Image Stitching

Computer Vision
CSE 576, Spring 2008
Richard Szeliski
Microsoft Research

Panoramic Image Mosaics
Full screen panoramas (cubic): http://www.panoramas.dk/

Gigapixel panoramas & images

Mapping / Tourism / WWT
Medical Imaging

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

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 filtering: change \textit{range} of image
\[ g(x) = h(f(x)) \]

image warping: change \textit{domain} of image
\[ g(x) = f(h(x)) \]

Parametric (global) warping

Examples of parametric warps:
- translation
- rotation
- aspect
- affine
- perspective
- cylindrical
2D coordinate transformations

translation: \[ x' = x + t \]
\[ x = (x,y) \]

rotation: \[ x' = Rx + t \]

similarity: \[ x' = sR x + t \]

affine: \[ x' = Ax + t \]

perspective: \[ x' = H x \]
\[ x = (x,y,1) \]

(x is a homogeneous coordinate)

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

Image Warping

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

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?

- 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 downampling (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

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

```
\[
\begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix} =
[ R ]_{3 \times 3}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} + t
\]
```

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

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
   \[(x_1,y_1,z_1) = R_{0f}(x_0,y_0,z_0)\]
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 \((n, \theta)\)
   \[
   R = I + \sin \theta [n]_x + (1 - \cos \theta) [n]_x^2
   \]
   (Rodriguez Formula), with \([n]_x\) = cross product matrix (see paper)
2. Unit quaternions [Shoemake SIGG’85]
   \[
   q = (n \sin \theta/2, \cos \theta/2) = (w, s)
   \]
   quaternion multiplication (division is easy)
   \[
   q_0 q_1 = (s_1 w_0 + s_0 w_1, s_0 s_1 w_0 - w_1)
   \]

Incremental rotation update

1. Small angle approximation
   \[
   \Delta R = I + \sin \theta [n]_x + (1 - \cos \theta) [n]_x^2
   \approx \theta [n]_x = [\omega]_x
   \]
   linear in \(\omega\)
2. Update original \(R\) matrix
   \[
   R \leftarrow R \Delta R
   \]

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]

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 consistent 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 $R$

$$p_i' = Rp_i$$

3D rays

$$A = \Sigma_i p_i p_i'^T = \Sigma_i p_i p_i^T R^T = USV^T = (USU^T)R^T$$

$$V^T = U^TR^T$$

$$R = VU^T$$

Panoramas

What if you want a $360^\circ$ 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

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

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

Cylindrical projection

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

unit cylinder

unwrapped cylinder
Cylindrical warping

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

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

Spherical warping

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

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

3D rotation

Rotate image before placing on unrolled sphere

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

Radial distortion

Correct for “bending” in wide field of view lenses

\[
\begin{align*}
\begin{align*}
\theta &= \frac{x_{cyl} - x_c}{f} \\
\varphi &= \frac{y_{cyl} - y_c}{f} \\
\hat{x} &= \sin \theta \cos \varphi \\
\hat{y} &= \sin \varphi \\
\hat{z} &= \cos \theta \cos \varphi \\
x &= \frac{f \hat{x}}{\hat{z}} + x_c \\
y &= \frac{f \hat{y}}{\hat{z}} + y_c
\end{align*}
\end{align*}
\]

Project \((\hat{x}, \hat{y}, \hat{z})\) to “normalized” image coordinates

\[
\begin{align*}
x_n' &= \frac{\hat{x}}{\hat{z}} \\
y_n' &= \frac{\hat{y}}{\hat{z}}
\end{align*}
\]

Apply radial distortion

\[
\begin{align*}
x_d' &= x_n'^2 + y_n'^2 \left(1 + \kappa_1 r^2 + \kappa_2 r^4 \right) \\
y_d' &= y_n'^2 \left(1 + \kappa_1 r^2 + \kappa_2 r^4 \right)
\end{align*}
\]

Apply focal length translate image center

\[
\begin{align*}
x' &= f x_d' + x_c \\
y' &= f y_d' + y_c
\end{align*}
\]

To model lens distortion

- Use above projection operation instead of standard projection matrix multiplication
Fisheye lens

Extreme “bending” in ultra-wide fields of view

\[ \hat{r}^2 = \hat{x}^2 + \hat{y}^2 \]

\[
\begin{align*}
\cos \theta \sin \phi, \sin \theta \sin \phi, \cos \phi &= s \ (x, y, z) \\
\text{Transformations become} \\
x' &= s \phi \cos \theta = s \frac{x}{r} \tan^{-1} \frac{r}{z}, \\
y' &= s \phi \sin \theta = s \frac{y}{r} \tan^{-1} \frac{r}{z}.
\end{align*}
\]

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

What do we do about the “bad” matches?
**RAAndom SAample 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.)

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
  - compute a global warp: \( y' = y + ax \)
  - run a big optimization problem, incorporating this

Full-view Panorama

Full-view (360° spherical) panoramas
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
Fully automated 2D stitching

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


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 \((x, y, s) + \text{orientation } (\theta)\)

Probabilistic Feature Matching

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 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:
   \[ w'_i = \frac{w_i}{\sum_i w_i} \]
Image Feathering

Effect of window size

Feathering

Effect of window size
Good window size

“Optimal” window: smooth but not ghosted
  • Doesn’t always work...

Pyramid Blending


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 $L_\infty$ norm: $(p = 10)$

$$w_i' = \left( \frac{w_ip}{\sum_i w_ip} \right)^{1/p}$$

Poisson Image Editing

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

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]

Region-based de-ghosting

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

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]

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

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

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 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?