Image Stitching

Computer Vision

CSE 576, Spring 2008

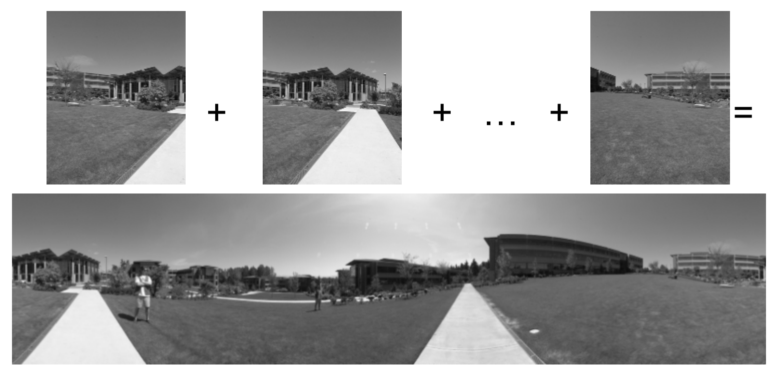
Richard Szeliski
Microsoft Research

Panoramic Image Mosaics

Full screen panoramas (cubic): http://www.panoramas.dk/
Mars: http://www.panoramas.dk/fullscreen3/f2_mars97.html
2003 New Years Eve: http://www.panoramas.dk/fullscreen3/f1.html

Gigapixel panoramas & images

Image Mosaics



Goal: Stitch together several images into a seamless composite

Today's lecture

Image alignment and stitching

- motion models
- image warping
- point-based alignment
- complete mosaics (global alignment)
- compositing and blending
- ghost and parallax removal

Readings

- Szeliski, *CVAA*:
  - Chapter 3.5: Image warping
  - Chapter 5.1: Feature-based alignment (in preparation)
  - Chapter 8.1: Motion models
  - Chapter 8.2: Global alignment
  - Chapter 8.3: Compositing

- Recognizing Panoramas, Brown & Lowe, ICCV'2003
- Szeliski & Shum, SIGGRAPH'97

Motion models

Motion models

What happens when we take two images with a camera and try to align them?

- translation?
- rotation?
- scale?
- affine?
- perspective?

… see interactive demo (VideoMosaic)

Image Warping

Image Warping

image filtering: change *range* of image

$g(x) = h(f(x))$

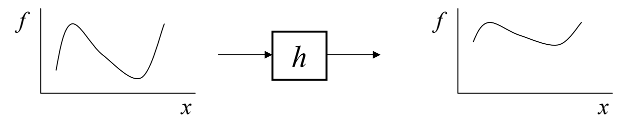


image warping: change *domain* of image

$g(x) = f(h(x))$

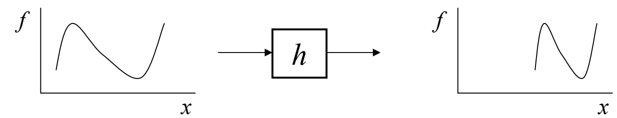


Image Warping

image filtering: change *range* of image

$g(x) = h(f(x))$

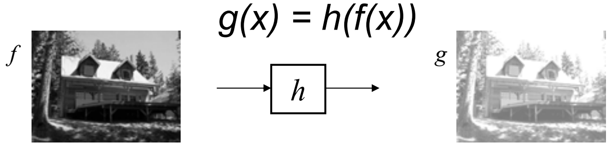
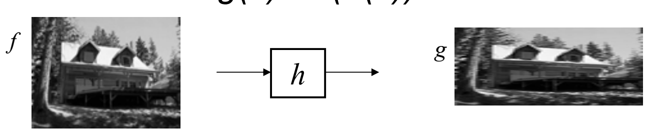


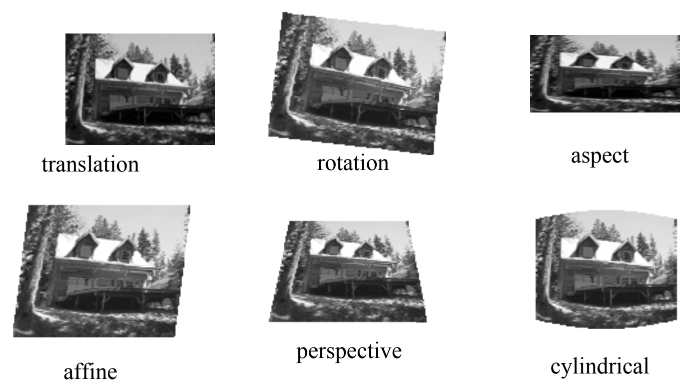
image warping: change *domain* of image

$g(x) = f(h(x))$



Parametric (global) warping

Examples of parametric warps:



2D coordinate transformations

translation: $\mathbf{x'} = \mathbf{x} + \mathbf{t}$ $\mathbf{x} = (x,y)$

rotation: $\mathbf{x'} = \mathbf{R}\,\mathbf{x} + \mathbf{t}$

similarity: $\mathbf{x'} = s\,\mathbf{R}\,\mathbf{x} + \mathbf{t}$

affine: $\mathbf{x'} = \mathbf{A}\,\mathbf{x} + \mathbf{t}$

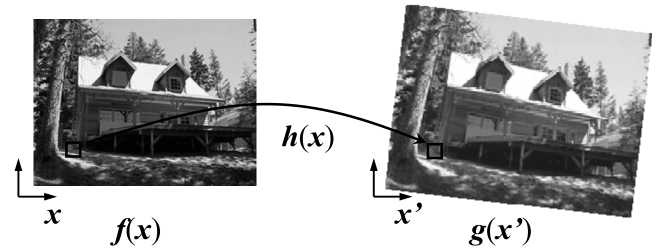
perspective: $\underline{\mathbf{x}}' \cong \mathbf{H}\,\underline{\mathbf{x}}$ $\underline{\mathbf{x}} = (x,y,1)$

($\underline{\mathbf{x}}$ is a *homogeneous* coordinate)

These all form a nested *group* (closed w/ inv.)

Image Warping

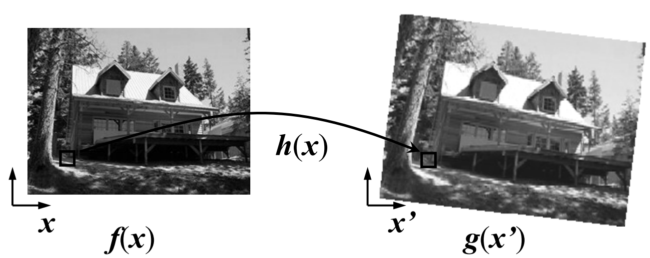
Given a coordinate transform $\mathbf{x'} = \mathbf{h}(\mathbf{x})$ and a source image $\mathbf{f}(\mathbf{x})$, how do we compute a transformed image $\mathbf{g}(\mathbf{x'}) = \mathbf{f}(\mathbf{h}(\mathbf{x}))$?



Forward Warping

Send each pixel $\mathbf{f}(\mathbf{x})$ to its corresponding location $\mathbf{x'} = \mathbf{h}(\mathbf{x})$ in $\mathbf{g}(\mathbf{x'})$

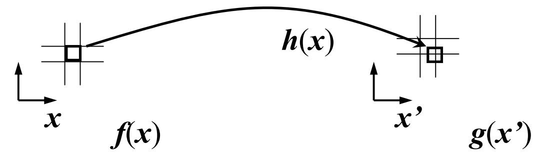
- What if pixel lands "between" two pixels?



Forward Warping

Send each pixel $\mathbf{f}(\mathbf{x})$ to its corresponding location $\mathbf{x'} = \mathbf{h}(\mathbf{x})$ in $\mathbf{g}(\mathbf{x'})$

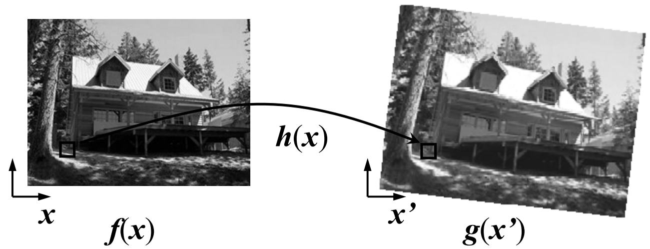
- What if pixel lands "between" two pixels?
- Answer: add "contribution" to several pixels, normalize later (*splatting*)



Inverse Warping

Get each pixel ***g***(***x'***) from its corresponding location ***x'*** = ***h***(***x***) in ***f***(***x***)

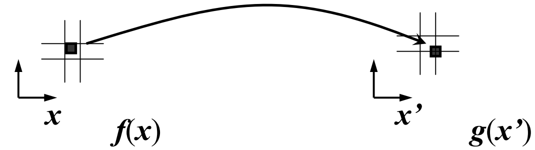
- What if pixel comes from "between" two pixels?



Inverse Warping

Get each pixel ***g***(***x'***) from its corresponding location ***x'*** = ***h***(***x***) in ***f***(***x***)

- What if pixel comes from "between" two pixels?
- Answer: *resample* color value from *interpolated* (*prefiltered*) source image



Interpolation

Possible interpolation filters:

- nearest neighbor
- bilinear
- bicubic (interpolating)
- sinc / FIR

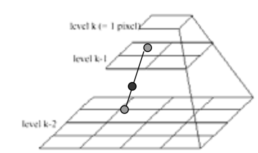
Needed to prevent "jaggies" and "texture crawl" (see demo)

Prefiltering

Essential for *downsampling* (*decimation*) to prevent *aliasing*

MIP-mapping [Williams'83]:

1. build pyramid (but what decimation filter?):
   - block averaging
   - Burt & Adelson (5-tap binomial)
   - 7-tap wavelet-based filter (better)
2. *trilinear* interpolation
   - bilinear within each 2 adjacent levels
   - linear blend *between* levels (determined by pixel size)



Prefiltering

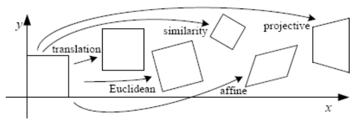
Essential for *downsampling* (*decimation*) to prevent *aliasing*

Other possibilities:

- summed area tables
- elliptically weighted Gaussians (EWA) [Heckbert'86]

Motion models (reprise)

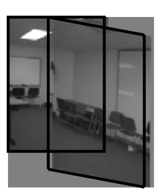
Motion models



Translation

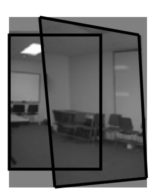
2 unknowns

Affine



6 unknowns

Perspective



8 unknowns

3D rotation

3 unknowns

Plane perspective mosaics

- 8-parameter generalization of affine motion
  - works for pure rotation or planar surfaces
- Limitations:
  - local minima
  - slow convergence
  - difficult to control interactively

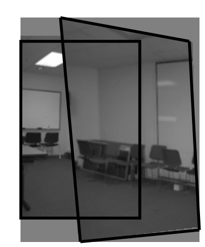


Image warping with homographies

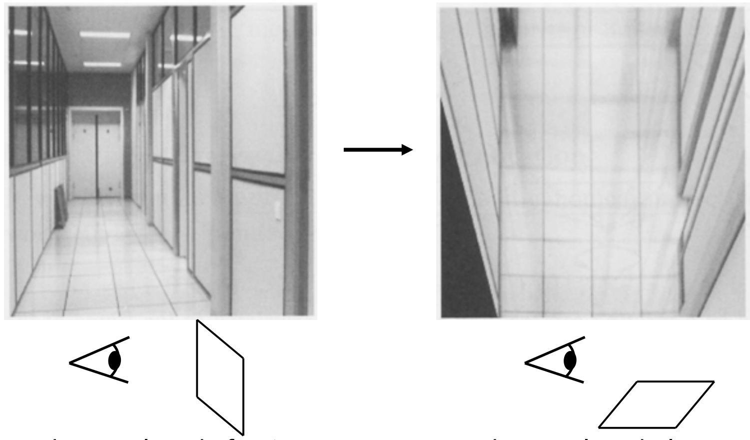
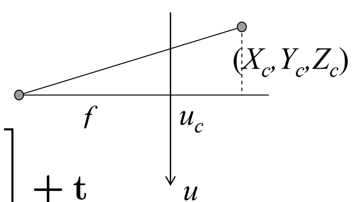


image plane in front | image plane below

Rotational mosaics

- Directly optimize rotation and focal length
- Advantages:
  - ability to build full-view panoramas
  - easier to control interactively
  - more stable and accurate estimates

3D → 2D Perspective Projection



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Rotational mosaic

Projection equations

1. Project from image to 3D ray

   $(x_0,y_0,z_0) = (u_0-u_c, v_0-v_c, f)$

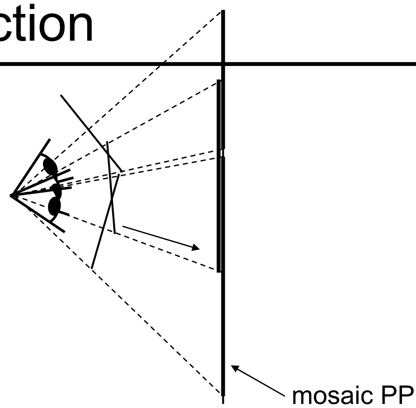
2. Rotate the ray by camera motion

   $(x_1,y_1,z_1) = \boldsymbol{R}_{01}\,(x_0,y_0,z_0)$

3. Project back into new (source) image

   $(u_1,v_1) = (fx_1/z_1+u_c, fy_1/z_1+v_c)$

Image reprojection



The mosaic has a natural interpretation in 3D

- The images are reprojected onto a common plane
- The mosaic is formed on this plane

Rotations and quaternions

How do we represent rotation matrices?

1. Axis / angle $(\boldsymbol{n},\theta)$
   $\boldsymbol{R} = \boldsymbol{I} + \sin\theta\, [\boldsymbol{n}]_\times + (1-\cos\theta)\, [\boldsymbol{n}]_\times^2$
   (Rodriguez Formula), with
   $[\boldsymbol{n}]_\times$ = cross product matrix (see paper)
2. Unit quaternions [Shoemake SIGG'85]
   $\boldsymbol{q} = (\boldsymbol{n} \sin\theta/2, \cos\theta/2) = (\boldsymbol{w},s)$
   quaternion multiplication (division is easy)
   $\boldsymbol{q}_0\, \boldsymbol{q}_1 = (s_1\boldsymbol{w}_0 + s_0\boldsymbol{w}_1,\, s_0\, s_1 - \boldsymbol{w}_0 \cdot \boldsymbol{w}_1)$

Incremental rotation update

1. Small angle approximation
   $\Delta\boldsymbol{R} = \boldsymbol{I} + \sin\theta\, [\boldsymbol{n}]_\times + (1-\cos\theta)\, [\boldsymbol{n}]_\times^2$
   $\approx \theta\, [\boldsymbol{n}]_\times = [\boldsymbol{\omega}]_\times$
   linear in $\boldsymbol{\omega}$
2. Update original $\boldsymbol{R}$ matrix
   $\boldsymbol{R} \leftarrow \boldsymbol{R}\, \Delta\boldsymbol{R}$

Perspective & rotational motion

Solve 8x8 or 3x3 system (see papers for details), and iterate (non-linear)

Patch-based approximation:

1. break up image into patches (say 16x16)
2. accumulate 2x2 linear system in each (local translational assumption)
3. compose larger system from smaller 2x2 results [Shum & Szeliski, ICCV'98]

Image Mosaics (Stitching)

[Szeliski & Shum, SIGGRAPH'97]

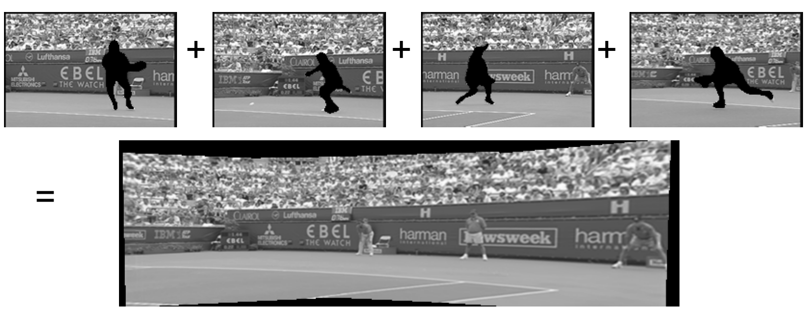
[Szeliski, FnT CVCG, 2006]

Image Mosaics (stitching)

Blend together several overlapping images into one seamless *mosaic* (composite)

Mosaics for Video Coding

Convert masked images into a background sprite for content-based coding



Establishing correspondences

1. Direct method:
   - Use generalization of affine motion model [Szeliski & Shum '97]
2. Feature-based method
   - Extract features, match, find consisten *inliers* [Lowe ICCV'99; Schmid ICCV'98, Brown&Lowe ICCV'2003]
   - Compute ***R*** from correspondences (absolute orientation)

Absolute orientation

[Arun *et al.*, PAMI 1987] [Horn *et al.*, JOSA A 1988]
Procrustes Algorithm [Golub & VanLoan]

Given two sets of matching points, compute R

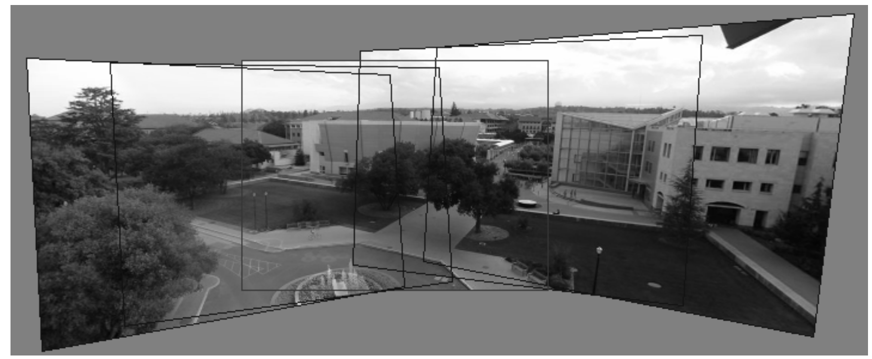
$p_i' = \boldsymbol{R}\, p_i$ 3D rays

$\boldsymbol{A} = \Sigma_{\mathbf{i}}\, p_i\, p_i'^T = \Sigma_{\mathbf{i}}\, p_i\, p_i^T \boldsymbol{R}^T = \boldsymbol{U}\,\boldsymbol{S}\,\boldsymbol{V}^T = (\boldsymbol{U}\,\boldsymbol{S}\,\boldsymbol{U}^T)\,\boldsymbol{R}^T$

$\boldsymbol{V}^T = \boldsymbol{U}^T \boldsymbol{R}^T$

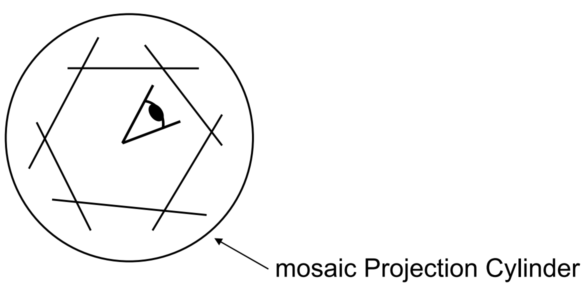
$\boldsymbol{R} = \boldsymbol{V}\,\boldsymbol{U}^T$

Stitching demo

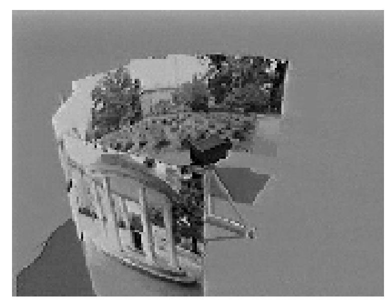


Panoramas

What if you want a 360° field of view?



Cylindrical panoramas



Steps

- Reproject each image onto a cylinder
- Blend
- Output the resulting mosaic

Cylindrical Panoramas

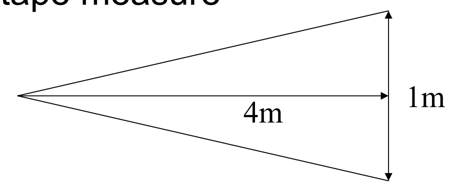
Map image to cylindrical or spherical coordinates

- need *known* focal length

Image 384x300 **f = 180 (pixels)** **f = 280** **f = 380**

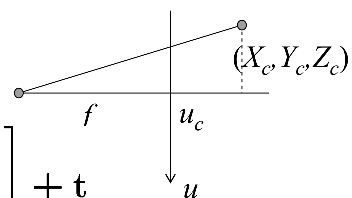
Determining the focal length

1. Initialize from homography $\boldsymbol{H}$ (see text or [SzSh'97])
2. Use camera's EXIF tags (approx.)
3. Use a tape measure



4. Ask your instructor

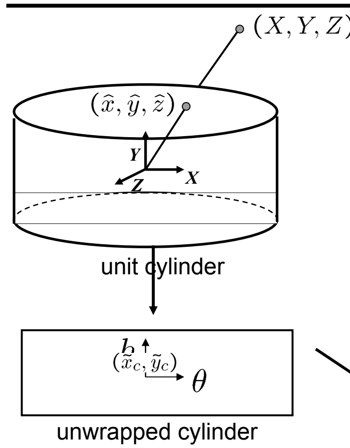
3D → 2D Perspective Projection



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

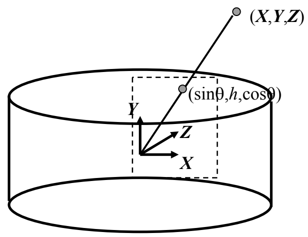
Cylindrical projection



- Map 3D point (X,Y,Z) onto cylinder
  $$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X, Y, Z)$$
- Convert to cylindrical coordinates
  $$(sin\theta, h, cos\theta) = (\hat{x}, \hat{y}, \hat{z})$$
- Convert to cylindrical image coordinates
  $$(\tilde{x}, \tilde{y}) = (s\theta, sh) + (\tilde{x}_c, \tilde{y}_c)$$
  – s defines size of the final image

Cylindrical warping

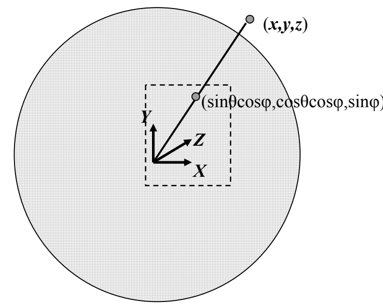
Given focal length f and image center (x_c,y_c)



$$\begin{aligned}\theta &= (x_{cyl} - x_c)/f \\ h &= (y_{cyl} - y_c)/f \\ \hat{x} &= \sin\theta \\ \hat{y} &= h \\ \hat{z} &= \cos\theta \\ x &= f\hat{x}/\hat{z} + x_c \\ y &= f\hat{y}/\hat{z} + y_c\end{aligned}$$

Spherical warping

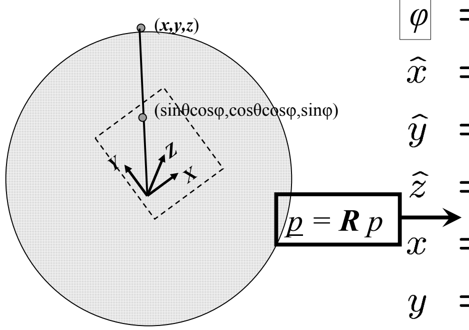
Given focal length f and image center (x_c,y_c)



$$\begin{aligned}\theta &= (x_{cyl} - x_c)/f \\ \varphi &= (y_{cyl} - y_c)/f \\ \hat{x} &= \sin\theta \cos\varphi \\ \hat{y} &= \sin\varphi \\ \hat{z} &= \cos\theta \cos\varphi \\ x &= f\hat{x}/\hat{z} + x_c \\ y &= f\hat{y}/\hat{z} + y_c\end{aligned}$$

3D rotation

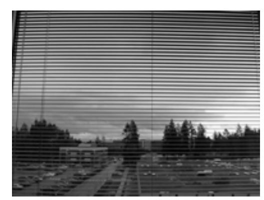
Rotate image before placing on unrolled sphere



$$\begin{aligned}\theta &= (x_{cyl} - x_c)/f \\ \varphi &= (y_{cyl} - y_c)/f \\ \hat{x} &= \sin\theta \cos\varphi \\ \hat{y} &= \sin\varphi \\ \hat{z} &= \cos\theta \cos\varphi \\ x &= f\underline{\hat{x}}/\underline{\hat{z}} + x_c \\ y &= f\underline{\hat{y}}/\underline{\hat{z}} + y_c\end{aligned}$$

Radial distortion

Correct for "bending" in wide field of view lenses



$$\begin{aligned}\hat{r}^2 &= \hat{x}^2 + \hat{y}^2 \\ \hat{x}' &= \hat{x}/(1 + \kappa_1\hat{r}^2 + \kappa_2\hat{r}^4) \\ \hat{y}' &= \hat{y}/(1 + \kappa_1\hat{r}^2 + \kappa_2\hat{r}^4) \\ x &= f\hat{x}'/\hat{z} + x_c \\ y &= f\hat{y}'/\hat{z} + y_c\end{aligned}$$

Fisheye lens

Extreme "bending" in ultra-wide fields of view

$$\hat{r}^2 = \hat{x}^2 + \hat{y}^2$$

$$(\cos\theta\sin\phi, \sin\theta\sin\phi, \cos\phi) = s\,(x, y, z)$$

uations become

$$x' = s\phi\cos\theta = s\frac{x}{r}\tan^{-1}\frac{r}{z},$$
$$y' = s\phi\sin\theta = s\frac{y}{r}\tan^{-1}\frac{r}{z},$$

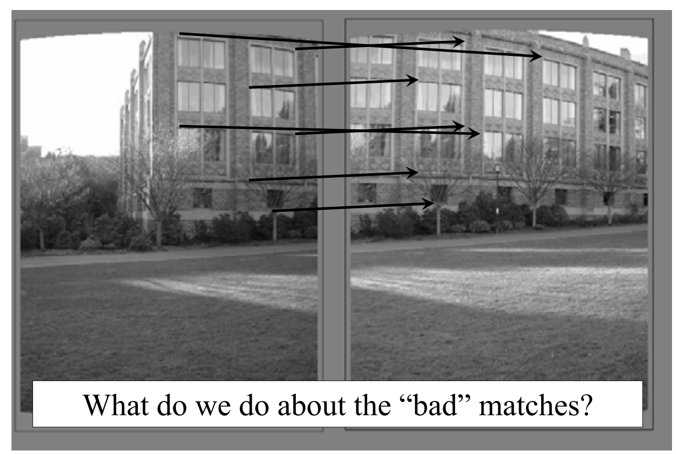
Image Stitching

1. Align the images over each other
   - camera pan ↔ translation on cylinder
2. Blend the images together (demo)

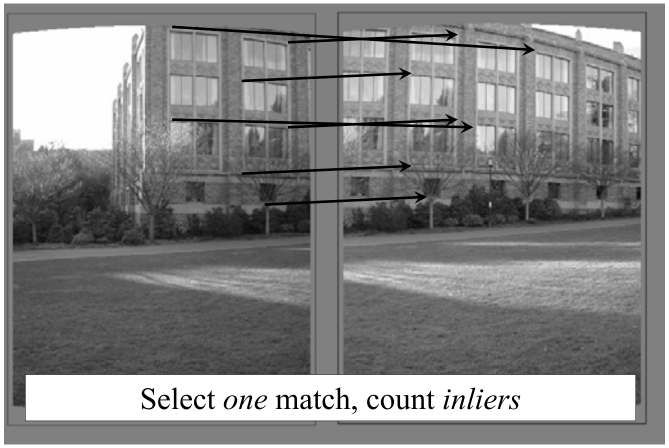
Project 2 – image stitching

1. Take pictures on a tripod (or handheld)
2. Warp images to spherical coordinates
3. Extract features
4. Align neighboring pairs using RANSAC
5. Write out list of neighboring translations
6. Correct for drift
7. Read in warped images and blend them
8. Crop the result and import into a viewer

Matching features

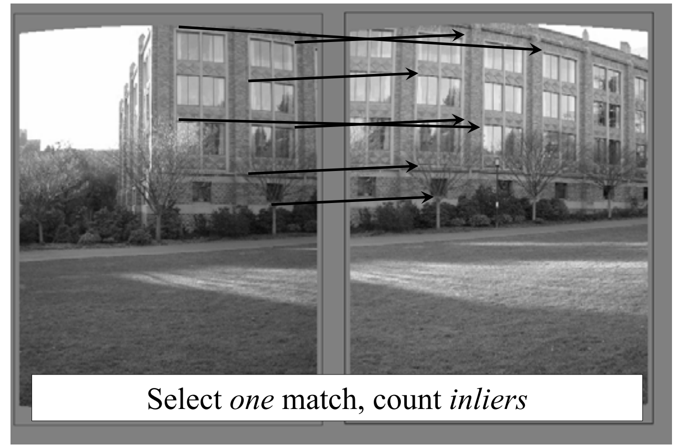


RAndom SAmple Consensus



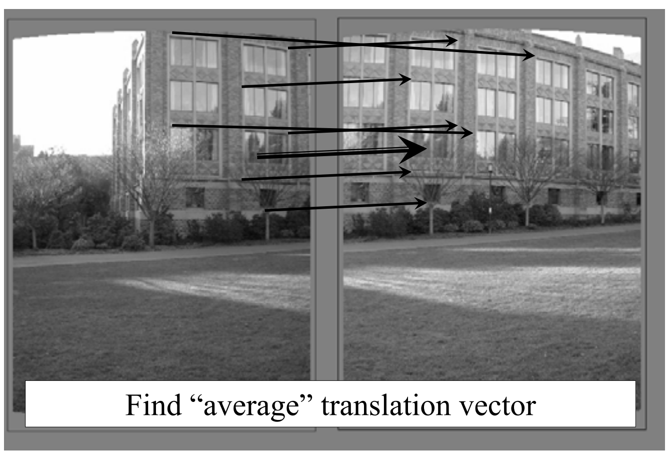
Select *one* match, count *inliers*

RAndom SAmple Consensus



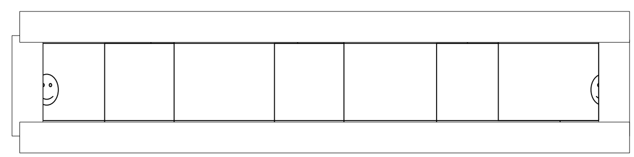
Select *one* match, count *inliers*

Least squares fit



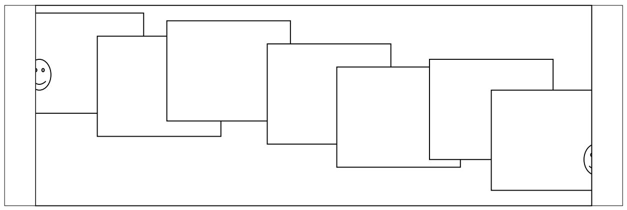
Find “average” translation vector

Assembling the panorama



Stitch pairs together, blend, then crop

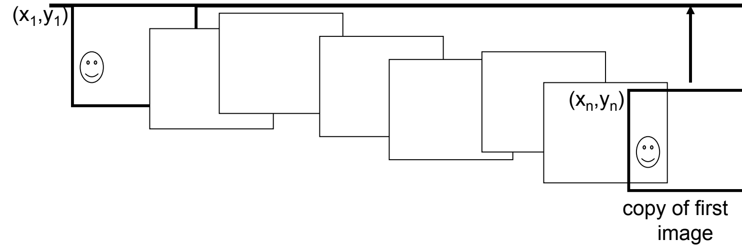
Problem: Drift



Error accumulation

- small (vertical) errors accumulate over time
- apply correction so that sum = 0 (for 360° pan.)

Problem: Drift

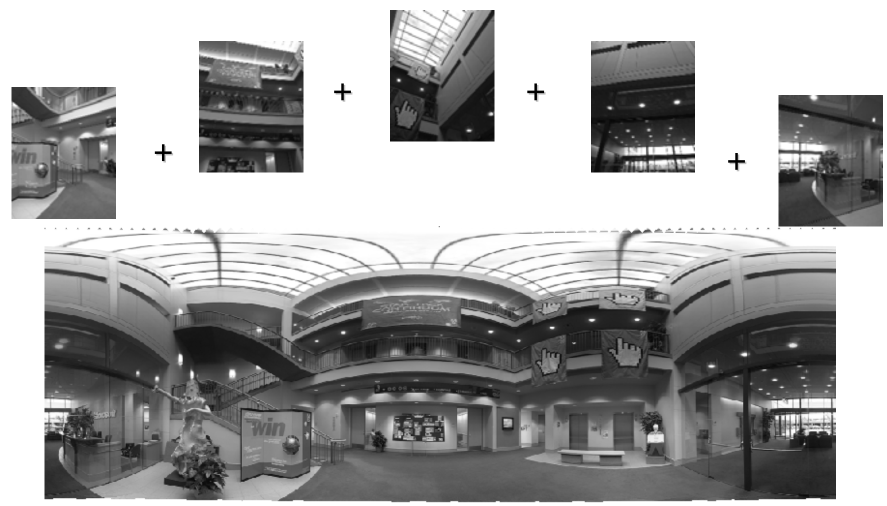


Solution

- add another copy of first image at the end
- this gives a constraint: $y_n = y_1$
- there are a bunch of ways to solve this problem
  - add displacement of $(y_1 – y_n)/(n -1)$ to each image after the first
  - compute a global warp: $y' = y + ax$
  - run a big optimization problem, incorporating this

Full-view (360° spherical) panoramas

Full-view Panorama



Texture Mapped Model

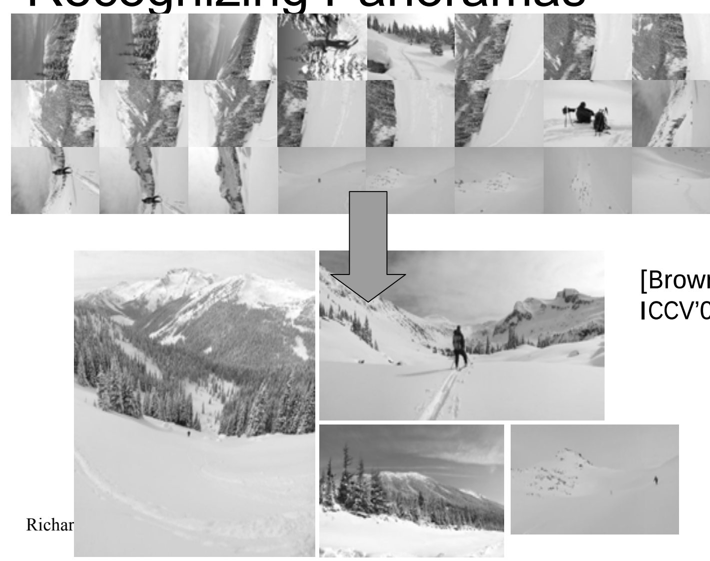
Global alignment

- Register *all* pairwise overlapping images
- Use a 3D rotation model (one R per image)
- Use direct alignment (patch centers) or feature based
- *Infer* overlaps based on previous matches (incremental)
- Optionally *discover* which images overlap other images using feature selection (RANSAC)

Recognizing Panoramas

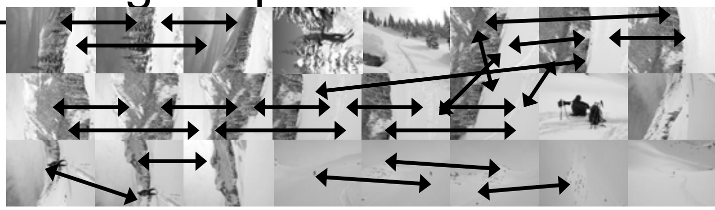
Matthew Brown & David Lowe
ICCV'2003

Recognizing Panoramas

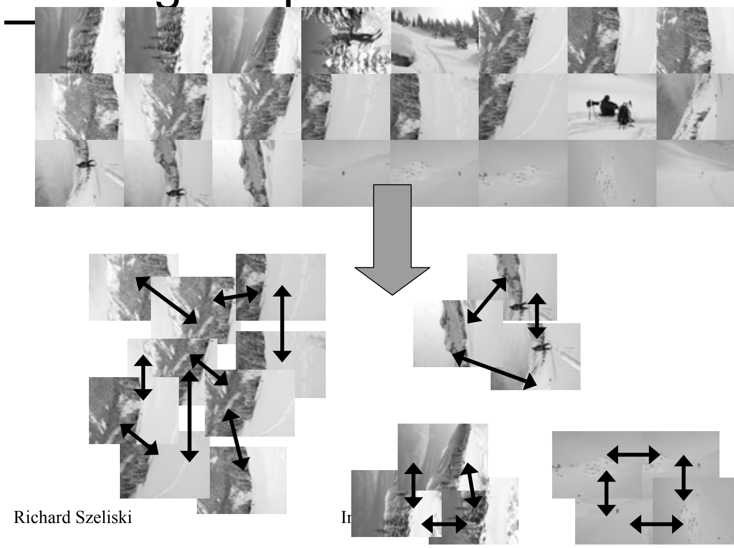


[Brown & Lowe, ICCV'03]

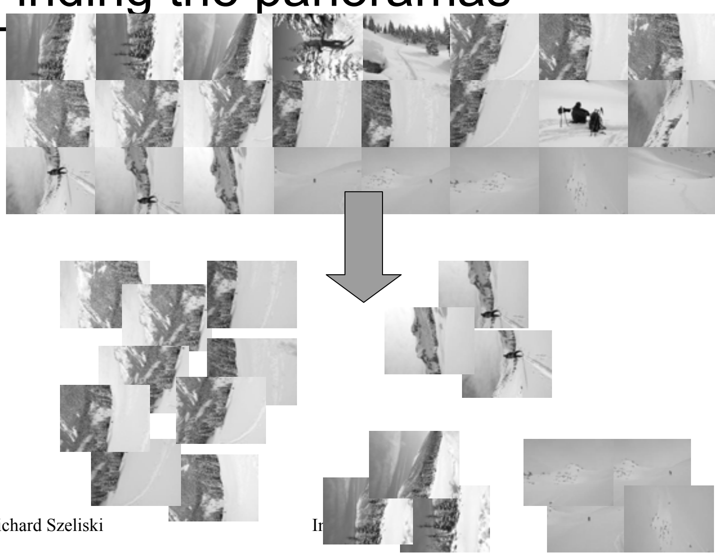
Finding the panoramas



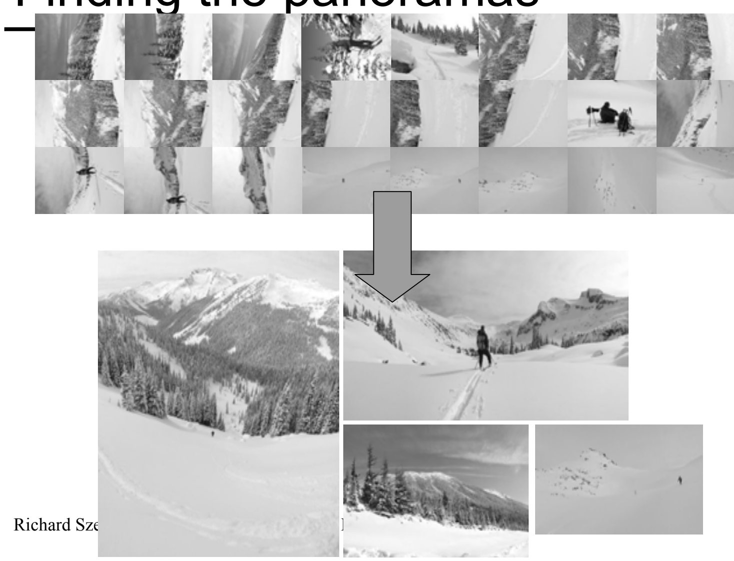
Finding the panoramas



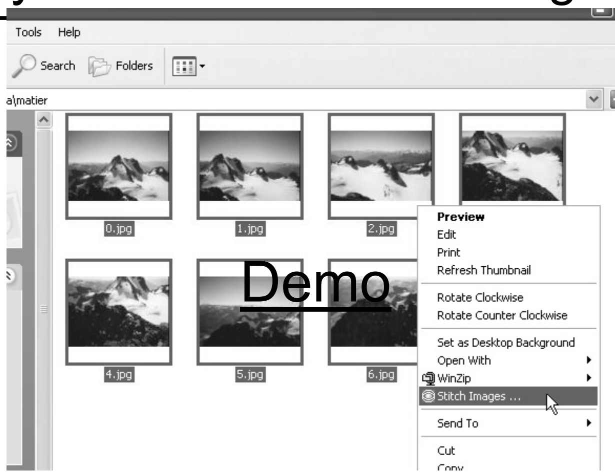
Finding the panoramas



Finding the panoramas

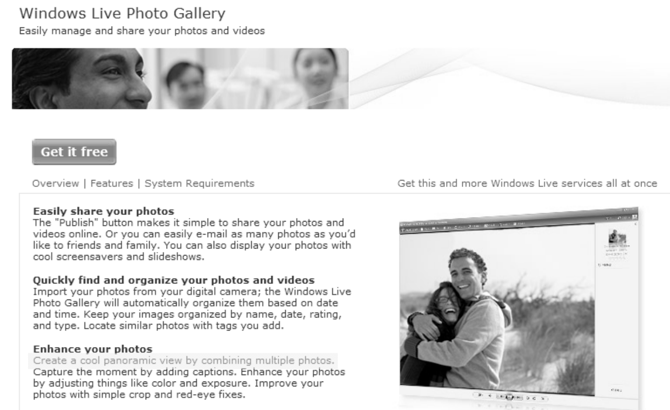


Fully automated 2D stitching



Demo

Get you own free copy



http://get.live.com/photogallery/overview

Rec.pano.: system components

1. Feature detection and description
   - more uniform point density
2. Fast matching (hash table)
3. RANSAC filtering of matches
4. Intensity-based verification
5. Incremental bundle adjustment

[M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches, CVPR'2005]

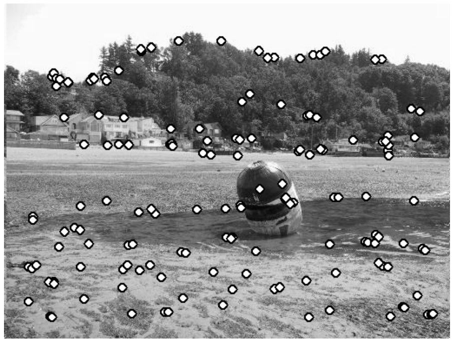
Multi-Scale Oriented Patches

Interest points
- Multi-scale Harris corners
- Orientation from blurred gradient
- Geometrically invariant to similarity transforms

Descriptor vector
- Bias/gain normalized sampling of local patch (8x8)
- Photometrically invariant to affine changes in intensity

Feature irregularities

Distribute points evenly over the image



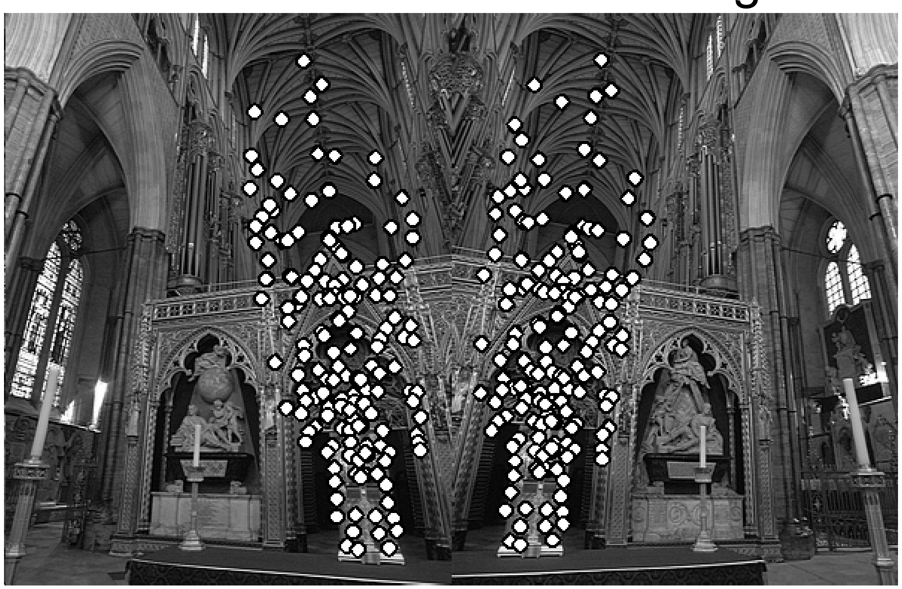
Descriptor Vector

Orientation = blurred gradient

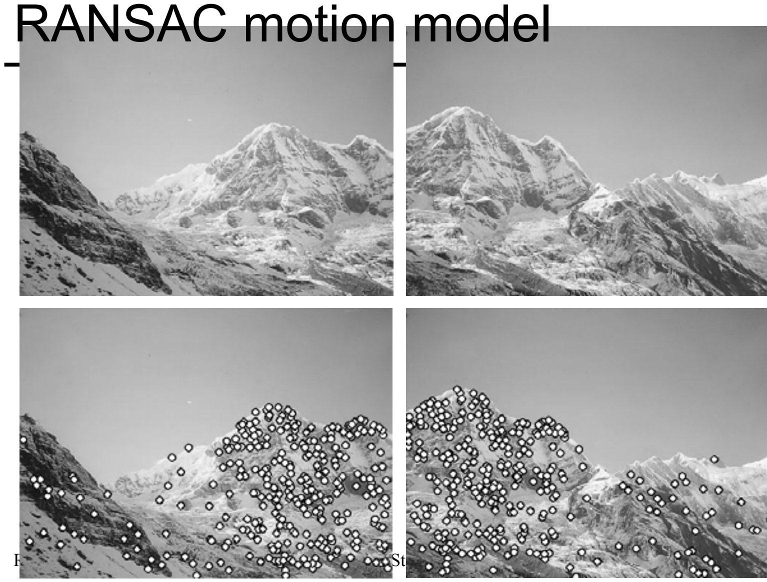
Similarity Invariant Frame

- Scale-space position (x, y, s) + orientation (θ)

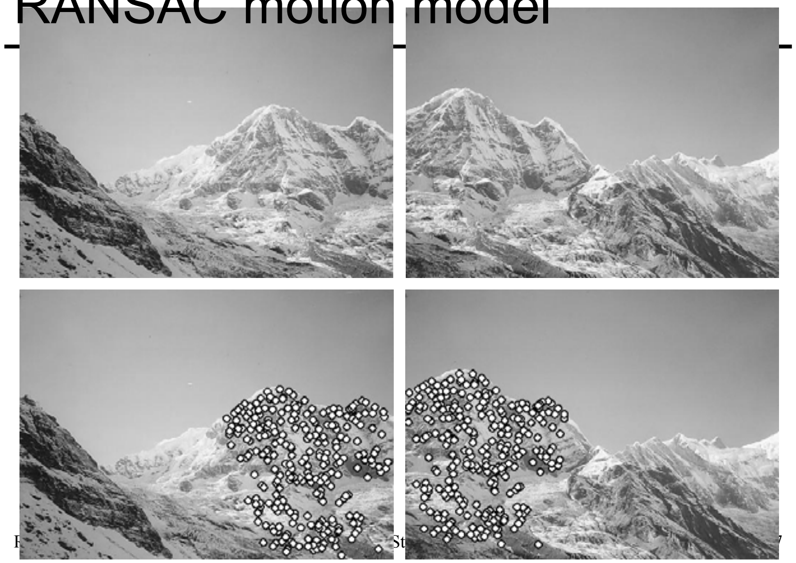
Probabilistic Feature Matching



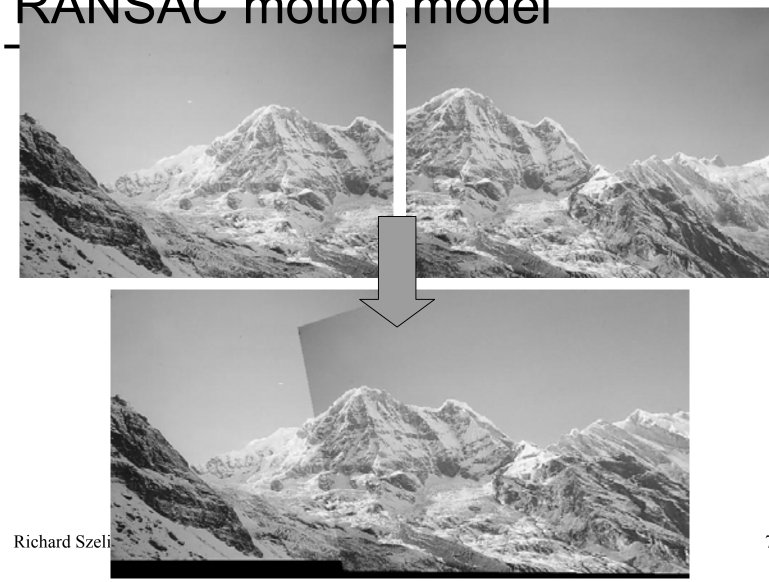
RANSAC motion model



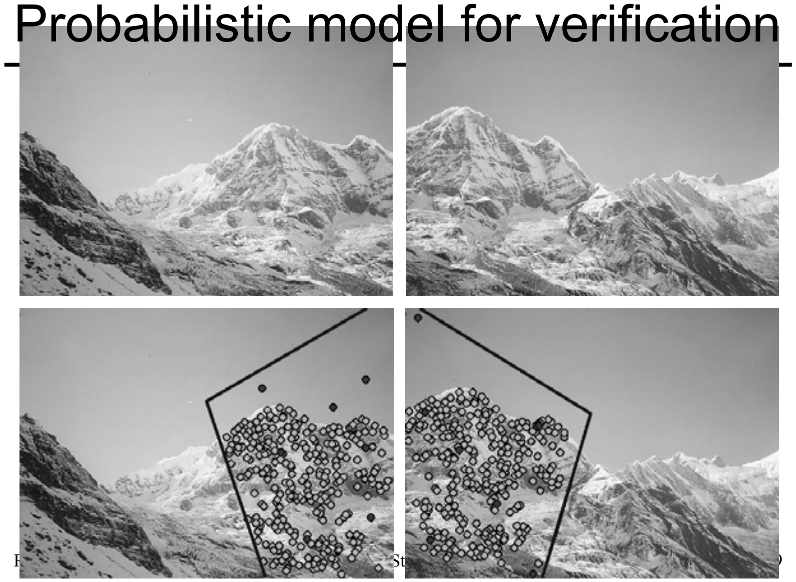
RANSAC motion model



RANSAC motion model



Probabilistic model for verification



How well does this work?

Test on 100s of examples…

How well does this work?

Test on 100s of examples…

…still too many failures (5-10%)
for <u>consumer</u> application

Matching Mistakes: False Positive

Matching Mistakes: False Positive

Matching Mistake: False Negative

Moving objects: large areas of disagreement

Matching Mistakes

Accidental alignment

- repeated / similar regions

Failed alignments

- moving objects / parallax
- low overlap
- "feature-less" regions (more variety?)

No 100% reliable algorithm?

How can we fix these?

Tune the feature detector

Tune the feature matcher (cost metric)

Tune the RANSAC stage (motion model)

Tune the verification stage

Use "higher-level" knowledge

- e.g., typical camera motions

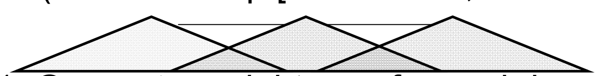
→ Sounds like a big "learning" problem

- Need a large training/test data set (panoramas)

Image Blending

Image feathering

Weight each image proportional to its distance from the edge
(distance map [Danielsson, CVGIP 1980]



1. Generate *weight map* for each image
2. Sum up all of the weights and divide by sum: weights sum up to 1: $w_i' = w_i / (\sum_i w_i)$

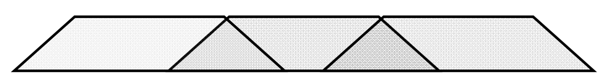
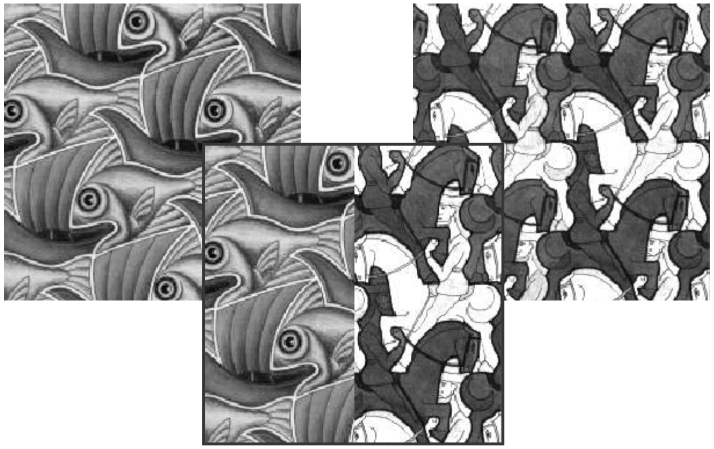
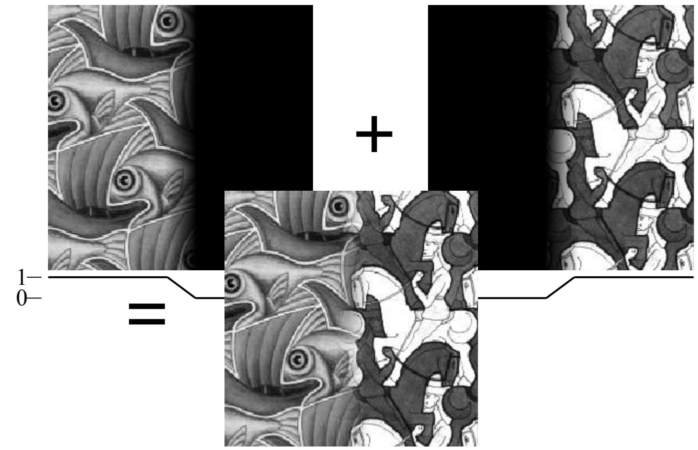


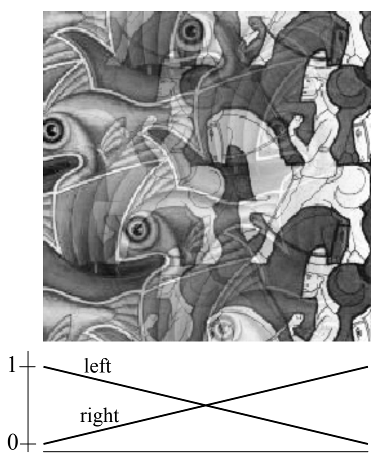
Image Feathering

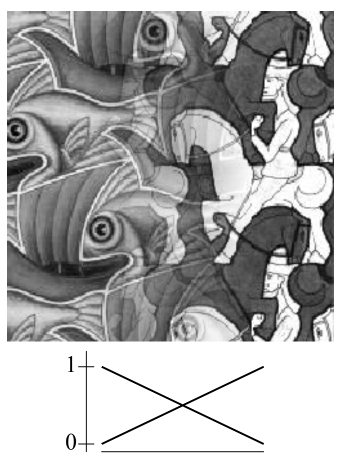


Feathering

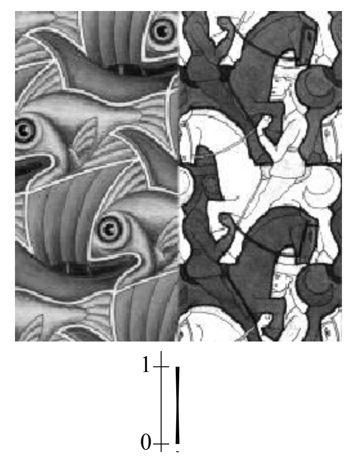


Effect of window size

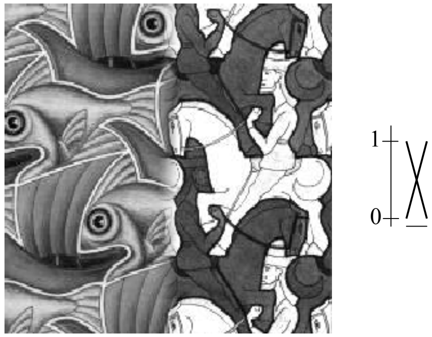




Effect of window size



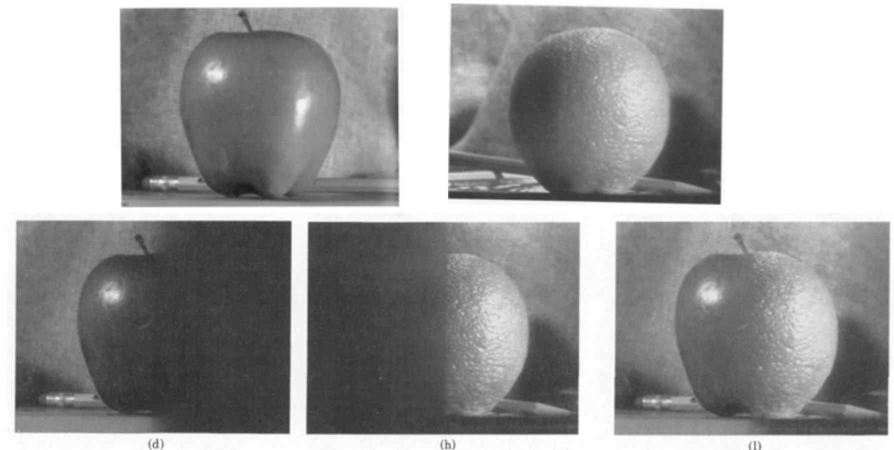
Good window size



"Optimal" window: smooth but not ghosted

- Doesn't always work...

Pyramid Blending

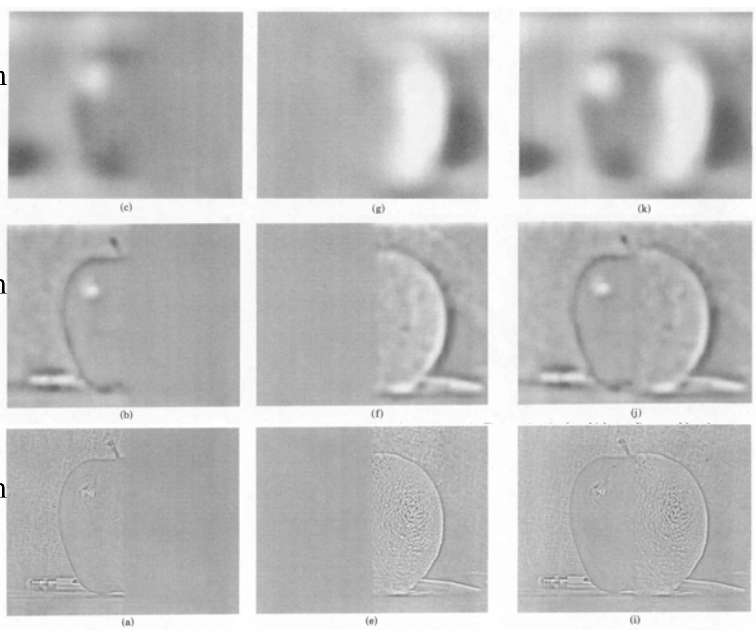


Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.

Laplacian level 4

Laplacian level 2

Laplacian level 0



left pyramid right pyramid blended pyramid

Laplacian image blend

1. Compute Laplacian pyramid
2. Compute Gaussian pyramid on *weight* image (can put this in A channel)
3. Blend Laplacians using Gaussian blurred weights
4. Reconstruct the final image

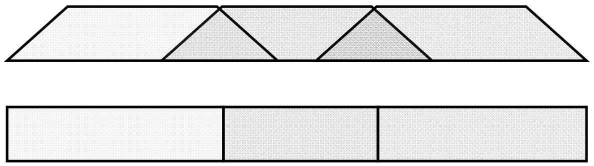
Q: How do we compute the original weights?

A: For horizontal panorama, use *mid-lines*

Q: How about for a general "3D" panorama?

Weight selection (3D panorama)

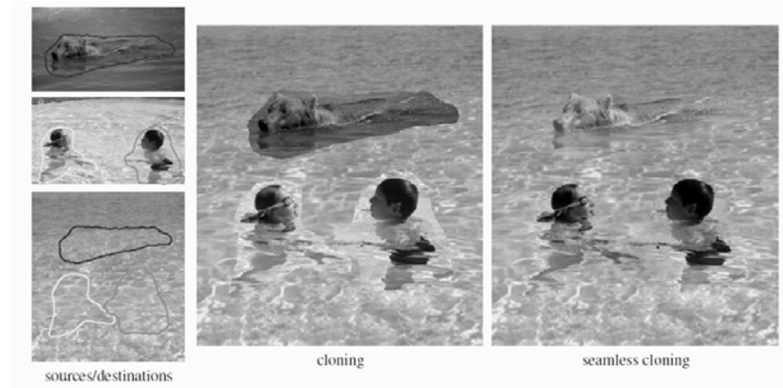
Idea: use original feather weights to select *strongest* contributing image



Can be implemented using L-∞ norm: ($p = 10$)

$$w_i' = [w_i^p / (\sum_i w_i^p)]^{1/p}$$

Poisson Image Editing



Blend the gradients of the two images, then integrate
For more info: Perez et al, SIGGRAPH 2003

De-Ghosting

Local alignment (deghosting)

Use local optic flow to compensate for small motions [Shum & Szeliski, ICCV'98]

Figure 3: Deghosting a mosaic with motion parallax: (a) with parallax; (b) after single deghosting step (patch size 32); (c) multiple steps (sizes 32, 16 and 8).

Local alignment (deghosting)

Use local optic flow to compensate for radial distortion [Shum & Szeliski, ICCV'98]

Figure 4: Deghosting a mosaic with optical distortion: (a) with distortion; (b) after multiple steps.

Region-based de-ghosting

Select only one image in *regions-of-difference* using weighted vertex cover [Uyttendaele *et al.*, CVPR'01]

(A) (B)

Figure 5 – (A) Ghosted mosaic. (B) Result of de-ghosting algorithm.

Region-based de-ghosting

Select only one image in *regions-of-difference* using weighted vertex cover [Uyttendaele *et al.*, CVPR'01]

(A) (B)

Figure 6 – (A) Ghosted mosaic. (B) Result of de-ghosting algorithm.

Cutout-based de-ghosting

- Select only one image per output pixel, using spatial continuity
- Blend across seams using gradient continuity ("Poisson blending")

[Agarwala *et al.*, SG'2004]

Cutout-based compositing

Photomontage [Agarwala *et al.*, SG'2004]

- Interactively blend *different* images:
  group portraits

Figure 1 From a set of five source images (of which four are shown on the left), we quickly create a composite family portrait in which everyone is smiling and looking at the camera (right). We simply flip through the stack and coarsely draw strokes using the *designated source* image objective over the people we wish to add to the composite. The user-applied strokes and computed regions are color-coded by the borders of the source images on the left (middle).

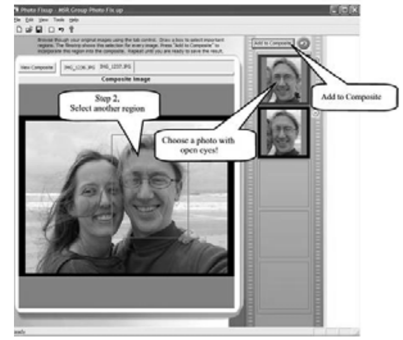
PhotoMontage

Technical details:

- use Graph Cuts to optimize seam placement

Demo:

- GroupShot application



Cutout-based compositing

Photomontage [Agarwala *et al.*, SG'2004]

- Interactively blend *different* images:
  focus settings

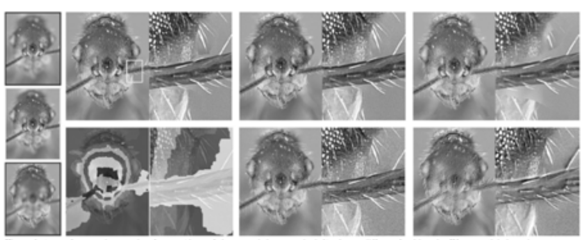


Figure 2 A set of macro photographs of an ant (three of eleven used shown on the left) taken at different focal lengths. We use a global *maximum contrast* image objective to compute the graph-cut composite automatically (top left, with an inset to show detail, and the labeling shown directly below). A small number of remaining artifacts disappear after gradient-domain fusion (top, middle). For comparison we show composites made by Auto-Montage (top, right), by Haeberli's method (bottom, middle), and by Laplacian pyramids (bottom, right). All of these other approaches have artifacts; Haeberli's method creates excessive noise, Auto-Montage fails to attach some hairs to the body, and Laplacian pyramids create halos around some of the hairs.

Cutout-based compositing

Photomontage [Agarwala *et al.*, SG'2004]

- Interactively blend *different* images:
  people's faces

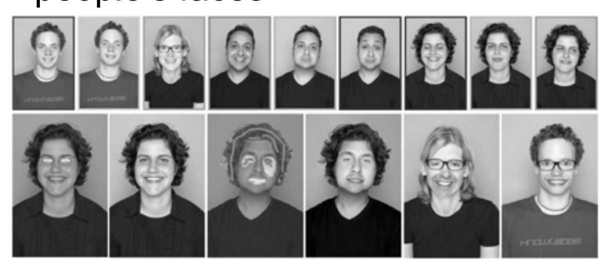


Figure 6 We use a set of portraits (first row) to mix and match facial features, to either improve a portrait, or create entirely new people. The faces are first hand-aligned, for example, to place all the noses in the same location. In the first two images in the second row, we replace the closed eyes of a portrait with the open eyes of another. The user paints strokes with the *designated source* objective to specify desired features. Next, we create a fictional person by combining three source portraits. Gradient-domain fusion is used to smooth out skin tone differences. Finally, we show two additional mixed portraits.

More stitching possibilities

- Video stitching
- High dynamic range image stitching
  - see demo…
- Flash + Non-Flash
- Video-based rendering

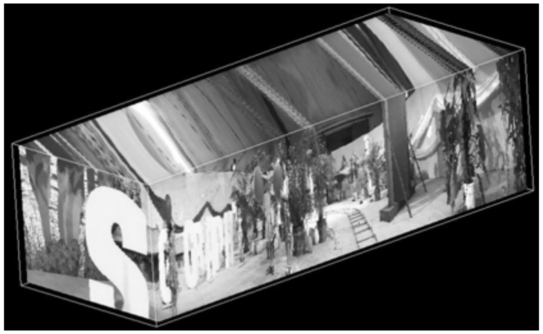
Next week's lecture:
Computational Photography

Other types of mosaics

Can mosaic onto *any* surface if you know the geometry

- See NASA's Visible Earth project for some stunning earth mosaics
  - http://earthobservatory.nasa.gov/Newsroom/BlueMarble/

Slit images

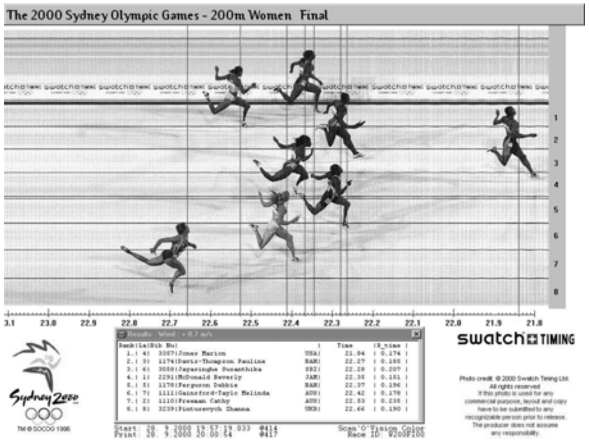


y-t slices of the video volume are known as *slit images*

- take a single column of pixels from each input image

Slit images: cyclographs

Slit images: photofinish



Final thought: What is a "panorama"?

Tracking a subject

Repeated (best) shots

Multiple exposures

"Infer" what photographer wants?