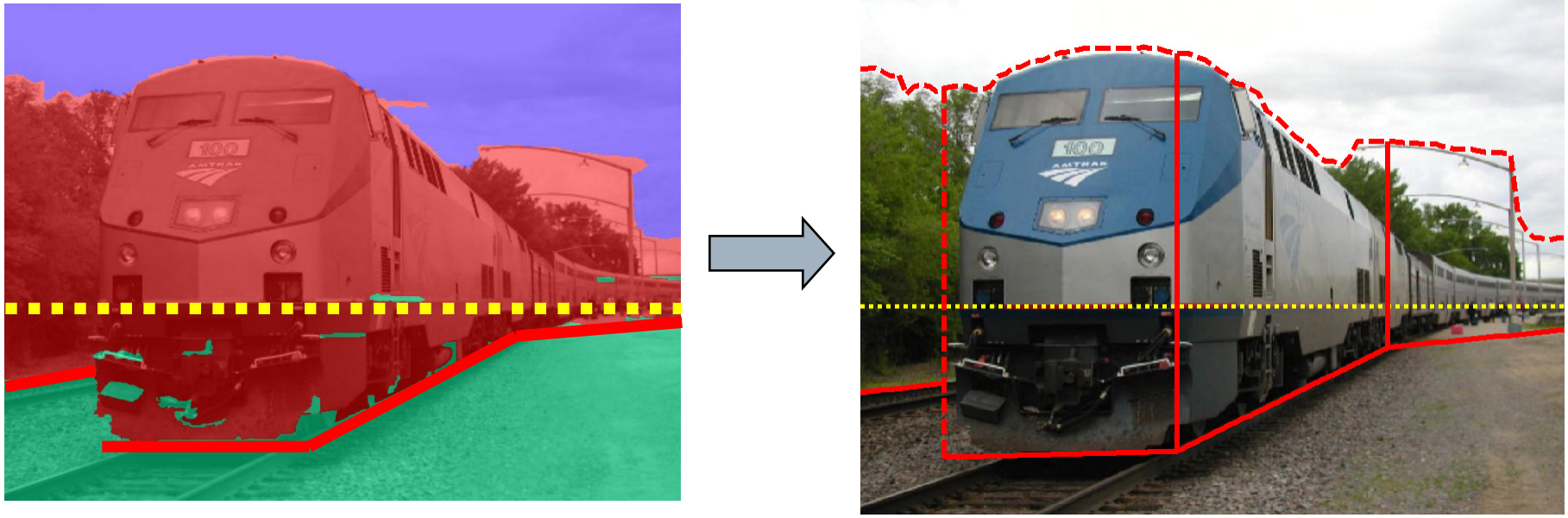
- Fit ground-vertical boundary
  - Iterative Hough transform

Cutting and Folding



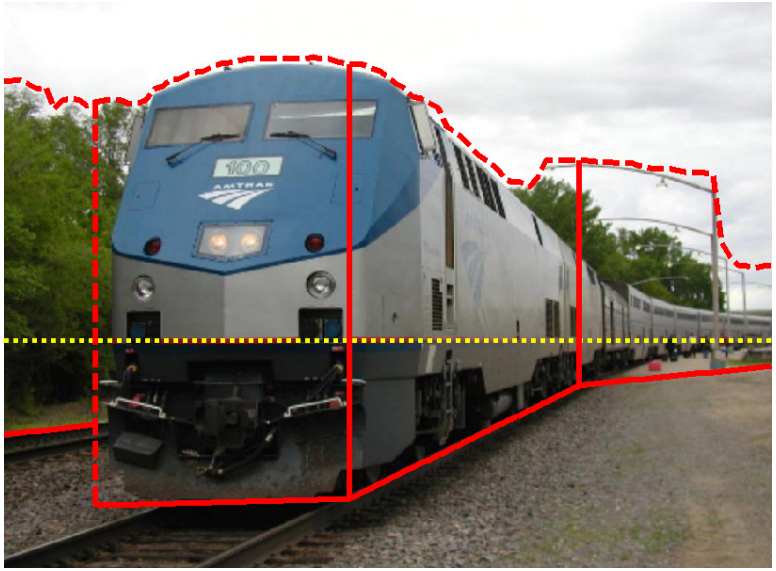
- Form polylines from boundary segments
  - Join segments that intersect at slight angles
  - Remove small overlapping polylines
- Estimate horizon position from perspective cues

Cutting and Folding



- ``Fold'' along polylines and at corners
- ``Cut'' at ends of polylines and along vertical-sky boundary

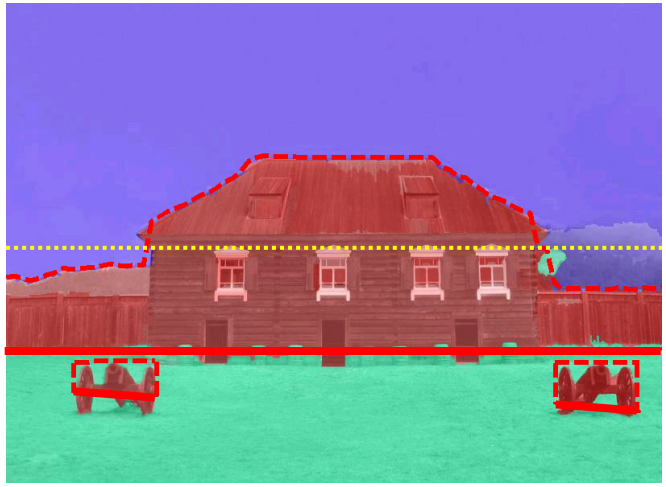
Cutting and Folding



- Construct 3D model
- Texture map

Results

Input Image

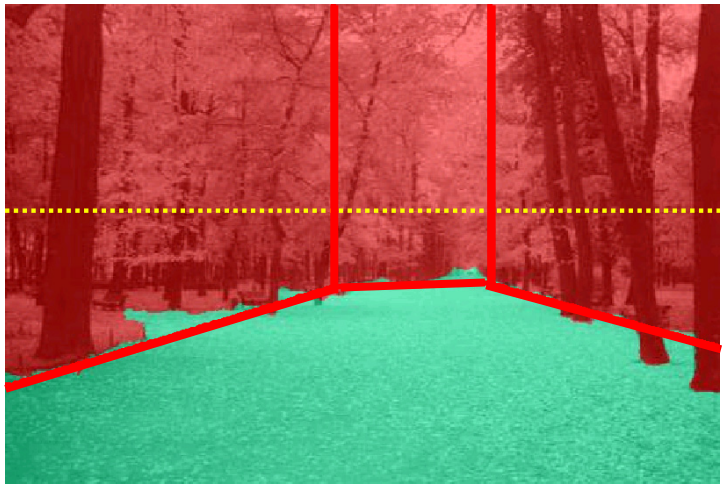


Cut and Fold

Automatic Photo Pop-

Results

Input Image



Cut and Fold

Automatic Photo Pop-

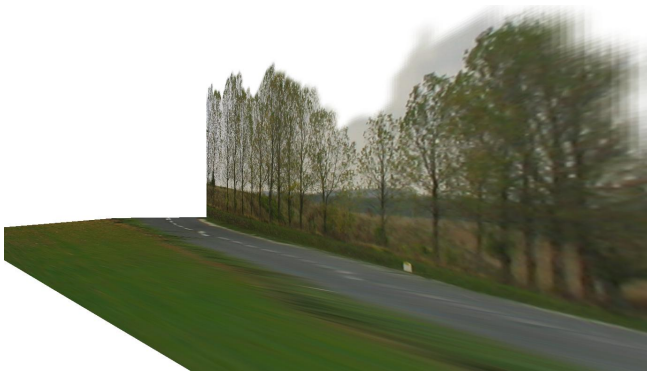
Results

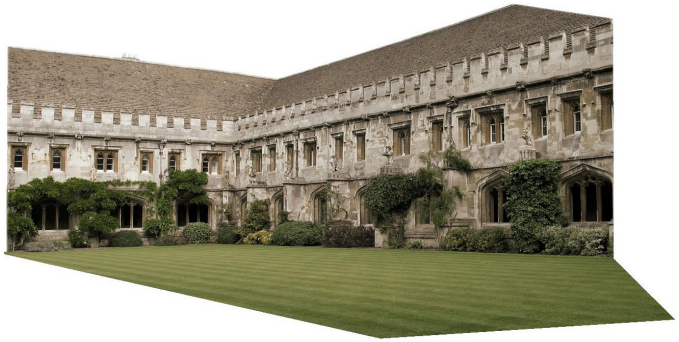
Input Image

Automatic Photo Pop-

Results








Input Images

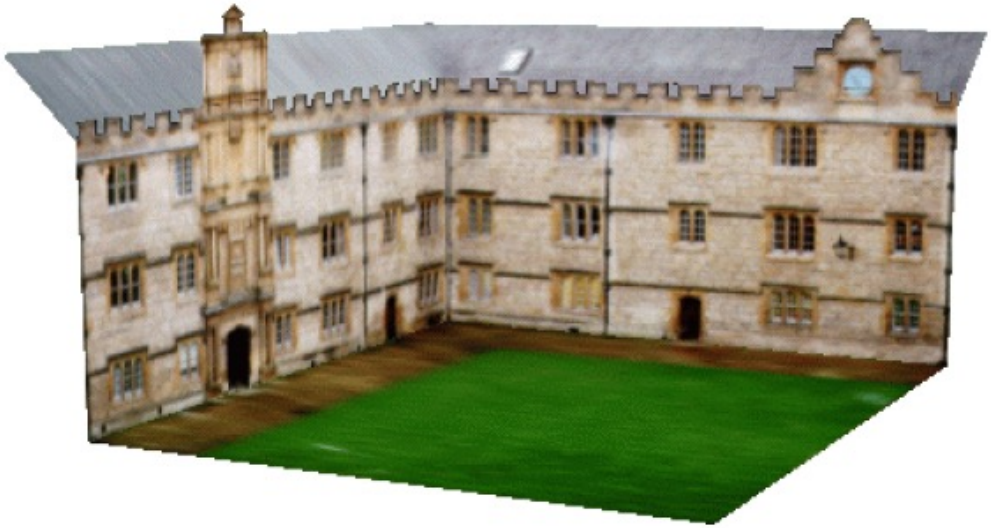
Automatic Photo Pop-up

Results

Input Image

Automatic Photo Pop-

Comparison with Manual Method



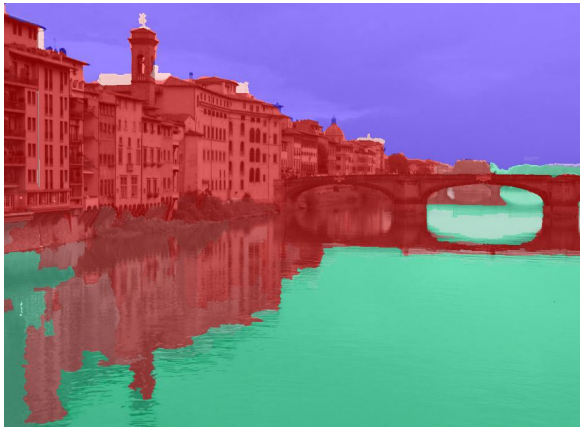
[Liebowitz et al. 1999]

Input Image

Automatic Photo Pop-up (30 sec)!

Failures

Labeling Errors



Failures

Foreground Objects