Single-view 3D Reconstruction

Computational Photography

Derek Hoiem, University of Illinois

Some slides from Alyosha Efros, Steve Seitz

Project 3 extension (one day)

- EWS down Fri 9pm to Sun 10am
- Project 3 now due Tues

Take-home question

Suppose you have estimated three vanishing points corresponding to orthogonal directions. How can you recover the rotation matrix that is aligned with the 3D axes defined by these points?

- Assume that intrinsic matrix K has three parameters
- Remember, in homogeneous coordinates, we can write a 3d point at infinity as (X, Y, Z, 0)

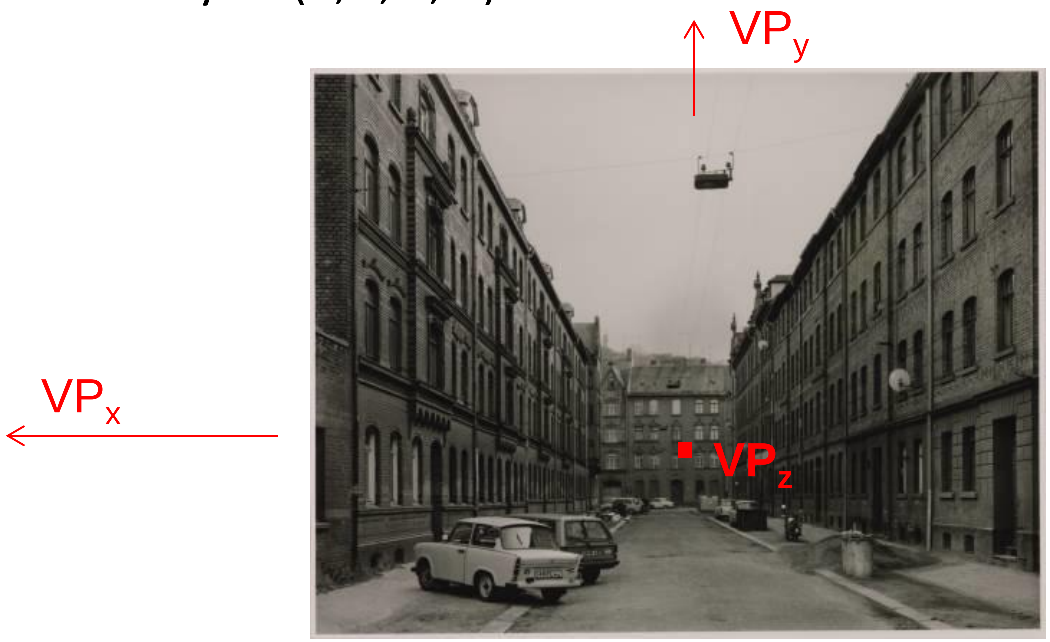
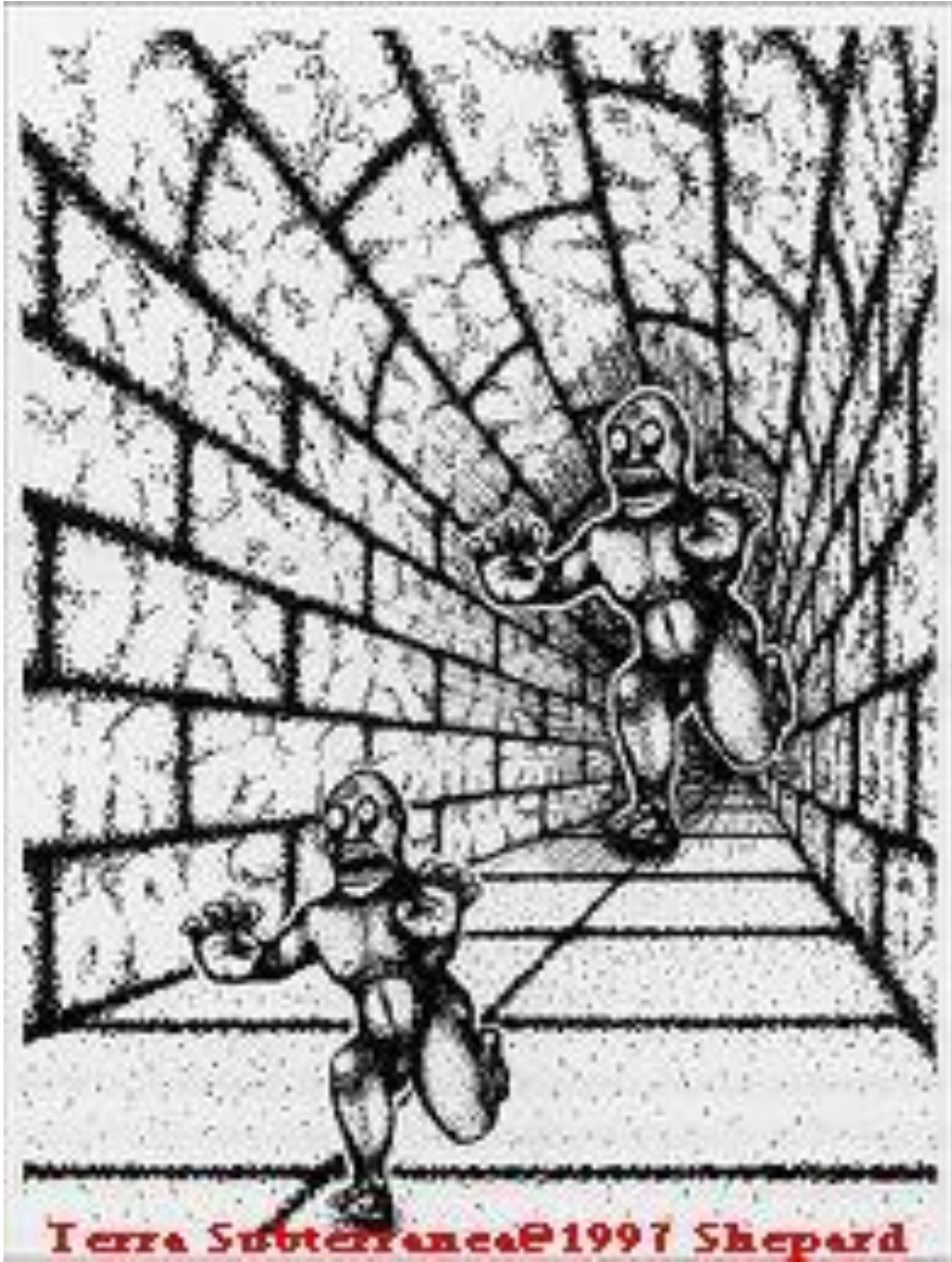


Photo from online Tate collection

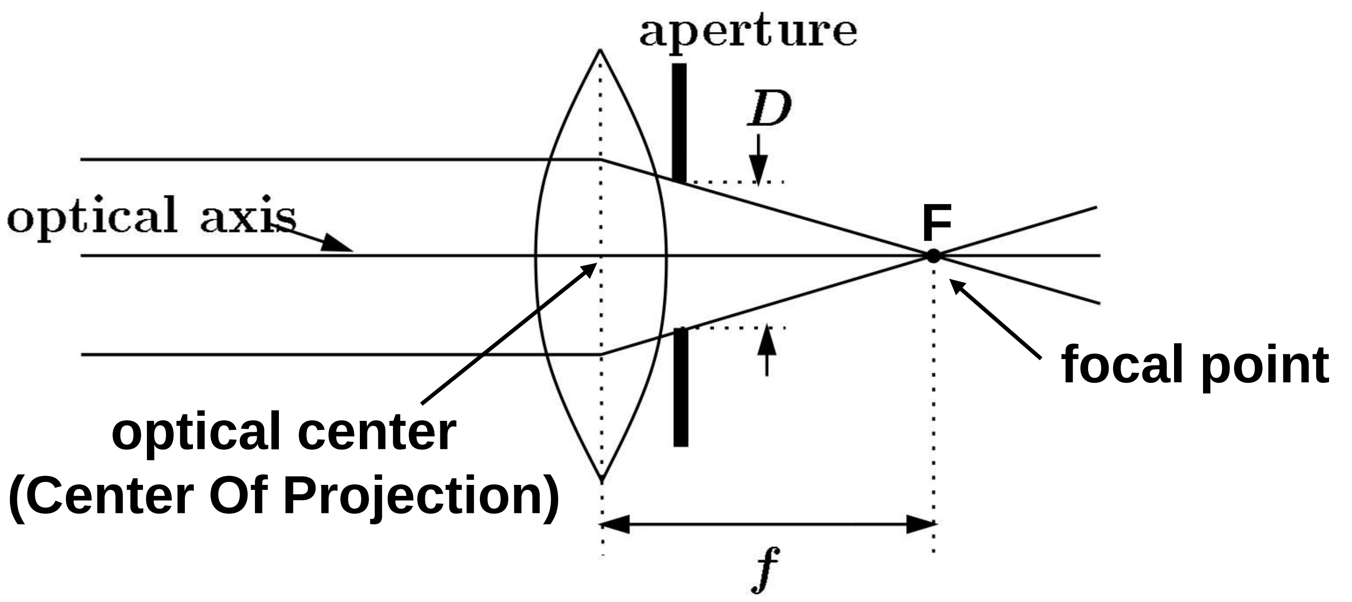
Take-home question

Assume that the camera height is 5 ft.

- What is the height of the man?
- What is the height of the building?


Terra Subterranea©1997 Shepard

Focal length, aperture, depth of field



A lens focuses parallel rays onto a single focal point

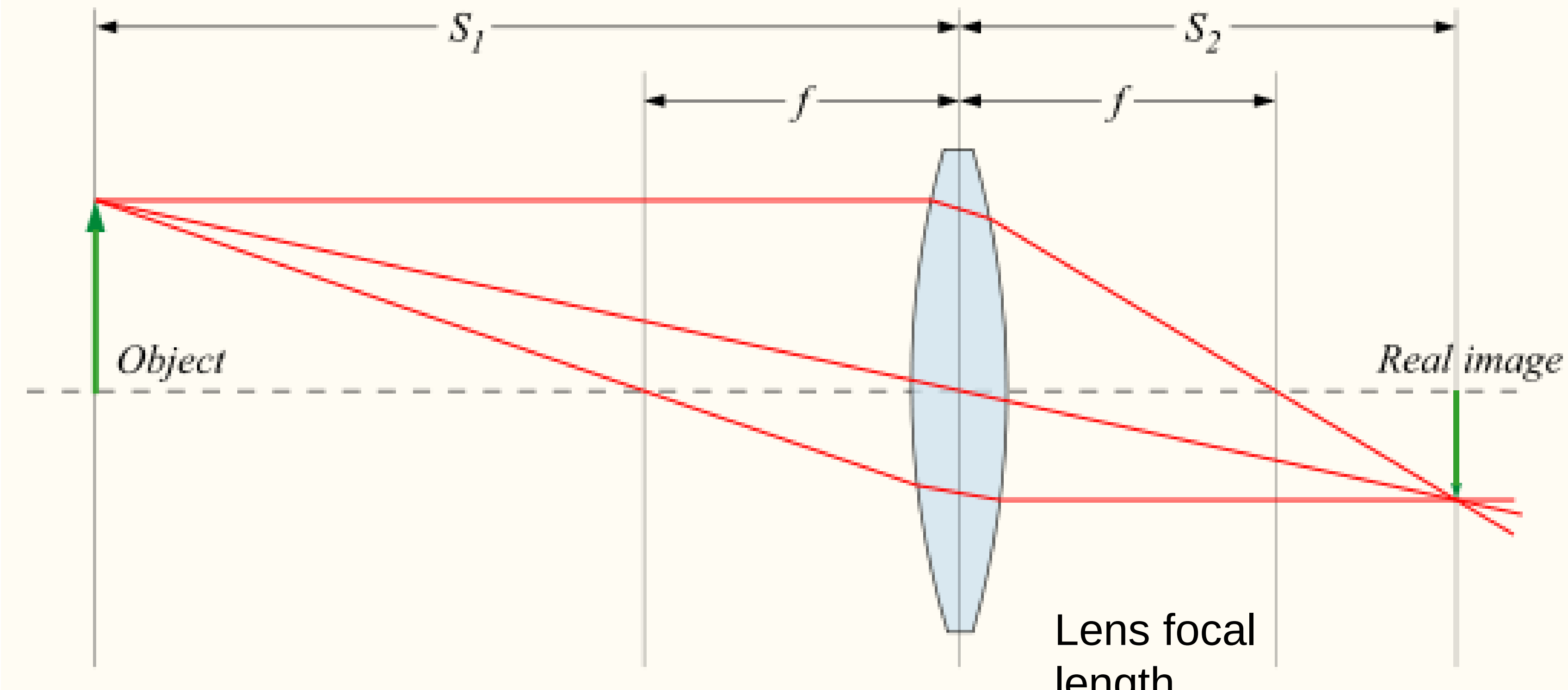
- focal point at a distance *f* beyond the plane of the lens
- Aperture of diameter D restricts the range of rays

Slide source: Seitz

Focus with lenses

Distance to object

Distance to sensor



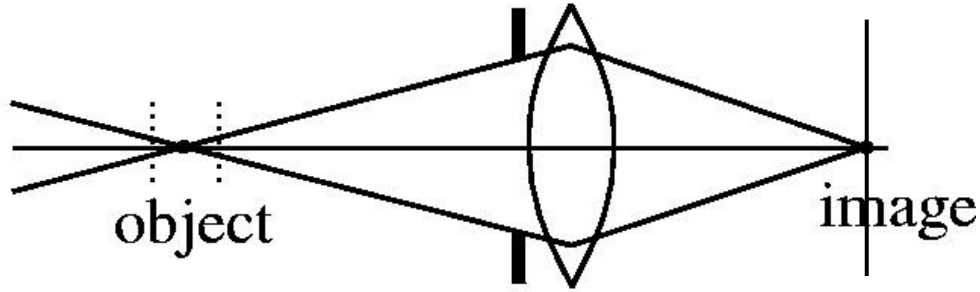
Equation for objects in focus

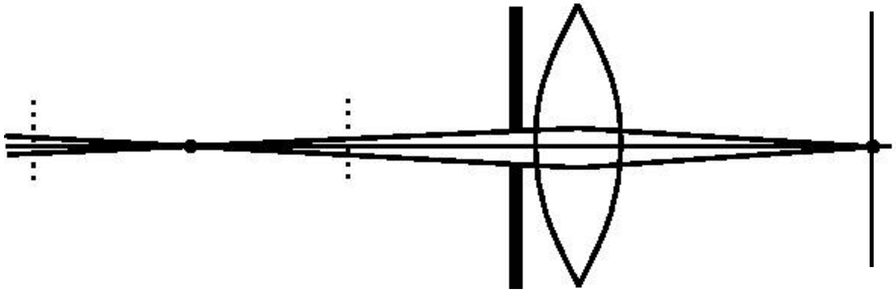
$$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}$$

Source: http://en.wikipedia.org/wiki/File:Lens3.svg

Slide source: Seitz

The aperture and depth of field





f / 5.6

f / 32

Changing the aperture size or focusing distance affects depth of field

- f-number (f/#) =focal_length / aperture_diameter (e.g., f/16 means that the focal length is 16 times the diameter)
- When you change the f-number, you are changing the aperture

Flower images from Wikipedia http://en.wikipedia.org/wiki/Depth_of_field

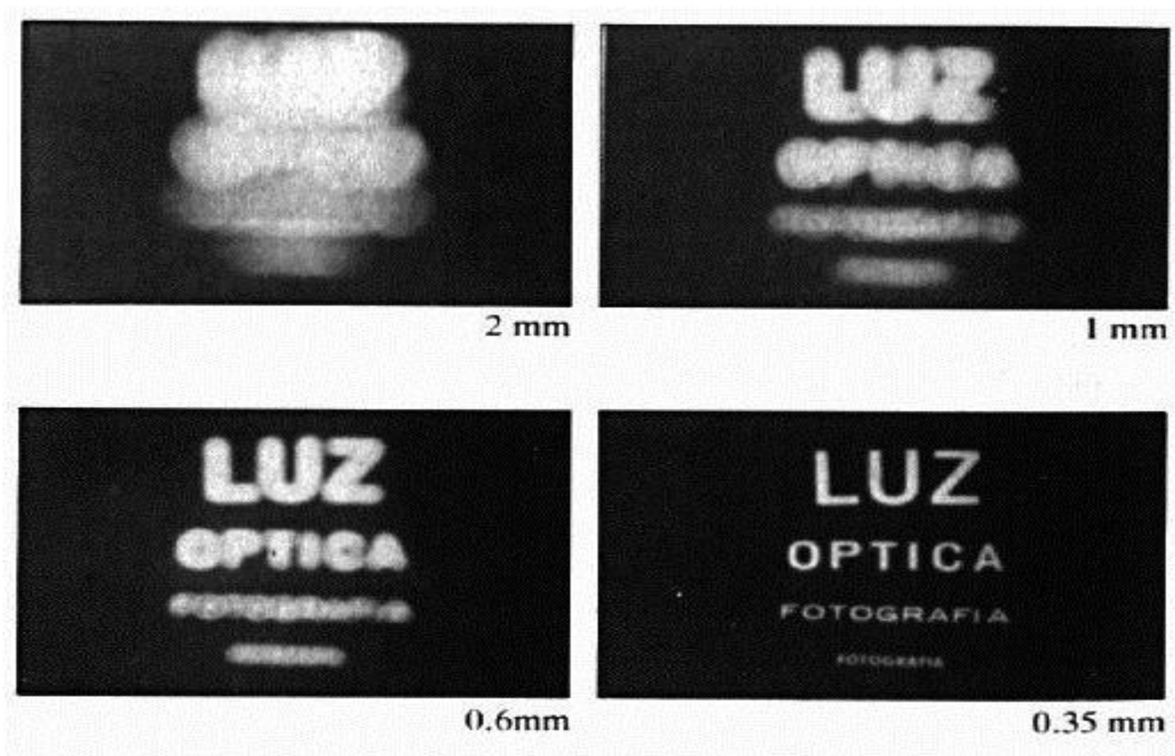
Slide from Efros

Varying the aperture

Large aperture = small DOF

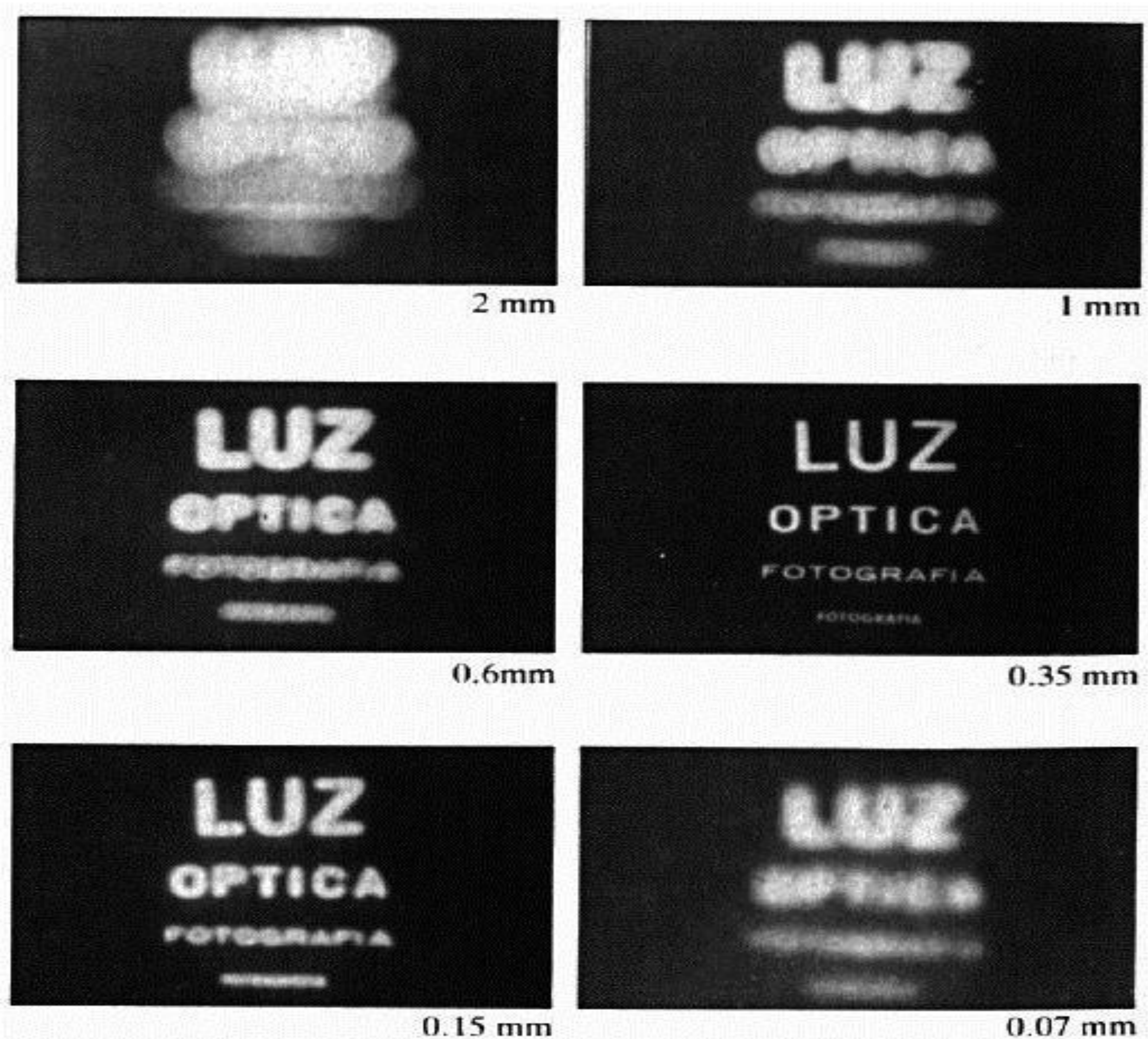
Small aperture = large DOF

Shrinking the aperture



- Why not make the aperture as small as possible?
  - Less light gets through
  - Diffraction effects

Shrinking the aperture



Slide by Steve Seitz

The Photographer’s Great Compromise

What we want	How we get it	Cost
More spatial resolution	Increase focal length	Light, FOV
Broader field of view	Decrease focal length	Resolution, DOF
More depth of field	Decrease aperture	Light
	Increase aperture	DOF
More temporal resolution	Shorten exposure	Light
More light	Lengthen exposure	Temporal Res

Difficulty in macro (close-up) photography

- For close objects, we have a small relative DOF
- Can only shrink aperture so far

How to get both bugs in focus?

Solution: Focus stacking

1. Take pictures with varying focal length

Example from
http://www.wonderfulphotos.com/articles/macro/focus_stacking/

Solution: Focus stacking

1. Take pictures with varying focal length
2. Combine

Focus stacking

http://www.wonderfulphotos.com/articles/macro/focus_stacking/

Focus stacking

How to combine?

Web answer: With software (Photoshop, CombineZM)

How to do it automatically?

Focus stacking

How to combine?

1. Align images (e.g., using corresponding points)
2. Two ideas
   a) Mask regions by hand and combine with pyramid blend
   b) Gradient domain fusion (mixed gradient) without masking

Automatic solution would make a very interesting final project

Recommended Reading:

http://www.digital-photography-school.com/an-introduction-to-focus-stacking

http://www.zen20934.zen.co.uk/photography/Workflow.htm#Focus%20Stacking

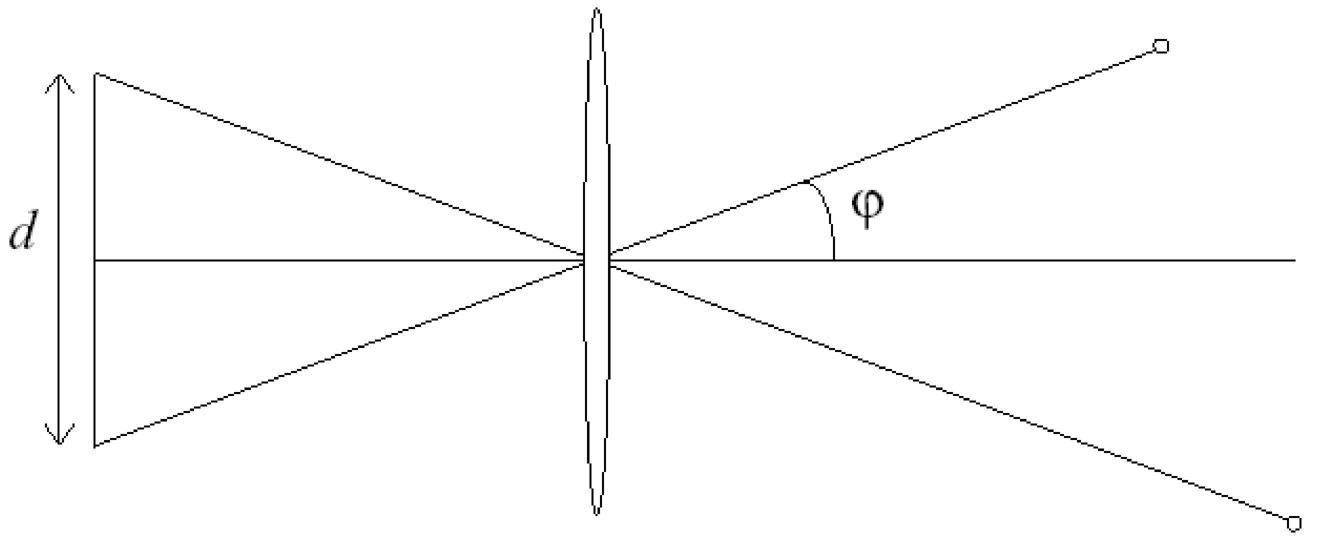
Relation between field of view and focal length

Field of view (angle width)

Film/Sensor Width

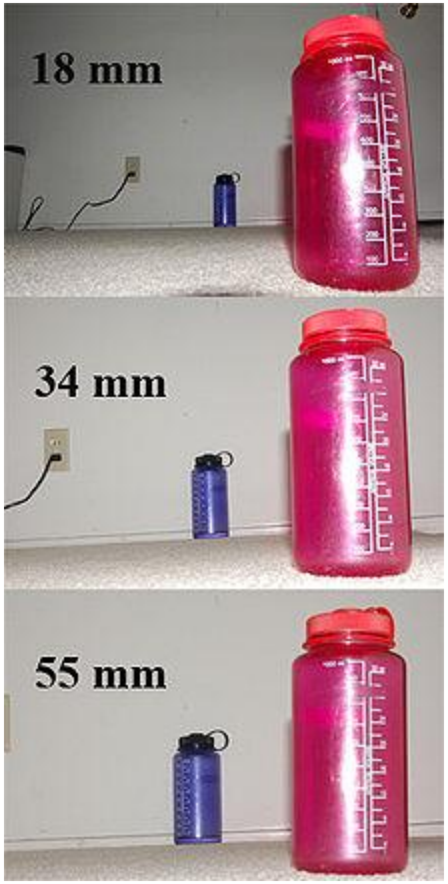
$$fov = 2\tan^{-1}\frac{d}{2f}$$

Focal length



Dolly Zoom or "Vertigo Effect"

http://www.youtube.com/watch?v=NB4bikrNzMk



How is this done?

Zoom in while moving away

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

Dolly zoom (or "Vertigo effect")

Field of view (angle width)

Film/Sensor Width

$$fov = 2\tan^{-1}\frac{d}{2f}$$

Focal length

width of object

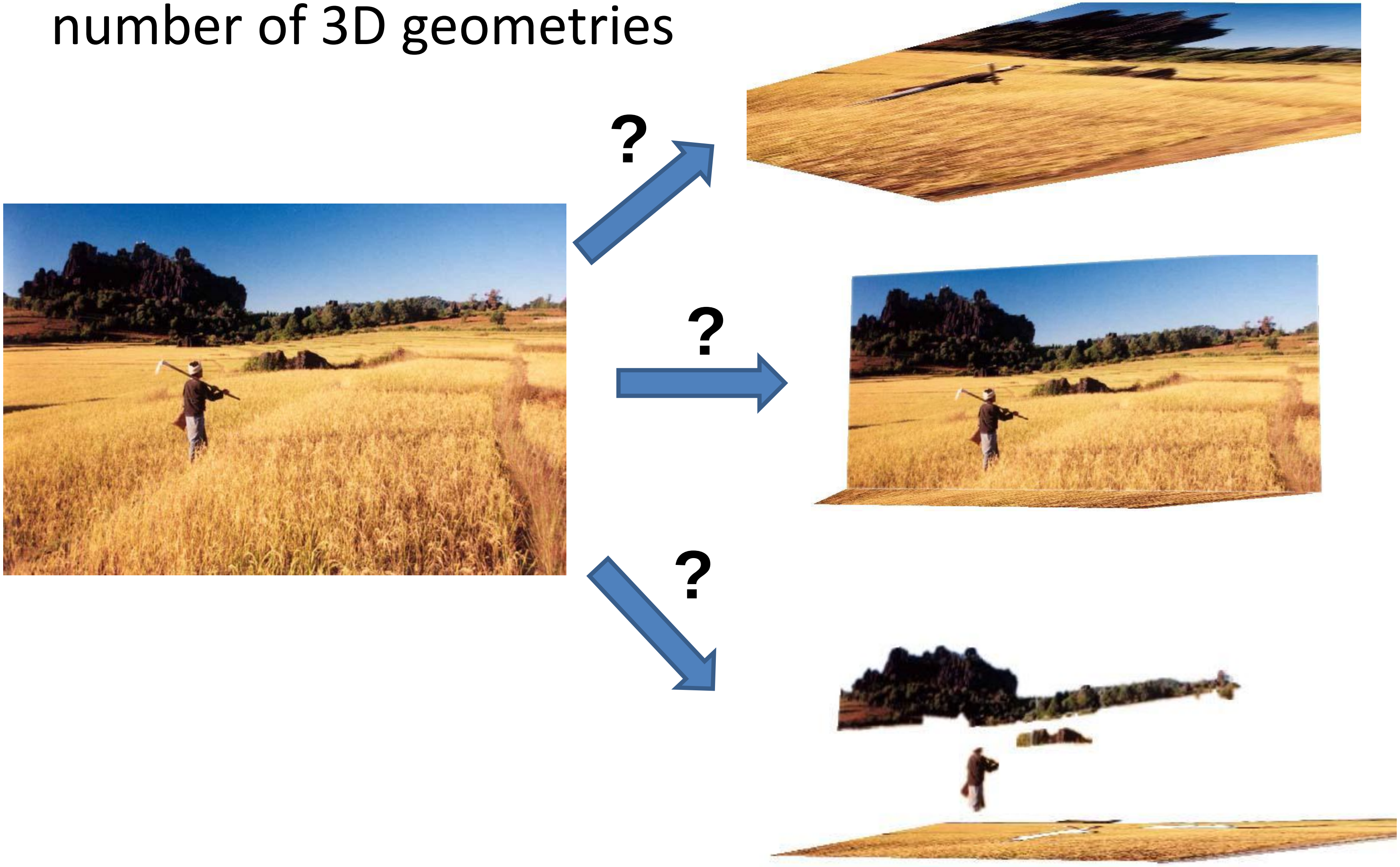
$$2\tan\frac{fov}{2} = \frac{width}{\text{distance}}$$

Distance between object and camera

Today’s class: 3D Reconstruction

The challenge

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



The solution

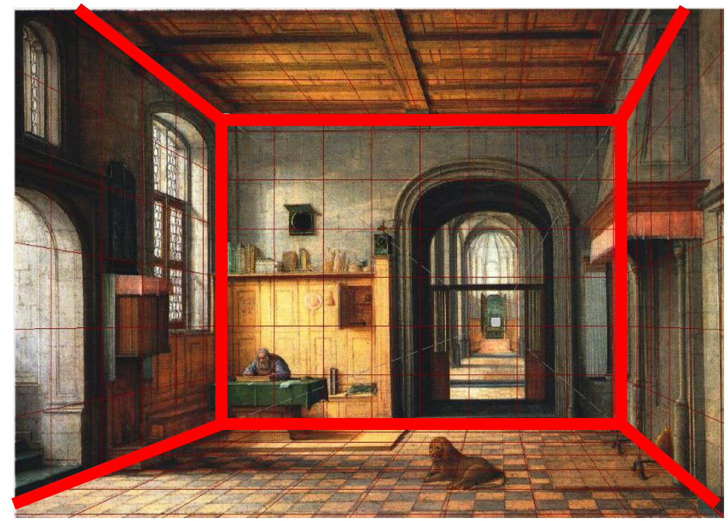
Make simplifying assumptions about 3D geometry

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 billboards

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)

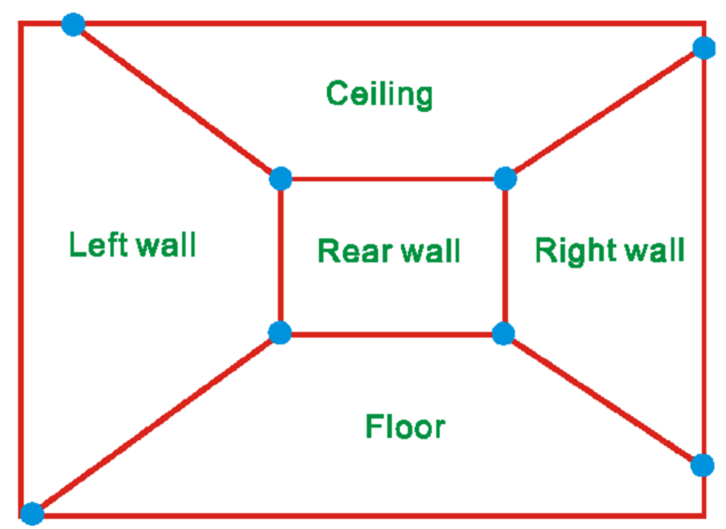
Key assumptions

- All walls are orthogonal
- Camera view plane is parallel to back of volume

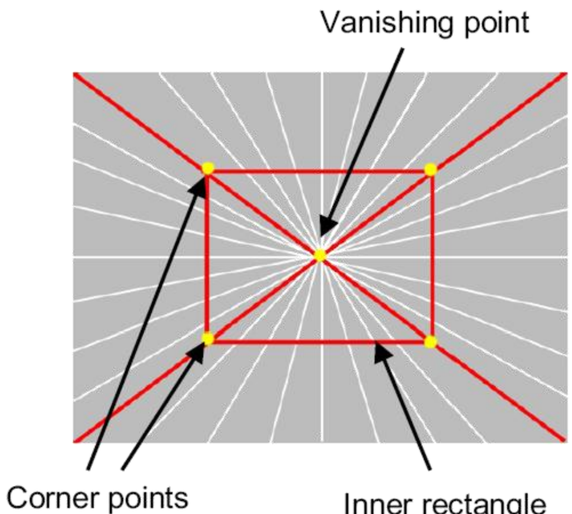
How many vanishing points does the box have?

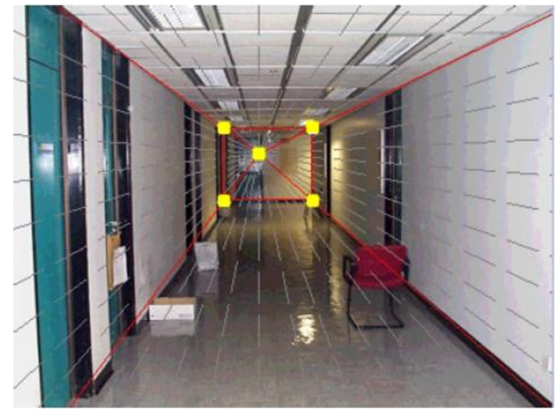
- Three, but two at infinity
- Single-point perspective

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



Step 1: specify scene geometry

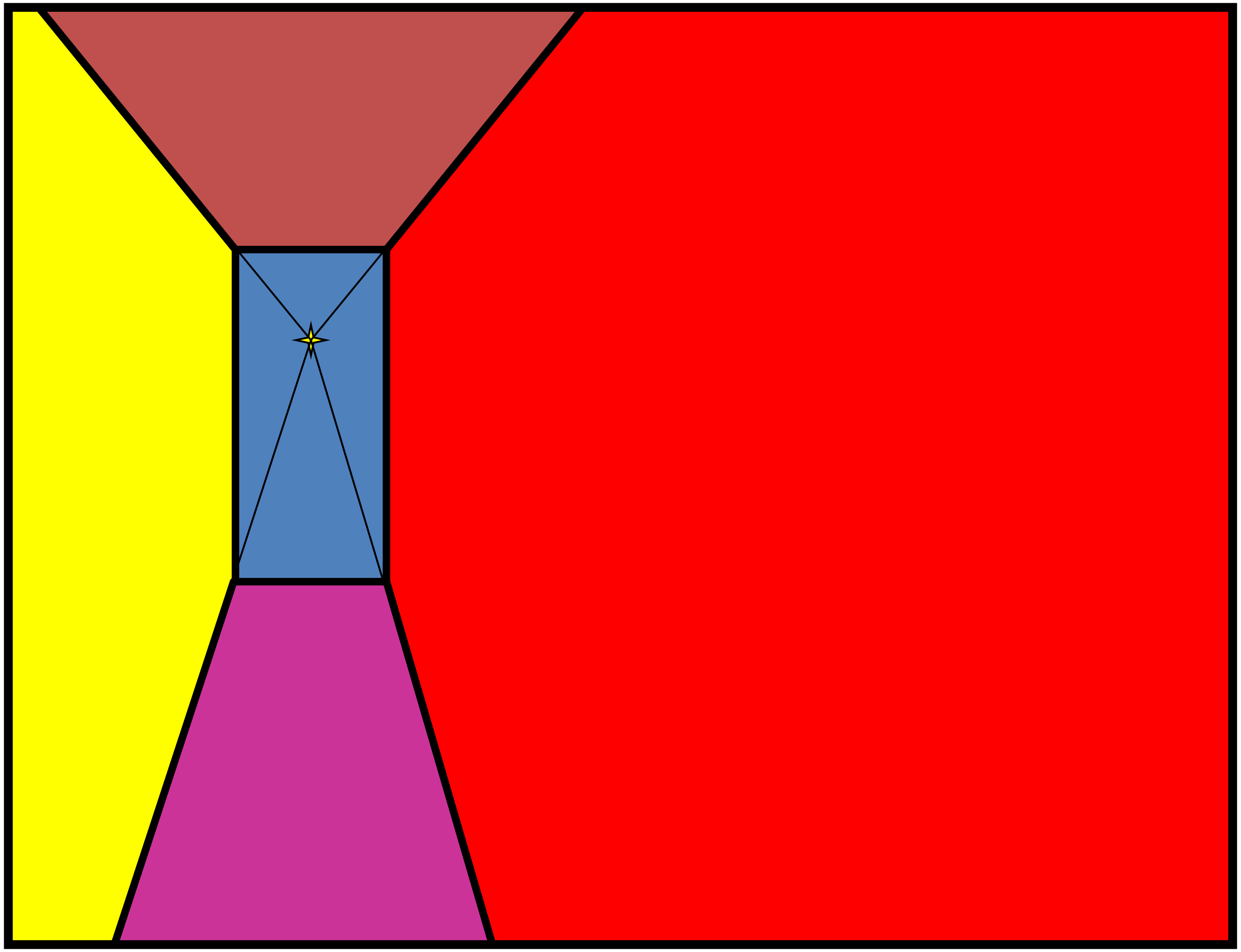




- User controls the inner box and the vanishing point placement (# of DOF?)

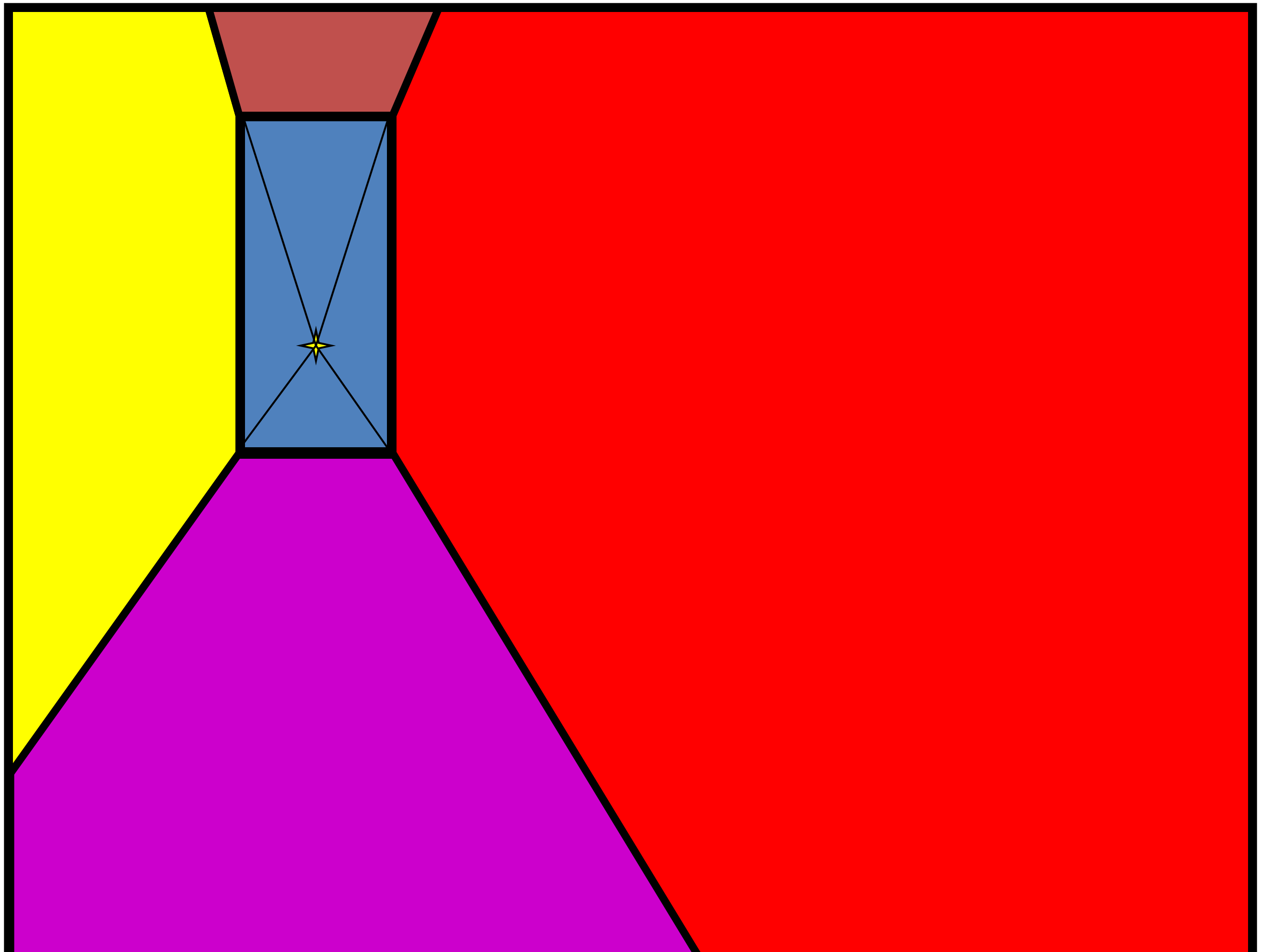
- Q: What's the significance of the vanishing point location?
- A: It's at eye (camera) level: ray from center of projection to VP is perpendicular to image plane
  - Under single-point perspective assumptions, the VP should be the principal point of the image

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



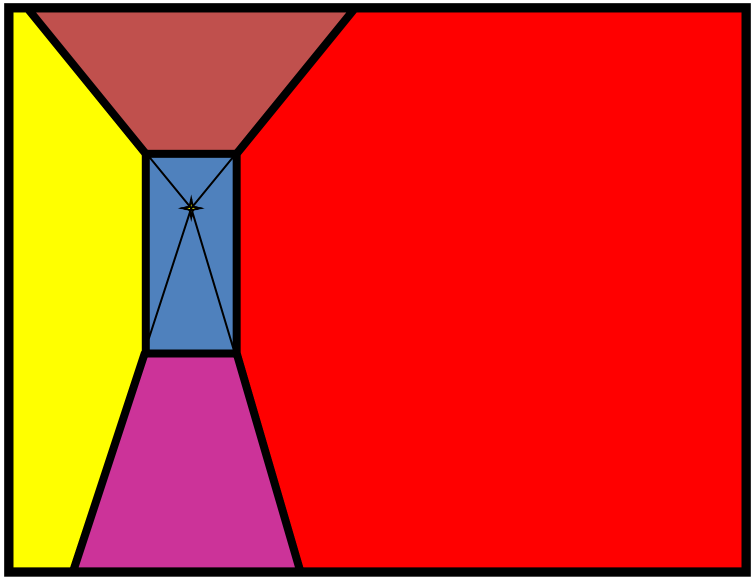
High Camera

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

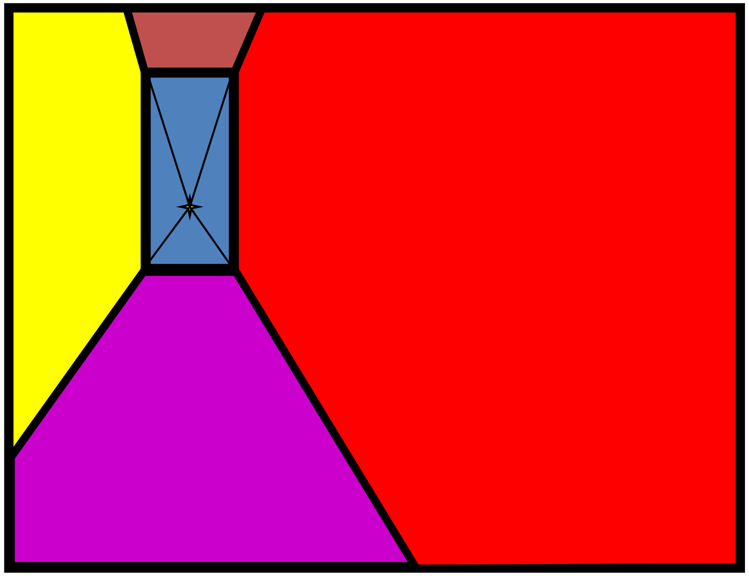


Low Camera

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.

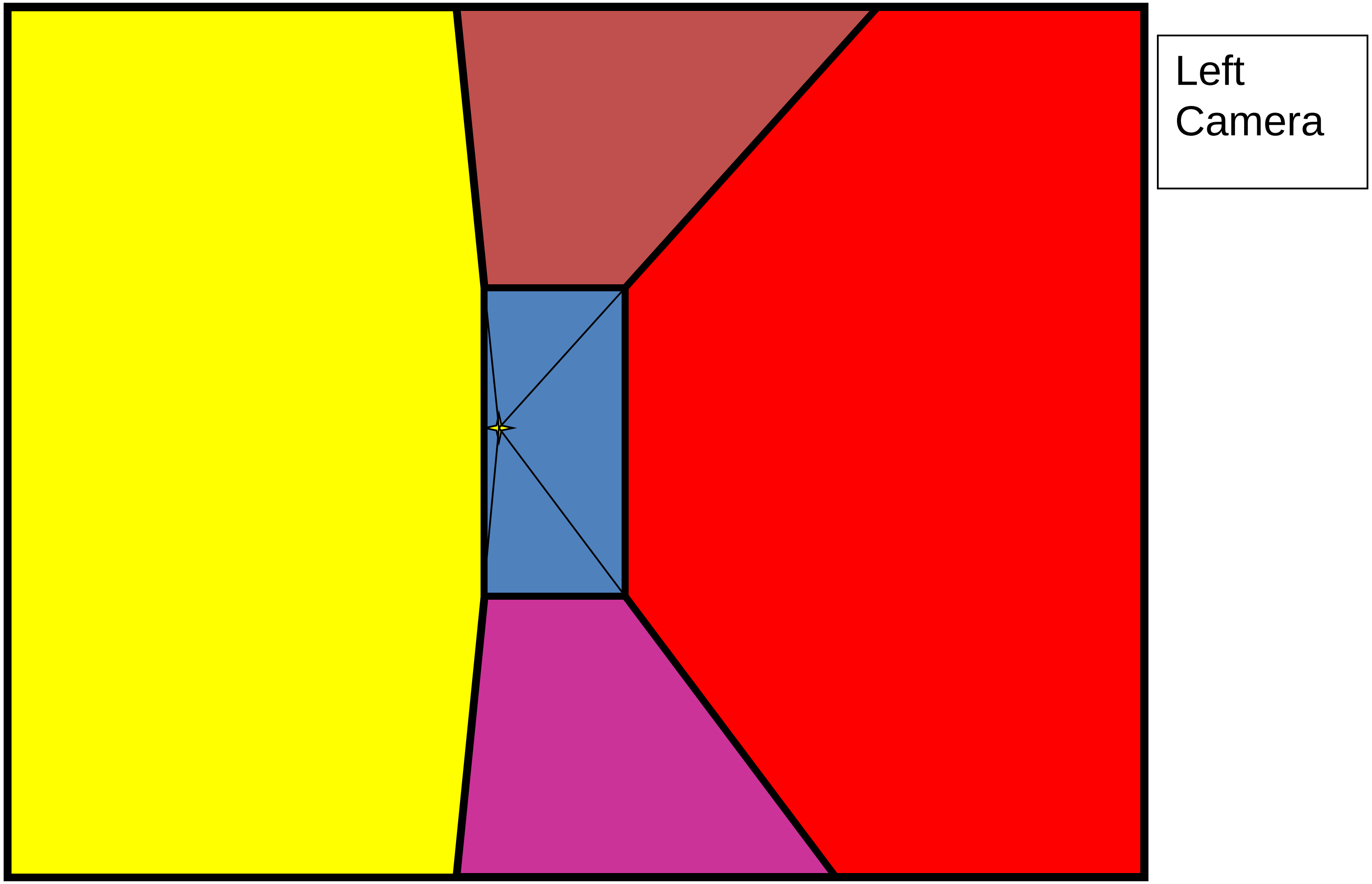


High Camera

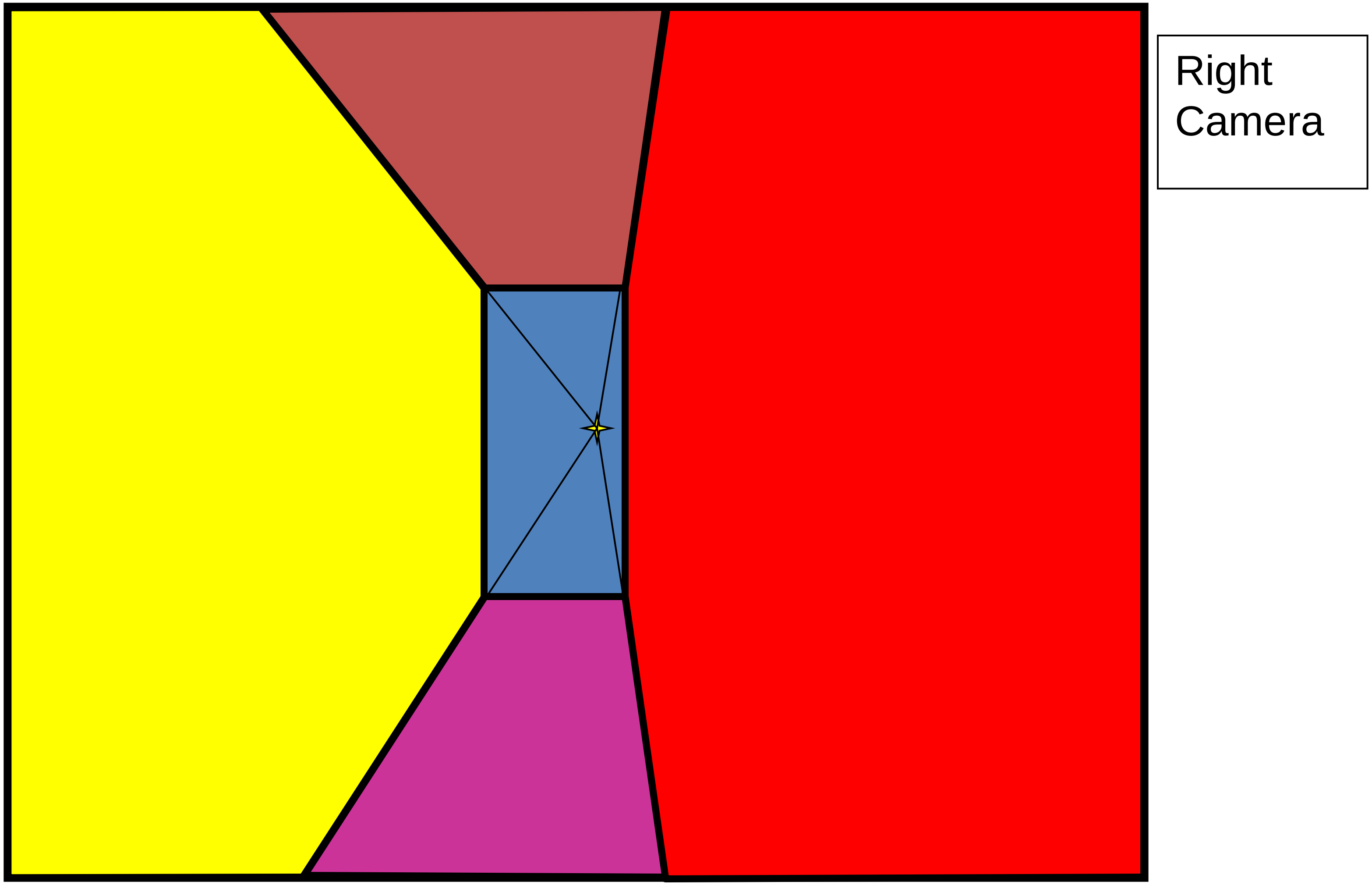


Low Camera

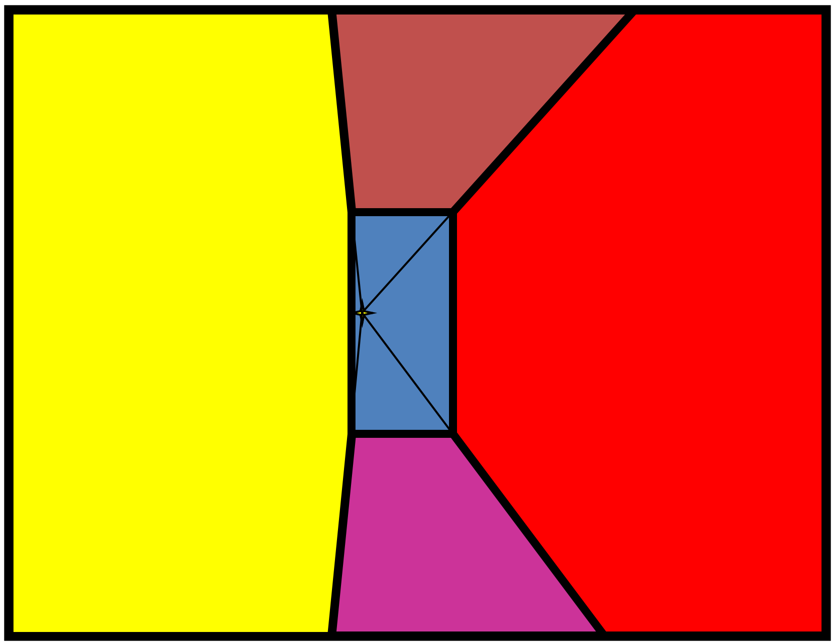
Another example of user input: vanishing point and back face of view volume are defined



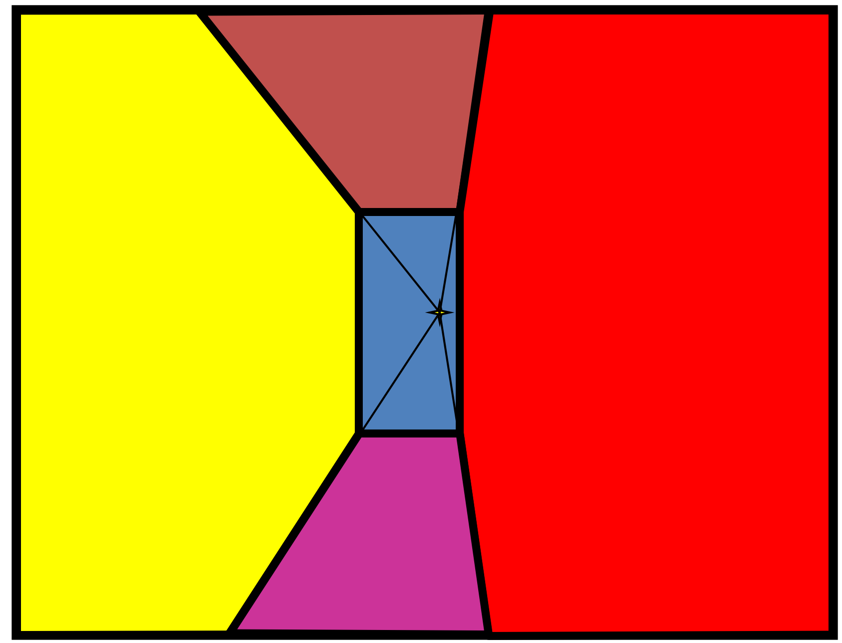
Another example of user input: vanishing point and back face of view volume are defined



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



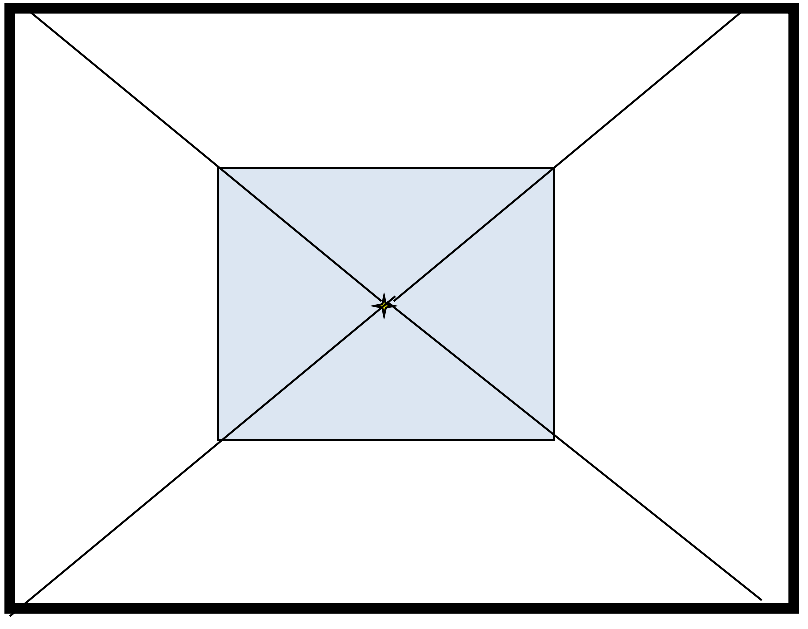
Left Camera



Right Camera

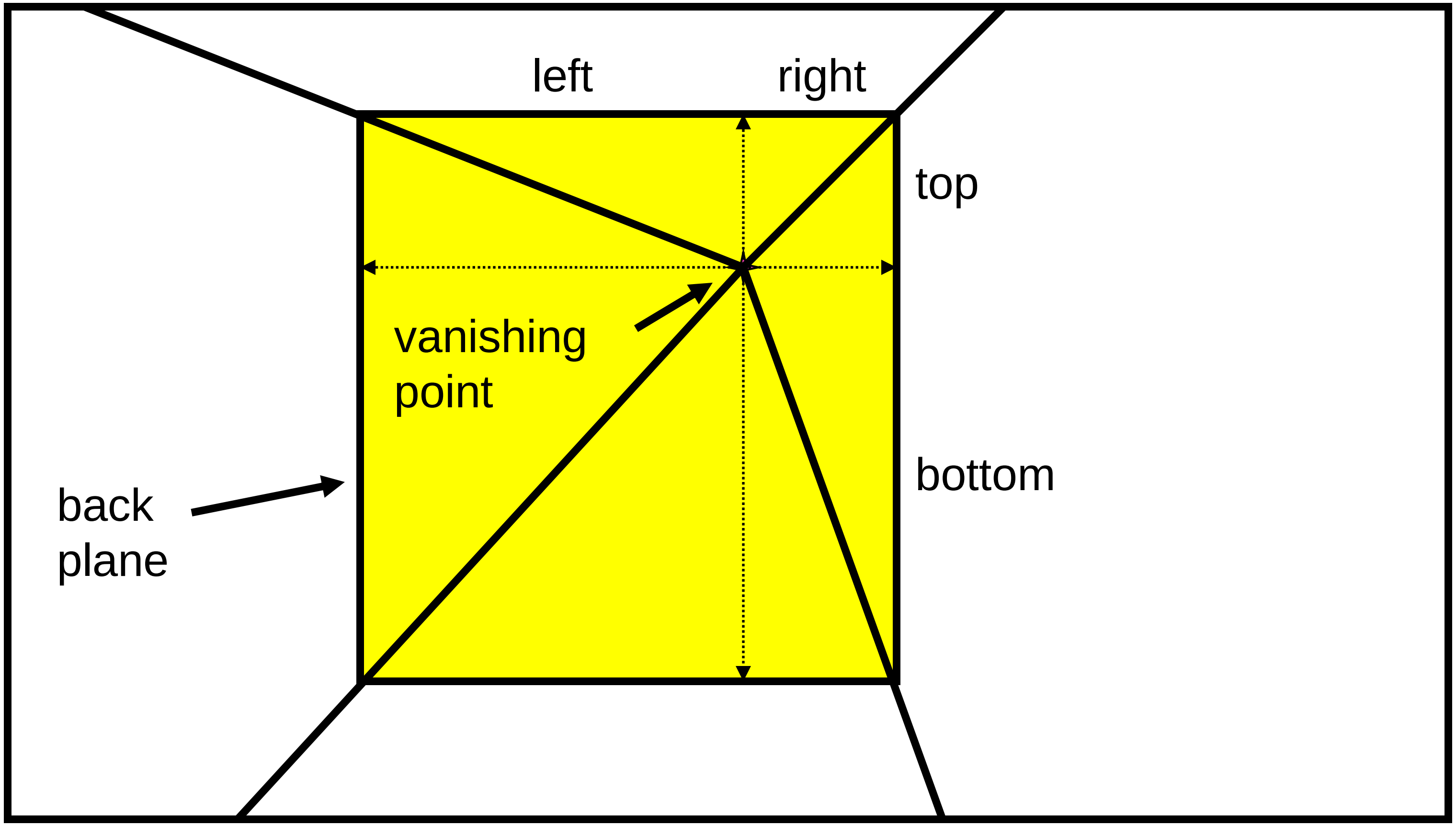
Question

- 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

- First, we can get ratios

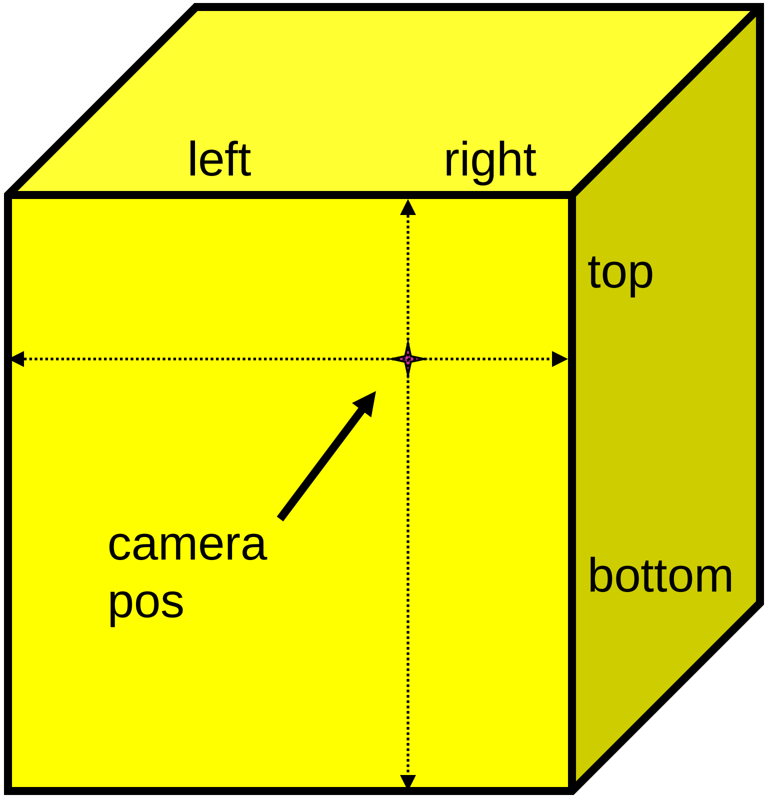


2D to 3D conversion

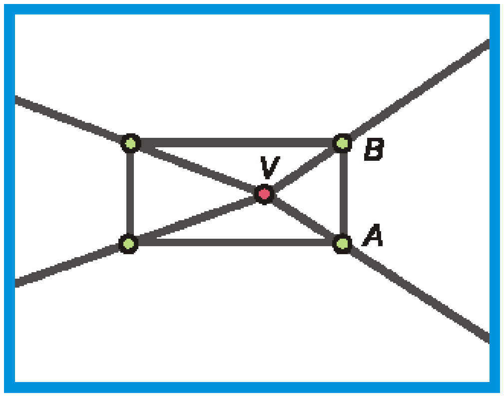
Size of user-defined back plane determines width/height throughout box (orthogonal sides)

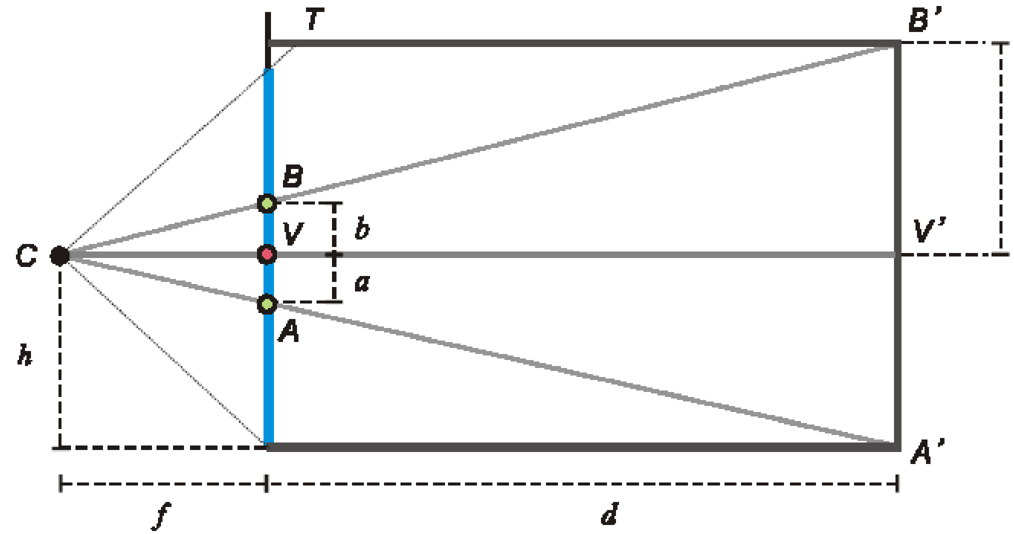
Use top versus side ratio to determine relative height and width dimensions of box

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?

Step 2: map image textures into frontal view

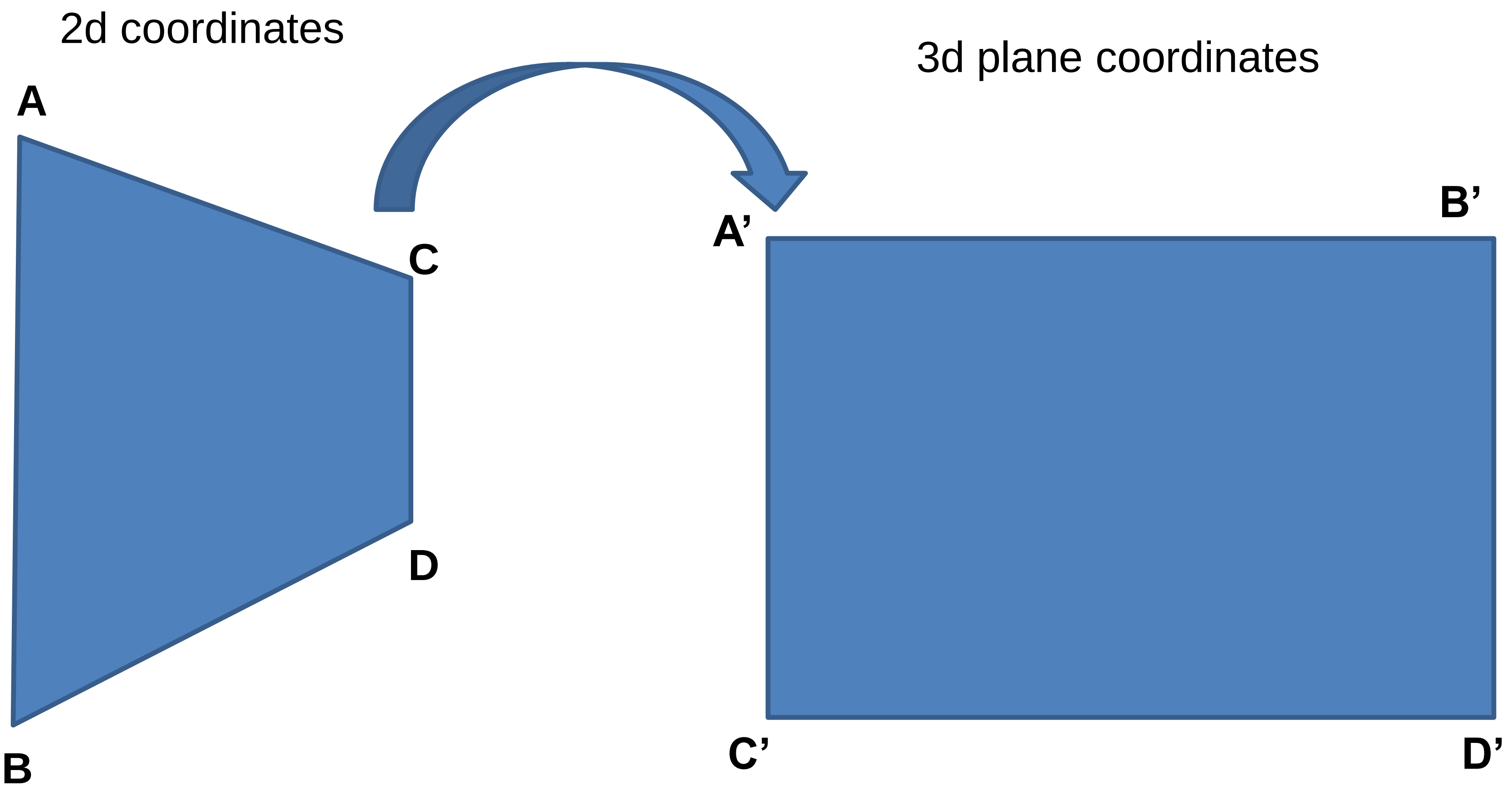
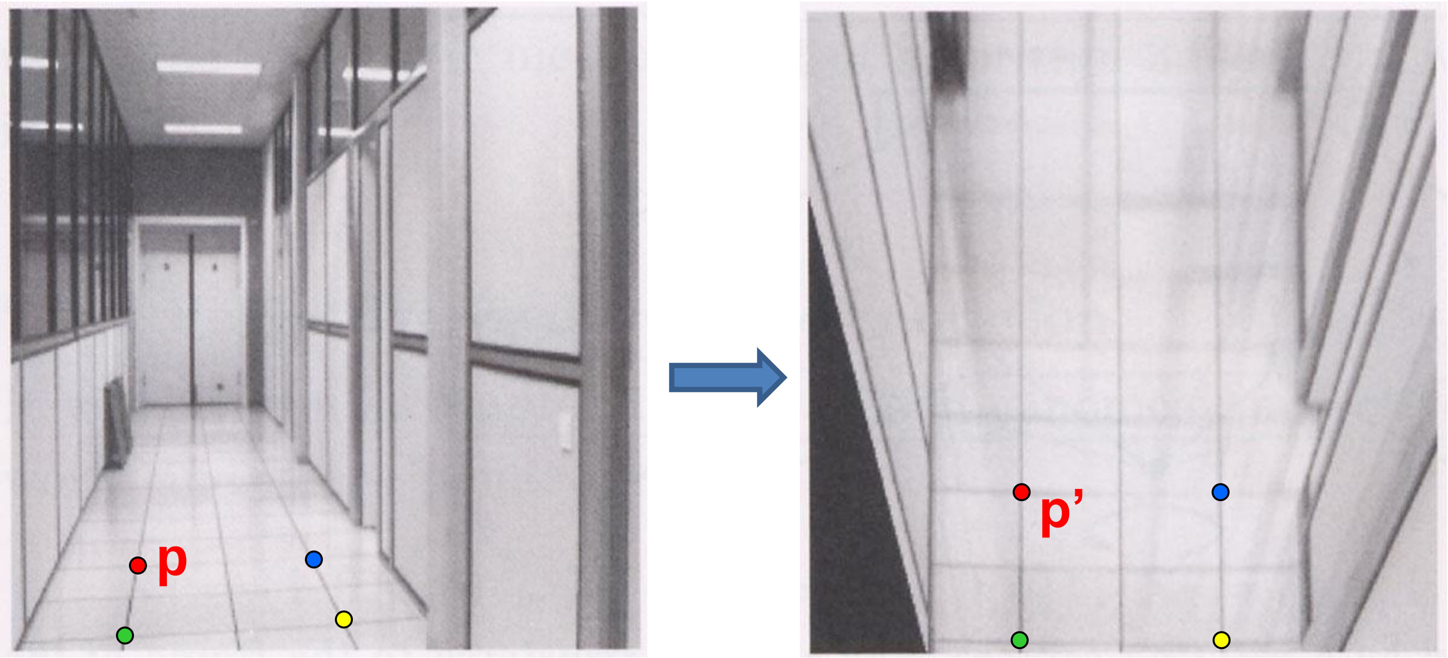


Image rectification



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)

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

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

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 $\boldsymbol{V}^T$ corr. to smallest singular value)

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

Matlab

```
[U, S, V] = svd(A);
h = V(:, end);
```

Explanation of SVD (also here) and solving systems of linear equations

Solving for homographies (more detail)

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

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

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

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

Solving for homographies (more detail)

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

A (2n × 9) **h** (9) **0** (2n)

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

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

Tour into the picture algorithm

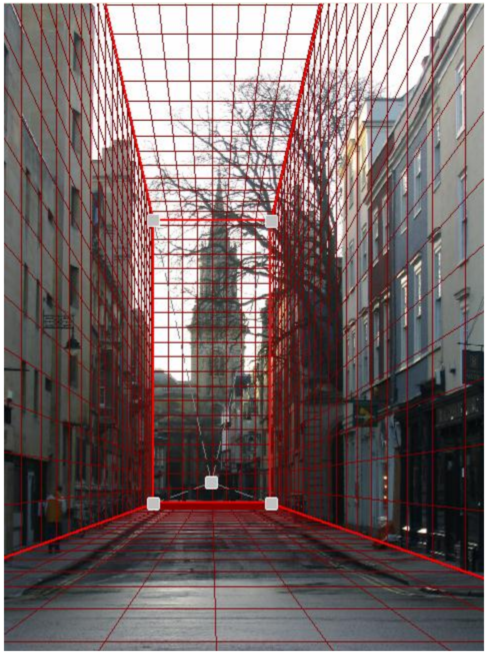
1. Set the box corners

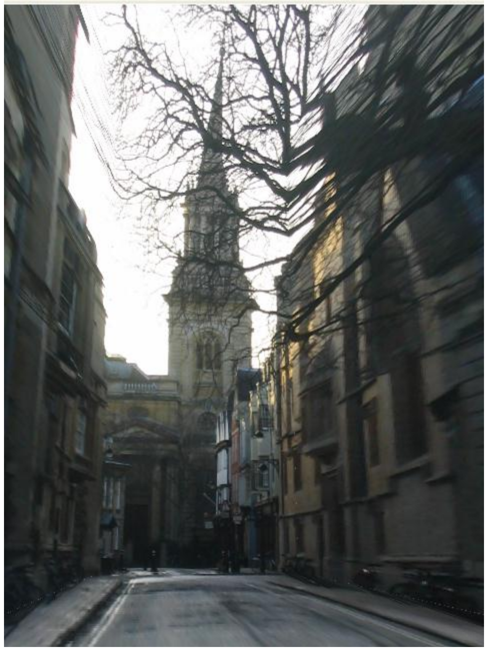
Tour into the picture algorithm

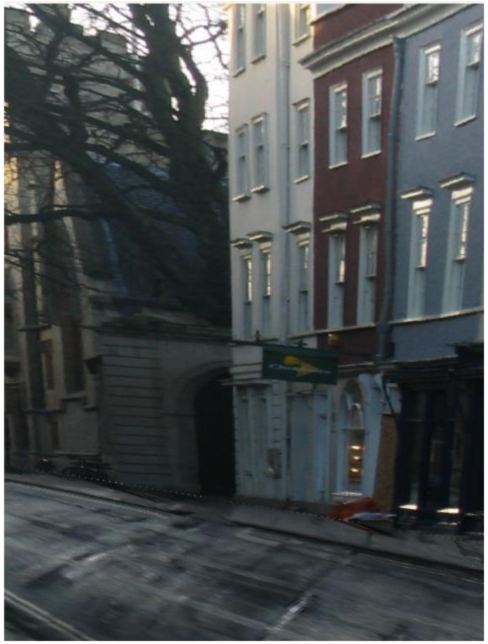
1. Set the box corners
2. Set the VP
3. Get 3D coordinates
   - Compute height, width, and depth of box
4. Get texture maps
   - homographies for each face
5. Create file to store plane coordinates and texture maps

Result

Render from new views





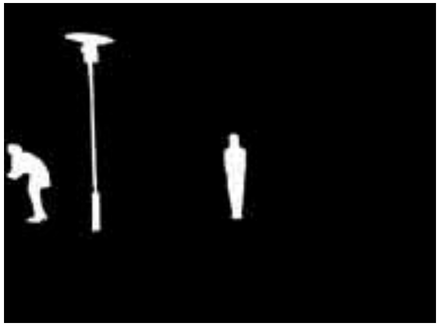


http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15463-f08/www/proj5/www/dmillett/

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 objects removed

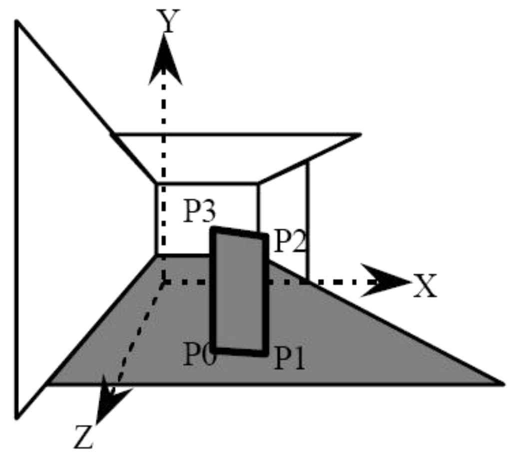
Foreground Objects

Add vertical rectangles for each foreground object

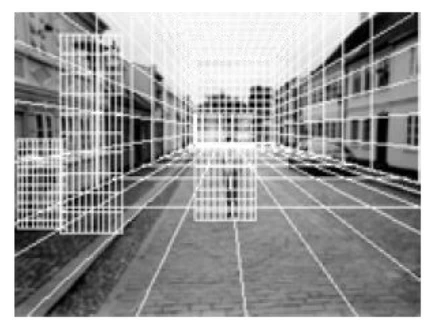
Can compute 3D coordinates P0, P1 since they are on known plane.

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

(a) Specifying of a foreground object

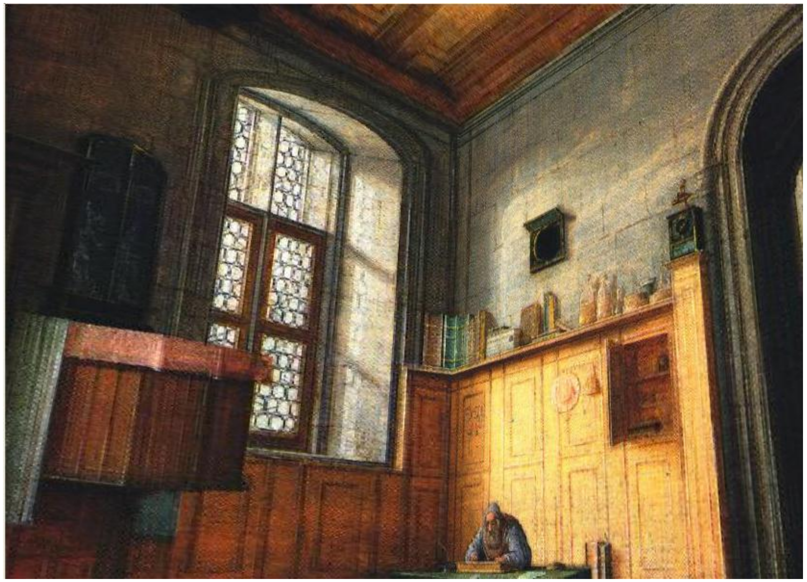


(b) Estimating the vertices of the foreground object model



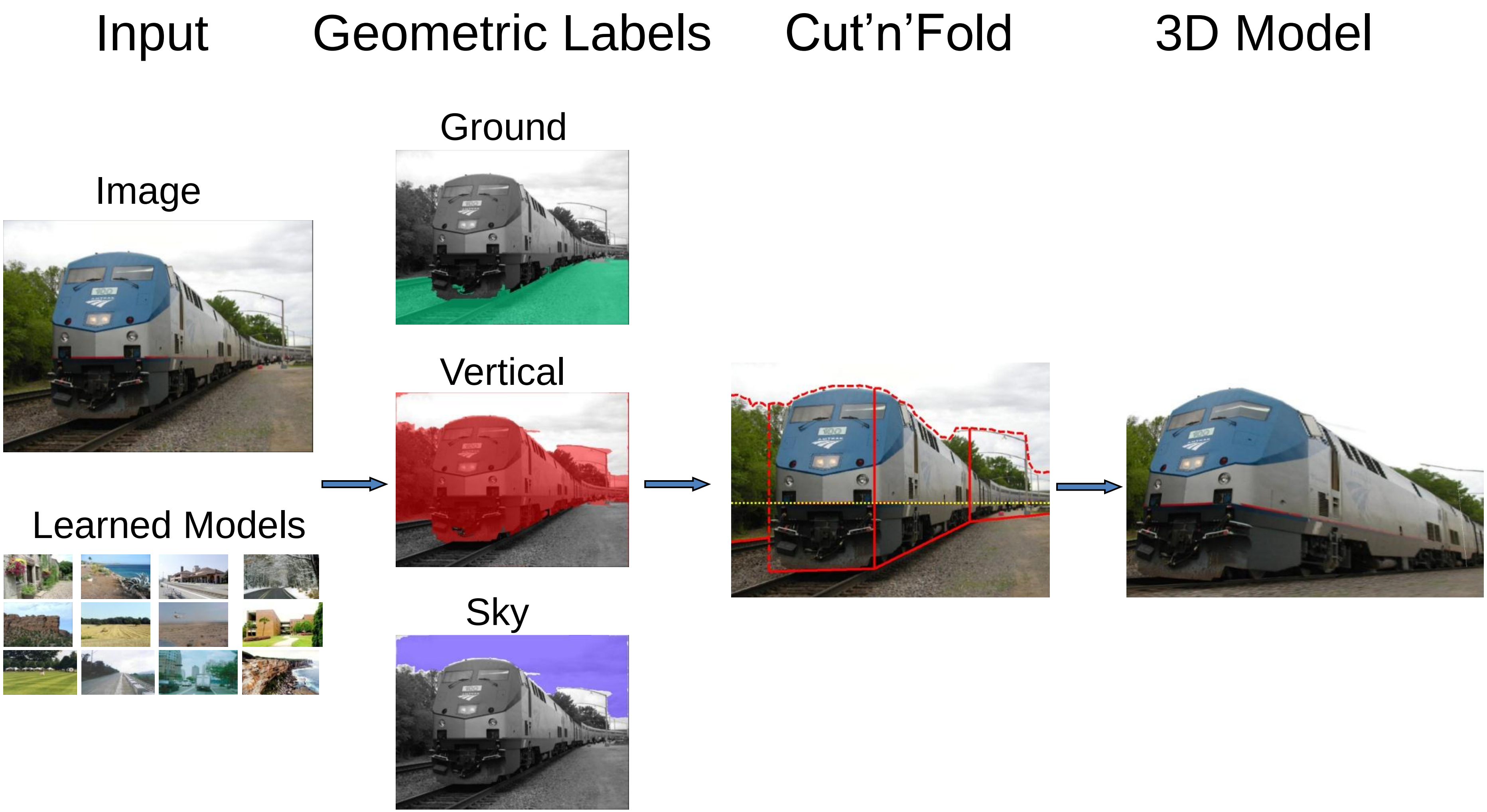
(c) Three foreground object models

Foreground Result



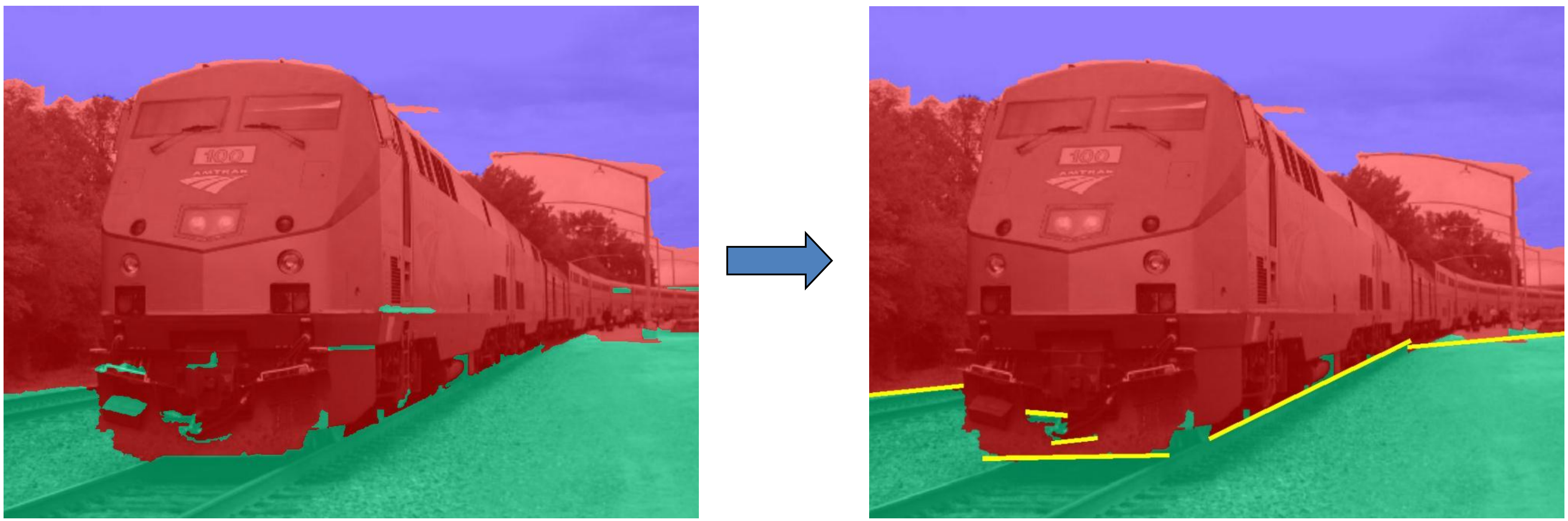
Video from CMU class:
http://www.youtube.com/watch?v=dUAtdmGwcuM

Automatic Photo Pop-up



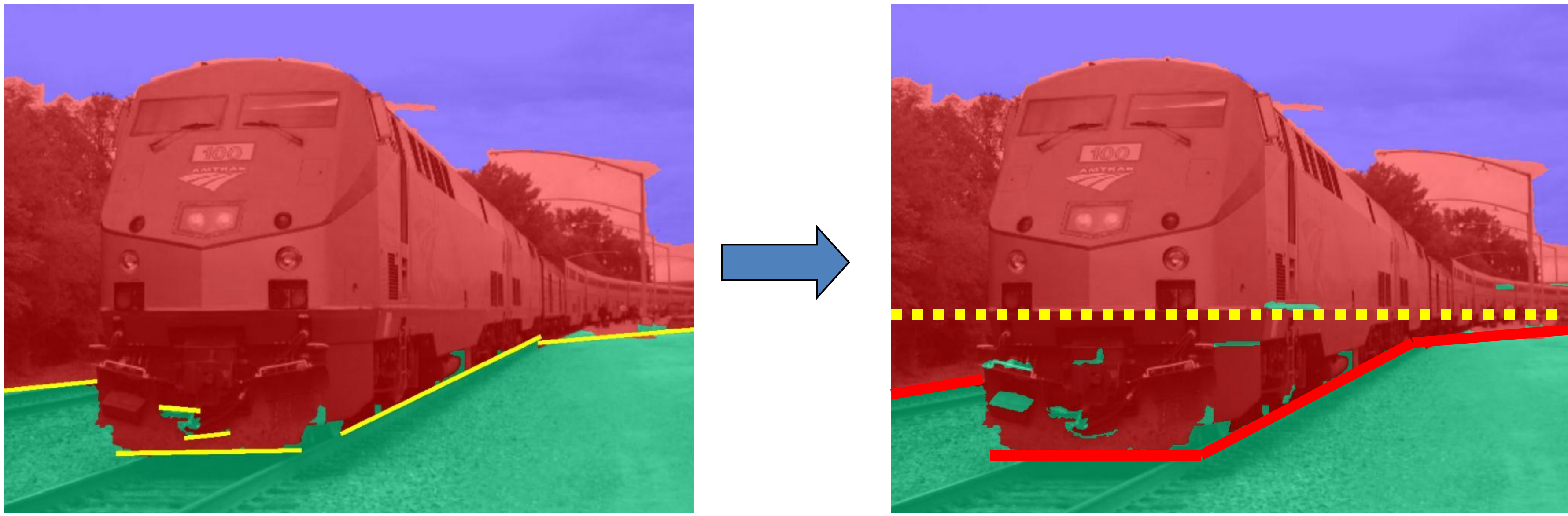
Hoiem et al. 2005

Cutting and Folding



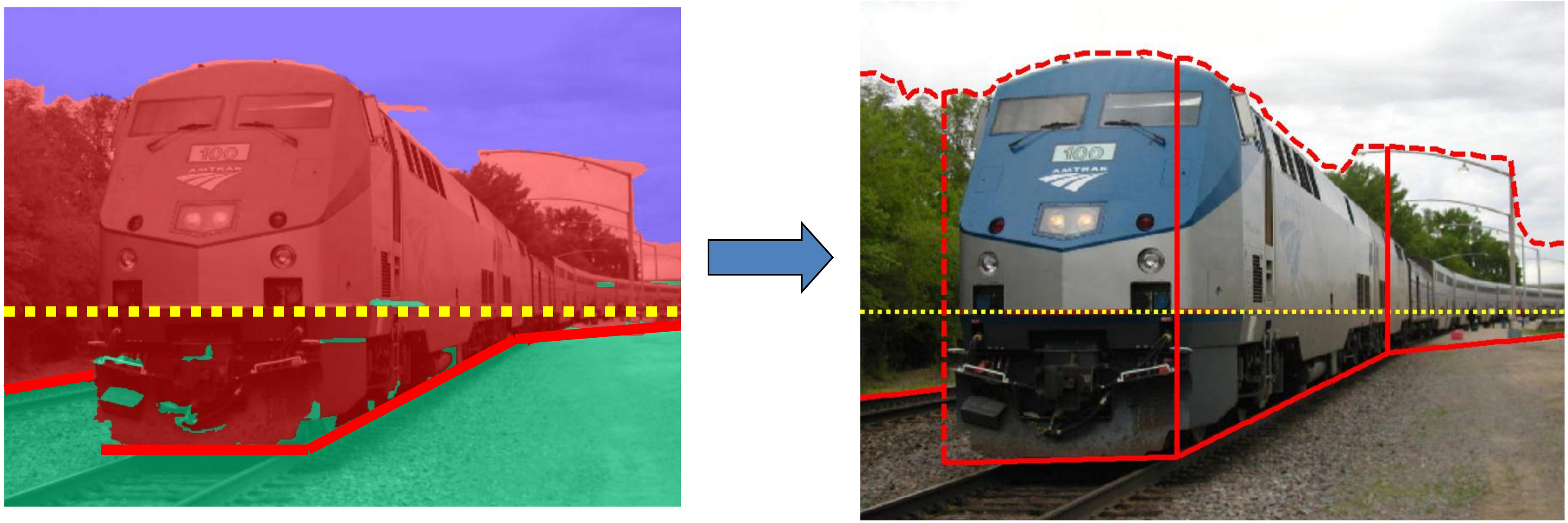
- Fit ground-vertical boundary
  - Iterative Hough transform

Cutting and Folding



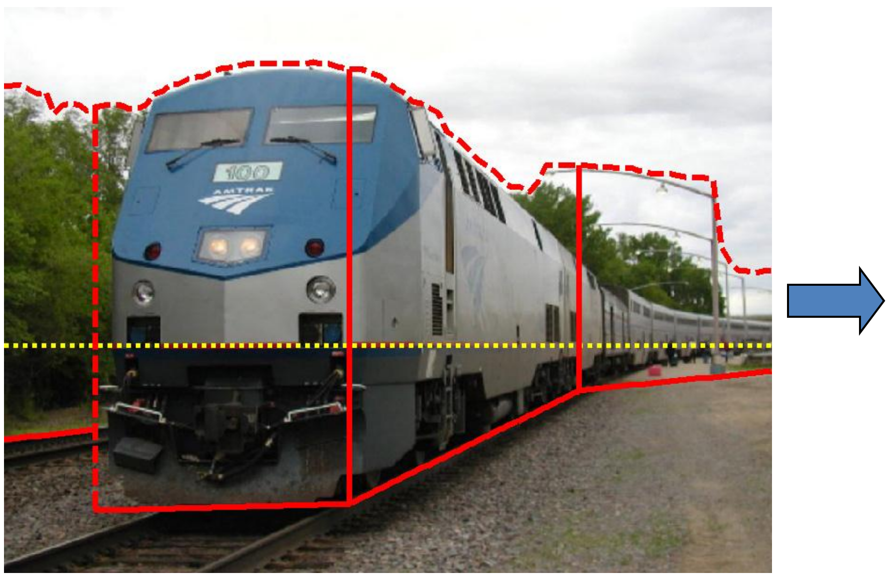
- Form polylines from boundary segments
  - Join segments that intersect at slight angles
  - Remove small overlapping polylines
- Estimate horizon position from perspective cues

Cutting and Folding



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

Cutting and Folding

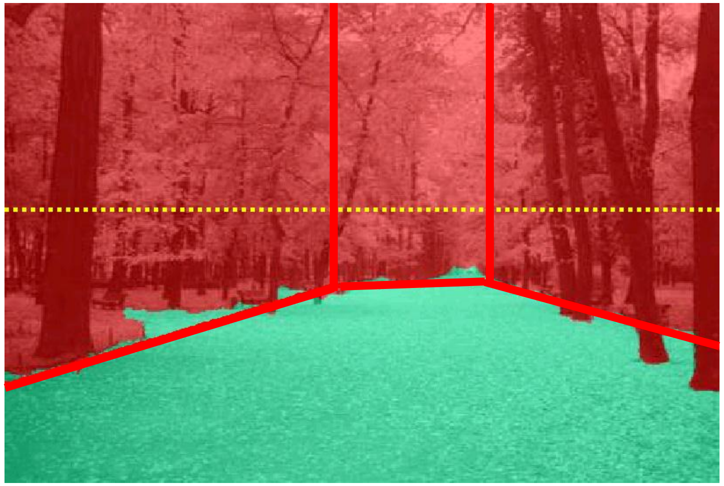


- Construct 3D model
- Texture map

Results

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

Input Image



Cut and Fold

Automatic Photo Pop-up

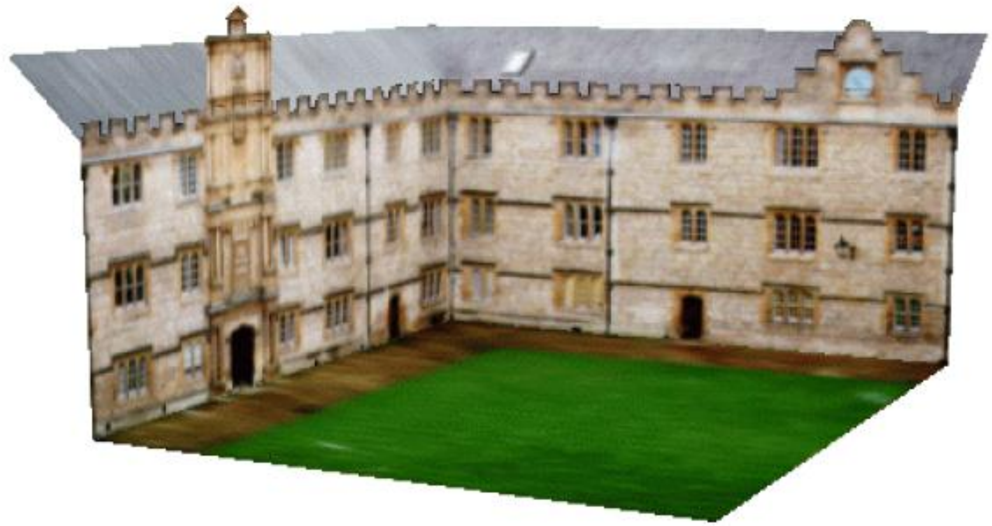
Results

Input Image

Automatic Photo Pop-up

Comparison with Manual Method

Input Image



[Liebowitz et al. 1999]

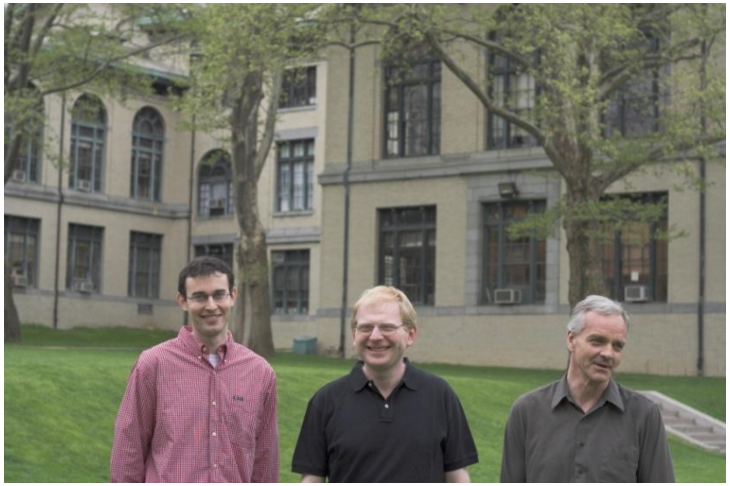
Automatic Photo Pop-up (15 sec)!

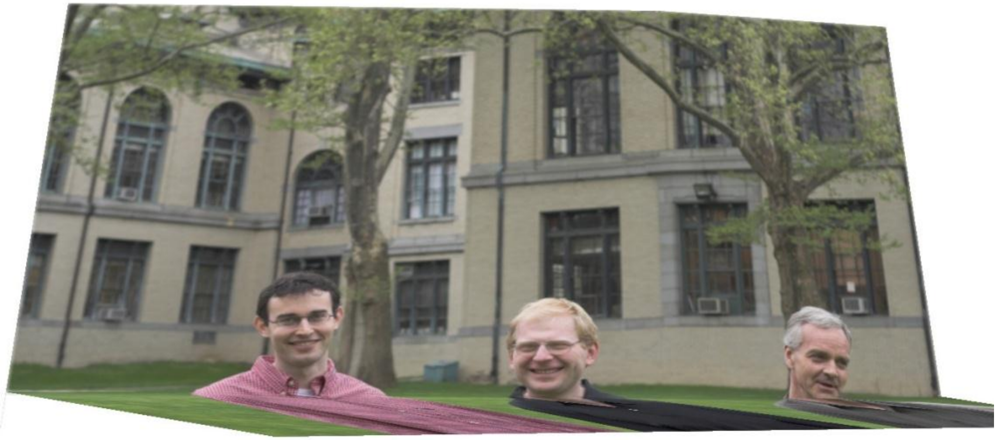
Failures

Labeling Errors

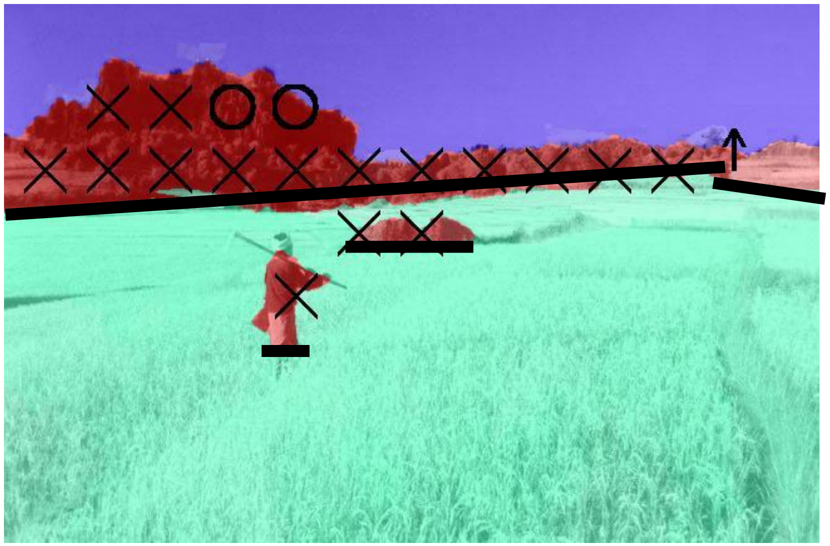
Failures

Foreground Objects

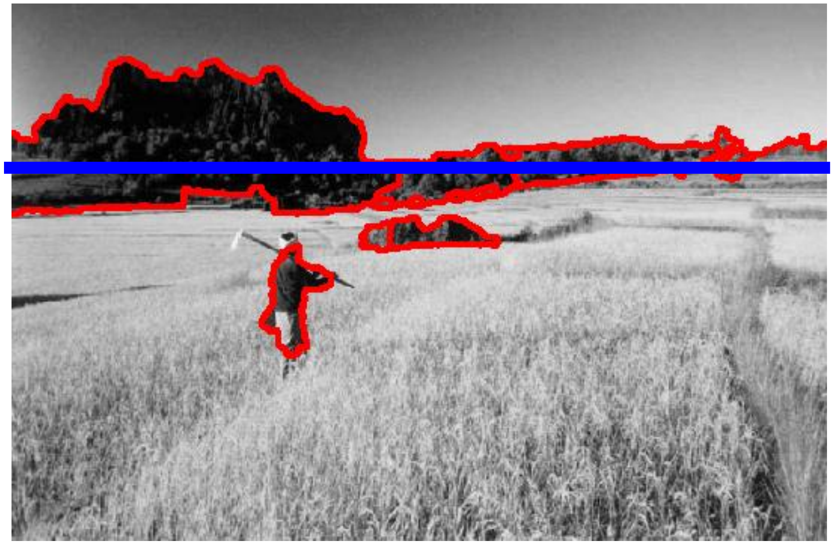




Adding Foreground Labels



Recovered Surface Labels +
Ground-Vertical Boundary Fit



Object Boundaries + Horizon

Final project ideas

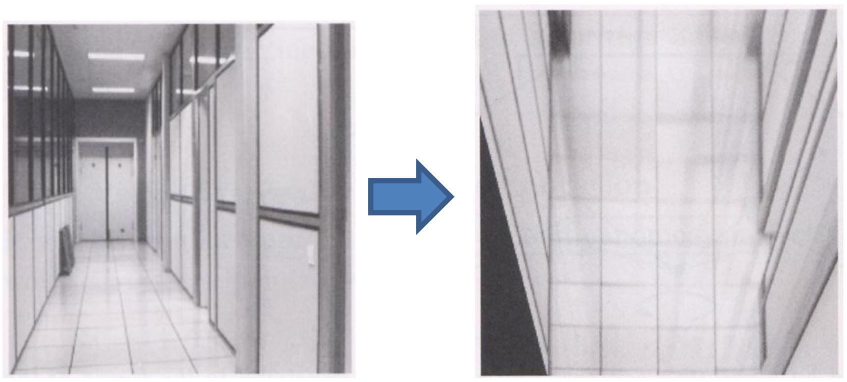
- If a one-person project:
  - Interactive program to make 3D model from an image (e.g., output in VRML, or draw path for animation)
- If a two-person team, 2nd person:
  - Add tools for cutting out foreground objects and automatic hole-filling

Summary

- 2D→3D is mathematically impossible

- Need right assumptions about the world geometry

- Important tools
  - Vanishing points
  - Camera matrix
  - Homography



Next Week

- Project 3 is due Tuesday (extension of 1 day)
- Next three classes: image-based lighting
  - How to model light
  - Recover HDR image from multiple LDR images
  - Recover lighting model from an image
  - Render object into a scene with correct lighting and geometry