Image Alignment and Stitching

Computer Vision
CSE576, Spring 2005
Richard Szeliski

Today's lecture

Image alignment and stitching
- motion models
- cylindrical and spherical warping
- point-based alignment
- global alignment
- automated stitching (recognizing panoramas)
- ghost and parallax removal
- compositing and blending

Readings

- Szeliski & Shum, SIGGRAPH'97 (Sections 1-4).
- Szeliski, Image Alignment and Stitching, MSR-TR-2004-92 (Sections 2, 4, 5).
- Recognizing Panoramas, Brown & Lowe, ICCV'2003

Motion models

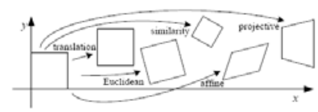
Motion models

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

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

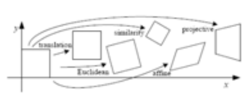
… see interactive demo (VideoMosaic)

Motion models

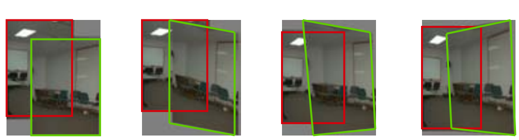


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\, I \mid t \,\right]_{2\times 3}$	2	orientation + ···	
rigid (Euclidean)	$\left[\, R \mid t \,\right]_{2\times 3}$	3	lengths + ···	
similarity	$\left[\, sR \mid t \,\right]_{2\times 3}$	4	angles + ···	
affine	$\left[\, A \,\right]_{2\times 3}$	6	parallelism + ···	
projective	$\left[\, \tilde{H} \,\right]_{3\times 3}$	8	straight lines	

Motion models



Translation **Affine** **Perspective** **3D rotation**



2 unknowns **6 unknowns** **8 unknowns** **3 unknowns**

Homographies

Perspective projection of a plane

- Lots of names for this:
  - **homography**, texture-map, colineation, planar projective map
- Modeled as a 2D warp using homogeneous coordinates

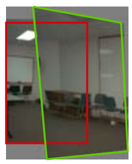
$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

To apply a homography **H** **p’** **H** **p**

- Compute **p’** = **Hp** (regular matrix multiply)
- Convert **p’** from homogeneous to image coordinates
  - divide by w (third) coordinate

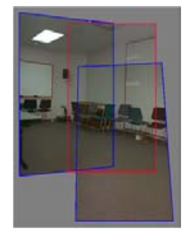
Plane perspective mosaics

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

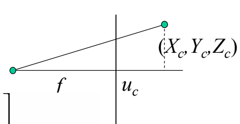


Rotational mosaics

- 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



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

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

3D Rotation Model

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

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

Rotations and quaternions

How do we represent rotation matrices?

1. Axis / angle ($\boldsymbol{n}$,θ)
   $\boldsymbol{R} = \boldsymbol{I} + \sin\theta\, [\boldsymbol{n}]_\times + (1 - \cos\theta)\, [\boldsymbol{n}]_\times^2$
   (Rodriguez Formula), with
   $[\boldsymbol{n}]_\times$ = cross product matrix (see paper)

Rotations and quaternions

How do we represent rotation matrices?

2. Unit quaternions [Shoemake SIGGRAPH'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
   $\boldsymbol{R} \leftarrow \boldsymbol{R}\, \Delta\boldsymbol{R}$

Image Mosaics (Stitching)

[Szeliski & Shum, SIGGRAPH'97]
[Szeliski, MSR-TR-2004-92]

Image Mosaics (Stitching)

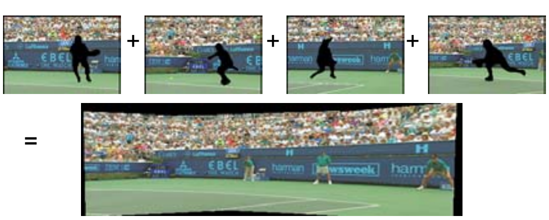
Full screen panoramas (cubic): http://www.panoramas.dk/
Mars: http://www.panoramas.dk/fullscreen3/f2_mars97.html
2003 New Years Eve: http://www.panoramas.dk/fullscreen3/f1.html

Image Mosaics (stitching)

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

Mosaics for Video Coding

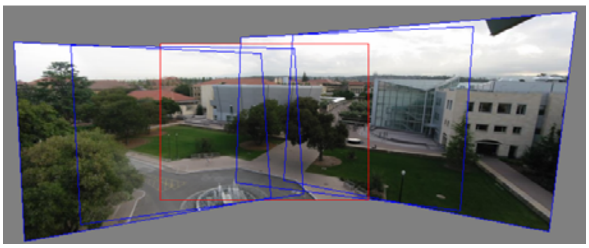
Convert masked images into a background sprite for content-based coding



Establishing correspondences

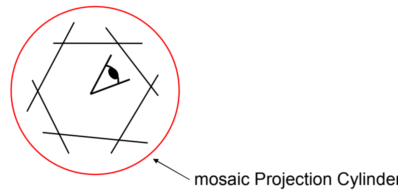
1. Direct method:
   - Use generalization of affine motion model [Szeliski & Shum '97]
2. Feature-based method
   - Compute feature-based correspondence [Lowe ICCV'99; Schmid ICCV'98, Brown&Lowe ICCV'2003]
   - Compute ***R*** from correspondences (absolute orientation)

Stitching demo

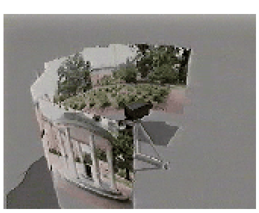


Panoramas

What if you want a 360° field of view?



Cylindrical panoramas



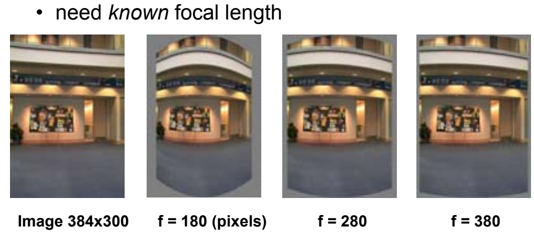
Steps

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

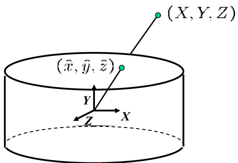
Cylindrical Panoramas

Map image to cylindrical or spherical coordinates

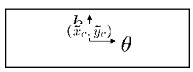
- need *known* focal length



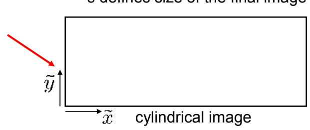
Cylindrical projection



unit cylinder



unwrapped cylinder

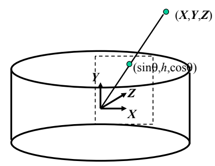


cylindrical image

- 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

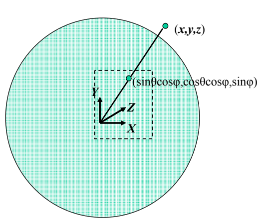
Given focal length *f* and image center (x_c,y_c)



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

Spherical warping

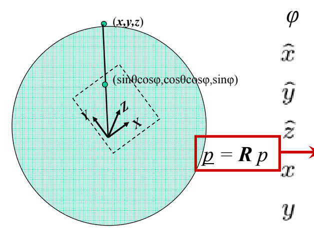
Given focal length *f* and image center (x_c,y_c)



$$\theta = (x_{sph} - x_c)/f$$
$$\varphi = (y_{sph} - y_c)/f$$
$$\hat{x} = \sin\theta\ \cos\varphi$$
$$\hat{y} = \sin\varphi$$
$$\hat{z} = \cos\theta\cos\varphi$$
$$x = f\hat{x}/\hat{z} + x_c$$
$$y = f\hat{y}/\hat{z} + y_c$$

3D rotation

Rotate image before placing on unrolled sphere



$$\theta = (x_{sph} - x_c)/f$$
$$\varphi = (y_{sph} - y_c)/f$$
$$\hat{x} = \sin\theta\ \cos\varphi$$
$$\hat{y} = \sin\varphi$$
$$\hat{z} = \cos\theta\cos\varphi$$
$$x = f\hat{x}/\hat{z} + x_c$$
$$y = f\hat{y}/\hat{z} + y_c$$

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

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

Inverse Warping

Get each pixel $\boldsymbol{I}_0(\boldsymbol{u}_0)$ from its corresponding location $\boldsymbol{u}_1 = \boldsymbol{h}(\boldsymbol{u}_0)$ in $\boldsymbol{I}_1(\boldsymbol{u}_1)$

- What if pixel comes from “between” two pixels?
- Answer in next lecture…

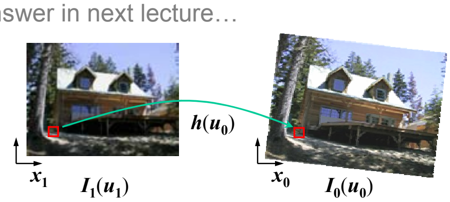
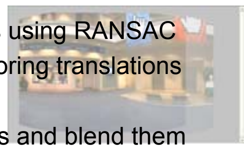


Image Stitching

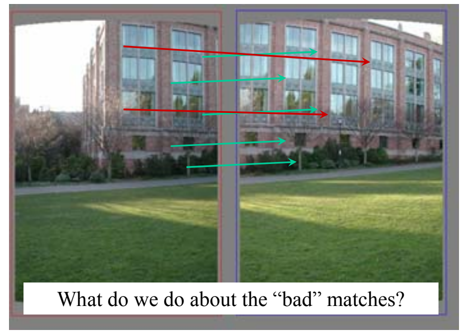
1. Align the images over each other
   - camera pan ↔ translation on cylinder!
2. Blend the images together (demo)

Project 2 – image stitching

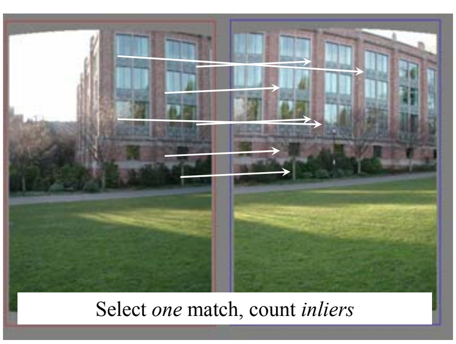


1. Take pictures on a tripod (or handheld)
2. Warp images to spherical coordinates
3. Extract features
4. Align neighboring pairs using RANSAC
5. Write out list of neighboring translations
6. Correct for drift
7. Read in warped images and blend them
8. Crop the result and import into a viewer

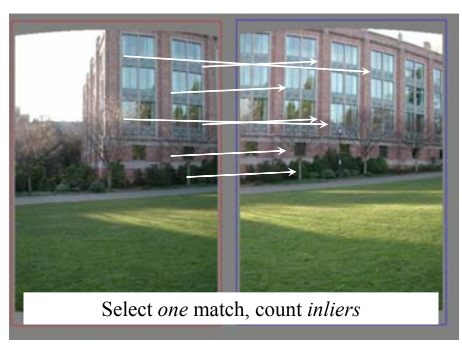
Matching features



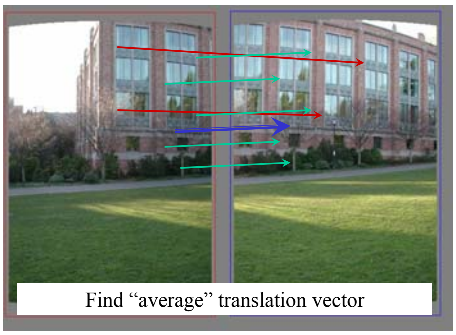
RAndom SAmple Consensus



RAndom SAmple Consensus

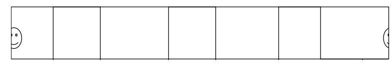


Least squares fit



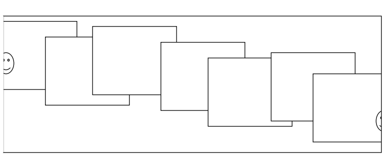
Find "average" translation vector

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.)

Full-view (360° spherical) panoramas

Full-view Panoramas

Global alignment

- Register *all* pairwise overlapping images
- Use a 3D rotation model (one R per image)
- Use feature based registration of *unwarped* images
- *Discover* which images overlap other images using feature selection (RANSAC)
- *Chain* together inter-frame rotations
- *Optimize* all R estimates together (next time)

3D Rotation Model

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

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

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

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

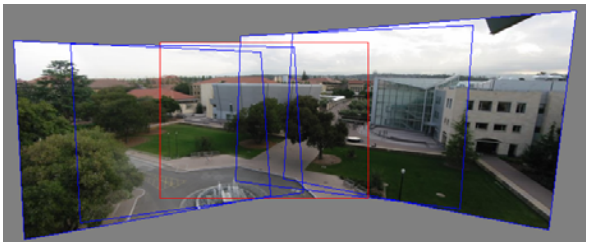
$p_i = (x_i,y_i,z_i) = (u_i-u_c,v_i-v_c,f)$

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

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

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

Stitching demo



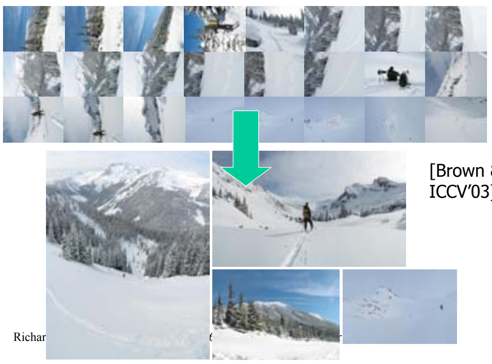
Texture Mapped Model (sphere)

Texture Mapped Model (cubical)

Recognizing Panoramas

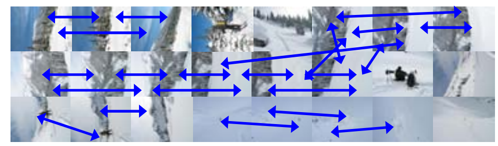
Matthew Brown & David Lowe
ICCV’2003

Recognizing Panoramas

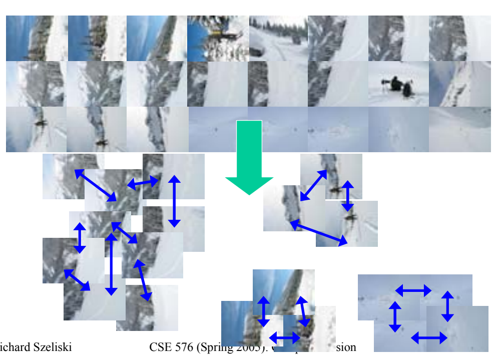


[Brown & Lowe, ICCV'03]

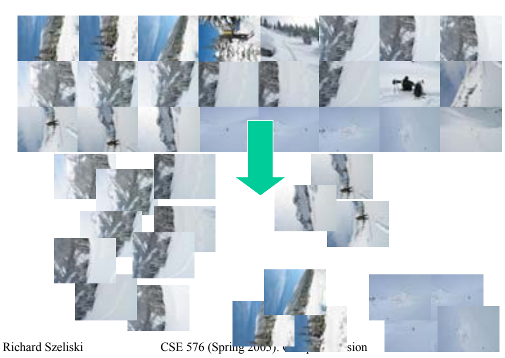
Finding the panoramas



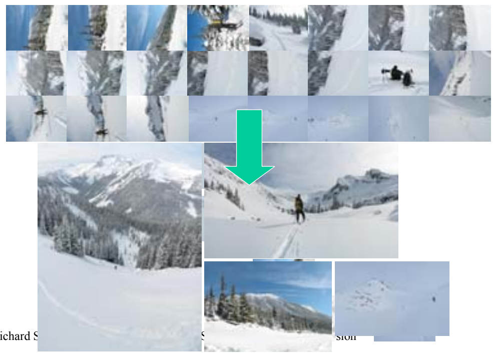
Finding the panoramas



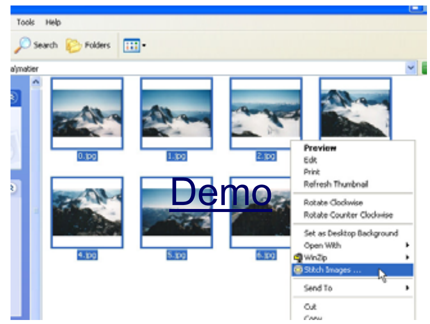
Finding the panoramas



Finding the panoramas



Fully automated 2D stitching



Get you own copy!

System components

Feature detection and description
- more uniform point density

Fast matching (hash table)
RANSAC filtering of matches
Intensity-based verification
Incremental bundle adjustment

[Brown, Szeliski, Winder, CVPR'05]

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

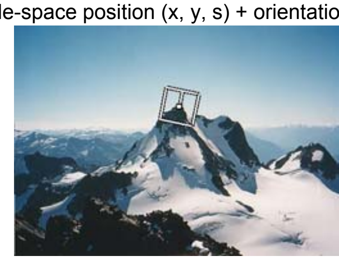
Distribute points evenly over the image

Descriptor Vector

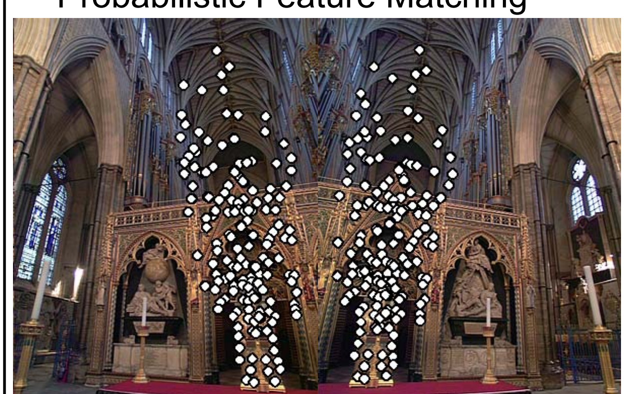
Orientation = blurred gradient

Similarity Invariant Frame

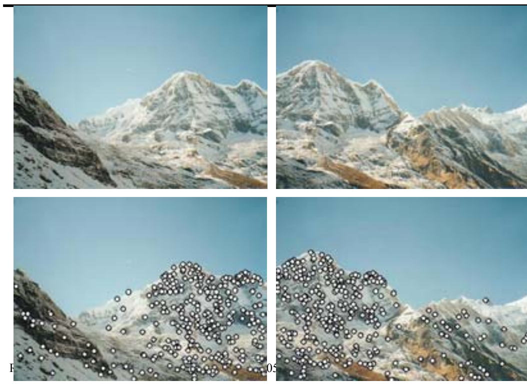
- Scale-space position (x, y, s) + orientation (θ)



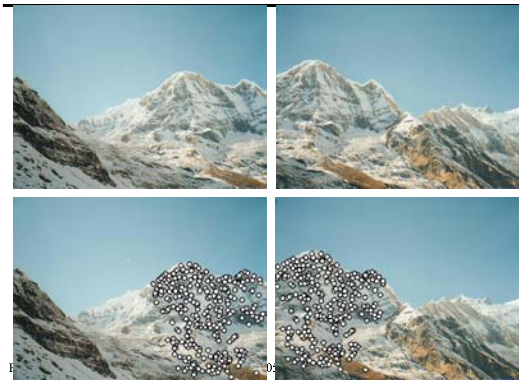
Probabilistic Feature Matching



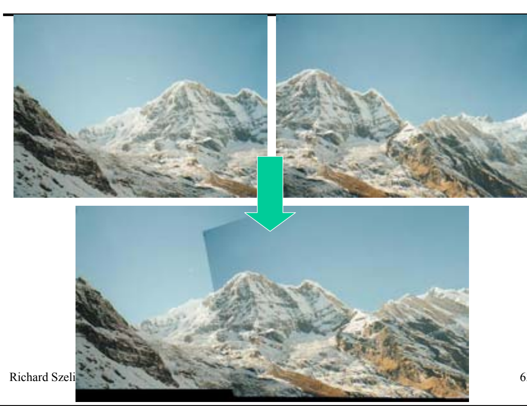
RANSAC motion model



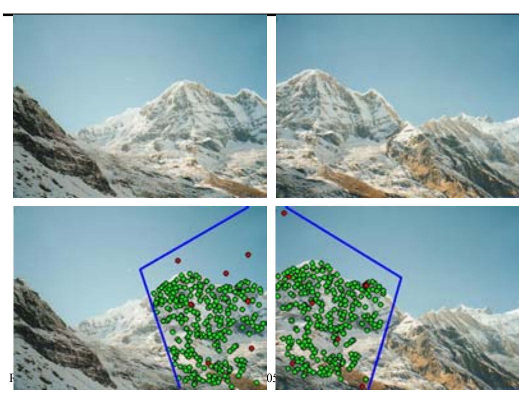
RANSAC motion model



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 consumer application

Matching Mistakes: False Positive

Matching Mistakes: False Positive

Matching Mistakes: 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)

Deghosting and blending

(optional material)

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.

Image feathering

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

Cut out the appropriate region from each image and then blend together

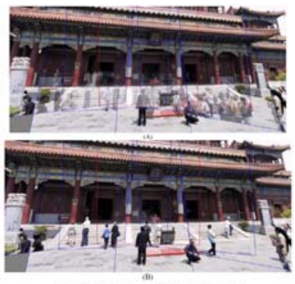
Region-based de-ghosting

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

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]

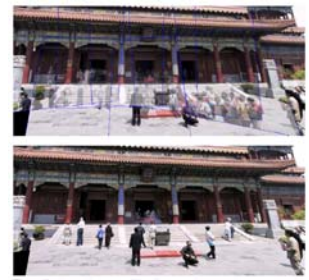


Cutout-based de-ghosting

•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).

Cutout-based compositing

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

- Interactively blend *different* images:
  focus settings

Cutout-based compositing

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

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

Final thought: What is a "panorama"?

Tracking a subject

Repeated (best) shots

Multiple exposures

"Infer" what photographer wants?