Image-based modeling

Digital Visual Effects

Yung-Yu Chuang

with slides by Richard Szeliski, Steve Seitz and Alexei Efros

Outline

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

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

Facade

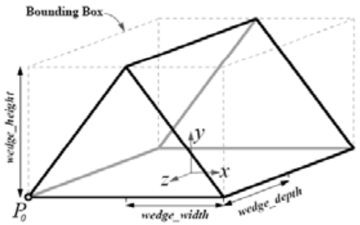
- Use a sparse set of images
- Calibrated camera (intrinsic only)
- Designed specifically for modeling architecture
- Use a set of blocks to approximate architecture

- Three components:
  - geometry reconstruction
  - texture mapping
  - model refinement

Idea

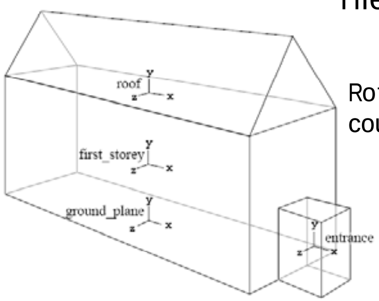
Idea

Geometric modeling

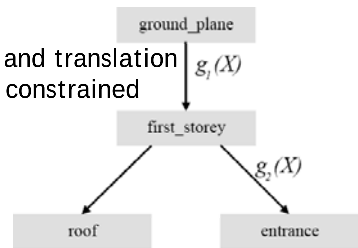


A block is a geometric primitive with a **small** set of parameters

Hierarchical modeling for a scene



Rotation and translation could be constrained



Reasons for block modeling

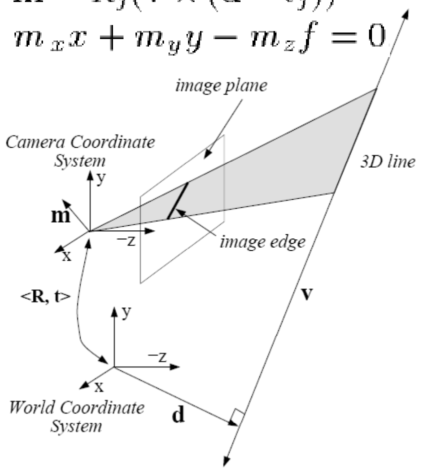
- Architectural scenes are well modeled by geometric primitives.
- Blocks provide a high level abstraction, easier to manage and add constraints.
- No need to infer surfaces from discrete features; blocks essentially provide prior models for architectures.
- Hierarchical block modeling effectively reduces the number of parameters for robustness and efficiency.

Reconstruction

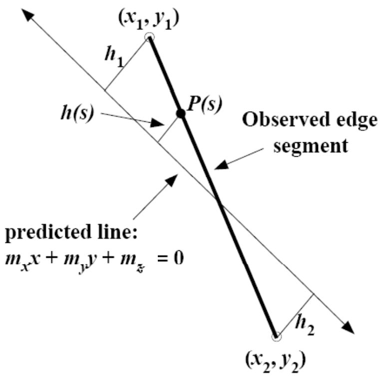
minimize $\mathcal{O} = \sum Err_i$

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

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

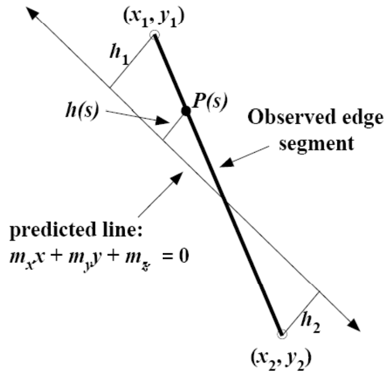


$$Err_i = \int_0^l h^2(s)ds$$



Reconstruction

$$Err_i = \int_0^l h^2(s)ds$$



$$h_1 = \frac{m_x x_1 + m_y y_1 + m_z}{\sqrt{m_x^2 + m_y^2}}$$

$$h_2 = \frac{m_x x_2 + m_y y_2 + m_z}{\sqrt{m_x^2 + m_y^2}}$$

$$h(s) = h_1 + s\frac{h_2 - h_1}{l}$$

$$Err_i = \int_0^l h^2(s)ds = \frac{l}{3}(h_1^2 + h_1 h_2 + h_2^2)$$

Reconstruction

$$Err_i = \int_0^l h^2(s)ds = \frac{l}{3}(h_1^2 + h_1 h_2 + h_2^2) = \mathbf{m}^T (A^T B A) \mathbf{m}$$

$$h_1 = \frac{m_x x_1 + m_y y_1 + m_z}{\sqrt{m_x^2 + m_y^2}}$$

$$h_2 = \frac{m_x x_2 + m_y y_2 + m_z}{\sqrt{m_x^2 + m_y^2}}$$

$$\mathbf{m} = (m_x, m_y, m_z)^T \qquad \mathbf{m} = R_j(\mathbf{v} \times (\mathbf{d} - t_j))$$

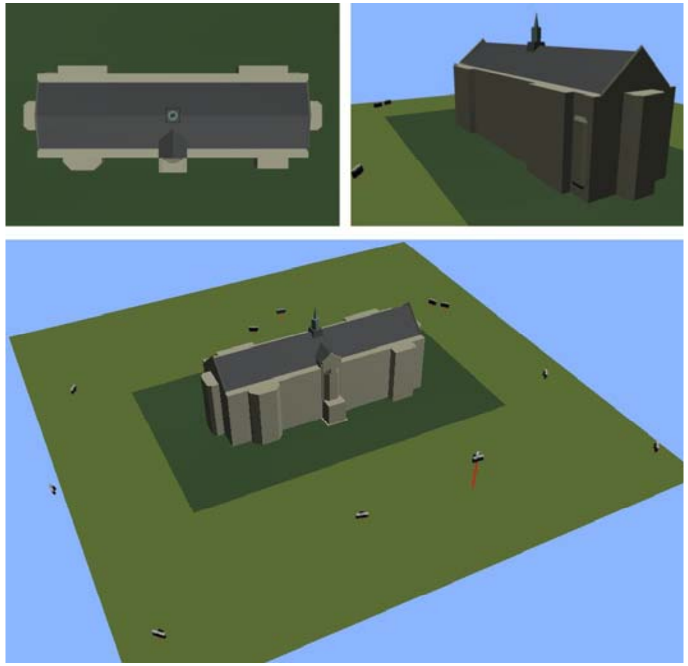
$$A = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix}$$

$$B = \frac{l}{3(m_x^2 + m_y^2)} \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$

nonlinear w.r.t. camera and model

Results

3 of 12 photographs

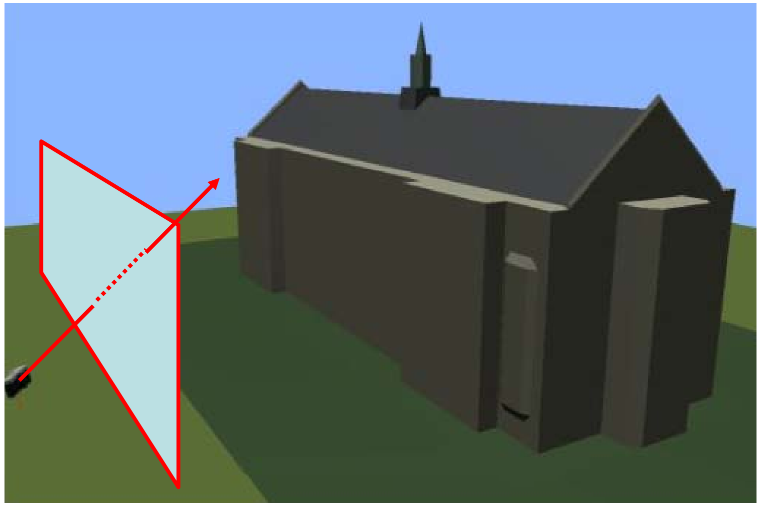


Texture mapping

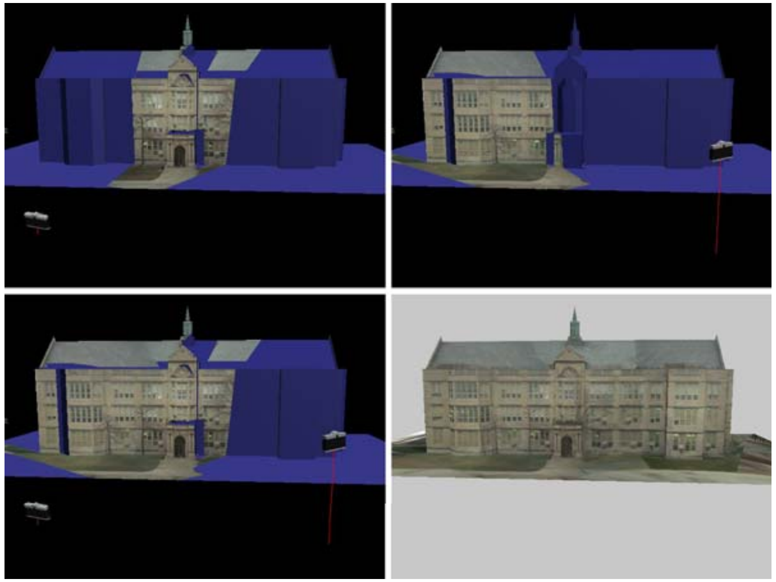
Texture mapping in real world

Demo movie
Michael Naimark,
San Francisco Museum
of Modern Art, 1984

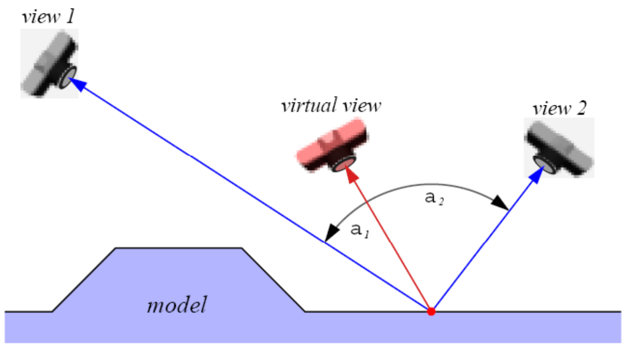
Texture mapping



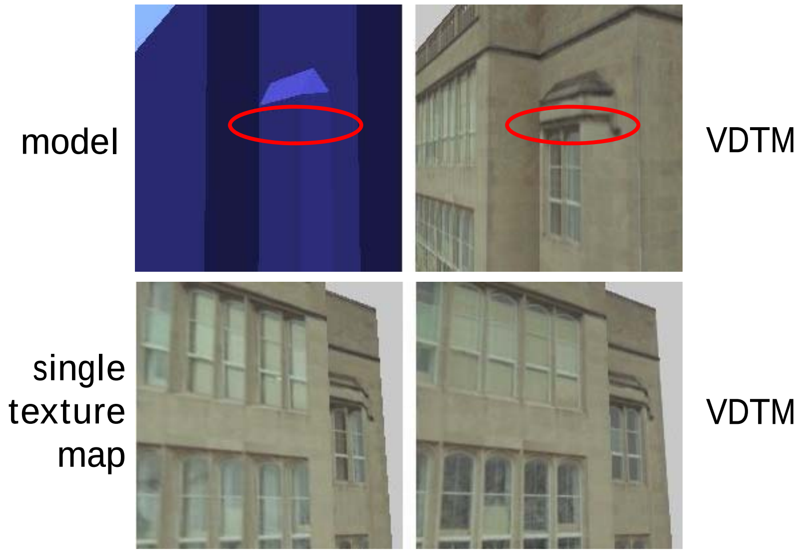
Texture mapping



View-dependent texture mapping



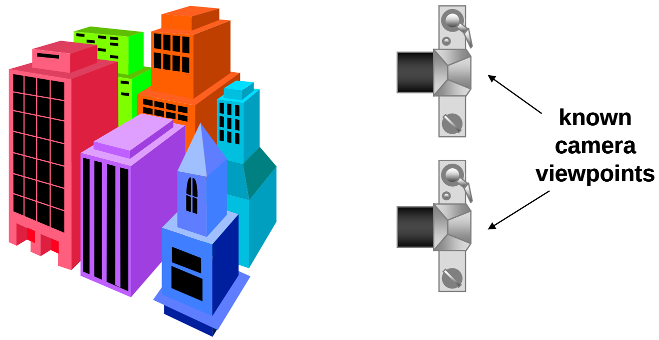
View-dependent texture mapping



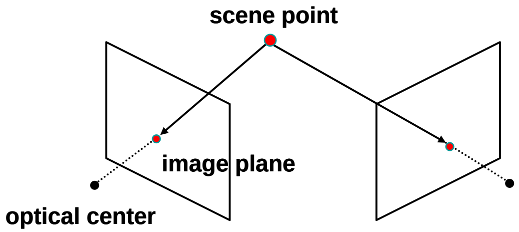
View-dependent texture mapping

Model-based stereo

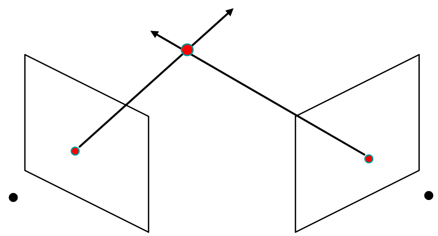
- Use stereo to refine the geometry



Stereo



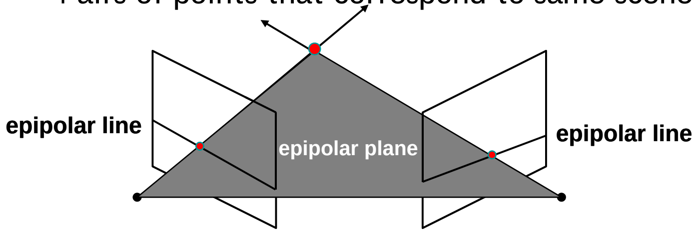
Stereo



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

Stereo correspondence

- 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

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

Image registration (revisited)

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

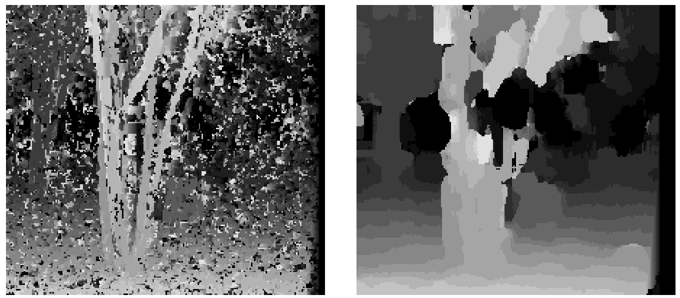
$$E(x,y;d)=\sum_{(x',y')\in N(x,y)}[I_L(x'+d,y')-I_R(x',y')]^2$$

d is the *disparity* (horizontal motion)

- How big should the neighborhood be?

Neighborhood size

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

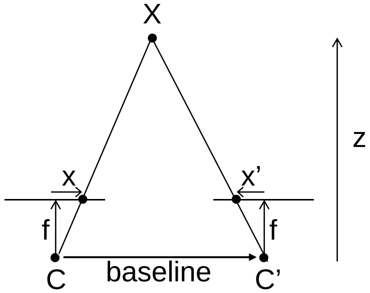


Depth from disparity

input image (1 of 2)

depth map
[Szeliski & Kang '95]

3D rendering



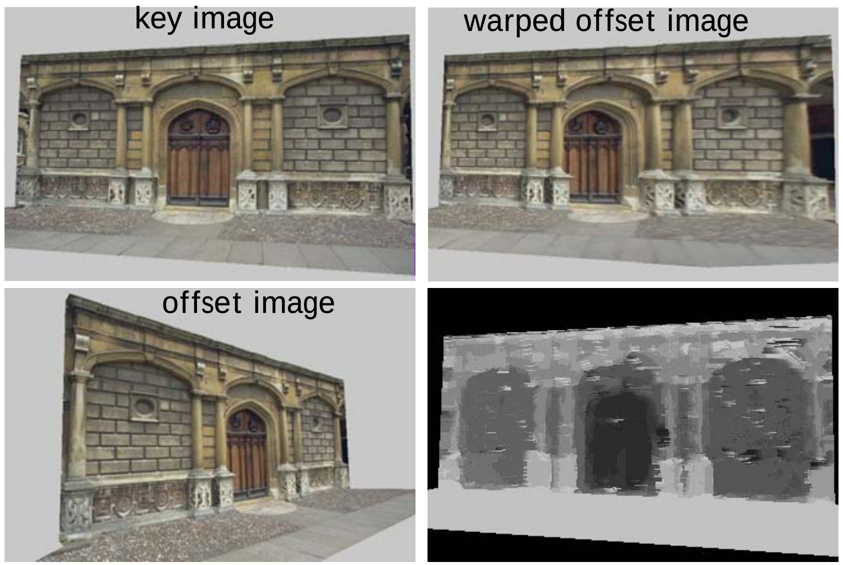
$$disparity = x - x' = \frac{baseline * f}{z}$$

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



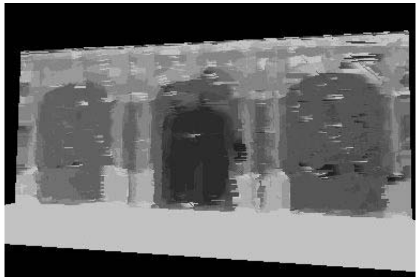
Results

Comparisons

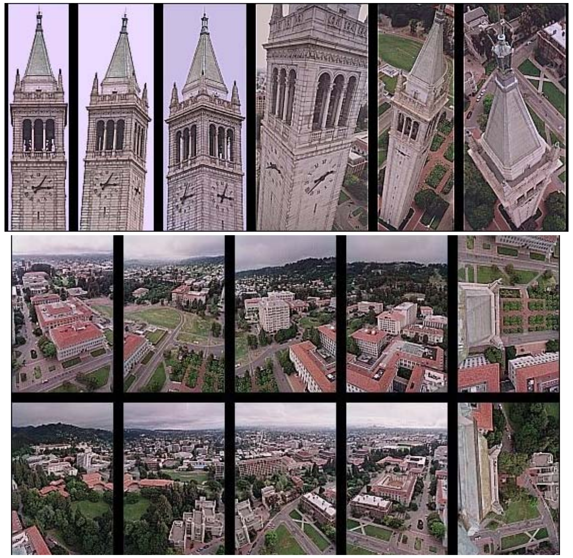

single texture, flat


VDTM, flat

VDTM, model-based stereo

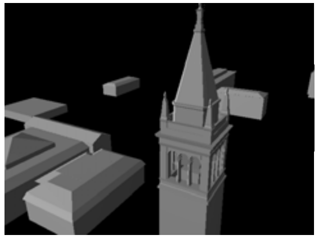


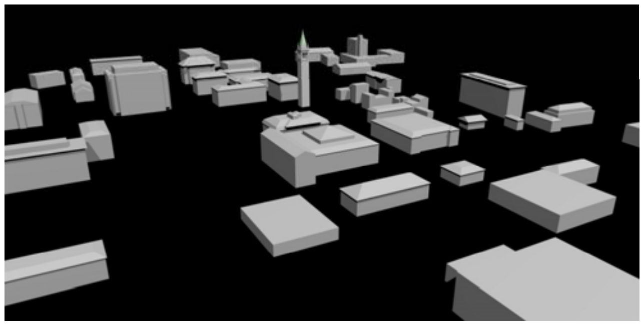
Final results

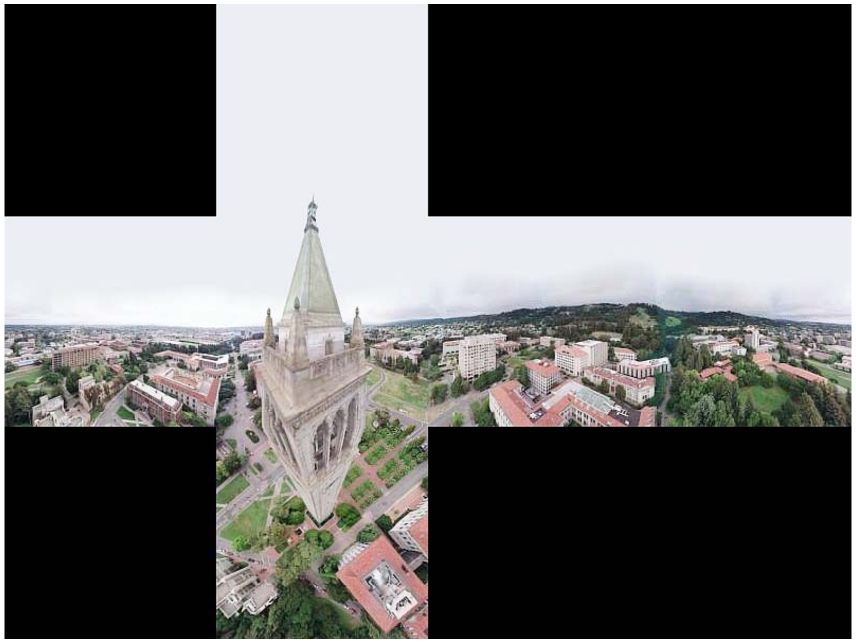


Kite photography

Final results







Results

Results

Commercial packages

- Autodesk ~~REALVIZ~~ ImageModeler

The Matrix

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."

The Matrix

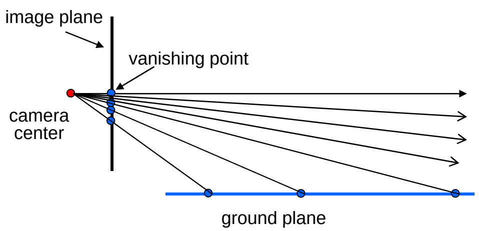
- *Academy Awards for Scientific and Technical achievement for 2000*

  To George Borshukov, Kim Libreri and Dan Piponi for the development of a system for image-based rendering allowing choreographed camera movements through computer graphic reconstructed sets.

This was used in The Matrix and Mission Impossible II; See The Matrix Disc #2 for more details

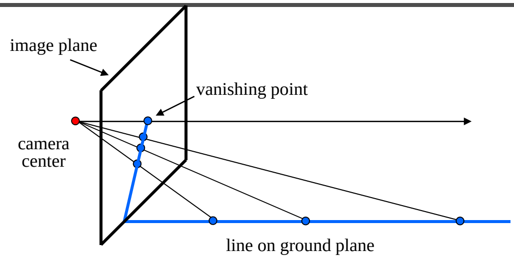
Models from single images

Vanishing points

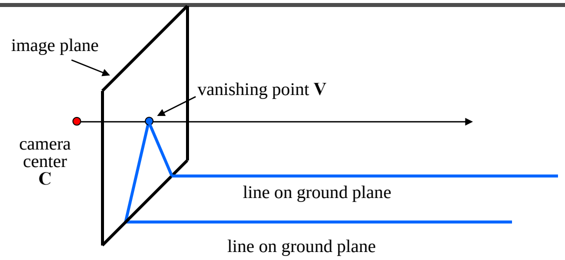


- Vanishing point
  - projection of a point at infinity

Vanishing points (2D)

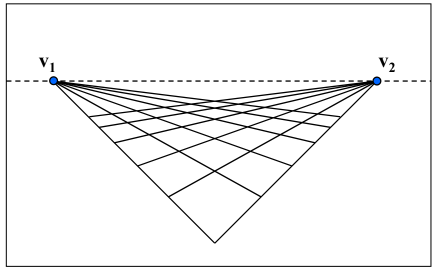


Vanishing points



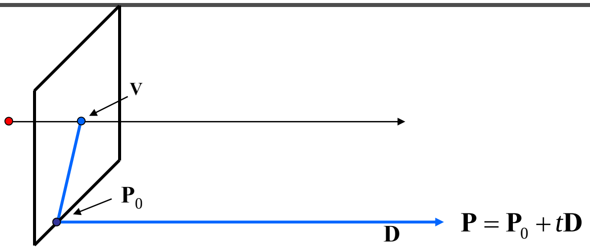
- Properties
  - Any two parallel lines have the same vanishing point **v**
  - The ray from **C** through **v** is parallel to the lines
  - An image may have more than one vanishing point

Vanishing lines



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

Computing vanishing points



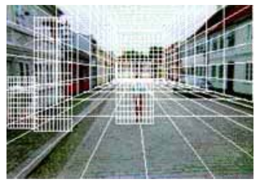
$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_X / t + D_X \\ P_Y / t + D_Y \\ P_Z / t + D_Z \\ 1/t \end{bmatrix} \qquad t \to \infty \qquad \mathbf{P}_\infty \cong \begin{bmatrix} D_X \\ D_Y \\ D_Z \\ 0 \end{bmatrix}$$

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

Tour into pictures

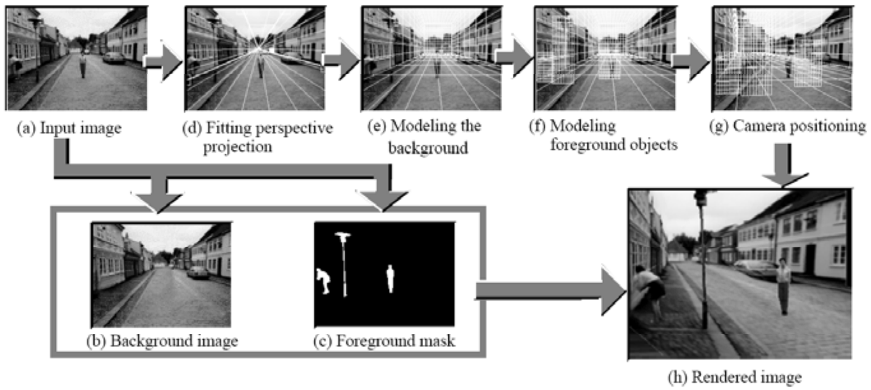
- Create a 3D “theatre stage” of five billboards

- Specify foreground objects through bounding polygons



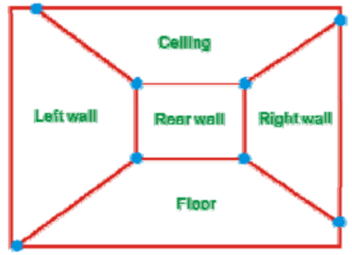
- Use camera transformations to navigate through the scene

Tour into pictures

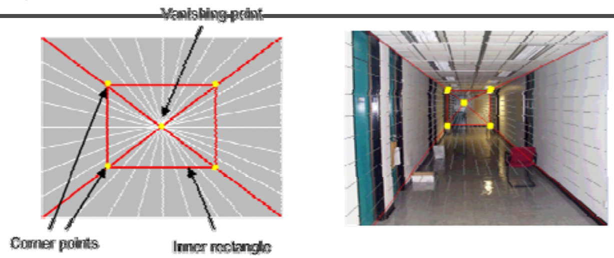


The idea

- 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!



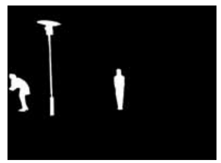
Fitting the box volume



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

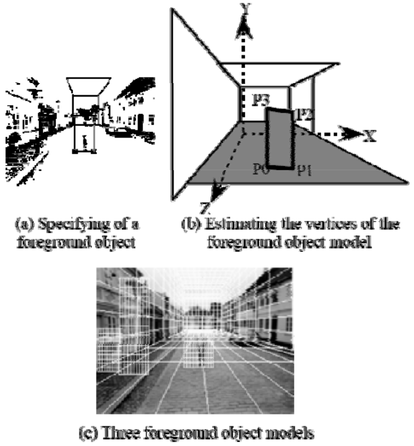
Foreground Objects

- 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

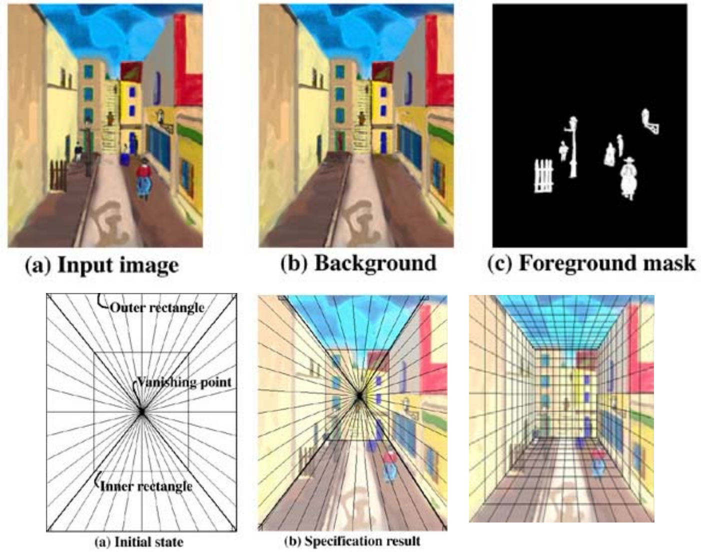


Foreground Objects

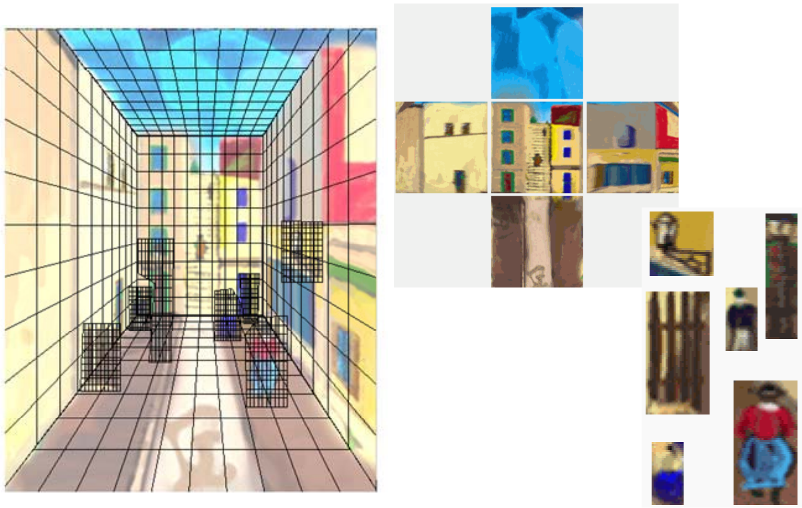
- Add vertical rectangles for each foreground object
- Can compute 3D coordinates P0, P1 since they are on known plane.
- P2, P3 can be computed as before (similar triangles)



Example



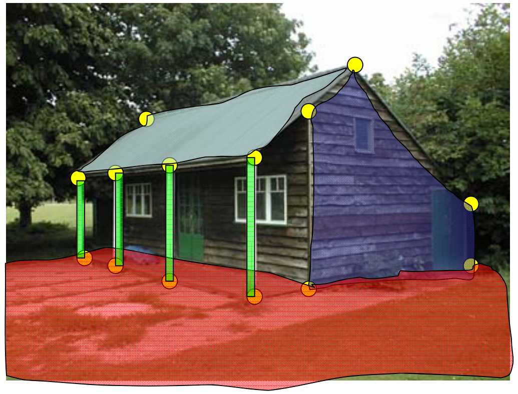
Example



glTip

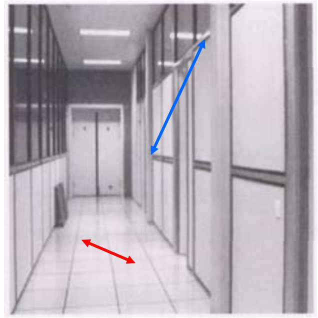
- http://www.cs.ust.hk/~cpegnel/glTIP/

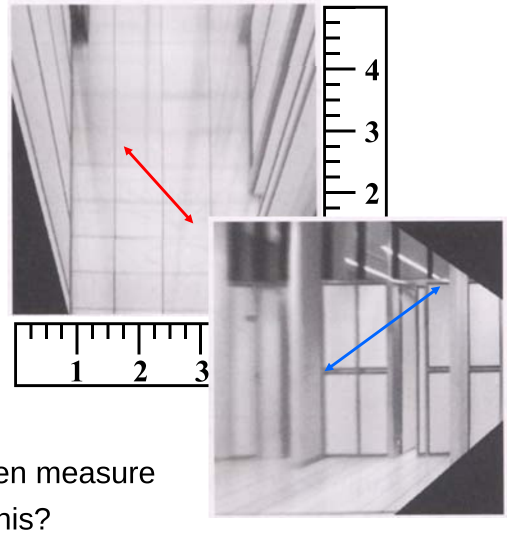
Criminisi *et al.* ICCV 1999



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

Measurements on planes

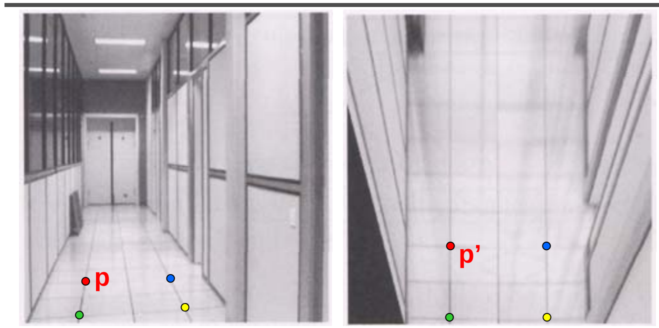




Approach: unwarp then measure

What kind of warp is this?

Image rectification



To unwarp (rectify) an image

- solve for homography **H** given **p** and **p'**
- 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

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

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

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

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

Solving for homographies

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

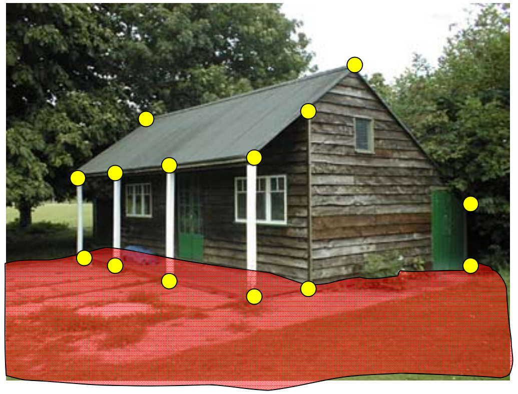
$\mathbf{A}$ (2n × 9) $\quad$ $\mathbf{h}$ (9) $\quad$ $\mathbf{0}$ (2n)

- Defines a least squares problem:

$$\text{minimize } \|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$$

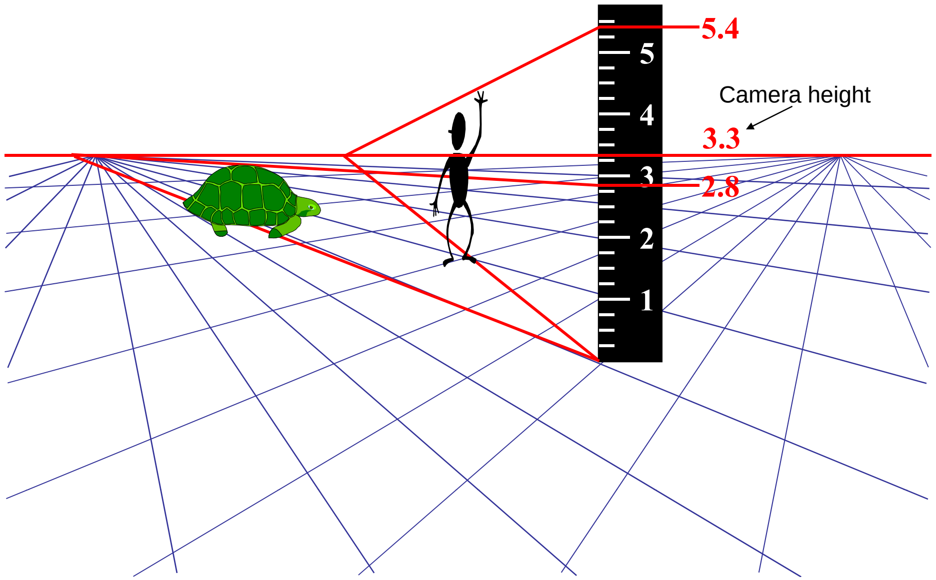
  - Since **h** is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
  - Works with 4 or more points

Finding world coordinates (X,Y,Z)

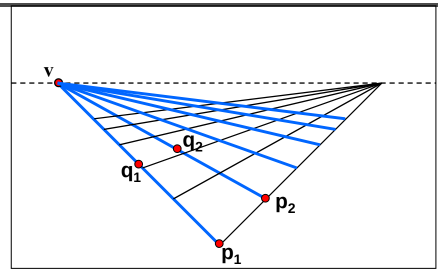


1. Define the ground plane (Z=0)
2. Compute points (X,Y,0) on that plane
3. Compute the *heights* Z of all other points

Measuring height



Computing vanishing points



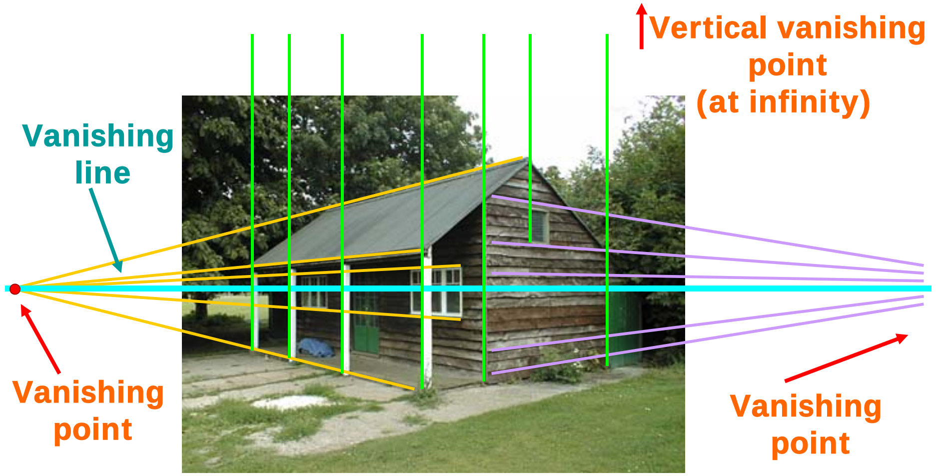
- Intersect p_1q_1 with p_2q_2

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

Criminisi et al., ICCV 99

- Load in an image
- Click on lines parallel to X axis
  - repeat for Y, Z axes
- Compute vanishing points

Criminisi et al., ICCV 99

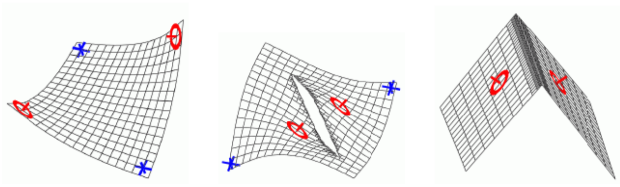


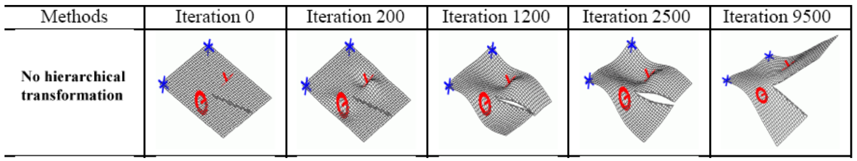
Criminisi et al., ICCV 99

- 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
- Output a VRML model

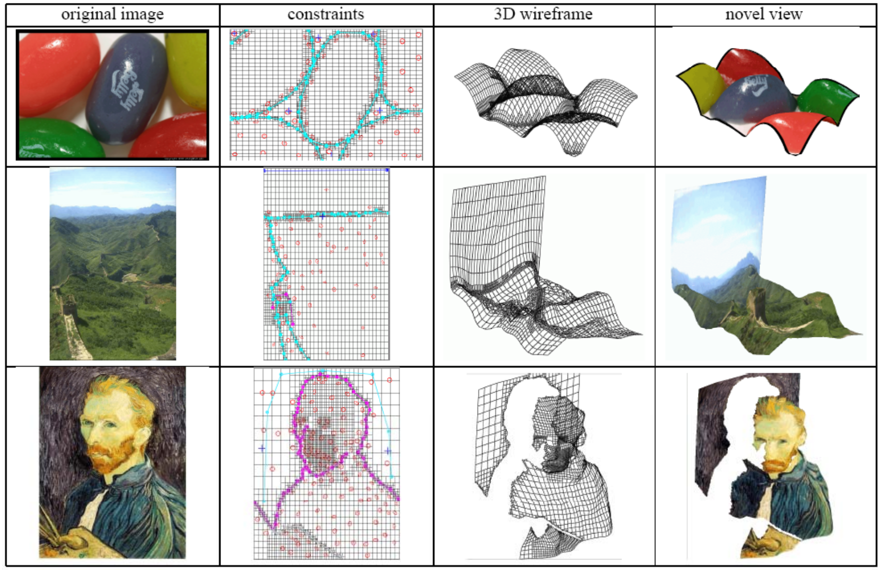
Results

Zhang *et. al.* CVPR 2001

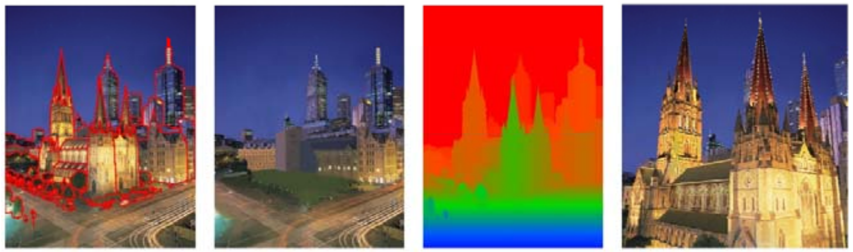


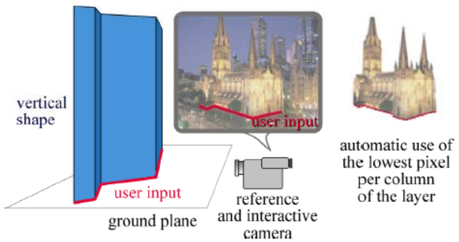


Zhang *et. al.* CVPR 2001

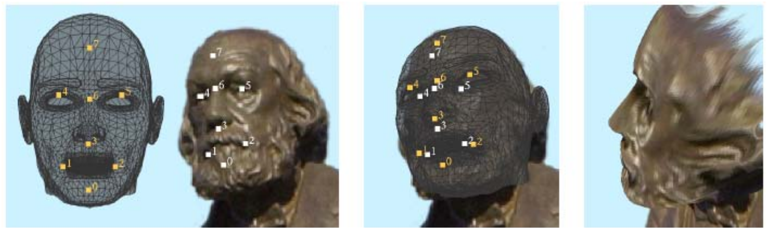


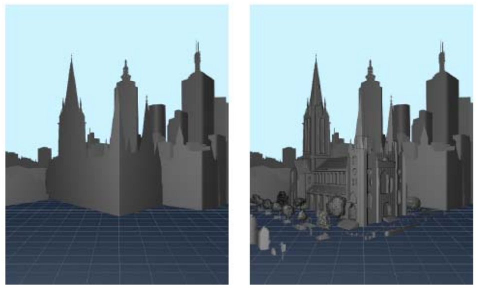
Oh *et. al.* SIGGRAPH 2001





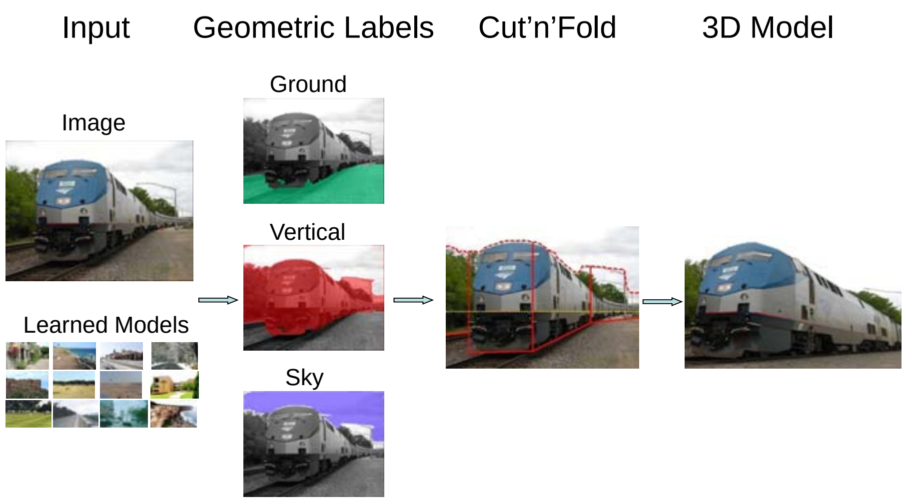
Oh *et. al.* SIGGRAPH 2001



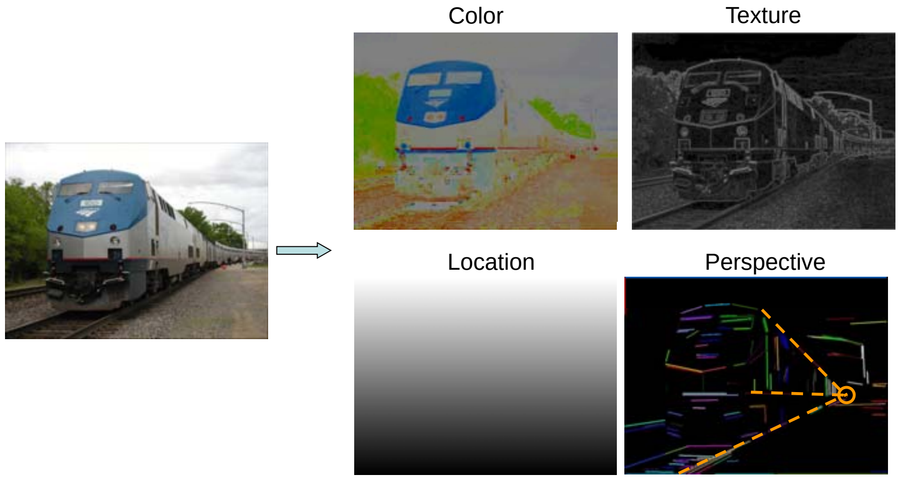


video

Automatic popup



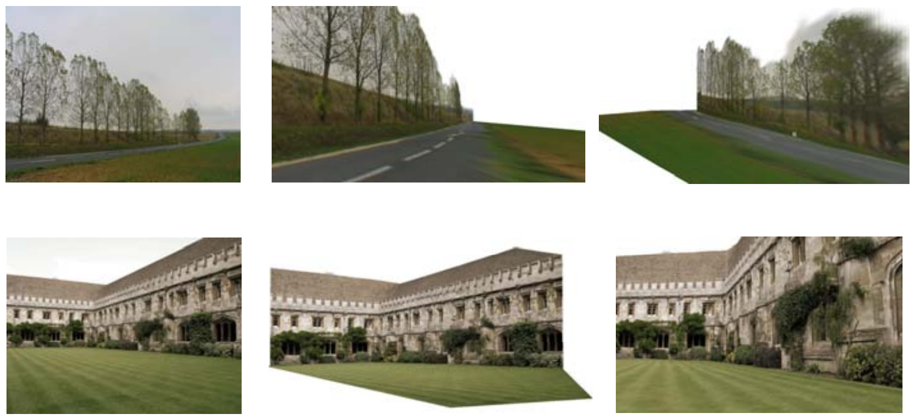
Geometric cues



Automatic popup

Feature Descriptions	Num	Used
Color	**15**	**15**
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
Texture	**29**	**13**
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
Location and Shape	**12**	**10**
L1. Location: normalized x and y, mean	2	2
L2. Location: norm. x and y, 10^{th} and 90^{th} percentile	4	4
L3. Location: norm. y wrt horizon, 10^{th} and 90^{th} pctl	2	2
L4. Shape: number of superpixels in constellation	1	1
L5. Shape: number of sides of convex hull	1	0
L6. Shape: *num pixels/area(convex hull)*	1	1
L7. Shape: whether the constellation region is contiguous	1	0
3D Geometry	**35**	**28**
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

Results



Input Images

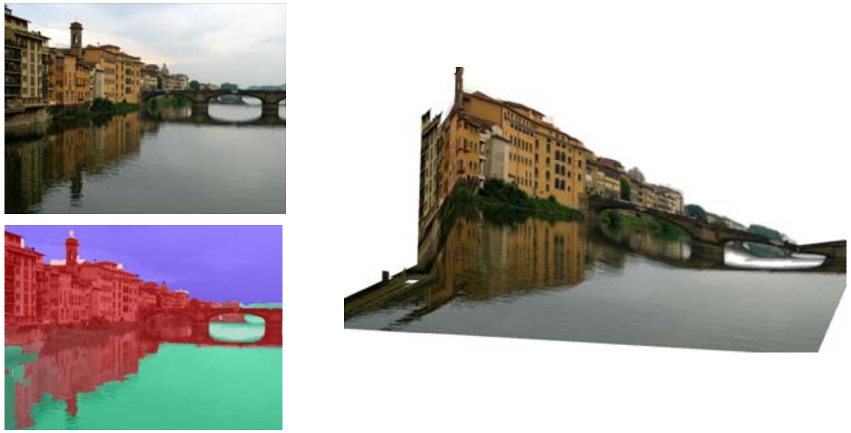
Automatic Photo Pop-up

Results

This approach works roughly for 35% of images.

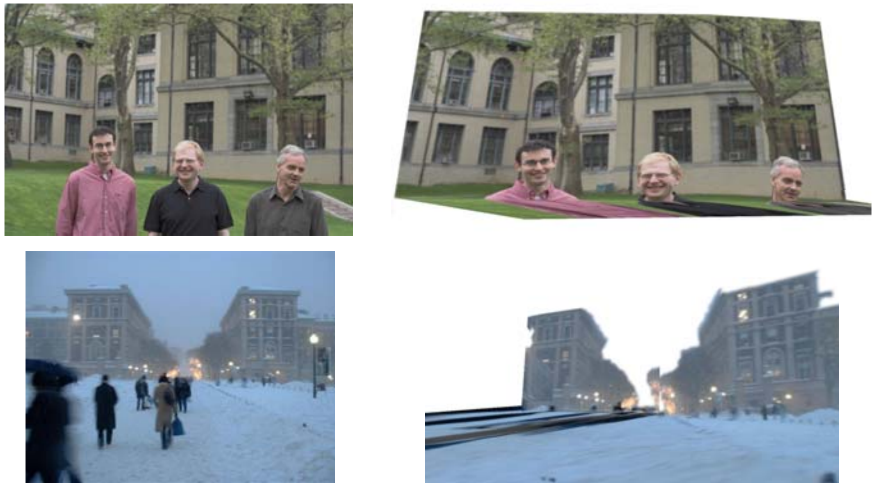
Failures

Labeling Errors



Failures

Foreground Objects



References

- 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.