

# Image-based modeling

Digital Visual Effects, Spring 2005

*Yung-Yu Chuang*

2005/5/11

*with slides by Richard Szeliski, Steve Seitz and Alexei Efros*

## Announcements

---

- Project #2 artifacts voting ends today
- Project #3 is online
- CGCG talk on 5/23



## Outline

---

- Models from multiple (sparse) images
  - Facade
- Models from single images
  - Tour into pictures
  - Single view metrology
  - Other approaches

## Models from multiple images (Façade, Debevec *et. al.* 1996)

### Facade

---

- Use a sparse set of images
  - Calibrated camera (intrinsic only)
  - Designed specifically for modeling architecture
  - Use a set of blocks to approximate architecture
- 
- 3 steps: geometry reconstruction, texture mapping and model refinement

### Idea

---

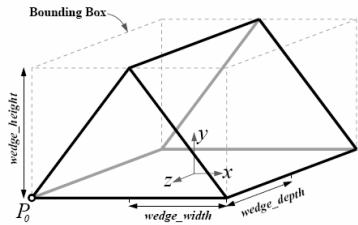


### Idea

---

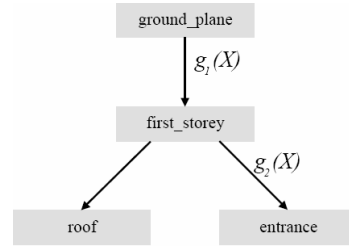
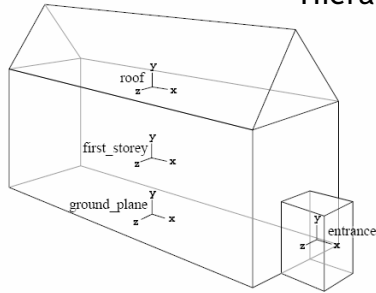


# Geometric modeling



A block is a geometric primitive with a small set of parameters

Hierarchical modeling for a scene

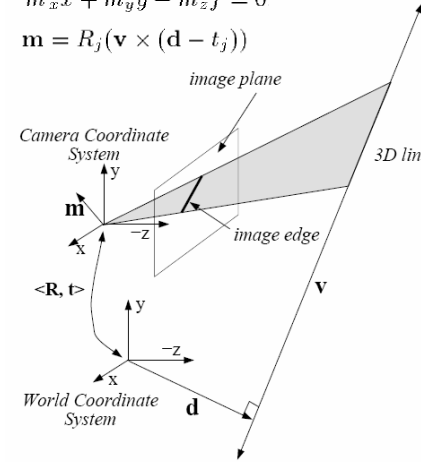


# Reconstruction

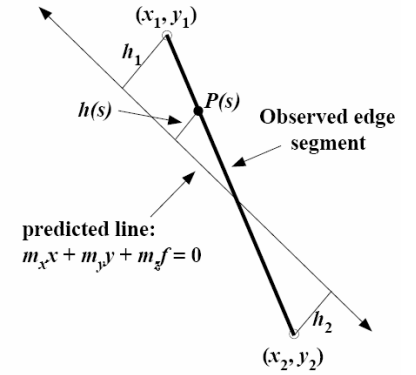
$$\text{minimize } \mathcal{O} = \sum \text{Err}_i$$

$$m_x x + m_y y - m_z f = 0$$

$$\mathbf{m} = R_j(\mathbf{v} \times (\mathbf{d} - t_j))$$



$$\text{Err}_i = \int_0^l h^2(s) ds$$

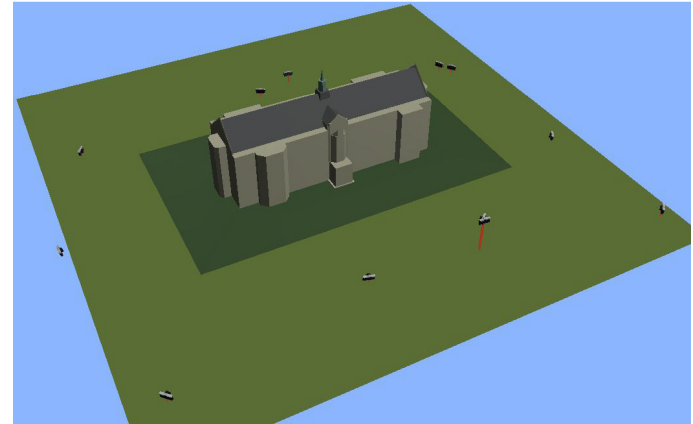
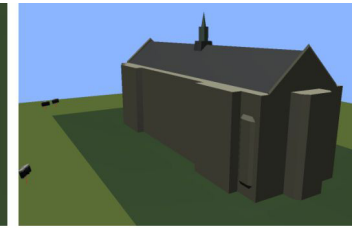
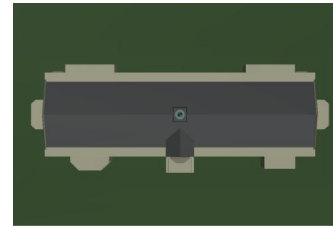
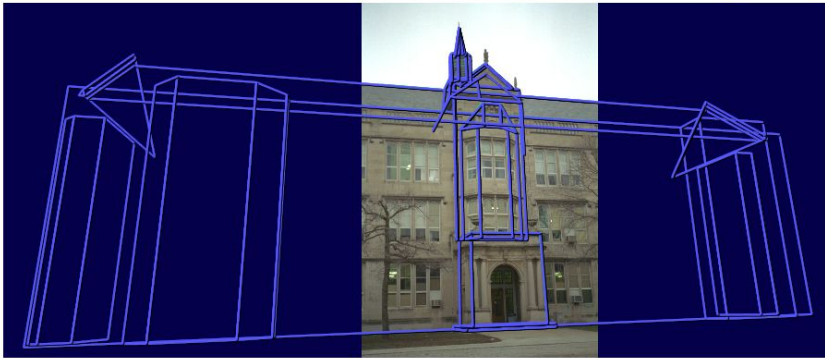
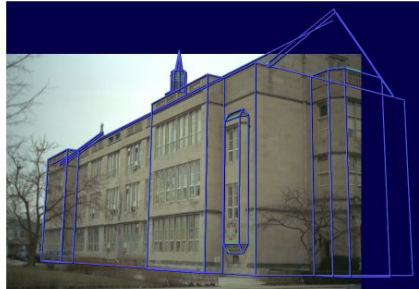
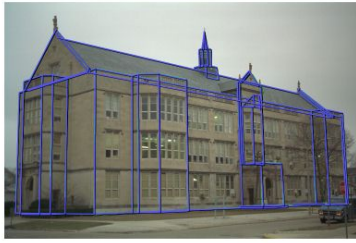


# Advantages

- Suitable for modeling architecture
- High level abstraction
- Reduce number of parameters (34 vs 240 vs 2896)
- Easy to add constraints in architecture

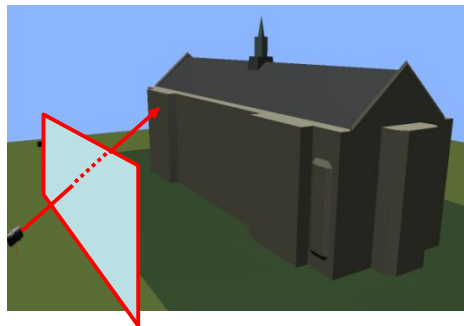
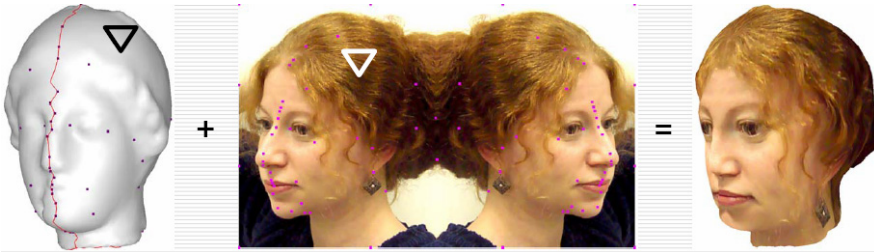
# Results





## Texture mapping

DigiVFX



## Texture mapping in real world

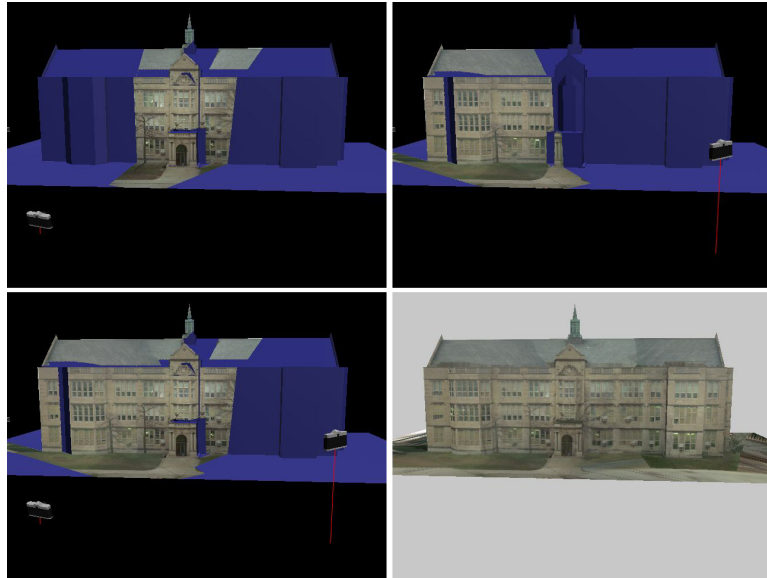
DigiVFX



[Demo movie](#)  
Michael Naimark,  
San Francisco Museum  
of Modern Art, 1984

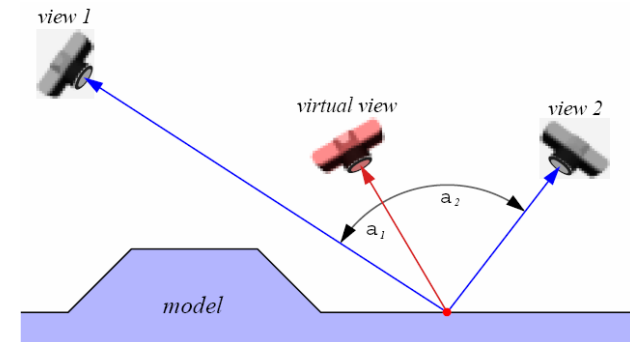
## View-dependent texture mapping

DigiVFX



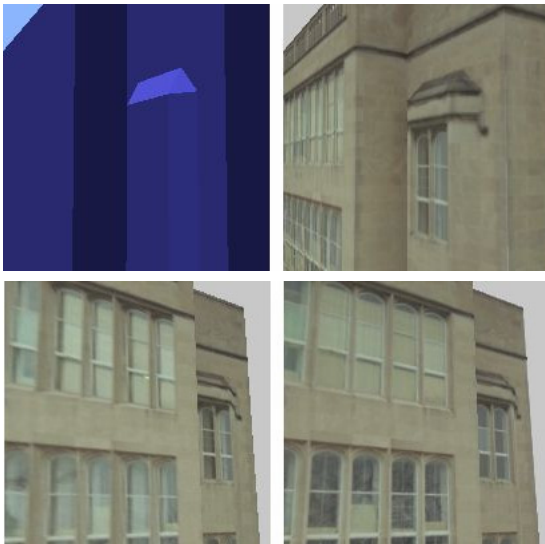
## View-dependent texture mapping

DigiVFX



## View-dependent texture mapping

DigiVFX



## View-dependent texture mapping

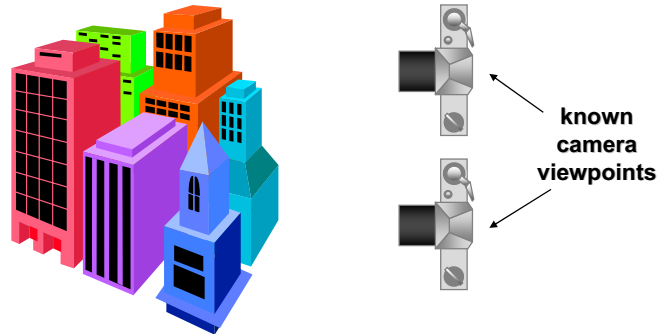
DigiVFX



## Model-based stereo

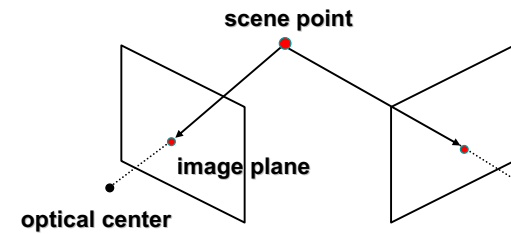
DigiVFX

- Use stereo to refine the geometry



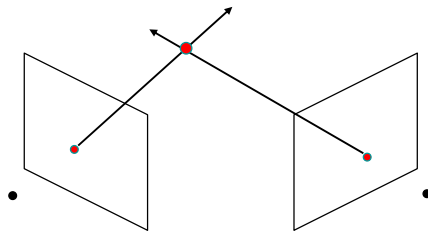
## Stereo

DigiVFX



## Stereo

DigiVFX

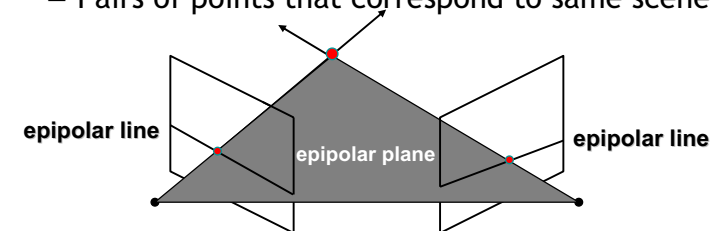


- Basic Principle: Triangulation
  - Gives reconstruction as intersection of two rays
  - Requires
    - calibration
    - *point correspondence*

## Stereo correspondence

DigiVFX

- Determine Pixel Correspondence
  - Pairs of points that correspond to same scene point



- Epipolar Constraint
  - Reduces correspondence problem to 1D search along *conjugate epipolar lines*

## Finding correspondences

DigiVFX

- apply feature matching criterion (e.g., correlation or Lucas-Kanade) at *all* pixels simultaneously
- search only over epipolar lines (many fewer candidate positions)



## Image registration (revisited)

DigiVFX

- How do we determine correspondences?
  - *block matching* or *SSD* (sum squared differences)

$$E(x, y; d) = \sum_{(x', y') \in N(x, y)} [I_L(x' + d, y') - I_R(x', y')]^2$$

$d$  is the *disparity* (horizontal motion)

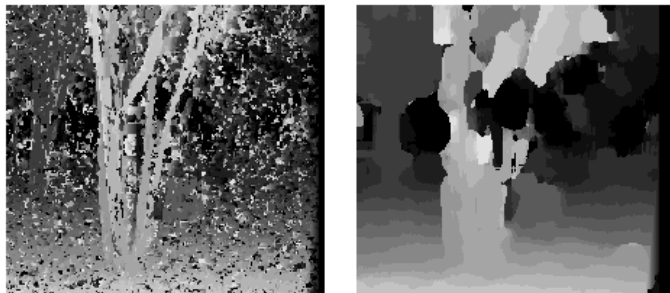


- How big should the neighborhood be?

## Neighborhood size

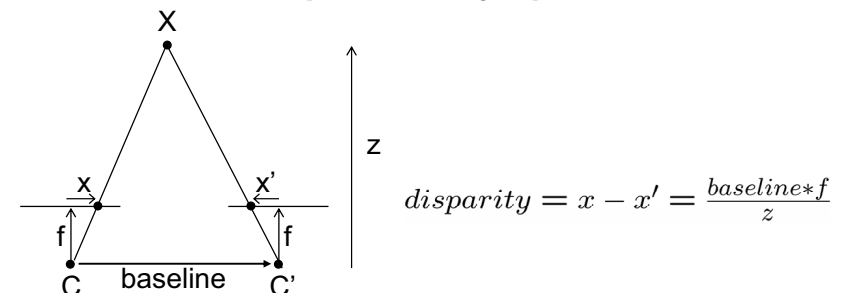
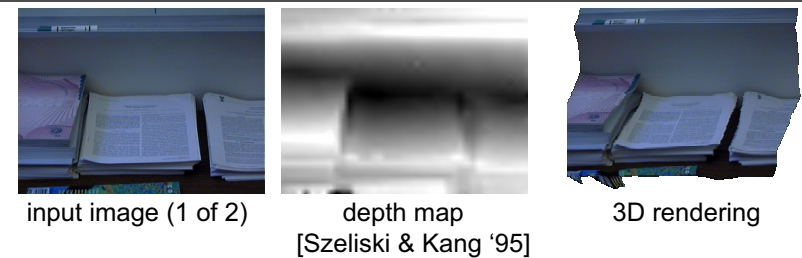
DigiVFX

- Smaller neighborhood: more details
- Larger neighborhood: fewer isolated mistakes



## Depth from disparity

DigiVFX



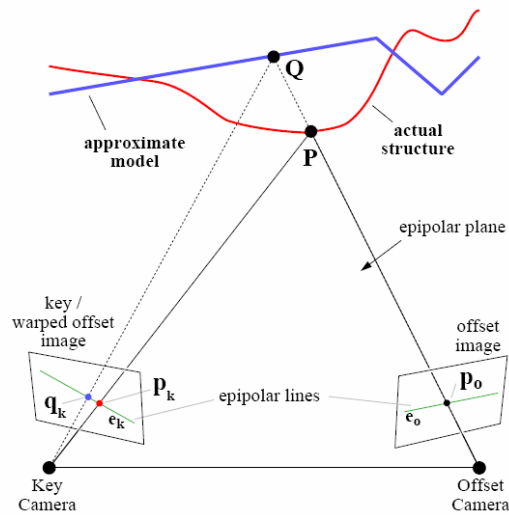
# Stereo reconstruction pipeline

- Steps
  - Calibrate cameras
  - Rectify images
  - Compute disparity
  - Estimate depth
- What will cause errors?
  - Camera calibration errors
  - Poor image resolution
  - Occlusions
  - Violations of brightness constancy (specular reflections)
  - Large motions
  - Low-contrast image regions

# Model-based stereo



# Epipolar geometry



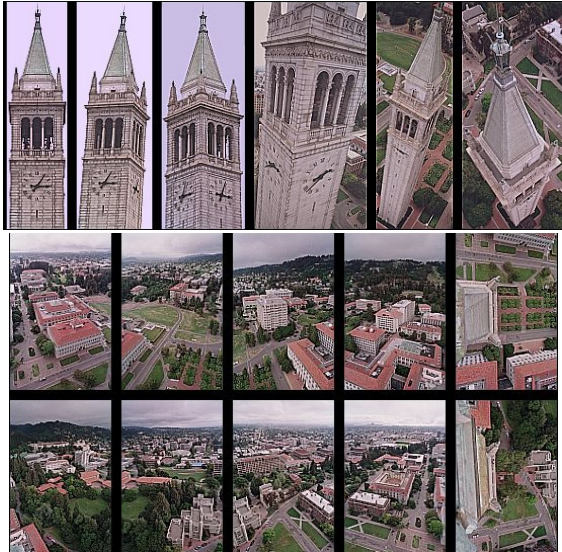
# Comparisons





## Final results

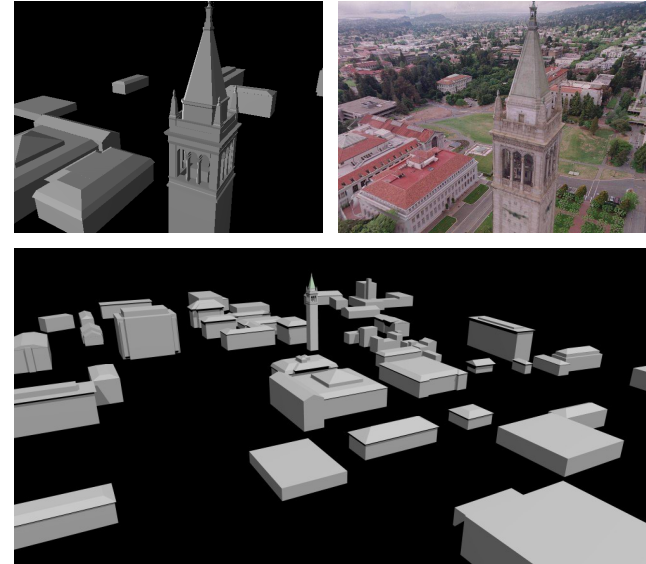
DigiVFX



[Kite photography](#)

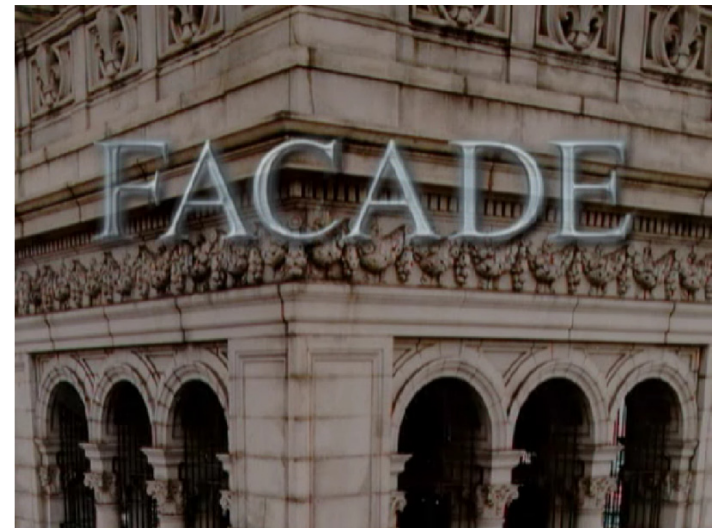
## Final results

DigiVFX



## Results

DigiVFX



## Results

DigiVFX



## Commercial packages

DigiVFX

- [REALVIZ ImageModeler](#)



## The Matrix

DigiVFX

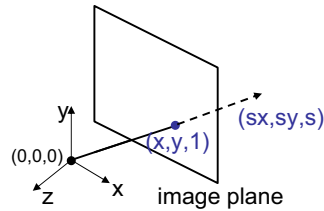
Cinefex #79, October 1999.

Since the bullet-time rig would be visible in shots featuring a 360-degree sweep of the characters, it was employed only for the shooting of the foreground subject – namely, the actors or their stunt doubles – necessitating a different approach for the backgrounds. Shot separately, the backgrounds used a virtual cinematography process that allowed a 360-degree environment to be constructed in the computer from stills taken on set. This approach for generating the backgrounds was based on the Berkeley Tower flyover, a novel image-based rendering technique presented at Siggraph '97 by George Borshukov and Paul Debevec, a researcher at UC Berkeley. The technique employed twenty stills of that town's college campus to create a virtual environment through which the camera could travel. "Instead of reinventing the background in traditional CG fashion – painting textures, shooting orthographic views of the set, and then proceeding to texture replication – we generated a completely free, high-resolution camera move that would have been impossible to achieve using traditional CG," Borshukov said, "and we did it working from just a handful of stills."

Models from single images

## The projective plane

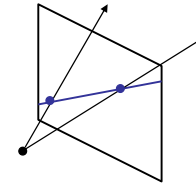
- Geometric intuition?
  - a point in the image is a *ray* in projective space



- Each *point*  $(x,y)$  on the plane is represented by a *ray*  $(sx, sy, s)$ 
  - all points on the ray are equivalent:  $(x, y, 1) \cong (sx, sy, s)$

## Projective lines

- What does a line in the image correspond to in projective space?



- A line is a *plane* of rays through origin
  - all rays  $(x,y,z)$  satisfying:  $ax + by + cz = 0$

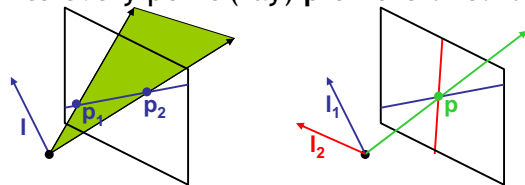
in vector notation:  $0 = [a \ b \ c] \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

**l p**

- A line is also represented as a homogeneous 3-vector **l**

## Point and line duality

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



What is the line **l** spanned by rays **p**<sub>1</sub> and **p**<sub>2</sub>?

- **l** is  $\perp$  to **p**<sub>1</sub> and **p**<sub>2</sub>  $\Rightarrow l = p_1 \times p_2$
- **l** is the plane normal

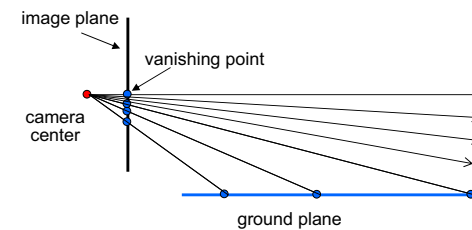
What is the intersection of two lines **l**<sub>1</sub> and **l**<sub>2</sub>?

- **p** is  $\perp$  to **l**<sub>1</sub> and **l**<sub>2</sub>  $\Rightarrow p = l_1 \times l_2$

Points and lines are *dual* in projective space

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

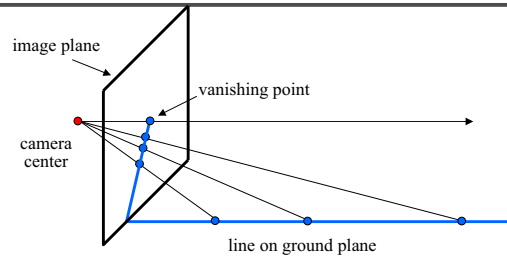
## Vanishing points



- Vanishing point
  - projection of a point at infinity

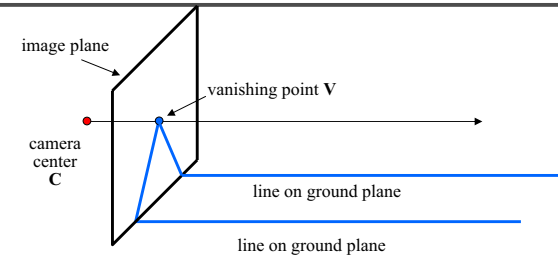
## Vanishing points (2D)

DigiVFX



## Vanishing points

DigiVFX

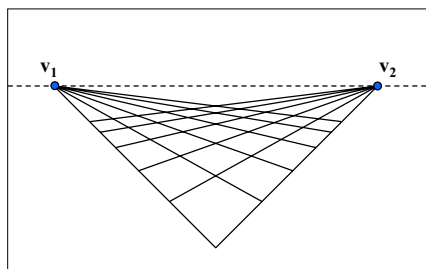


### • Properties

- Any two parallel lines have the same vanishing point  $v$
- The ray from  $C$  through  $v$  is parallel to the lines
- An image may have more than one vanishing point

## Vanishing lines

DigiVFX

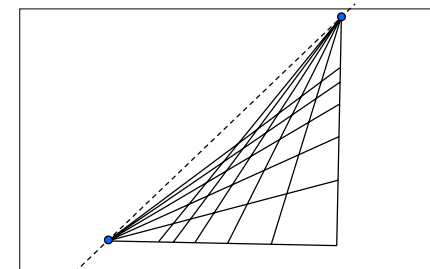


### • Multiple Vanishing Points

- 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

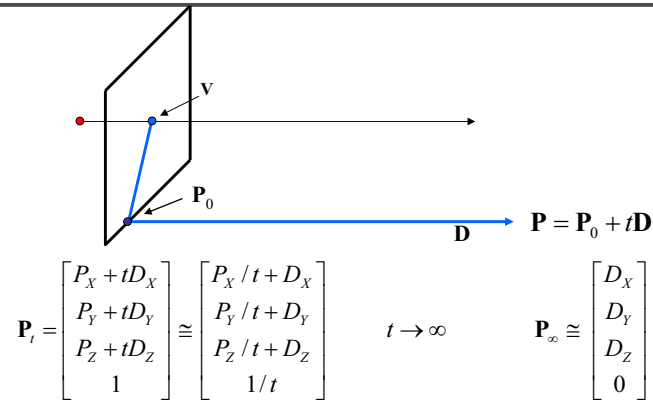
DigiVFX



### • Multiple Vanishing Points

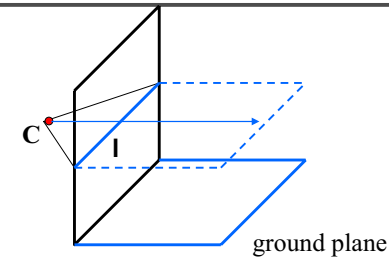
- Any set of parallel lines on the plane define a vanishing point
- The union of all of these vanishing points is the *horizon line*
  - also called *vanishing line*
- Note that different planes define different vanishing lines

## Computing vanishing points



- Properties  $v = \Pi P_\infty$ 
  - $P_\infty$  is a point at *infinity*,  $v$  is its projection
  - They depend only on line *direction*
  - Parallel lines  $P_0 + tD$ ,  $P_1 + tD$  intersect at  $P_\infty$

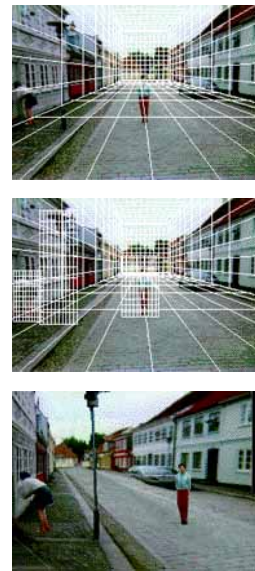
## Computing vanishing lines



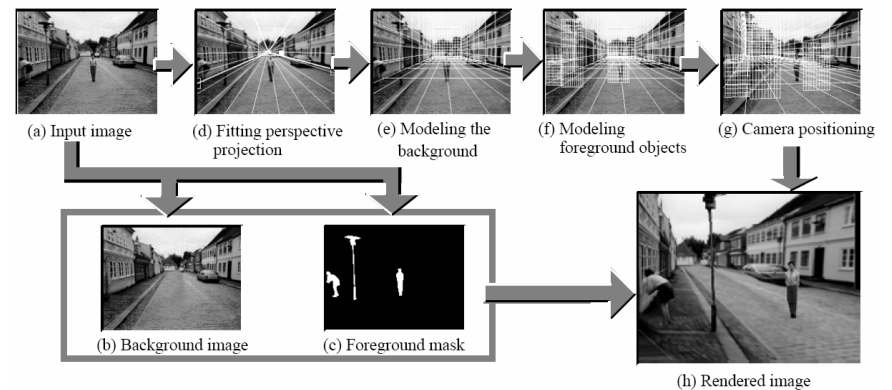
- Properties
  - $l$  is intersection of horizontal plane through  $C$  with image plane
  - Compute  $l$  from two sets of parallel lines on ground plane
    - points higher than  $C$  project above  $l$
  - Provides way of comparing height of objects in the scene

## Tour into pictures

- Create a 3D “theatre stage” of five billboards
- Specify foreground objects through bounding polygons
- Use camera transformations to navigate through the scene



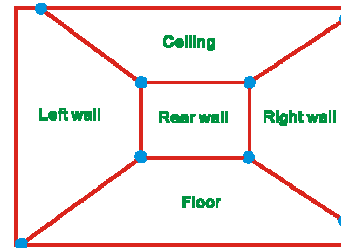
## Tour into pictures



## The idea

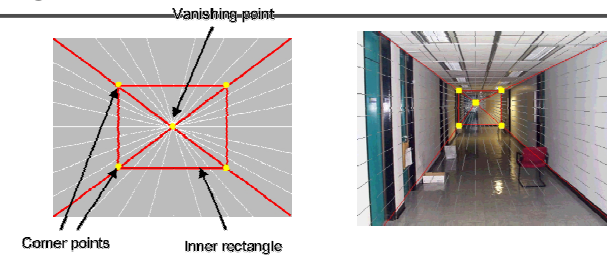
DigiVFX

- 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
  - Volume bottom is  $y=0$
- Can use the vanishing point to fit the box to the particular Scene!



## Fitting the box volume

DigiVFX

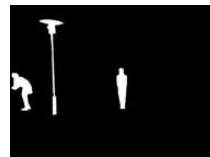


- User controls the inner box and the vanishing point placement (6 DOF)

## Foreground Objects

DigiVFX

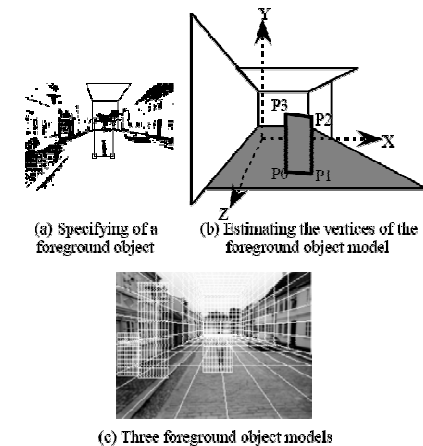
- Use separate billboard for each
- For this to work, three separate images used:
  - Original image.
  - Mask to isolate desired foreground images.
  - Background with objects removed



## Foreground Objects

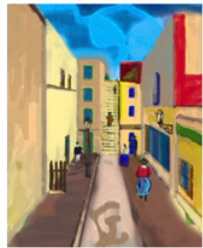
DigiVFX

- Add vertical rectangles for each foreground object
- Can compute 3D coordinates  $P_0$ ,  $P_1$  since they are on known plane.
- $P_2$ ,  $P_3$  can be computed as before (similar triangles)



## Example

DigiVFX



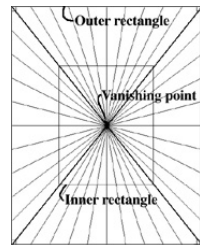
(a) Input image



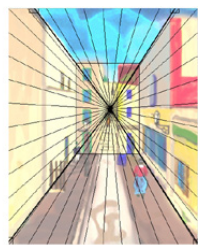
(b) Background



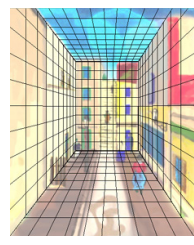
(c) Foreground mask



(a) Initial state

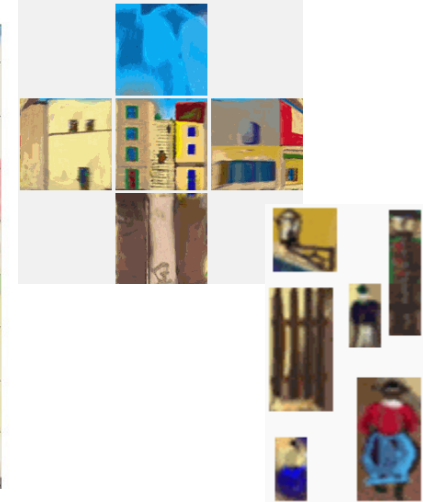
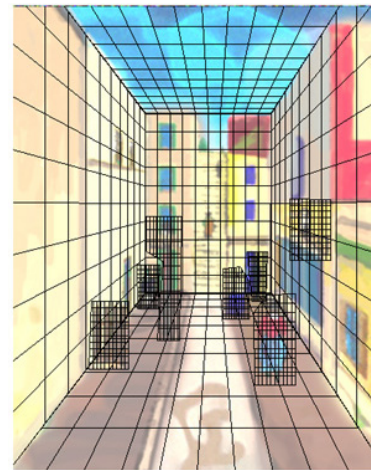


(b) Specification result



## Example

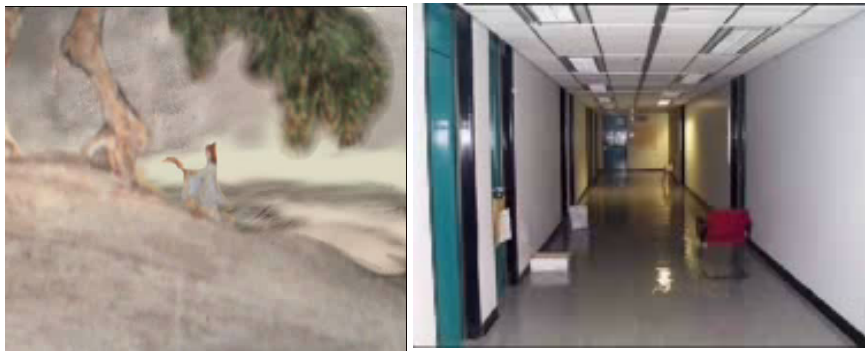
DigiVFX



## glTip

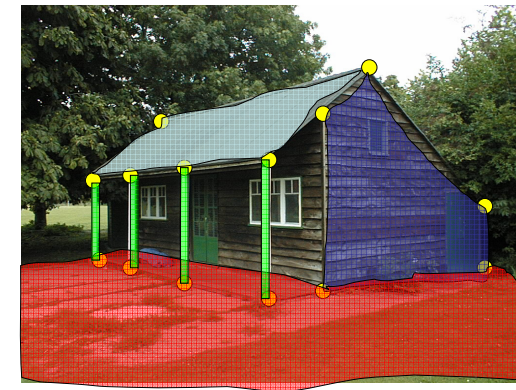
DigiVFX

- <http://www.cs.ust.hk/~cpegnel/glTIP/>



## Criminisi et al. ICCV 1999

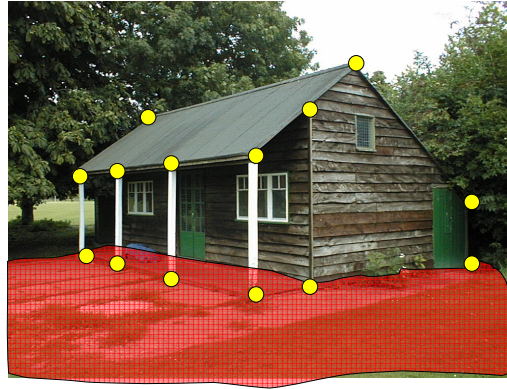
DigiVFX



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)

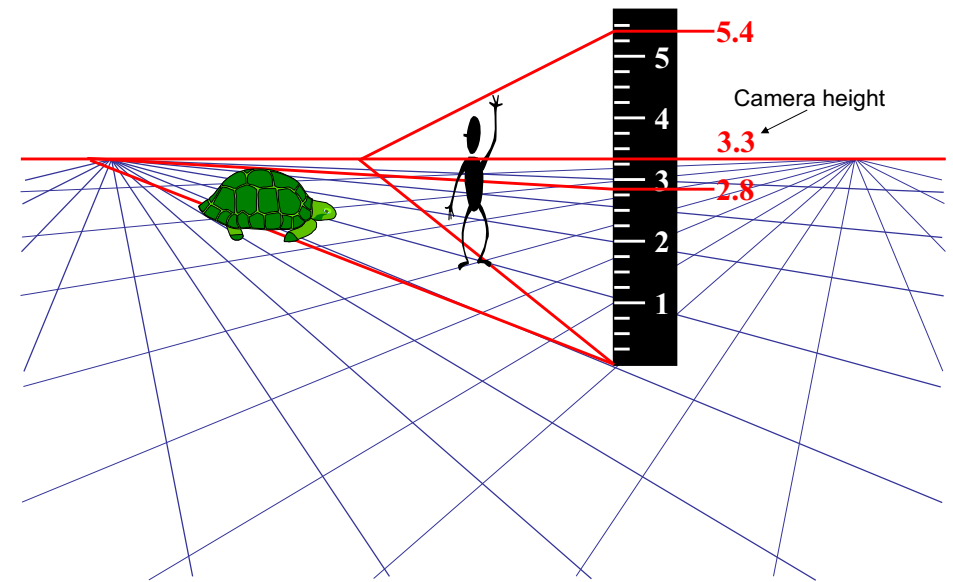
DigiVFX



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

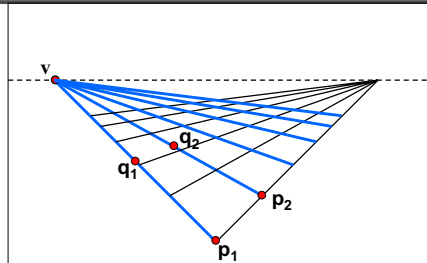
## Measuring height

DigiVFX



## Computing vanishing points

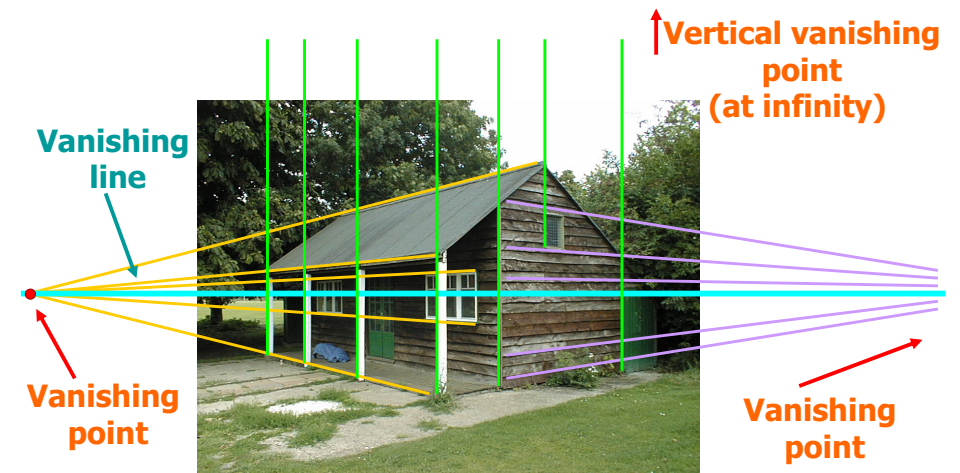
DigiVFX



- Intersect  $p_1q_1$  with  $p_2q_2$
- Least squares version
  - Better to use more than two lines and compute the “closest” point of intersection
  - See notes by [Bob Collins](http://www-2.cs.cmu.edu/~ph/869/www/notes/vanishing.txt) for one good way of doing this:
    - <http://www-2.cs.cmu.edu/~ph/869/www/notes/vanishing.txt>

## Criminisi '99

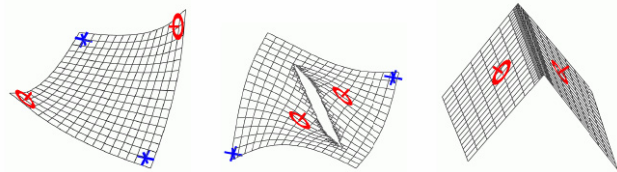
DigiVFX





# Zhang et. al. CVPR 2001

DigiVFX



Methods	Iteration 0	Iteration 200	Iteration 1200	Iteration 2500	Iteration 9500
No hierarchical transformation					

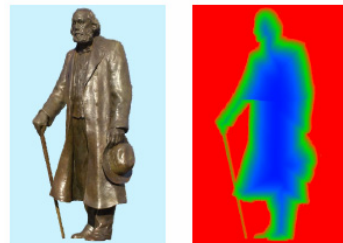
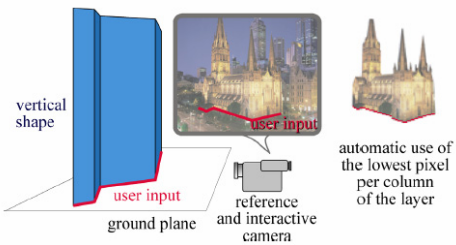
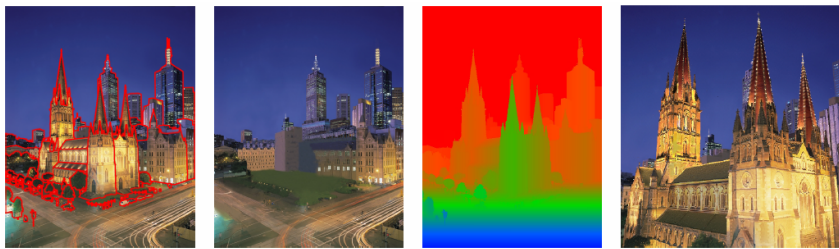
# Zhang et. al. CVPR 2001

DigiVFX

original image	constraints	3D wireframe	novel view

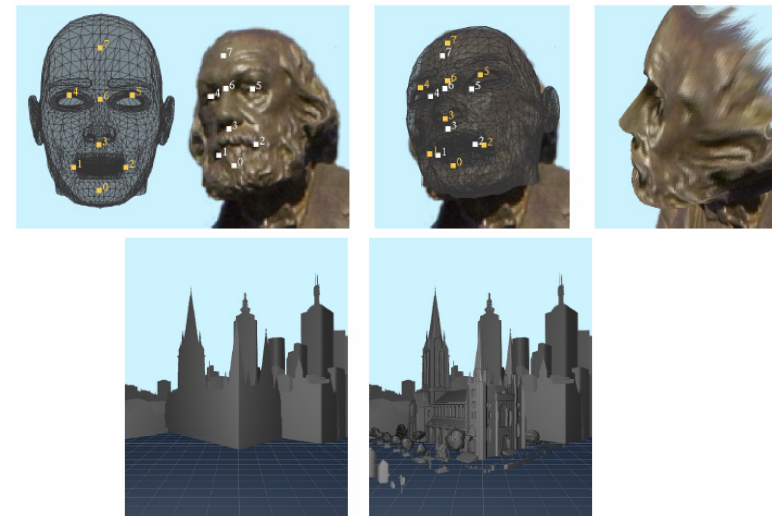
# Oh et. al. SIGGRAPH 2001

DigiVFX



# Oh et. al. SIGGRAPH 2001

DigiVFX



## Automatic popup (SIGGRAPH 2005)



Feature Descriptions	Num	Used
<b>Color</b>	<b>15</b>	<b>15</b>
C1. RGB values: mean	3	3
C2. HSV values: conversion from mean RGB values	3	3
C3. Hue: histogram (5 bins) and entropy	6	6
C4. Saturation: histogram (3 bins) and entropy	3	3
<b>Texture</b>	<b>29</b>	<b>13</b>
T1. DOOG Filters: mean abs response	12	3
T2. DOOG Filters: mean of variables in T1	1	0
T3. DOOG Filters: id of max of variables in T1	1	1
T4. DOOG Filters: (max - median) of variables in T1	1	1
T5. Textons: mean abs response	12	7
T6. Textons: max of variables in T5	1	0
T7. Textons: (max - median) of variables in T5	1	1
<b>Location and Shape</b>	<b>12</b>	<b>10</b>
L1. Location: normalized x and y, mean	2	2
L2. Location: norm. x and y, 10 <sup>th</sup> and 90 <sup>th</sup> percentile	4	4
L3. Location: norm. y wrt horizon, 10 <sup>th</sup> and 90 <sup>th</sup> pctl	2	2
L4. Shape: number of superpixels in constellation	1	1
L5. Shape: number of sides of convex hull	1	0
L6. Shape: <i>num pixels/area(convex hull)</i>	1	1
L7. Shape: whether the constellation region is contiguous	1	0
<b>3D Geometry</b>	<b>35</b>	<b>28</b>
G1. Long Lines: total number in constellation region	1	1
G2. Long Lines: % of nearly parallel pairs of lines	1	1
G3. Line Intersection: hist. over 12 orientations, entropy	13	11
G4. Line Intersection: % right of center	1	1
G5. Line Intersection: % above center	1	1
G6. Line Intersection: % far from center at 8 orientations	8	4
G7. Line Intersection: % very far from center at 8 orientations	8	5
G8. Texture gradient: x and y "edginess" (T2) center	2	2

## Automatic popup



## Reference

- P. Debevec, C. Taylor and J. Malik. [Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach](#), SIGGRAPH 1996.
- Y. Horry, K. Anjyo and K. Arai. [Tour Into the Picture: Using a Spidery Mesh Interface to Make Animation from a Single Image](#), SIGGRAPH 1997.
- A. Criminisi, I. Reid and A. Zisserman. [Single View Metrology](#), ICCV 1999.
- L. Zhang, G. Dugas-Phocion, J.-S. Samson and S. Seitz. [Single View Modeling of Free-Form Scenes](#), CVPR 2001.
- B. Oh, M. Chen, J. Dorsey and F. Durand. [Image-Based Modeling and Photo Editing](#), SIGGRAPH 2001.
- D. Hoiem, A. Efros and M. Hebert. [Automatic Photo Pop-up](#), SIGGRAPH 2005.