Image Stitching

Computer Vision
CSE P 576, Spring 2011

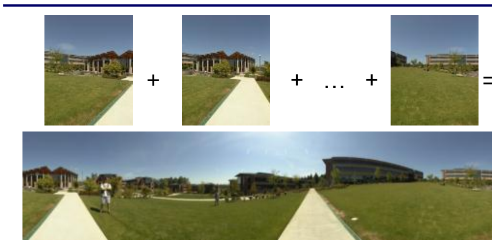
Richard Szeliski
Microsoft Research

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

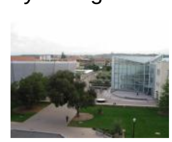
- Szeliski, *CVAA*:
  - Chapter 3.6: Image warping
  - Chapter 6.1: Feature-based alignment
  - Chapter 9.1: Motion models
  - Chapter 9.2: Global alignment
  - Chapter 9.3: Compositing
- Recognizing Panoramas, Brown & Lowe, ICCV'2003
- Szeliski & Shum, SIGGRAPH'97

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)

Projective transformations

(aka *homographies*)

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \qquad \begin{array}{l} x' = u/w \\ y' = v/w \end{array}$$

"keystone" distortions

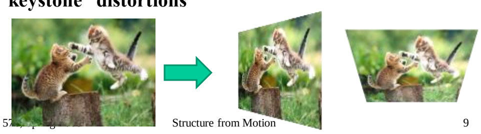


Image Warping

Image Warping

image filtering: change *range* of image

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

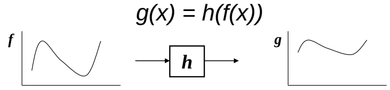


image warping: change *domain* of image

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

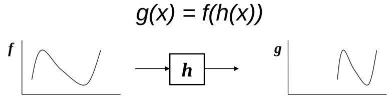


Image Warping

image filtering: change *range* of image

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

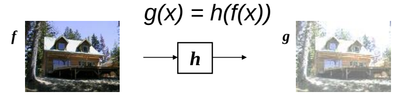
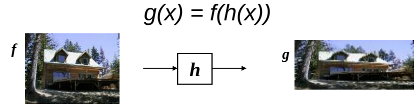


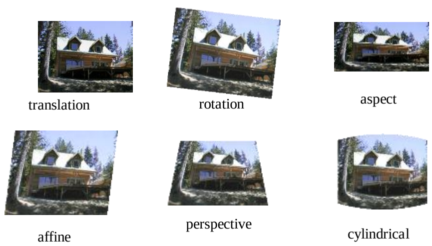
image warping: change *domain* of image

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



Parametric (global) warping

Examples of parametric warps:



2D coordinate transformations

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

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

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

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

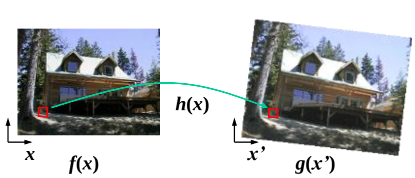
perspective: $\underline{\boldsymbol{x}}' \cong \boldsymbol{H}\,\underline{\boldsymbol{x}}$ $\underline{\boldsymbol{x}} = (x,y,1)$

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

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

Image Warping

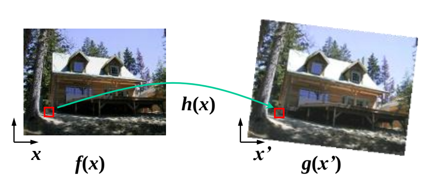
Given a coordinate transform $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$ and a source image $\boldsymbol{f}(\boldsymbol{x})$, how do we compute a transformed image $\boldsymbol{g}(\boldsymbol{x}') = \boldsymbol{f}(\boldsymbol{h}(\boldsymbol{x}))$?



Forward Warping

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

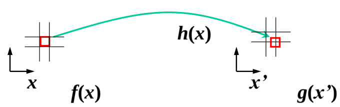
- What if pixel lands "between" two pixels?



Forward Warping

Send each pixel ***f***(***x***) to its corresponding location ***x'*** = ***h***(***x***) in ***g***(***x'***)

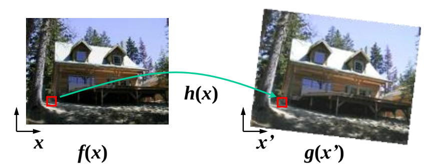
- What if pixel lands "between" two pixels?
- Answer: add "contribution" to several pixels, normalize later (*splatting*)



Inverse Warping

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

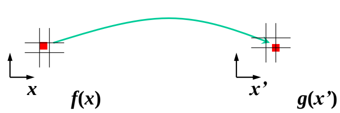
- What if pixel comes from "between" two pixels?



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

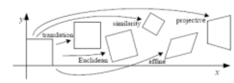
Possible interpolation filters:

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

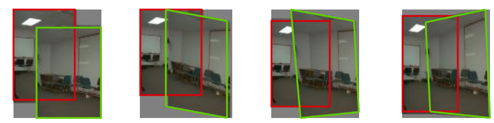
Needed to prevent "jaggies" and "texture crawl" (see demo)

Motion models (reprise)

Motion models



Translation	Affine	Perspective	3D rotation



2 unknowns	6 unknowns	8 unknowns	3 unknowns

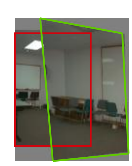
Finding the transformation

Translation	=	2 degrees of freedom
Similarity	=	4 degrees of freedom
Affine	=	6 degrees of freedom
Homography	=	8 degrees of freedom

How many corresponding points do we need to solve?

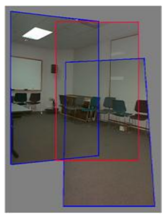
Plane perspective mosaics

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

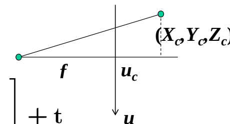


Rotational mosaics

- 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



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

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

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 Mosaics (Stitching)

[Szeliski & Shum, SIGGRAPH'97]

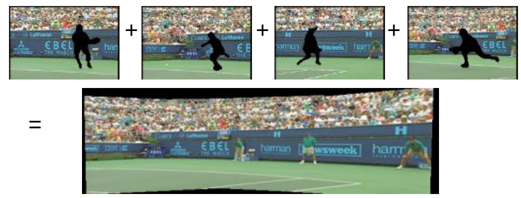
[Szeliski, FnT CVCG, 2006]

Image Mosaics (stitching)

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

Mosaics for Video Coding

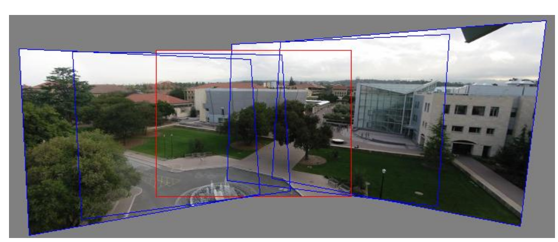
Convert masked images into a background sprite for content-based coding



Establishing correspondences

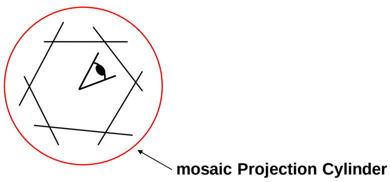
1. Direct method:
   - Use generalization of affine motion model [Szeliski & Shum ’97]
2. Feature-based method
   - Extract features, match, find consisten *inliers* [Lowe ICCV’99; Schmid ICCV’98, Brown&Lowe ICCV’2003]
   - Compute ***R*** from correspondences (absolute orientation)

Stitching demo

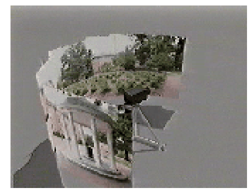


Panoramas

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

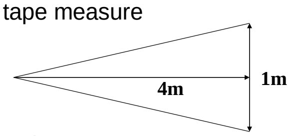
Map image to cylindrical or spherical coordinates

- need *known* focal length

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

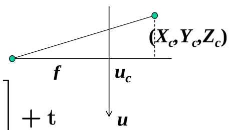
Determining the focal length

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



4. Ask your instructor

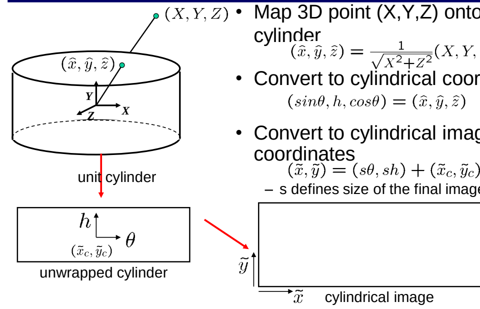
3D → 2D Perspective Projection



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

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

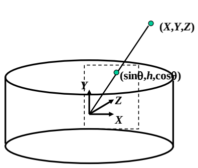
Cylindrical projection



- 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

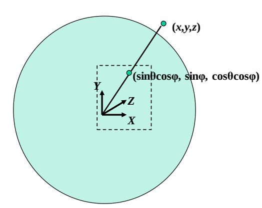
Given focal length *f* and image center (x_c,y_c)



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

Spherical warping

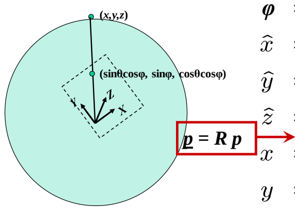
Given focal length *f* and image center (x_c,y_c)



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

3D rotation

Rotate image before placing on unrolled sphere



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

Radial distortion

Correct for "bending" in wide field of view lenses

Project $(\hat{x}, \hat{y}, \hat{z})$ to "normalized" image coordinates

$$
\begin{aligned}
x'_n &= \hat{x}/\hat{z} \\
y'_n &= \hat{y}/\hat{z}
\end{aligned}
$$

Apply radial distortion

$$
\begin{aligned}
r^2 &= x'^2_n + y'^2_n \\
x'_d &= x'_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \\
y'_d &= y'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)
\end{aligned}
$$

Apply focal length translate image center

$$
\begin{aligned}
x' &= f x'_d + x_c \\
y' &= f y'_d + y_c
\end{aligned}
$$

To model lens distortion

- Use above projection operation instead of standard projection matrix multiplication

Fisheye lens

Extreme "bending" in ultra-wide fields of view

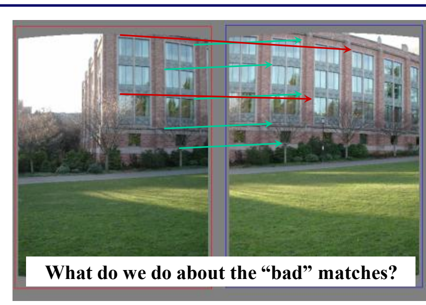
$$\hat{r}^2 = \hat{x}^2 + \hat{y}^2$$

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

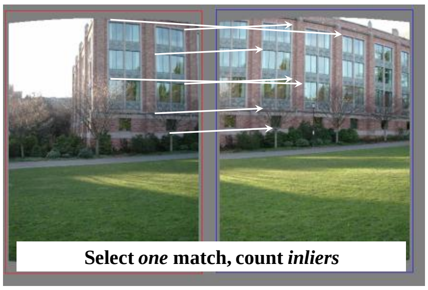
uations become

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

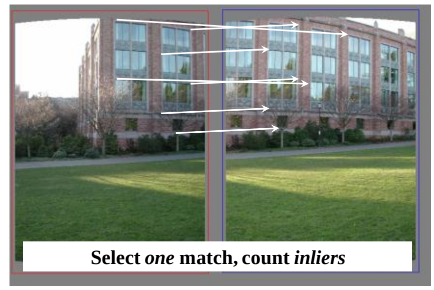
Matching features



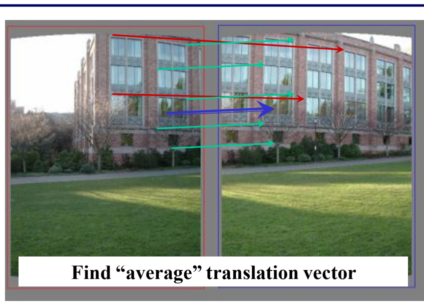
RAndom SAmple Consensus



RAndom SAmple Consensus



Least squares fit



RANSAC for estimating homography

RANSAC loop:

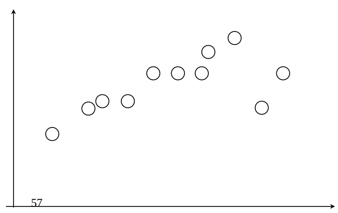
1. Select four feature pairs (at random)
2. Compute homography $\boldsymbol{H}$ (exact)
3. Compute inliers where $\|p_i', \boldsymbol{H}\, p_i\| < \varepsilon$

Keep largest set of inliers

Re-compute least-squares $\boldsymbol{H}$ estimate using all of the inliers

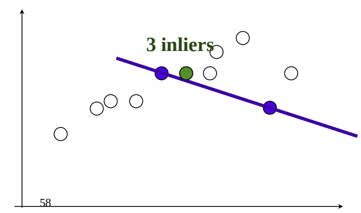
Simple example: fit a line

Rather than homography H (8 numbers)
fit y=ax+b (2 numbers a, b) to 2D pairs



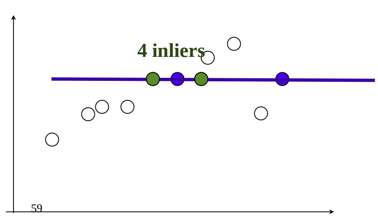
Simple example: fit a line

Pick 2 points
Fit line
Count inliers



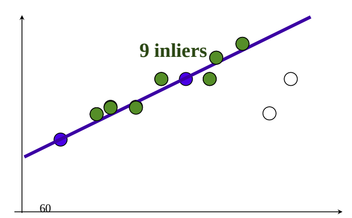
Simple example: fit a line

Pick 2 points
Fit line
Count inliers



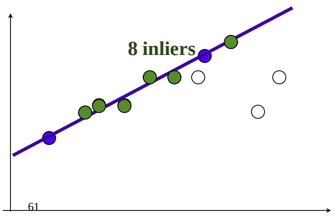
Simple example: fit a line

Pick 2 points
Fit line
Count inliers



Simple example: fit a line

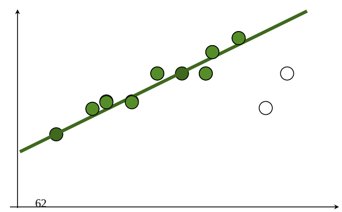
Pick 2 points
Fit line
Count inliers



61

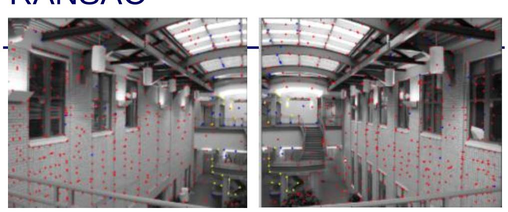
Simple example: fit a line

Use biggest set of inliers
Do least-square fit



62

RANSAC



red:
rejected by 2nd nearest neighbor criterion
blue:
Ransac outliers
yellow:
inliers

Image Stitching—review

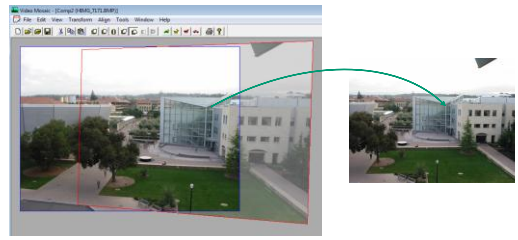
1. Align the images over each other
   - camera pan ↔ translation on cylinder
2. Blend the images together (demo)

Richard Szeliski Image Stitching 66

Assignment 2 – Creating Panoramas

1. Implement Harris corner detector
2. Implement MatchInterestPoints
3. Compute homography using RANSAC
4. Compute size of stitched images from projected corners
5. Inverse sample image and average

Resampling the final image



Full-view (360° spherical) panoramas

Full-view Panorama

Texture Mapped Model

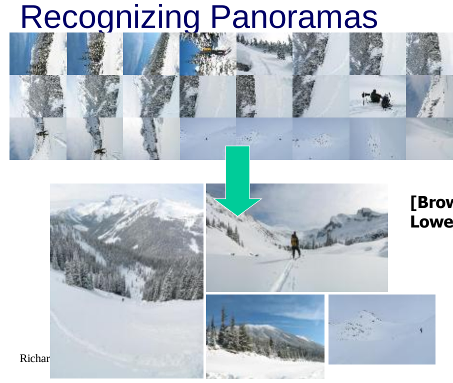
Global alignment

- 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

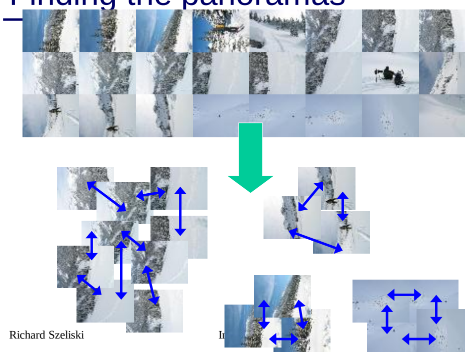
Recognizing Panoramas



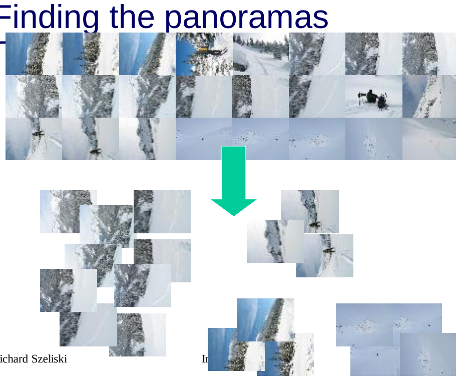
[Brown & Lowe, ICCV'03]

Finding the panoramas

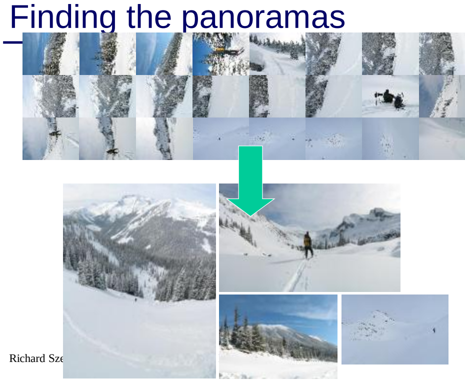
Finding the panoramas



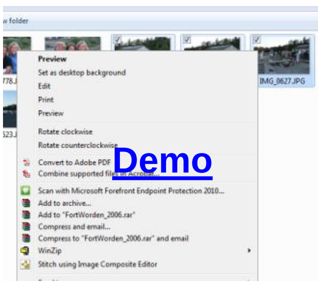
Finding the panoramas



Finding the panoramas



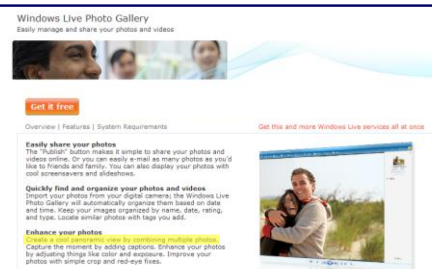
Fully automated 2D stitching



Demo

Richard Szeliski Image Stitching 82

Get you own copy



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

Richard Szeliski Image Stitching 83

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]

Richard Szeliski Image Stitching 84

Multi-Scale Oriented Patches

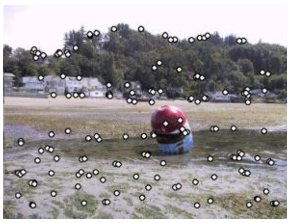
Interest points
- Multi-scale Harris corners
- Orientation from blurred gradient
- Geometrically invariant to similarity transforms

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

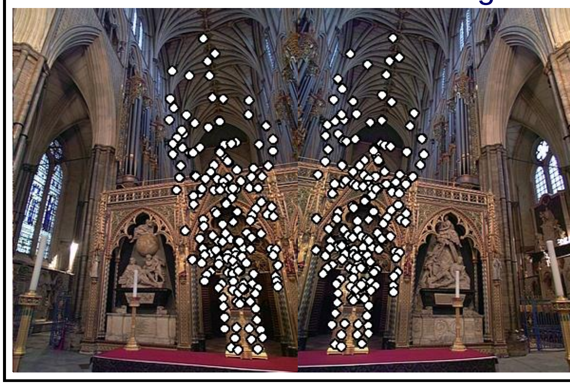
Richard Szeliski Image Stitching 85

Feature irregularities

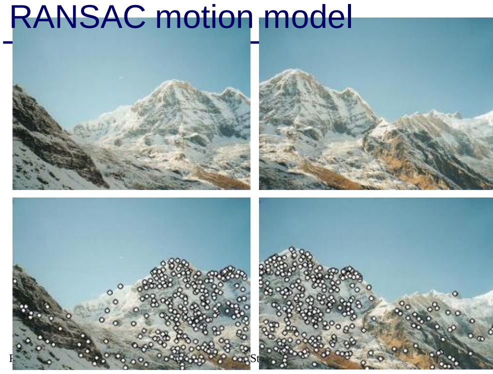
Distribute points evenly over the image



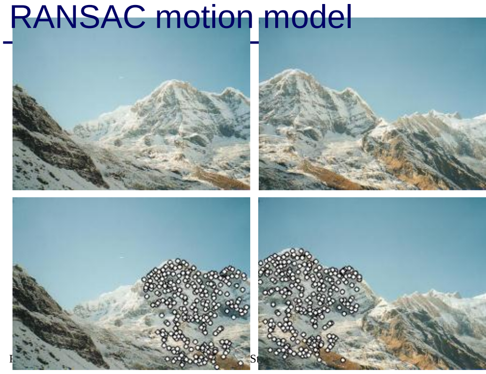
Probabilistic Feature Matching



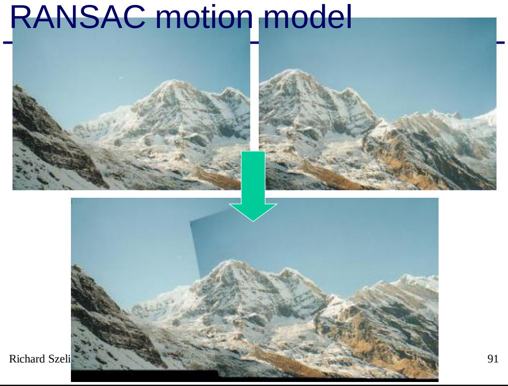
RANSAC motion model



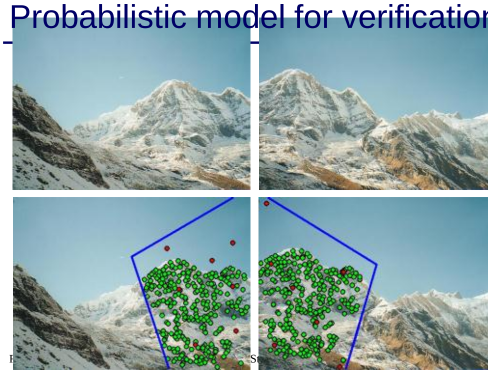
RANSAC motion model



RANSAC motion model



Probabilistic model for verification



How well does this work?

Test on 100s of examples…

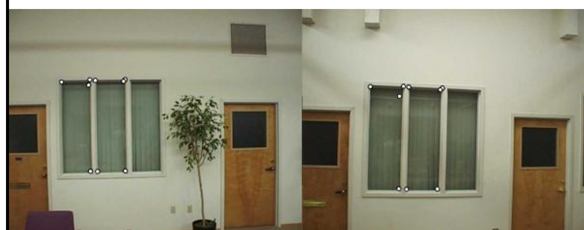
How well does this work?

Test on 100s of examples…

…still too many failures (5-10%)
for consumer application

Matching Mistakes: False Positive

Matching Mistakes: False Positive



Matching 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

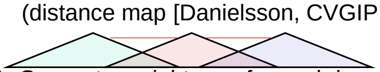
→ Sounds like a big “learning” problem

- 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: $w_i' = w_i / (\sum_i w_i)$

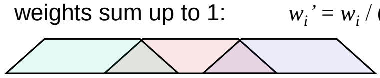
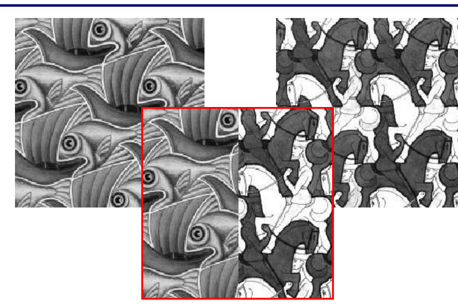
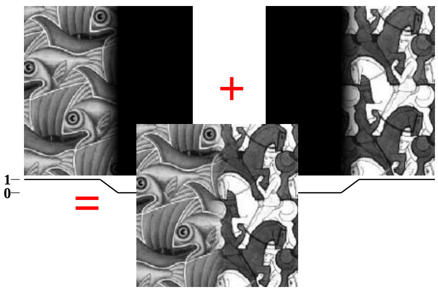


Image Feathering

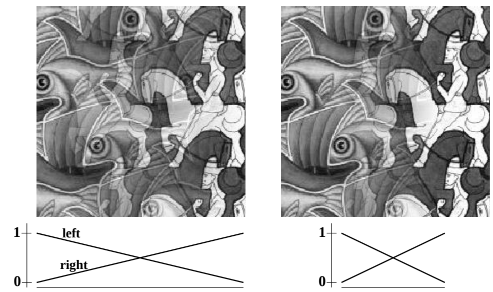


Feathering



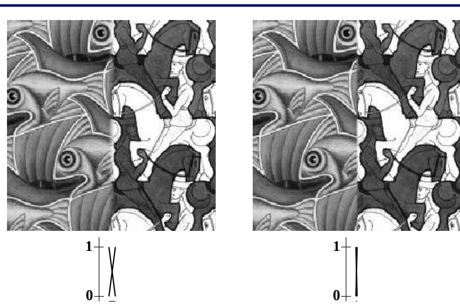
Richard Szeliski Image Stitching 103

Effect of window size



Richard Szeliski Image Stitching 104

Effect of window size



Richard Szeliski Image Stitching 105

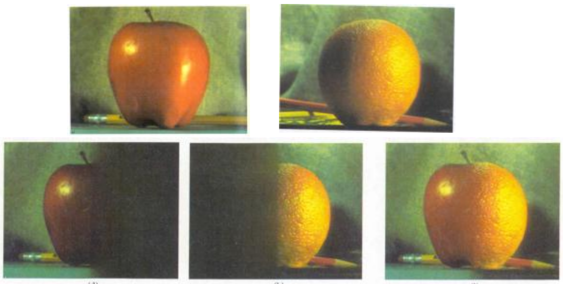
Good window size

"Optimal" window: smooth but not ghosted

- **Doesn't always work...**

Richard Szeliski Image Stitching 106

Pyramid Blending

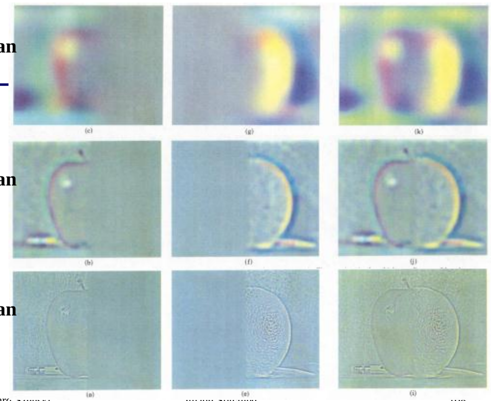


Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.

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

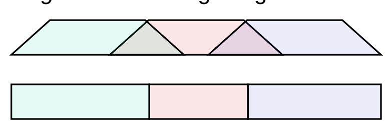
Q: How do we compute the original weights?

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
- see Computational Photography lecture

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.

Richard Szeliski Image Stitching 115

Region-based de-ghosting

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

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

Richard Szeliski Image Stitching 116

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]

Richard Szeliski Image Stitching 117

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).

Richard Szeliski Image Stitching 118

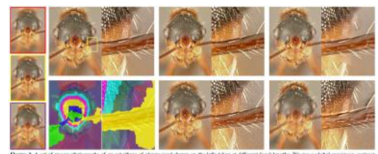
PhotoMontage

Technical details:
- use Graph Cuts to optimize seam placement

Demo:
- Windows Live Photo Gallery *Photo Fuse*

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

More stitching possibilities

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

Next-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/

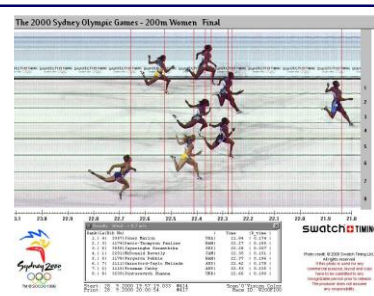
Slit images

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

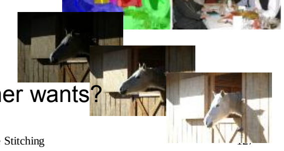


Final thought: What is a "panorama"?

Tracking a subject

Repeated (best) shots

Multiple exposures



"Infer" what photographer wants?