Single-view metrology

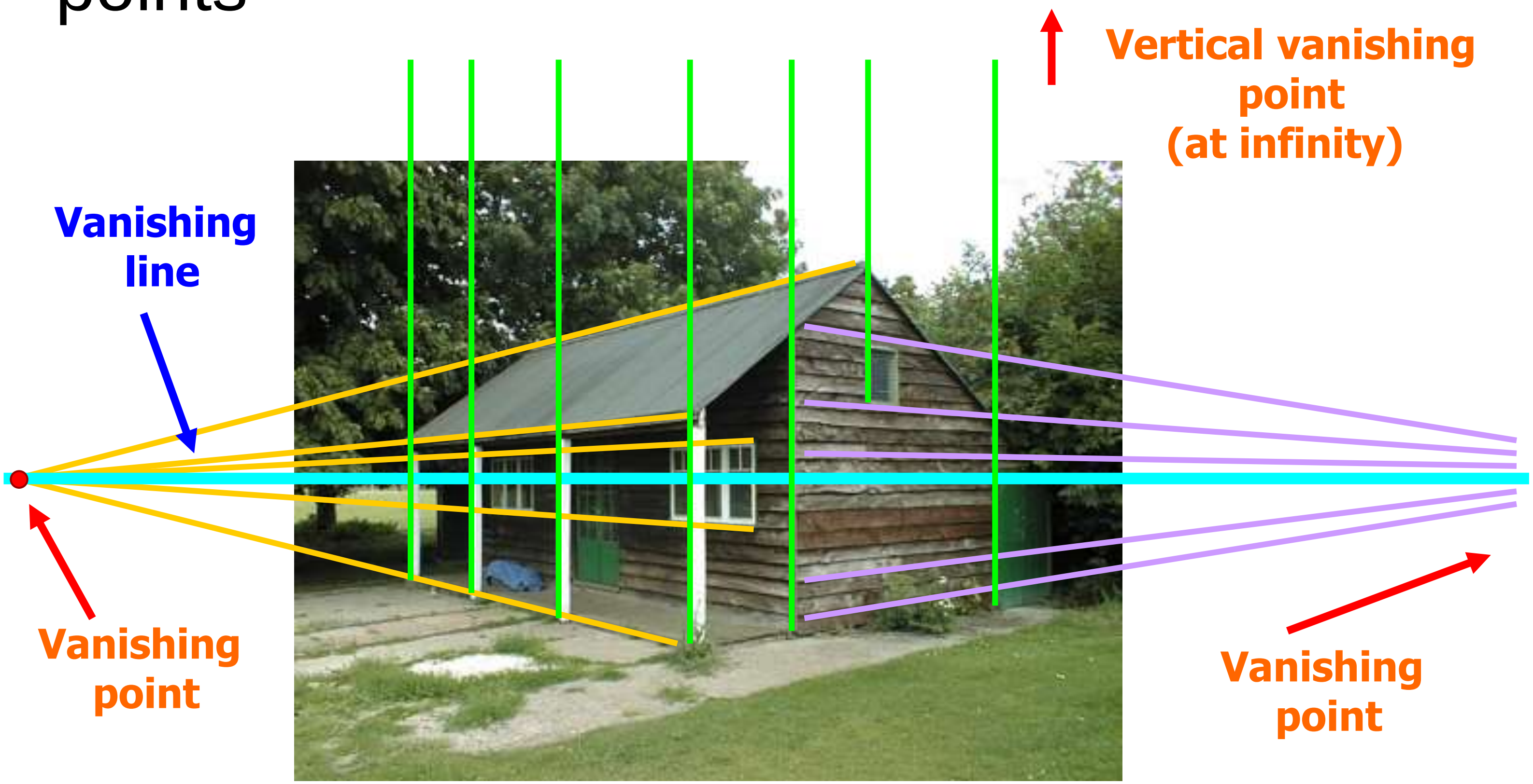
Magritte, *Personal Values*, 1952

Section 6.3.2, R. Szeliski

Many slides from S. Seitz, D. Hoiem

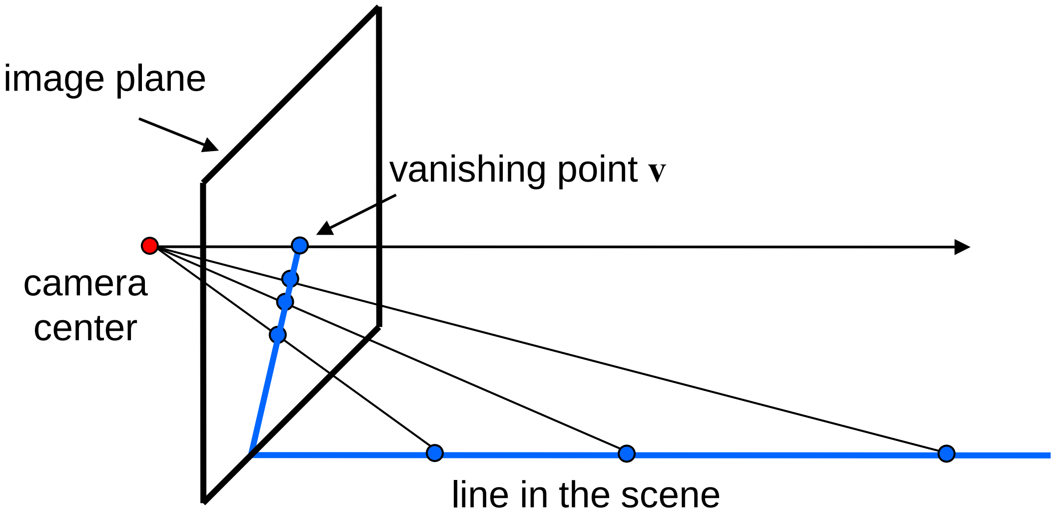
Camera calibration revisited

- What if world coordinates of reference 3D points are not known?
- We can use scene features such as vanishing points



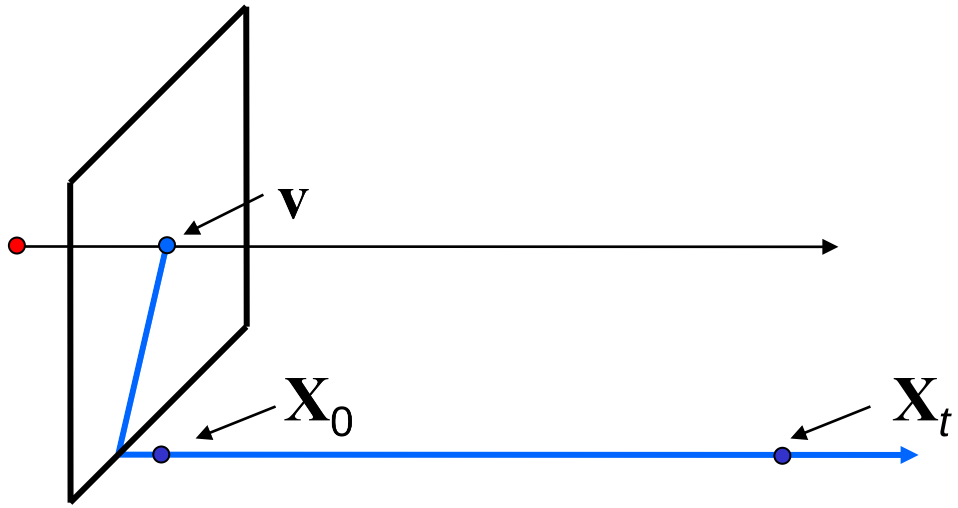
Slide from Efros, Photo from Criminisi

Recall: Vanishing points



- All lines having the same direction share the same vanishing point

Computing vanishing points

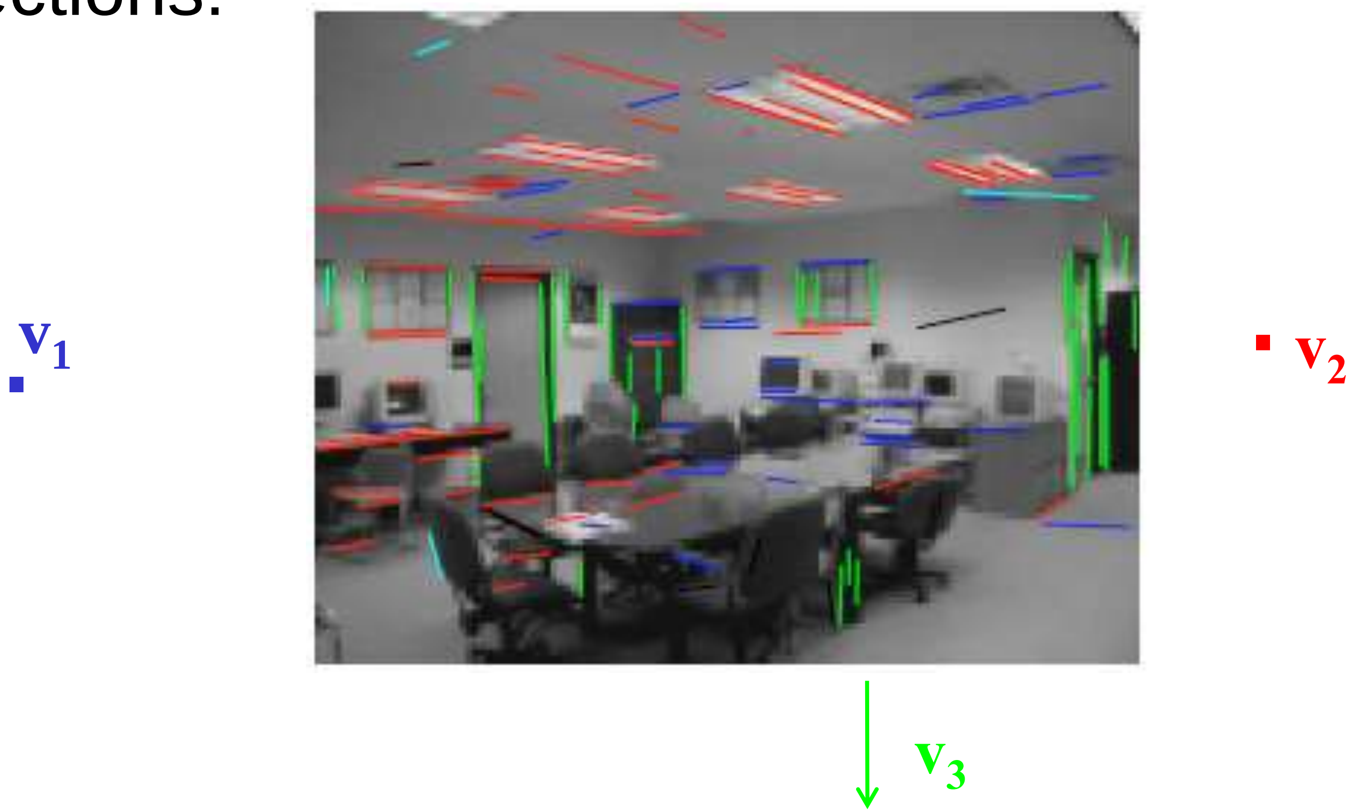


$$\mathbf{X}_t = \begin{bmatrix} x_0 + td_1 \\ y_0 + td_2 \\ z_0 + td_3 \\ 1 \end{bmatrix} = \begin{bmatrix} x_0/t + d_1 \\ y_0/t + d_2 \\ z_0/t + d_3 \\ 1/t \end{bmatrix} \qquad \mathbf{X}_\infty = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ 0 \end{bmatrix}$$

- $\mathbf{X}_\infty$ is a *point at infinity,* $\mathbf{v}$ is its projection: $\mathbf{v} = \mathbf{P}\mathbf{X}_\infty$
- The vanishing point depends only on *line direction*
- All lines having direction $\mathbf{d}$ intersect at $\mathbf{X}_\infty$

Calibration from vanishing points

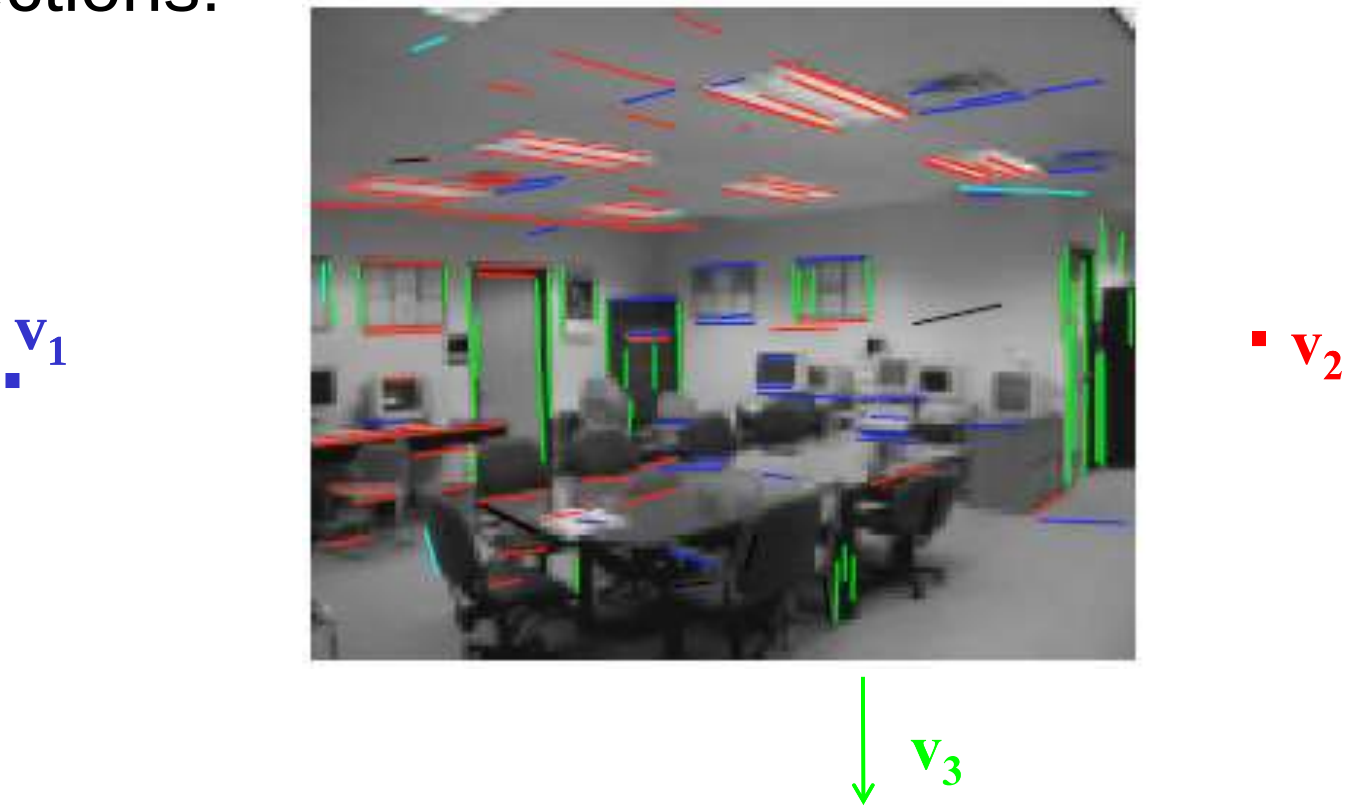
- Consider a scene with three orthogonal vanishing directions:



- Note: $\mathbf{v}_1$, $\mathbf{v}_2$ are *finite* vanishing points and $\mathbf{v}_3$ is an *infinite* vanishing point

Calibration from vanishing points

- Consider a scene with three orthogonal vanishing directions:



- We can align the world coordinate system with these directions

Calibration from vanishing points

$$\mathbf{P}X = \begin{bmatrix} \boxed{\begin{matrix} * \\ * \\ * \end{matrix}} & \boxed{\begin{matrix} * \\ * \\ * \end{matrix}} & \boxed{\begin{matrix} * \\ * \\ * \end{matrix}} & \boxed{\begin{matrix} * \\ * \\ * \end{matrix}} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \end{bmatrix}$$

- $\mathbf{p}_1 = \mathbf{P}(1,0,0,0)^T$ – the vanishing point in the x direction
- Similarly, $\mathbf{p}_2$ and $\mathbf{p}_3$ are the vanishing points in the y and z directions
- $\mathbf{p}_4 = \mathbf{P}(0,0,0,1)^T$ – projection of the origin of the world coordinate system
- Problem: we can only know the four columns up to independent scale factors, additional constraints needed to solve for them

Calibration from vanishing points

- Let us align the world coordinate system with three orthogonal vanishing directions in the scene:

$$\mathbf{e_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e_2} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{e_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \lambda_i \mathbf{v}_i = \mathbf{K}\big[\mathbf{R} \,|\, \mathbf{t}\big]\begin{bmatrix} \mathbf{e}_i \\ 0 \end{bmatrix} = \mathbf{K}\mathbf{R}\mathbf{e}_i$$

$$\mathbf{e}_i = \lambda_i \mathbf{R}^T \mathbf{K}^{-1} \mathbf{v}_i, \quad \mathbf{e}_i^T \mathbf{e}_j = 0$$

$$\mathbf{v}_i^T \mathbf{K}^{-T} \mathbf{R} \mathbf{R}^T \mathbf{K}^{-1} \mathbf{v}_j = \mathbf{v}_i^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{v}_j = 0$$

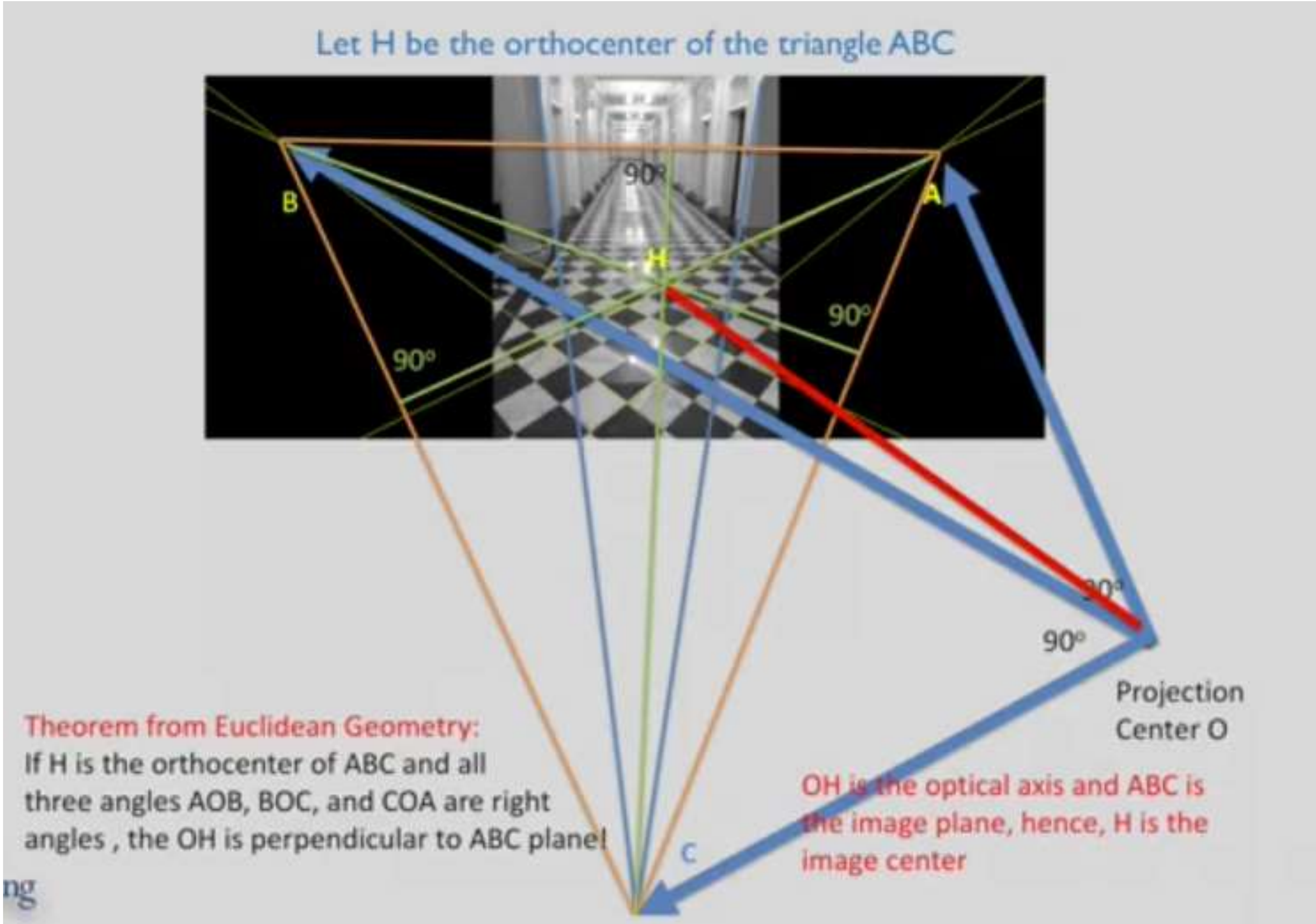
- Each pair of vanishing points gives us a constraint on the focal length and principal point

- zero skew, unit aspect ratio

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} K^{-1} = \begin{bmatrix} 1/f & 0 & -u_0/f \\ 0 & 1/f & -v_0/f \\ 0 & 0 & 1 \end{bmatrix}$$

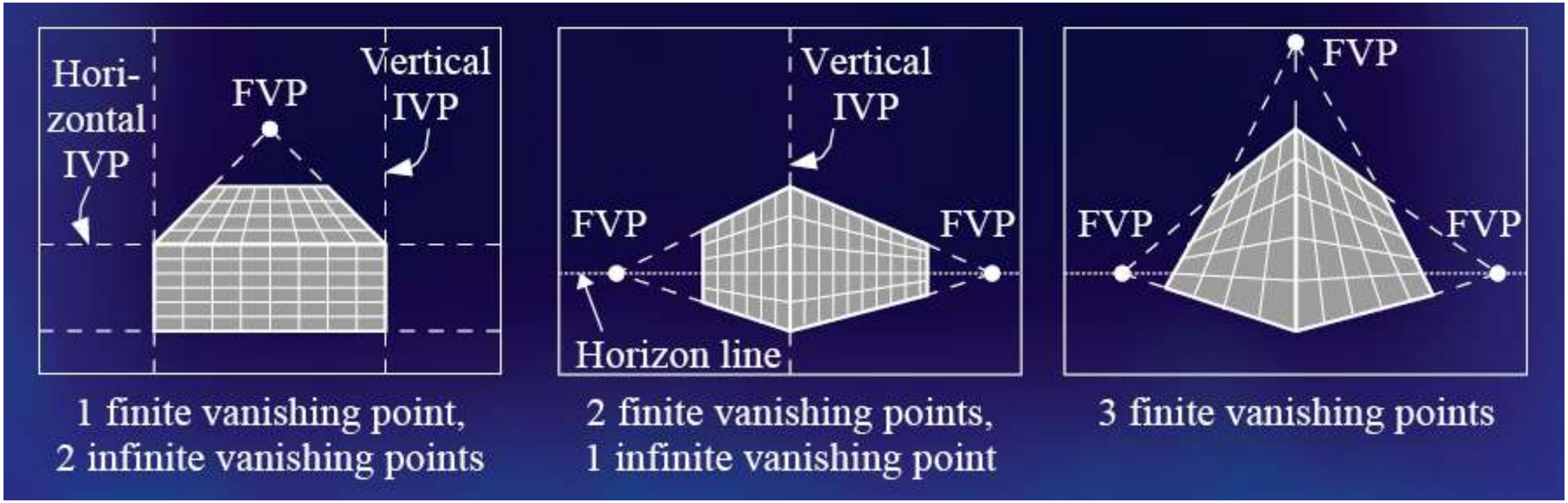
$$v_i^T K^{-T} K^{-1} v_j = 0$$
$$v_j^T K^{-T} K^{-1} v_k = 0$$
$$v_i^T K^{-T} K^{-1} v_k = 0$$

- 3 finite vanishing points: get f, u0, v0
- 2 finite and one infinite : u0,v0 as point on vf1 vf2 closest to image center, get f
- 2 infinite vanishing points : f cant be recovered u0, v0 is at the third vanishing point

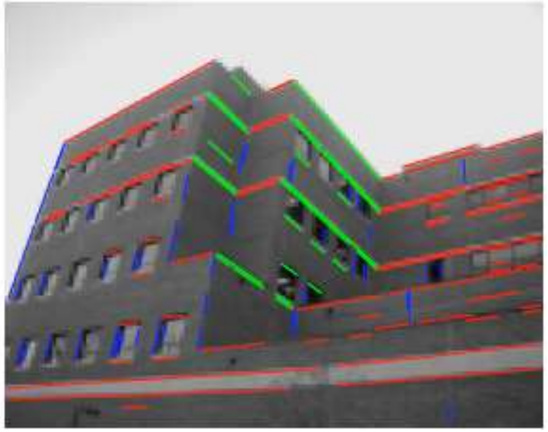


https://www.coursera.org/learn/robotics-perception/lecture/jnaLs/how-to-compute-intrinsics-from-vanishing-points

Calibration from vanishing points







Cannot recover focal length, principal point is the third vanishing point

Can solve for focal length, principal point

Rotation from vanishing points

$$\lambda_i \mathbf{v}_i = \mathbf{K}\left[\mathbf{R} \mid \mathbf{t}\right]\begin{bmatrix} \mathbf{e}_i \\ 0 \end{bmatrix} = \mathbf{KRe}_i$$

$$\lambda_1 \mathbf{K}^{-1}\mathbf{v}_1 = \mathbf{Re}_1 = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{r}_1$$

Thus, $\lambda_i \mathbf{K}^{-1}\mathbf{v}_i = \mathbf{r}_i$.

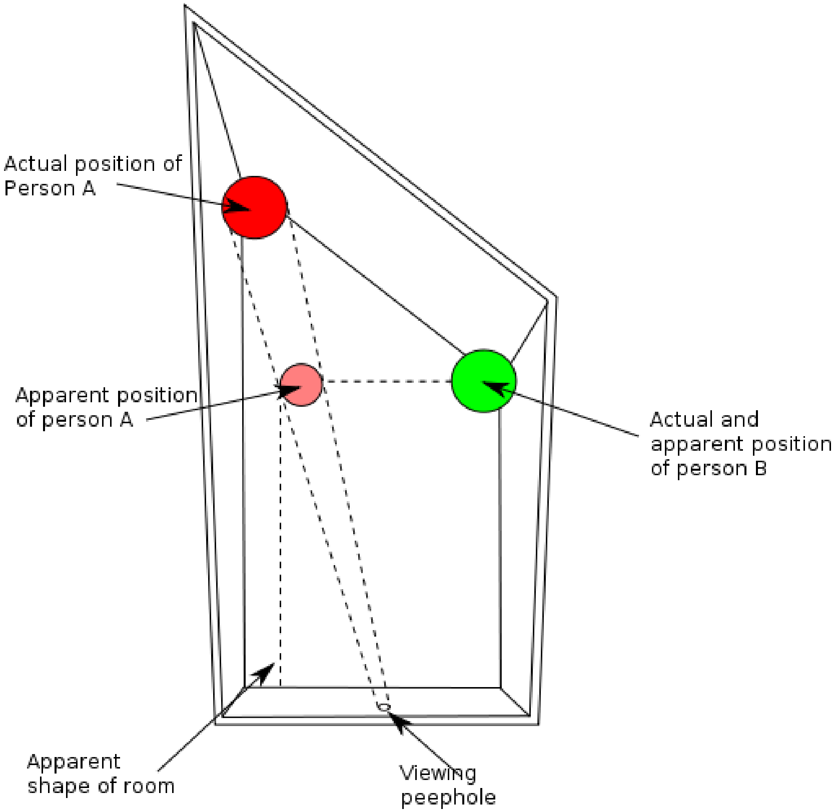
Get λ_i by using the constraint $\|\mathbf{r}_i\|^2=1$.

Calibration from vanishing points: Summary

- Solve for K (focal length, principal point) using three orthogonal vanishing points
- Get rotation directly from vanishing points once calibration matrix is known

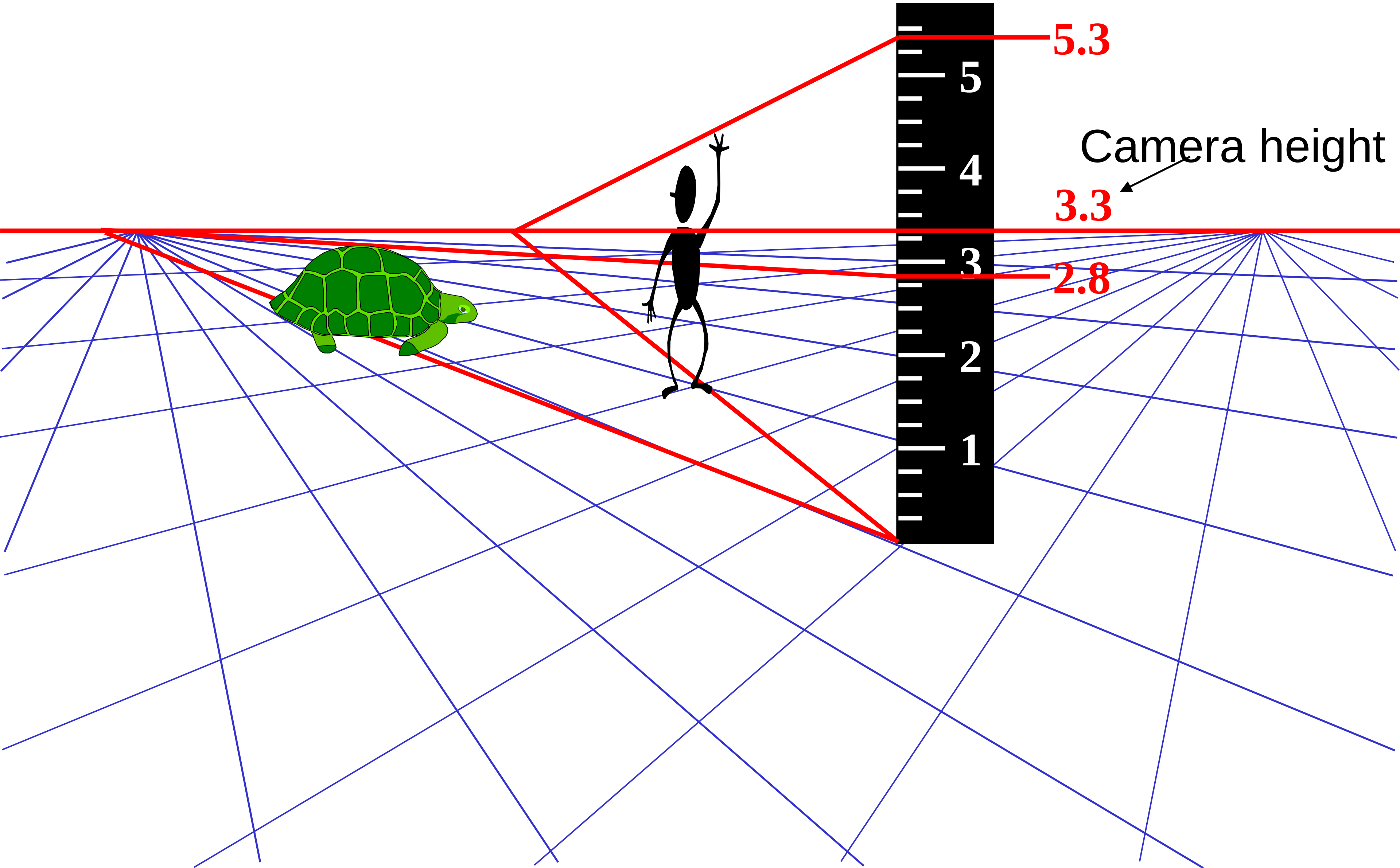
- Advantages
  - No need for calibration chart, 2D-3D correspondences
  - Could be completely automatic
- Disadvantages
  - Only applies to certain kinds of scenes
  - Inaccuracies in computation of vanishing points
  - Problems due to infinite vanishing points

Making measurements from a single image

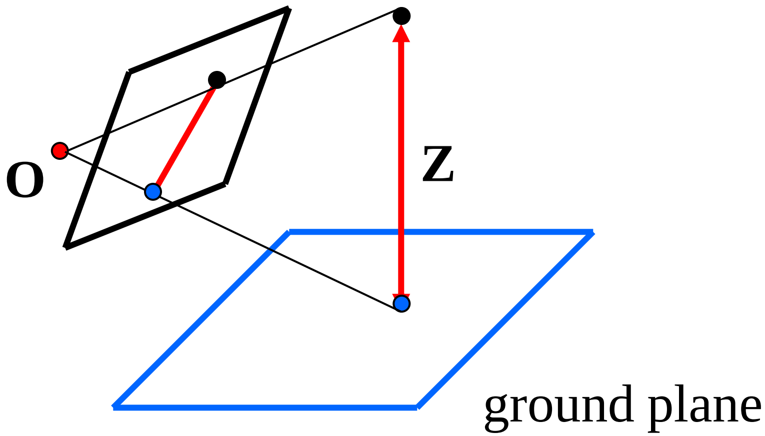


http://en.wikipedia.org/wiki/Ames_room

Recall: Measuring height



Measuring height without a ruler



Compute Z from image measurements

- Need more than vanishing points to do this

Projective invariant

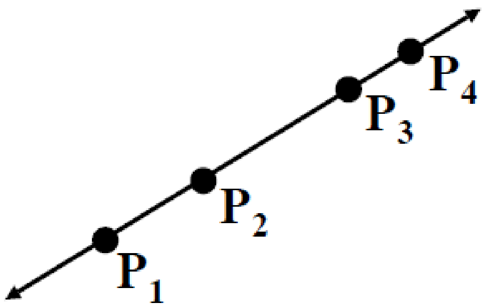
- We need to use a *projective invariant*: a quantity that does not change under projective transformations (including perspective projection)
  - What are some invariants for similarity, affine transformations?

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

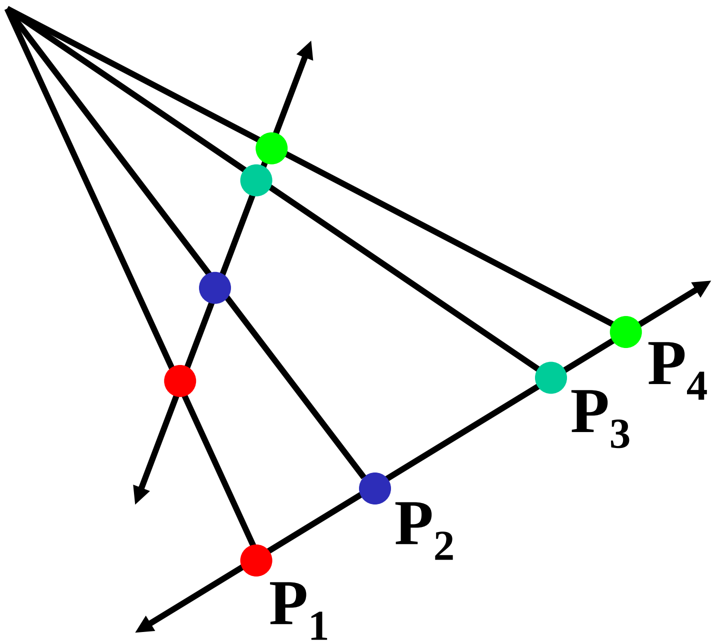
$$\frac{\|\mathbf{P}_1 - \mathbf{P}_3\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_1 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_3\|}$$

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

This is the fundamental invariant of projective geometry

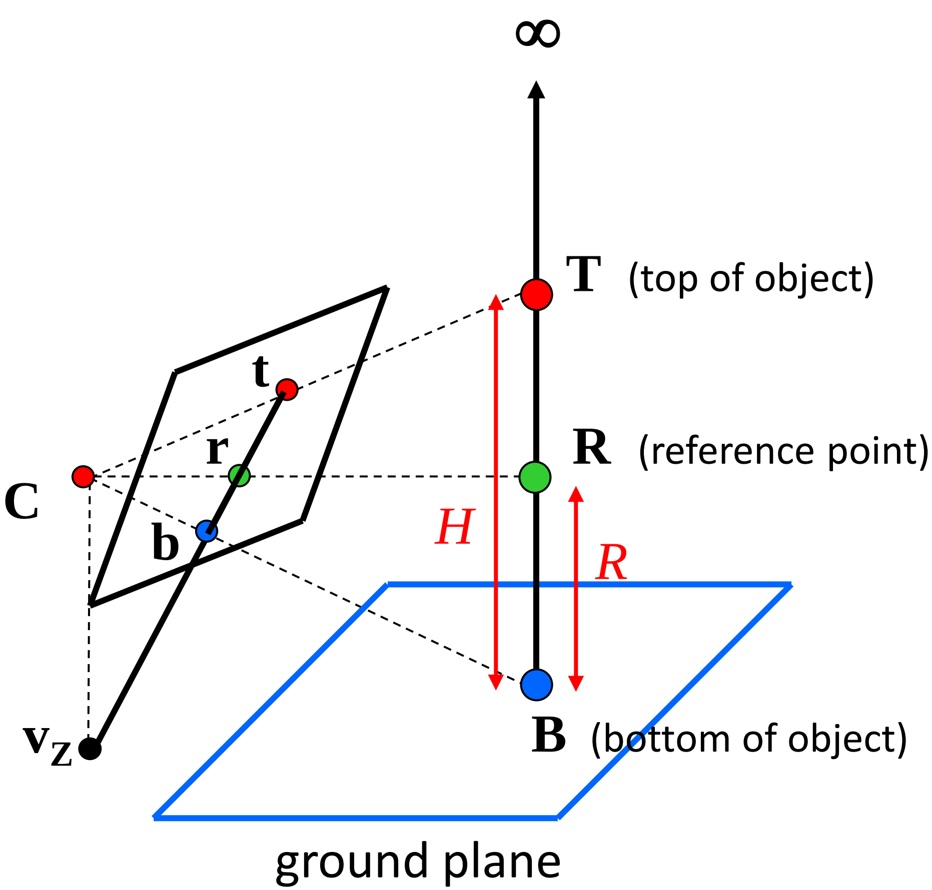
Projective invariant

- We need to use a *projective invariant*: a quantity that does not change under projective transformations (including perspective projection)
- The *cross-ratio* of four points:



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

Measuring height



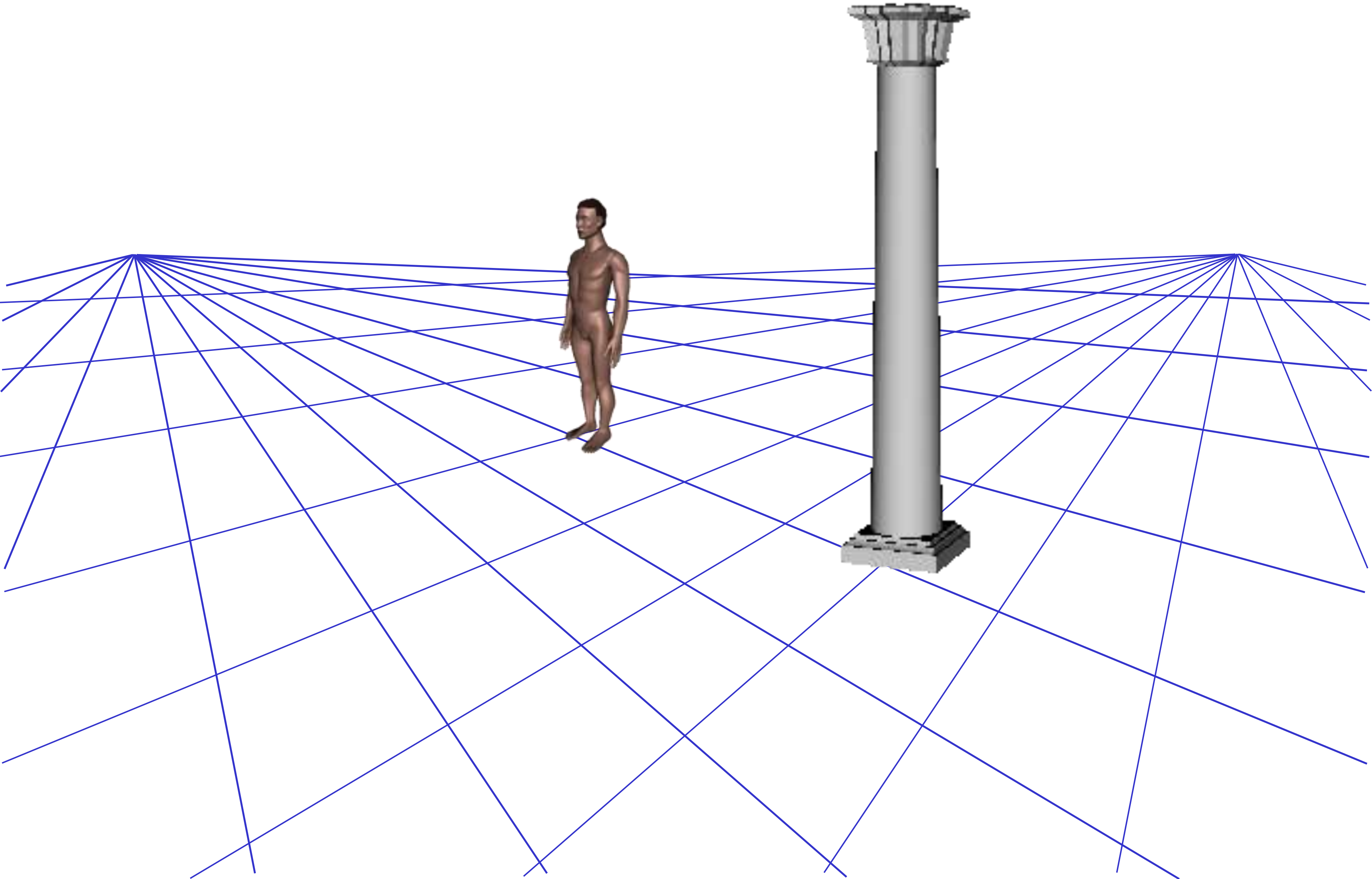
$$\frac{\|\mathbf{T}-\mathbf{B}\|\,\|\infty-\mathbf{R}\|}{\|\mathbf{R}-\mathbf{B}\|\,\|\infty-\mathbf{T}\|} = \frac{H}{R}$$

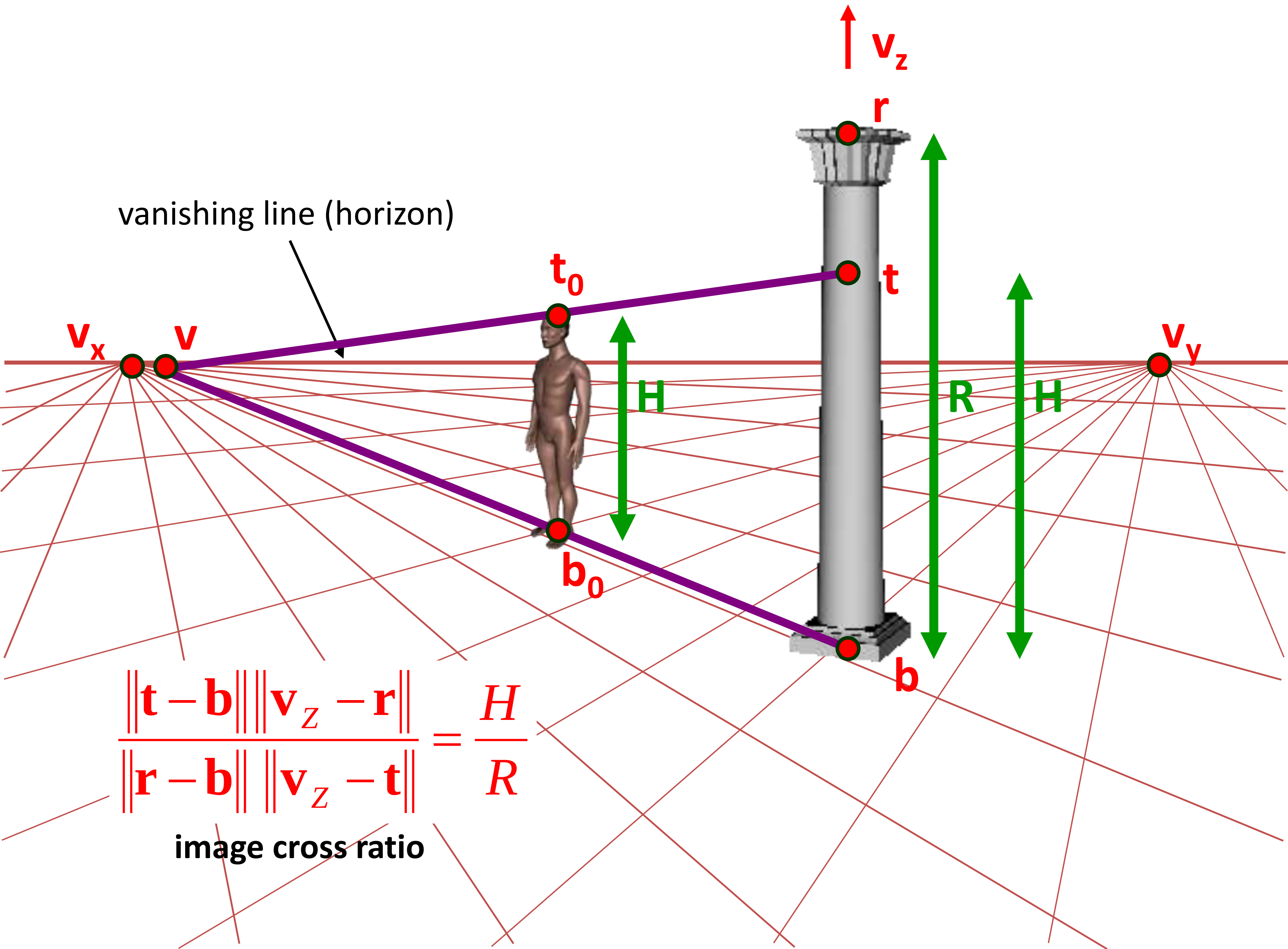
scene cross ratio

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

image cross ratio

Measuring height without a ruler





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

image cross ratio

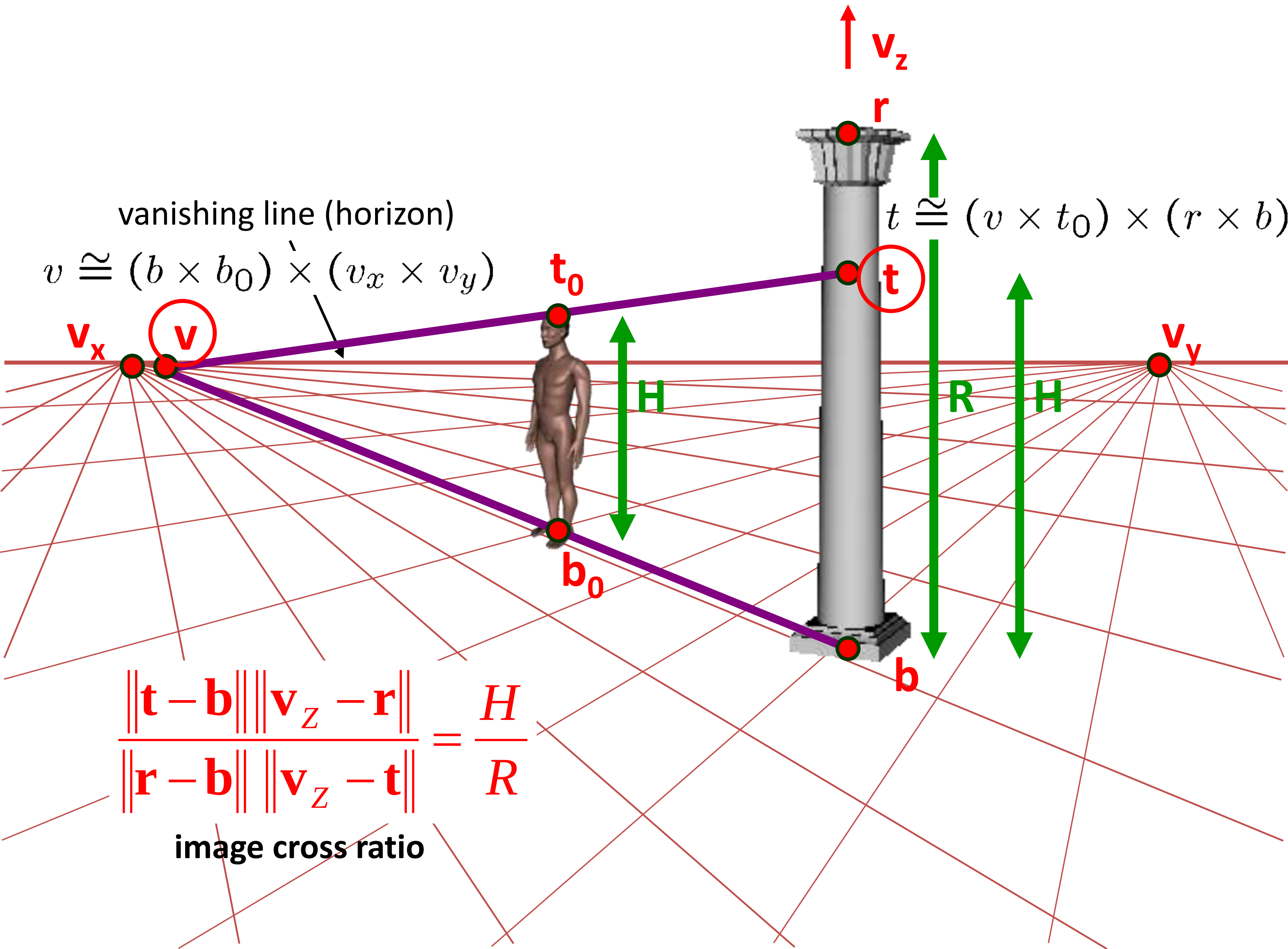
2D lines in homogeneous coordinates

- Line equation: $ax + by + c = 0$

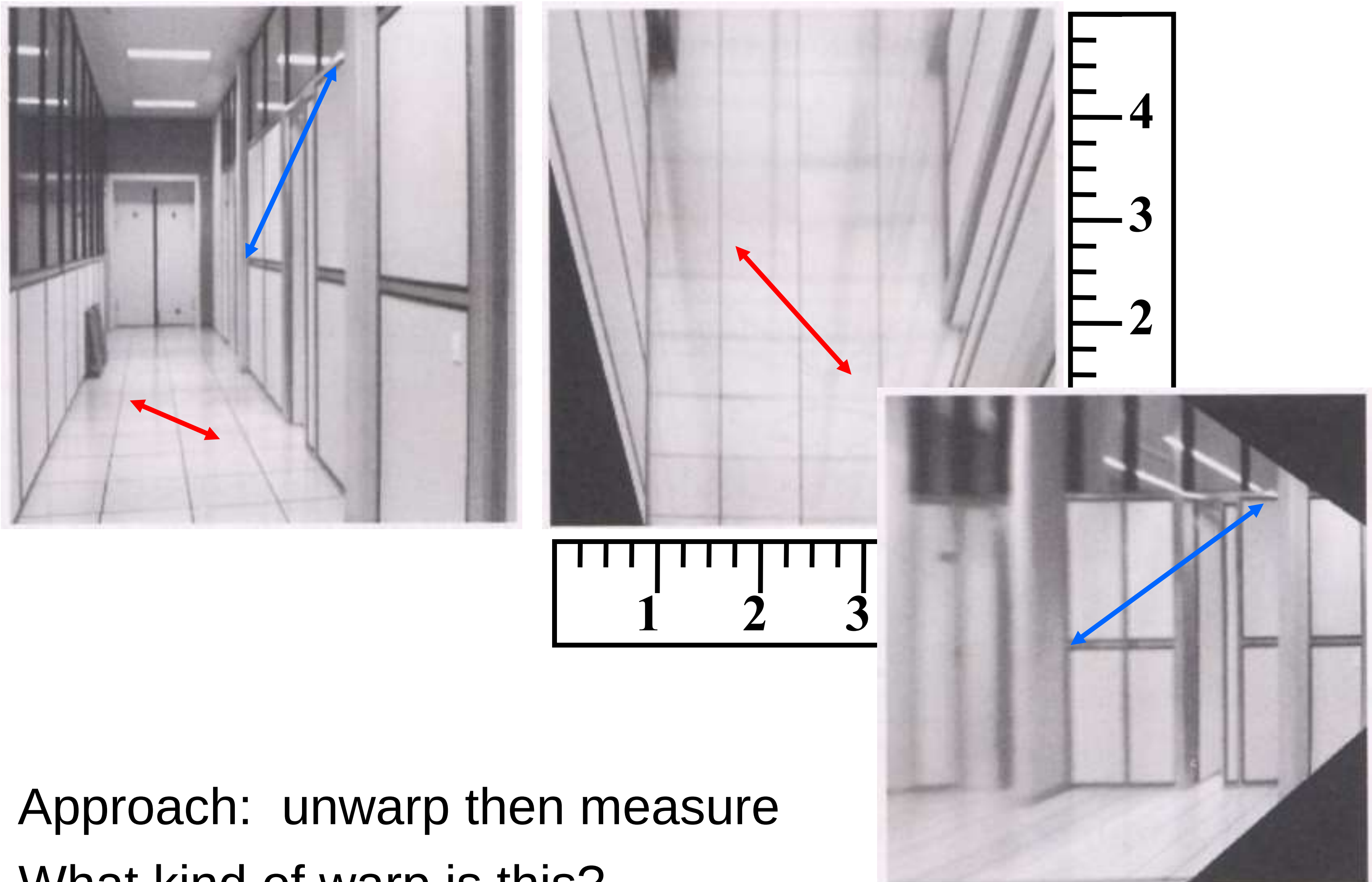
$$\mathbf{l}^T \mathbf{x} = 0 \quad \text{where} \quad \mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Line passing through two points: $\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2$

- Intersection of two lines: $\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2$


v_z
r
vanishing line (horizon)
$t \cong (v \times t_0) \times (r \times b)$
$v \cong (b \times b_0) \times (v_x \times v_y)$
t_0
t
v_x
v
v_y
H
R
H
b_0
b
$\frac{\|\mathbf{t}-\mathbf{b}\| \|\mathbf{v}_Z-\mathbf{r}\|}{\|\mathbf{r}-\mathbf{b}\| \|\mathbf{v}_Z-\mathbf{t}\|} = \frac{H}{R}$
image cross ratio

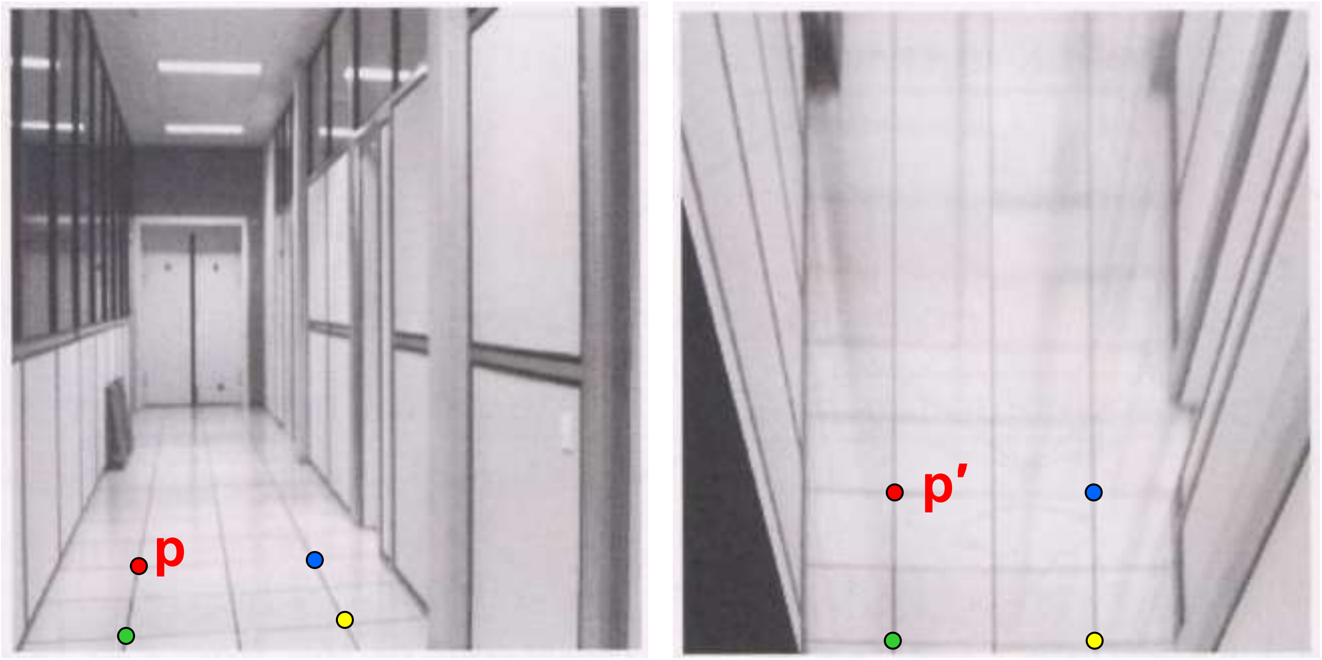
Measurements on planes



Approach: unwarp then measure

What kind of warp is this?

Image rectification



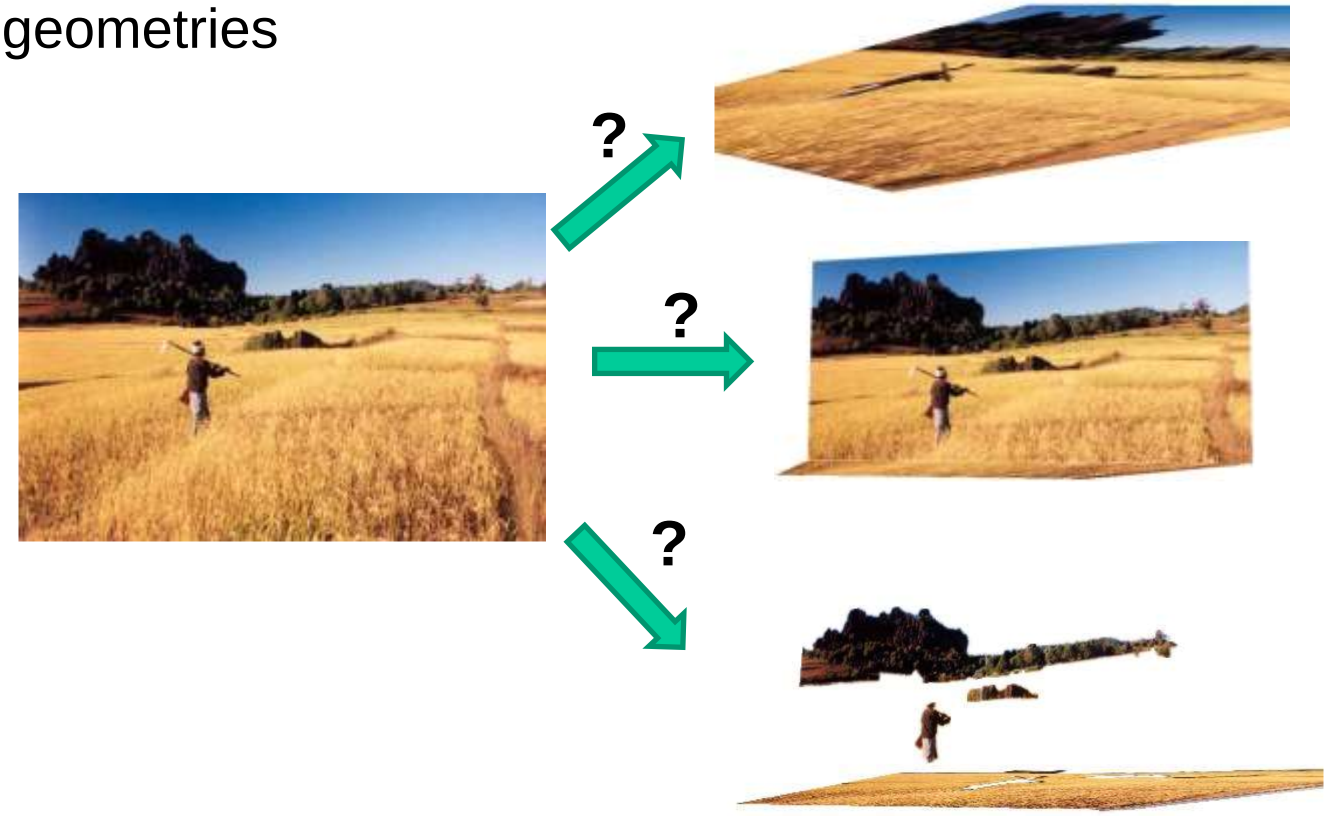
To unwarp (rectify) an image

- solve for homography **H** given **p** and **p′**
  - how many points are necessary to solve for **H**?

Today’s class: 3D Reconstruction

The challenge

One 2D image could be generated by an infinite number of 3D geometries



The solution

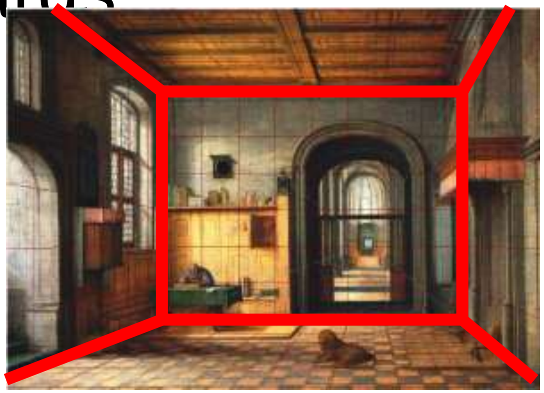
Make simplifying assumptions about 3D geometry

Unlikely

Likely

Today’s class: Two Models

Box + frontal billboards



Ground plane + non-frontal billboards

“Tour into the Picture” (Horry et al. SIGGRAPH ’97)

Create a 3D “theatre stage” of five planes

Specify foreground objects through bounding polygons

Use camera transformations to navigate through the scene

Following slides modified from Efros

The idea

Many scenes can be represented as an axis-aligned box volume (i.e. a stage)

Key assumptions

All walls are orthogonal

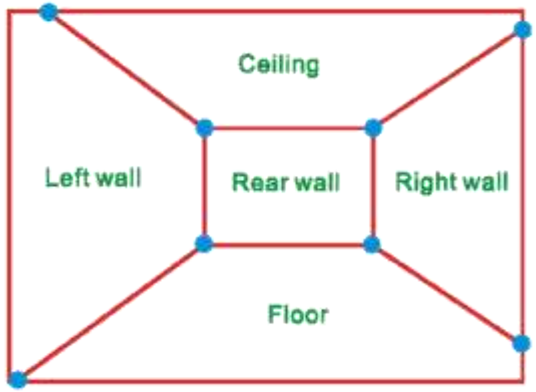
Camera view plane is parallel to back of volume

How many vanishing points does the box have?

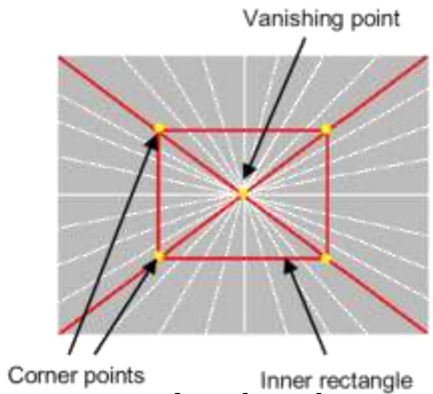
Three, but two at infinity

Single-point perspective

Can use the vanishing point to fit the box to the particular scene



Step 1: specify scene geometry

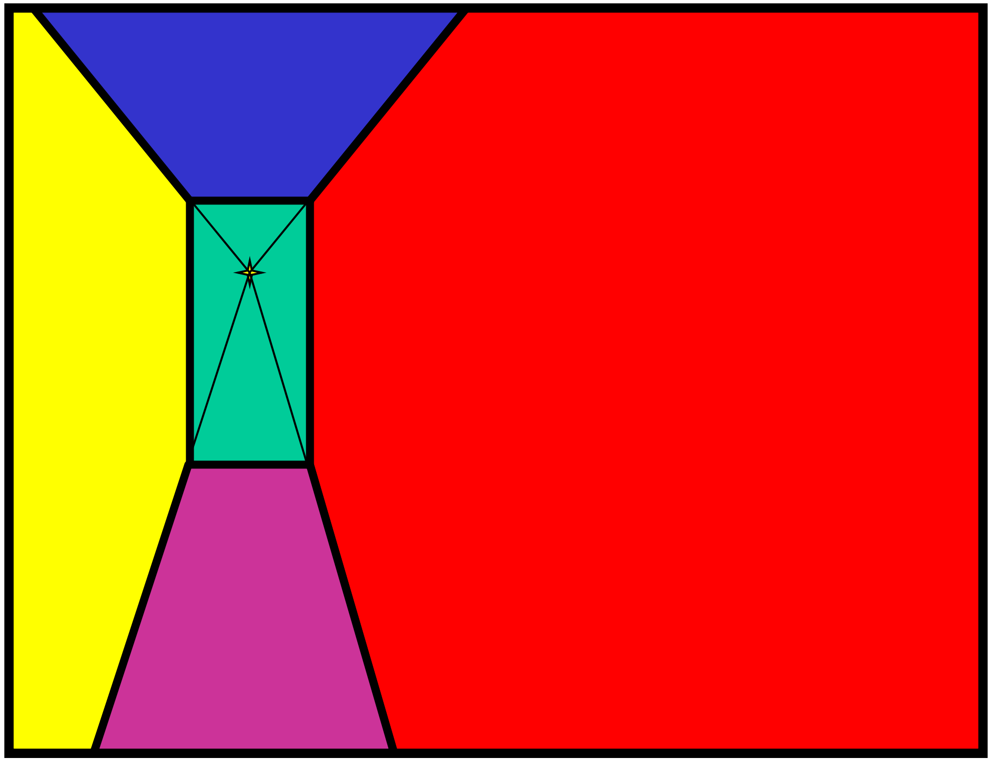


User controls the inner box and the vanishing point placement (# of DOF?)

Q: If we assume camera is looking straight at back wall, what camera parameter(s) does the vanishing point position provide?

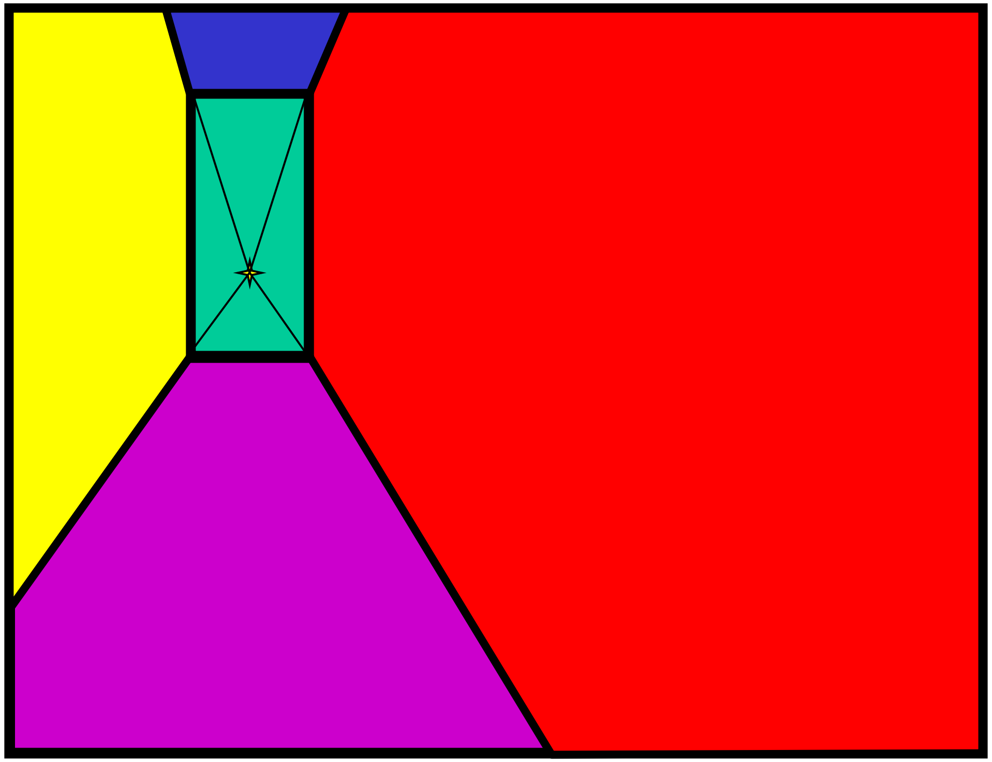
A: Vanishing point direction is perpendicular to image plane, so the vp is the principal point

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



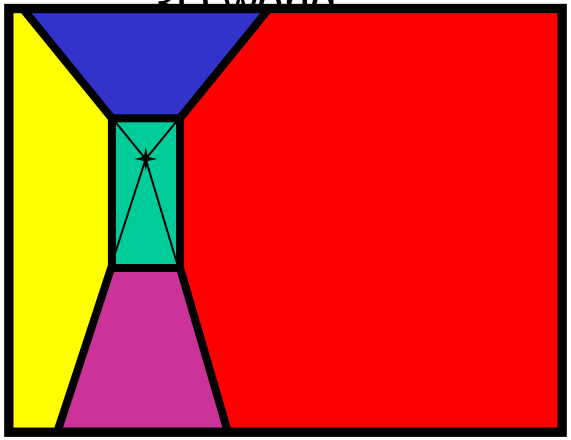
High Camera

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

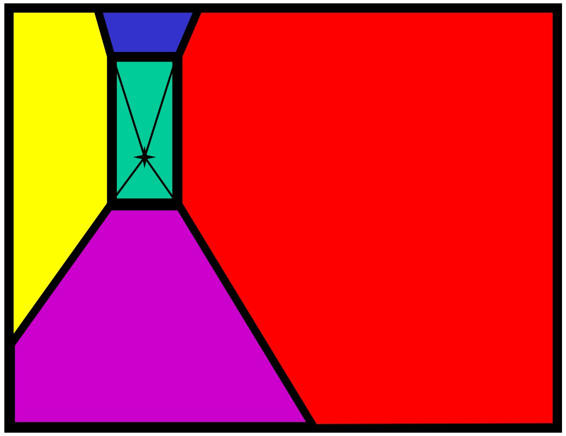


Low Camera

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

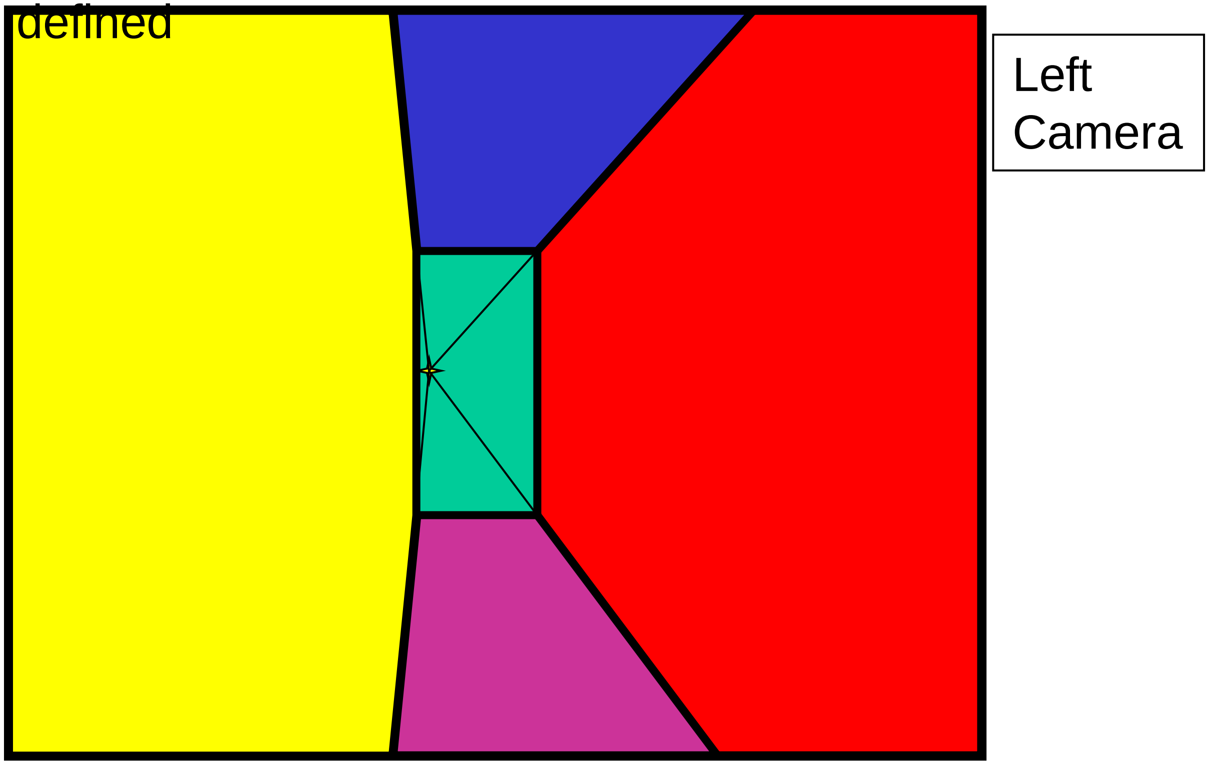


High Camera

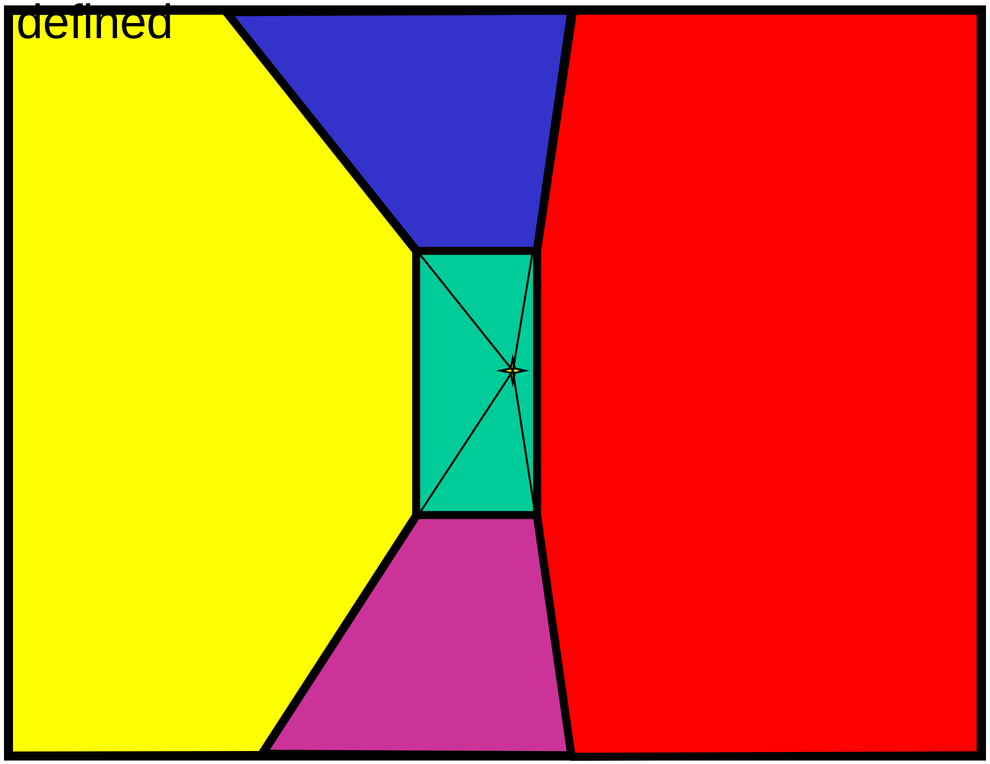


Low Camera

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

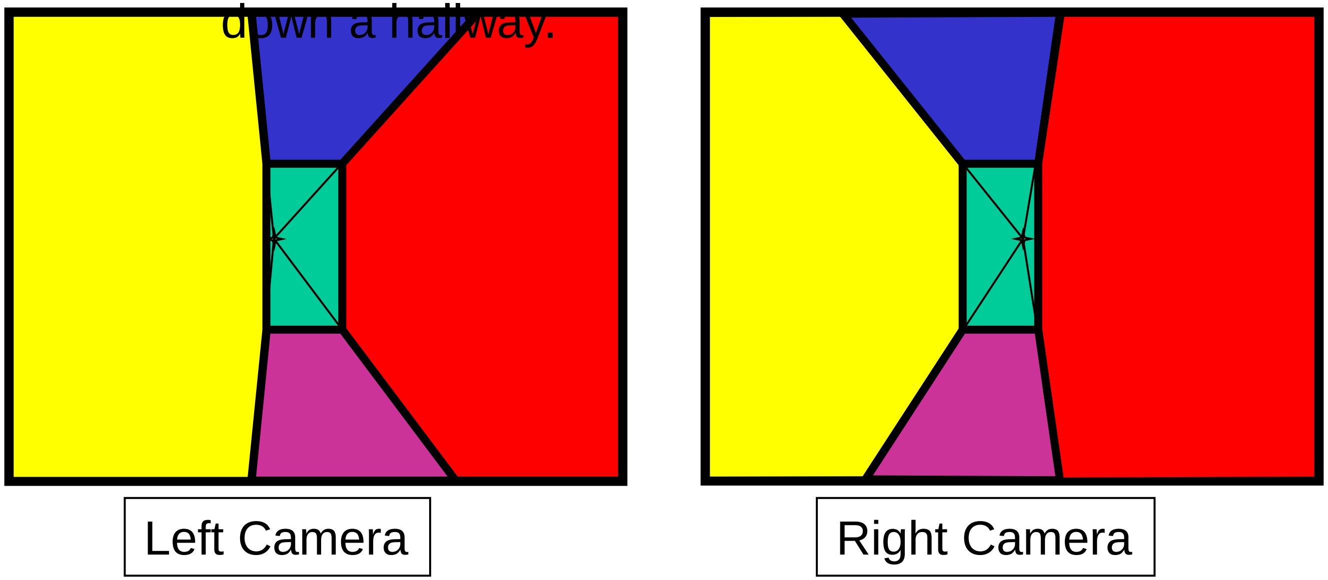


Another example of user input: vanishing point and back face of view volume are defined



Right Camera

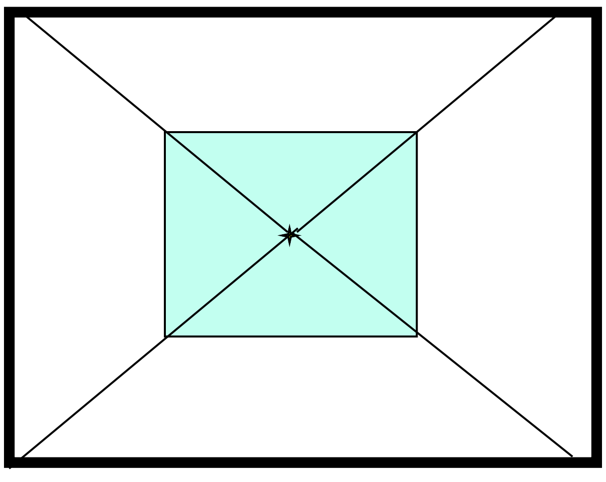
Comparison of two camera placements – left and right. Corresponding subdivisions match view you would see if you looked down a hallway.



Question

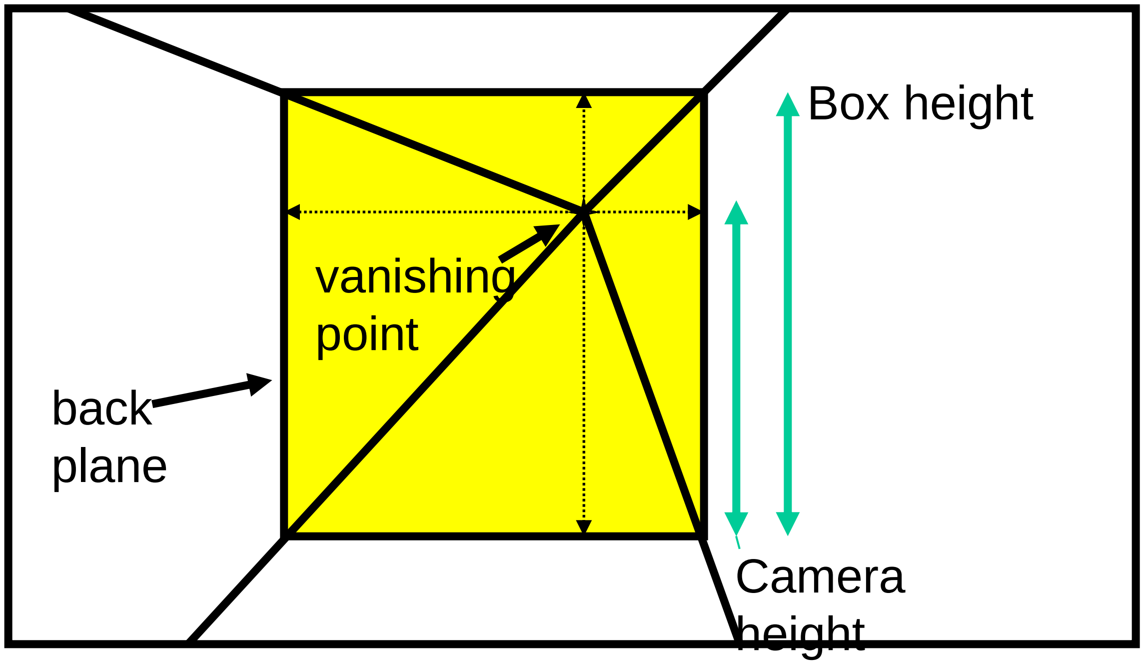
Think about the camera center and image plane…

- What happens when we move the box?
- What happens when we move the vanishing point?



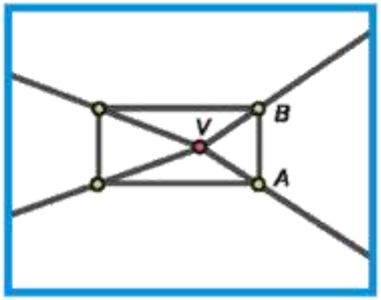
2D to 3D conversion

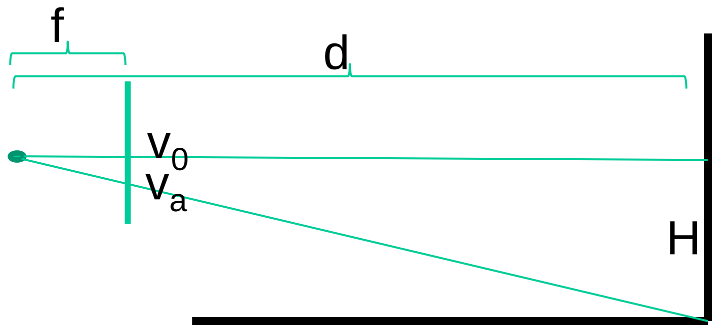
Use ratios



Box width / height in 3D is proportional to width over height in the image because back plane is parallel to image plane

Get depth using similar triangles





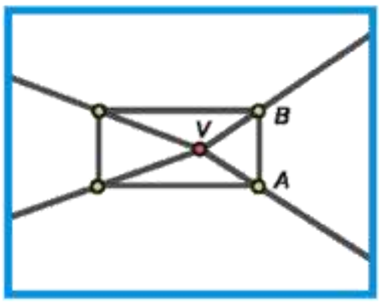
$f / (v_a - v_0) = d / H$

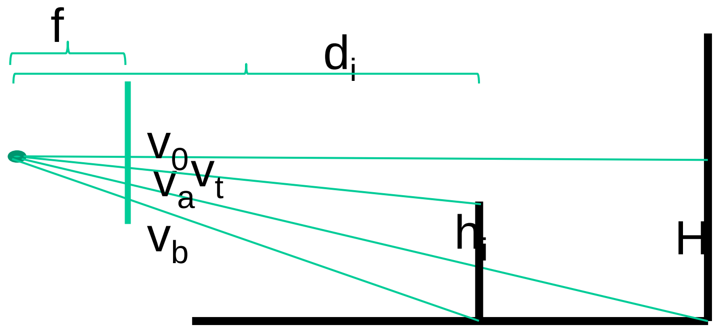
Can compute by similar triangles (CVA vs. CV'A')
Need to know focal length f (or FoV)

Note: can compute position of any object on the ground

- Simple unprojection
- What about things off the ground?

Get depth using similar triangles





$$f / (v_a - v_0) = d / h$$

$$h_i = (v_t - v_b) / (v_0 - v_a) * H$$

$$f / (v_b - v_t) = d_i / h_i$$

Can compute by similar triangles (CVA vs. CV'A')
Need to know focal length f (or FoV)

Can compute 3D position of any object on the ground
w/ unprojection

- What about things off the ground?

Step 2: map image textures into frontal view

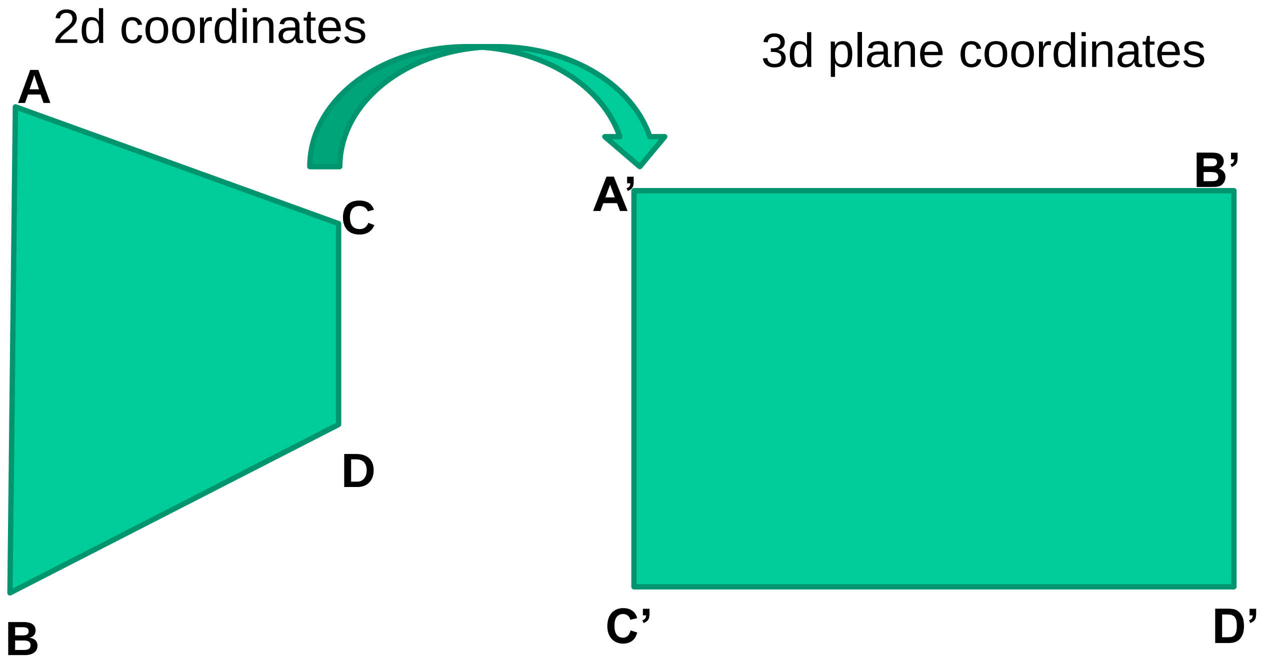
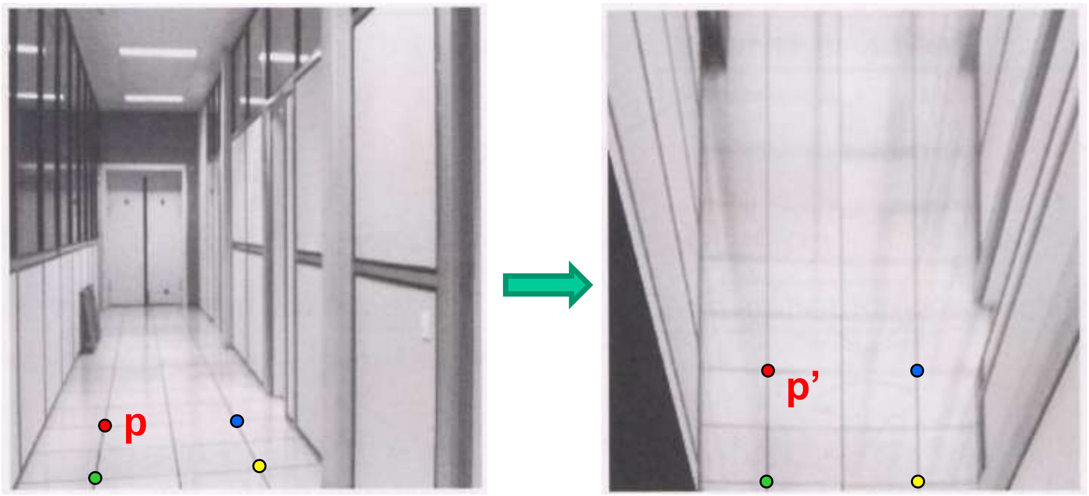


Image rectification by homography



To unwarp (rectify) an image solve for homography **H** given **p** and **p'**: w**p'=Hp**

Computing homography

Assume we have four matched points: How do we compute homography **H**?

Direct Linear Transformation (DLT)

$$\mathbf{p}' = \mathbf{H}\mathbf{p} \qquad \mathbf{p}' = \begin{bmatrix} w'u' \\ w'v' \\ w' \end{bmatrix} \qquad \mathbf{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

$$\begin{bmatrix} -u & -v & -1 & 0 & 0 & 0 & uu' & vu' & u' \\ 0 & 0 & 0 & -u & -v & -1 & uv' & vv' & v' \end{bmatrix} \mathbf{h} = \mathbf{0} \qquad \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}$$

Computing homography

Direct Linear Transform

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1u_1' & v_1u_1' & u_1' \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1v_1' & v_1v_1' & v_1' \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_nv_n' & v_nv_n' & v_n' \end{bmatrix}\mathbf{h} = \mathbf{0} \Rightarrow \mathbf{Ah} = \mathbf{0}$$

Apply SVD: $\boldsymbol{USV^T} = \boldsymbol{A}$

$\boldsymbol{h} = \boldsymbol{V}_{\text{smallest}}$ (column of $\boldsymbol{V^T}$ corresponds to smallest singular value)

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_9 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Python

```
U, S, Vt = scipy.linalg.svd(A)
# last column corr. to smallest singular value
h = Vt[:,-1];
```

Explanation of SVD, solving systems of linear equations, derivation of solution here

Solving for homography (another formulation)

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1u_1' & v_1u_1' & u_1' \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1v_1' & v_1v_1' & v_1' \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_nv_n' & v_nv_n' & v_n' \end{bmatrix} \mathbf{h} = \mathbf{0} \Rightarrow \mathbf{A}\mathbf{h} = \mathbf{0}$$

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

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

- Since **h** is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
    - Can derive using Lagrange multipliers method
- Works with 4 or more points

Tour into the picture algorithm

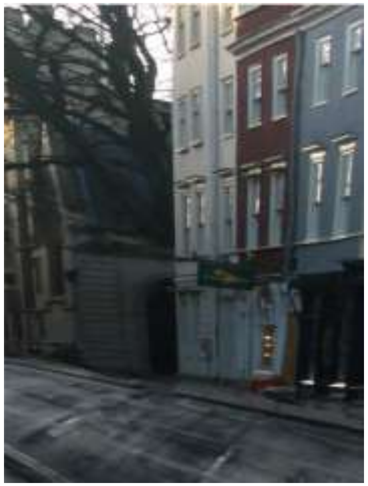
1. Set the box corners

Tour into the picture algorithm

1. Set the box corners
2. Set the VP
3. Get 3D coordinates
   - Compute height, width, and depth of box
4. Get texture maps
   - homographies for each face
5. Create file to store plane coordinates and texture maps

Result

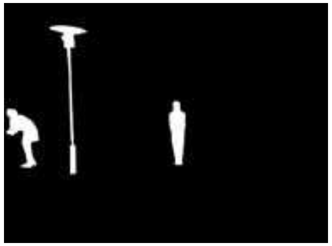
Render from new views



http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15463-f08/www/proj5/www/dmillett/

Foreground Objects

Use separate billboard for each



For this to work, three separate images used:

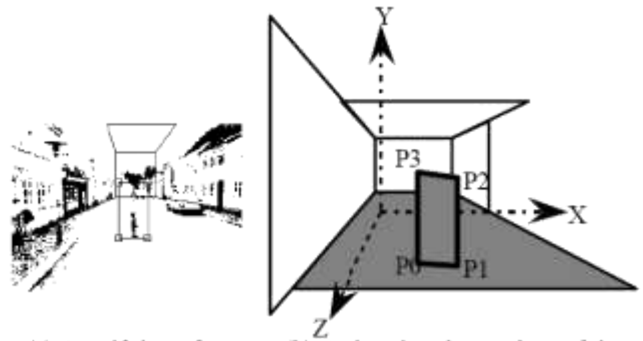
- Original image.
- Mask to isolate desired foreground images.
- Background with

Foreground Objects

Add vertical rectangles for each foreground object

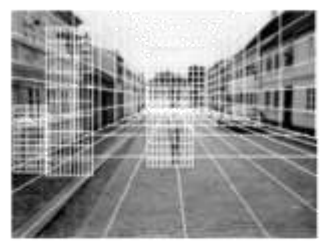
Can compute 3D coordinates P0, P1 since they are on known plane.

P2, P3 can be computed as before (similar triangles)



(a) Specifying of a foreground object

(b) Estimating the vertices of the foreground object model

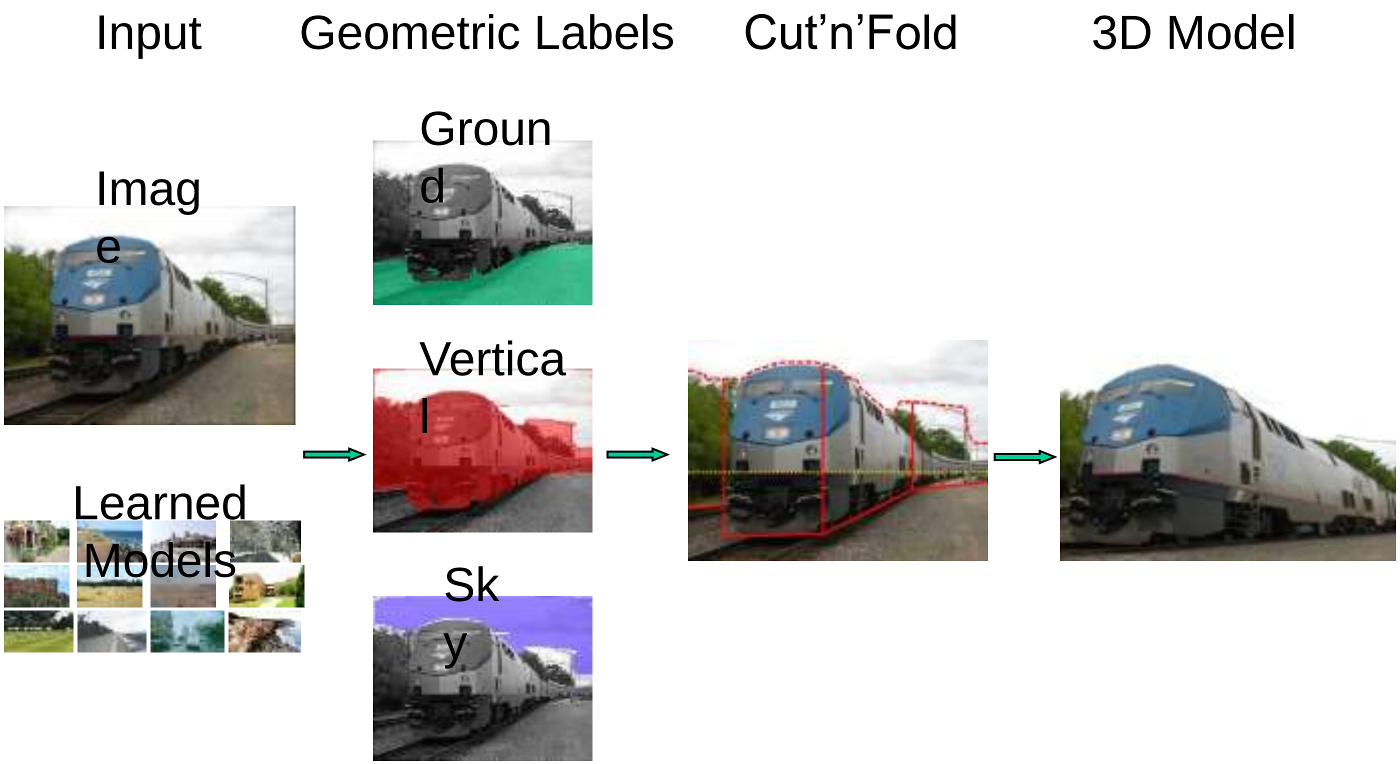


(c) Three foreground object models

Foreground Result

Video from CMU class:
http://www.youtube.com/watch?v=dUAtdmGwcuM

Automatic Photo Pop-up



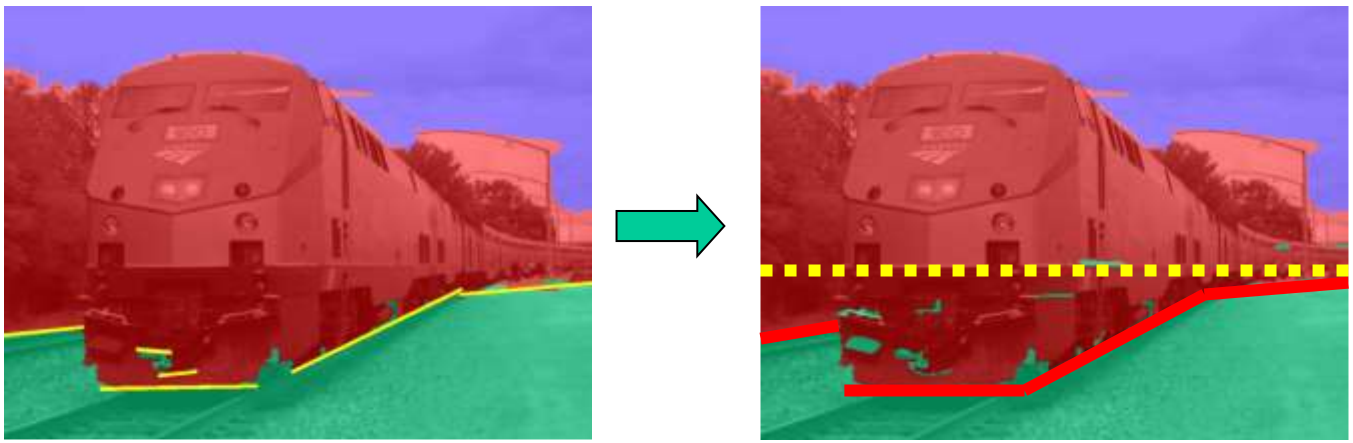
Hoiem et al. 2005

Cutting and Folding

Fit ground-vertical boundary

- Iterative Hough transform

Cutting and Folding

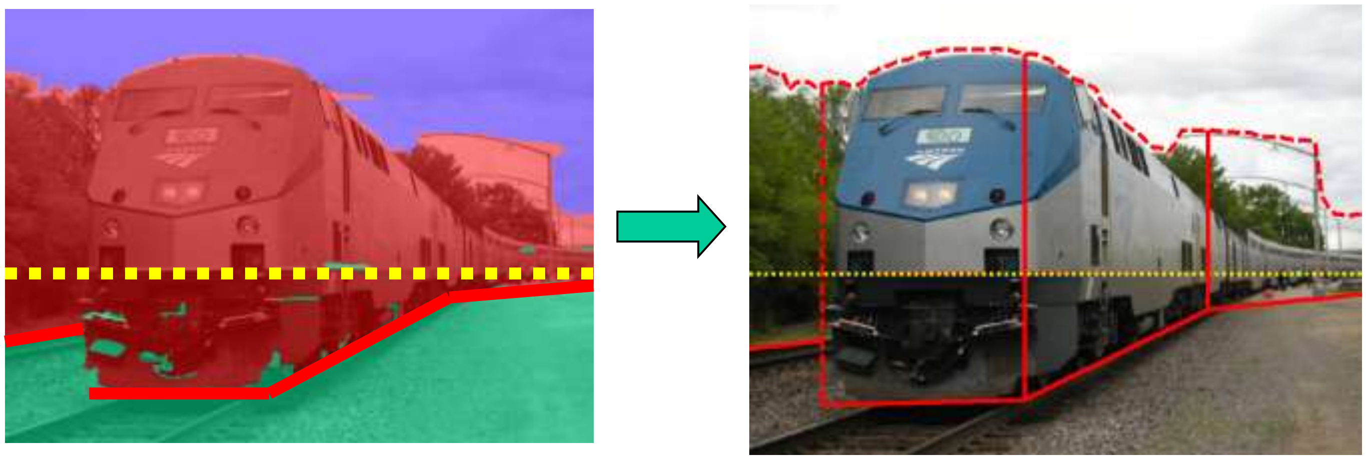


Form polylines from boundary segments

- Join segments that intersect at slight angles
- Remove small overlapping polylines

Estimate horizon position from perspective cues

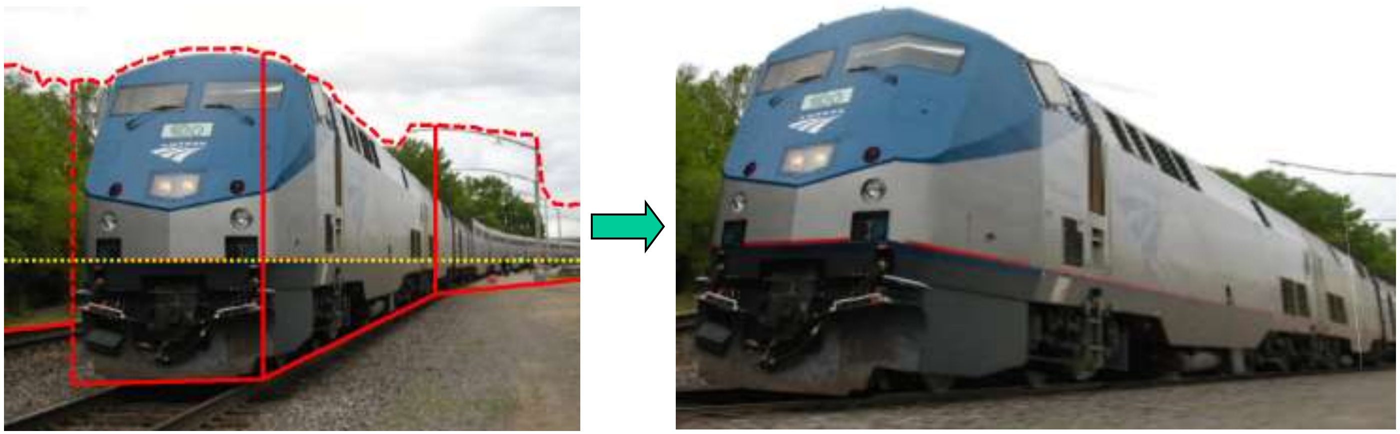
Cutting and Folding



``Fold'' along polylines and at corners

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

Cutting and Folding



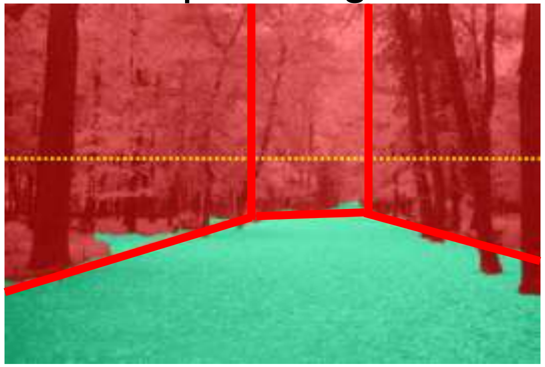
Construct 3D model

Texture map

Results

http://www.cs.illinois.edu/homes/dhoiem/projects/popup/

Input Image



Cut and Fold

Automatic Photo Pop-up

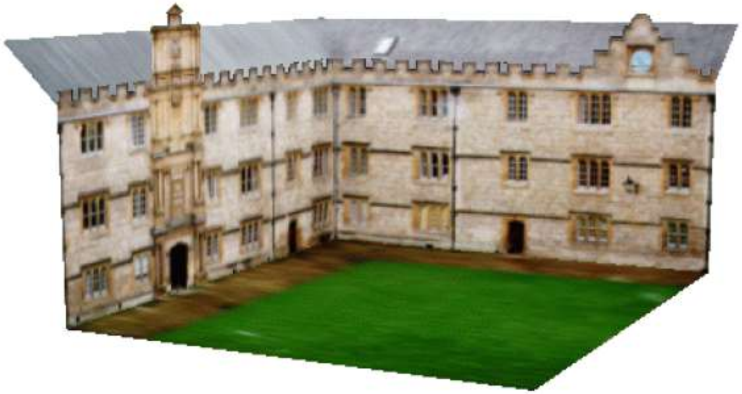
Results

Input Image

Automatic Photo Pop-up

Comparison with Manual Method

Input Image



[Liebowitz et al. 1999]

Automatic Photo Pop-up (15 sec)!

Failures

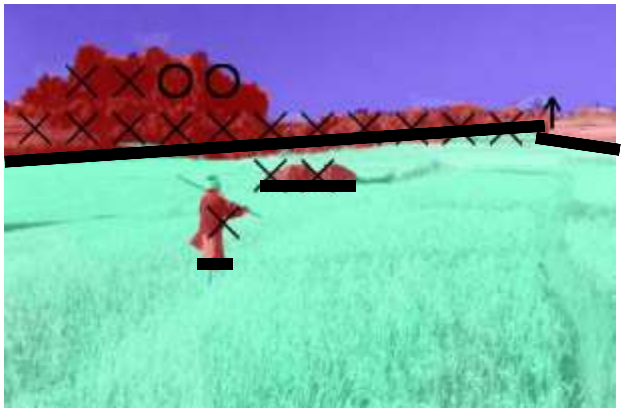
Labeling
Errors

Failures

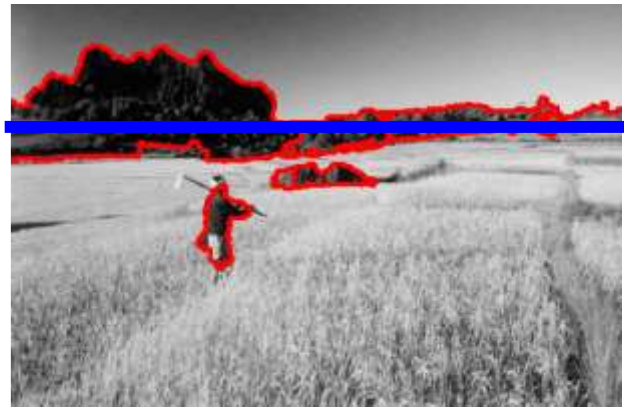
Foreground Objects




Adding Foreground Labels



Recovered Surface Labels +
Ground-Vertical Boundary Fit



Object Boundaries + Horizon

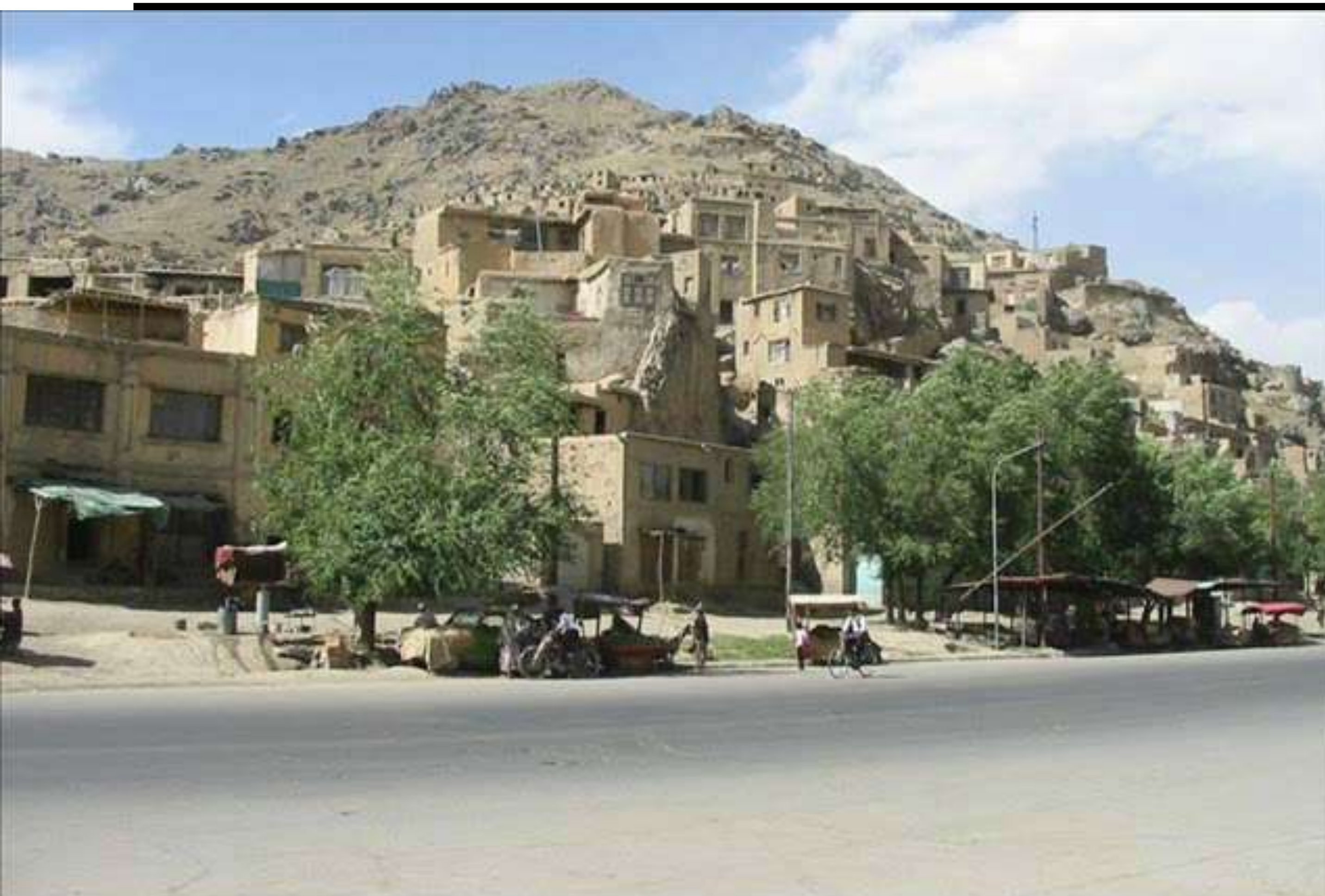
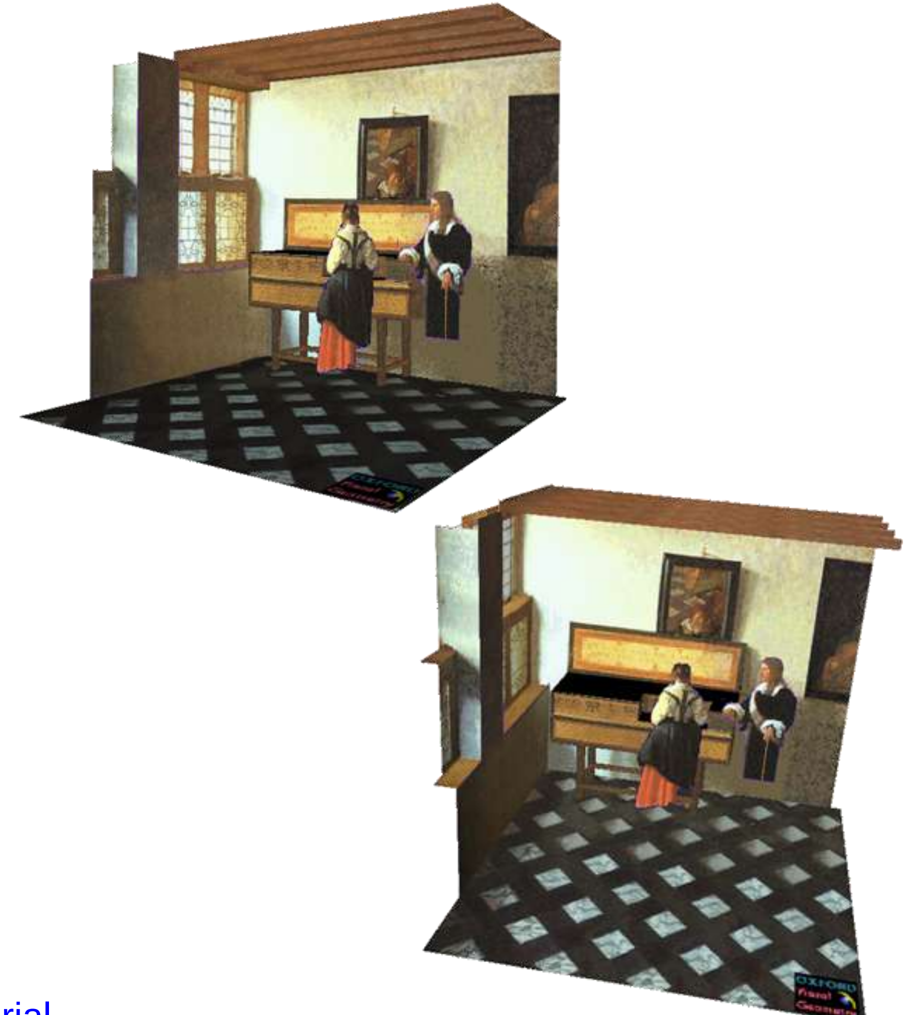


Image rectification: example

Piero della Francesca*, Flagellation,* ca. 1455

Application: 3D modeling from a single image

J. Vermeer, *Music Lesson*, 1662



A. Criminisi, M. Kemp, and A. Zisserman, [Bringing Pictorial Space to Life: computer techniques for the analysis of paintings](), *Proc. Computers and the History of Art*, 2002

[http://research.microsoft.com/en-us/um/people/antcrim/ACriminisi_3D_Museum.wmv]()

Application: 3D modeling from a single image

D. Hoiem, A.A. Efros, and M. Hebert, "Automatic Photo Pop-up", SIGGRAPH 2005.

http://dhoiem.cs.illinois.edu/projects/popup/popup_movie_450_250.mp4

Application: Image editing

Inserting synthetic objects into images:
http://vimeo.com/28962540

K. Karsch and V. Hedau and D. Forsyth and D. Hoiem, "Rendering Synthetic Objects into Legacy Photographs," *SIGGRAPH Asia* 2011

Application: Object recognition

D. Hoiem, A.A. Efros, and M. Hebert, "Putting Objects in Perspective", CVPR 2006