Image Stitching
Image Stitching

1. Align the images over each other
   – camera pan ↔ translation on cylinder
2. Blend the images together
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?
**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 constraint
      – best solution, but more complicated
      – known as “bundle adjustment”
Full-view (360° spherical) panoramas
Full-view Panorama
Global alignment

- Pairwise matching $\rightarrow$ global energy function that involves all of the per-image pose parameters
- Local adjustments to reduce blurring due to local mis-registration
- Panorama recognition
Bundle Adjustment

• Developed in photogrammetry in 50’s, for general structure from motion (Szeliski and Kang 1994)

• Pairwise alignment

\[ E_{\text{pairwise-LS}} = \sum_i \| r_i \|^2 = \| \tilde{x}_i'(x_i; p) - \hat{x}_i' \|^2 \]
Bundle Adjustment

• N features, appears in M images

\[ \tilde{x}_{ij} \sim K_j R_j x_i \]
\[ x_i \sim R_j^{-1} K_j^{-1} \tilde{x}_{ij} \]

\[ \tilde{x}_{ik} \sim \tilde{H}_{kj} \tilde{x}_{ij} = K_k R_k R_j^{-1} K_j^{-1} \tilde{x}_{ij} \]
Bundle Adjustment

- Given initial set of \{R_j, K_j\}, refine them together

\[ E_{\text{all-pairs-2D}} = \sum_i \sum_{jk} c_{ij} c_{ik} \| \hat{x}_{ik}(\hat{x}_{ij}; R_j, f_j, R_k, f_k) - \hat{x}_{ik} \|^2 \]

- \( X_{ik} \) depends on \( x_{ij} \), error-in-variable

- Overweighted for feature observed many times

- Predicted location of feature I in image k

- Observed location of feature I in image k
Bundle Adjustment

• True bundle adjustment to estimate camera pose and 3d points

\[ E_{BA-2D} = \sum_i \sum_j c_{ij} \| \tilde{x}_{ij}(x_i; R_j, f_j) - \hat{x}_{ij} \|^2 \]

• Disadvantages
  – A lot of variables to solve
  – Slow convergence
Bundle Adjustment

• Sparse matrix techniques
• Levenberg-Marquardt

http://www.ics.forth.gr/~lourakis/sba/
Non-linear least squares

• **Linear** approximation of residual $e_0 - J\Delta$
• allows **quadratic** approximation of sum-of-squares $(e_0 - J\Delta)^T(e_0 - J\Delta)$

Minimization corresponds to finding zeros of derivative

$$2J^TJ\Delta - 2J^Te_0 = 0$$

$$\Rightarrow \Delta = (J^TJ)^{-1}J^Te_0$$

Levenberg-Marquardt: extra term to deal with singular $N$
(decrease/increase $\lambda$ if success/failure to descent)

(extra term = descent term)

$$N' = J^TJ + \lambda\text{diag}(J^TJ)$$
Bundle Adjustment

- Jacobian of \( \sum_{i=1}^{m} \sum_{j=1}^{n} D(m_{ij}, \hat{P}_i(\hat{M}_j))^2 \) has sparse block structure
  - cameras independent of other cameras,
  - points independent of other points

\[
J = \begin{bmatrix}
\text{im.pts. view 1} & P_1 & P_2 & P_3 & M
\end{bmatrix}
\]

\[
N = J^T J = \begin{bmatrix}
U_1 & U_2 & U_3 & W & W^T & V
\end{bmatrix}
\]

Needed for non-linear minimization
Bundle Adjustment

Eliminate dependence of camera/motion parameters on structure parameters

Note in general $3n >> 11m$

$$\begin{bmatrix} I & -WV^{-1} \\ 0 & I \end{bmatrix} \times N = \begin{bmatrix} U-WV^{-1}W^T \\ \vdots \end{bmatrix}$$

Allows much more efficient computations

e.g. 100 views, 10000 points, solve $\pm1000\times1000$, not $\pm30000\times30000$

Often still band diagonal

use sparse linear algebra algorithms
Up Vector Selection

- Keep the upright direction at the final stitched image

\[ \tilde{x}_{ik} \sim K_k R_k x_i \]

- Multiply each rotation by a global rotation \( R_g \)

\[ \hat{i} = (1, 0, 0) \quad \hat{j} = (0, 1, 0) \quad \hat{i}^T R_k R_g \hat{j} = 0 \]

\[ r_{k0} = \hat{i}^T R_k \quad r_{g1} = R_g \hat{j} \]

\[ r_{g1} = \arg \min_r \sum_k (r^T r_{k0})^2 = \arg \min_r r^T \left[ \sum_k r_{k0} r_{k0}^T \right] r. \]
Up Vector Selection

\[ r_{g1} = \arg \min_r \sum_k (r^T r_{k0})^2 = \arg \min_r r^T \left[ \sum_k r_{k0} r_{k0}^T \right] r. \]

- \( r_{g1} \) is the smallest eigenvector of the scatter or moment matrix spanned by the individual camera rotation x-vectors
- Full rotation matrix \( R_g \)

1. \( r_{g1} = \min \) eigenvector \( (\sum_k r_{k0} r_{k0}^T) \)
2. \( r_{g0} = \mathcal{N}((\sum_k r_{k2}) \times r_{g1}); \)
3. \( r_{g2} = r_{g0} \times r_{g1}, \)
Parallax Removal

• Blurry or ghosting
  – Unmodeled radial distortion
  – 3D parallax: failure to rotate the camera around its optical center
  – Small scene motion, large scale scene motion

• Different approaches for them
Parallax Removal

• Based on bundle adjustment, compute 3D point location, then reproject it to images

\[
\bar{x}_i \sim \sum_j c_{ij} \bar{x}_i(\hat{x}_{ij}; R_j, f_j) / \sum_j c_{ij} \quad u_{ij} = \bar{x}_{ij} - x_{ij}
\]

Deghosting a mosaic with motion parallax (Shum and Szeliski 2000) c 2000 IEEE: (a) composite with parallax; (b) after a single deghosting step (patch size 32); (c) after multiple steps (sizes 32, 16 and 8)
Recognizing Panoramas

Matthew Brown & David Lowe

ICCV’2003
Recognizing Panoramas

[Brown & Lowe, ICCV’03]
Finding the panoramas

• Find all pairwise image overlaps
• Find connected components in the overlap graph
Finding the panoramas
Finding the panoramas
Get you own free copy

Windows Live Photo Gallery
Easily manage and share your photos and videos

Get it free

Overview | Features | System Requirements

Easily share your photos
The "Publish" button makes it simple to share your photos and videos online. Or you can easily e-mail as many photos as you'd like to friends and family. You can also display your photos with cool screensavers and slideshows.

Quickly find and organize your photos and videos
Import your photos from your digital camera; the Windows Live Photo Gallery will automatically organize them based on date and time. Keep your images organized by name, date, rating, and type. Locate similar photos with tags you add.

Enhance your photos
Create a cool panoramic view by combining multiple photos. Capture the moment by adding captions. Enhance your photos by adjusting things like color and exposure. Improve your photos with simple crop and red-eye fixes.

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

Probabilistic Feature Matching
RANSAC motion model
RANSAC motion model
RANSAC motion model
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)
Direct vs. Feature-based Alignment

• Feature-based
  – Confused in too textural or not textured enough region
  – Features should be distributed evenly over the image
  – Relies on robust feature detection and matching schemes

• Direct
  – Limited range of convergence, even hierarchical framework
  – Work for video, fail for partially overlapping images
Image Compositing

• How to produce the final mosaic image?
  – Compositing surface (flat, cylindrical, spherical, etc)
  – View (reference image)
  – Select and blend pixels to minimize visible seams, blur, and ghosting

• Surface parameterization, pixel and seam selection, blending, exposure compensation...
Compositing Surface

- If only a few images --- flat panorama
  - select one image as reference and warp all the others to the reference coordinate system
  - Still perspective projection
Compositing Surface

• Large field of view
  – Large distortion
  – Usually use a cylindrical or spherical projection
  – Environment map

• Tradeoff between
  – keeping local appearance undistorted, and
  – providing a reasonably uniform sampling of the environment
View Selection

• Which part of the scene will be centered in the final composition
  – Average depth
  – Middle image in a 360 panorama
  – Image that contains the object of greatest interest
  – User control
Coordinate Transformations

- Mapping between input and output pixels
  - Flat surface: homography
  - Analytic form (cylindrical, spherical...)
    • Pixel in panorama -> 3D view ray -> image
    • Precompute lookup tables
Sampling Issues

• Low-resolution panorama than input images
  – Prefilter input images to avoid aliasing
Pixel Selection and Weighting (De-ghosting)

- Without perfect registration and identically exposed
  - Visible seams due to exposure difference
  - Blurring due to mis-registration
  - Ghosting due to moving objects

- Which pixels to use? How to blend them?
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

![Diagram showing the effect of window size on image stitching.](image)
Effect of window size
Good window size

“Optimal” window: smooth but not ghosted
  • Doesn’t always work...
Feathering and Center-Weighting

Warped images

\[ C(x) = \frac{\sum_k w_k(x)\tilde{I}_k(x)}{\sum_k w_k(x)} \]

W_k: simple 0 or 1

Simple averaging: exposure difference, misregistration and scene movement are all visible

If rapidly moving objects are the only problem, taking median filter to remove
Feathering and Center-Weighting

• Weight on distance to image center
  \[ w_k(x) = \arg \min_y \|y\| \mid \tilde{I}_k(x + y) \text{ is in valid} \]

• Feathering: weighted averaging with a distance map

• Bluring, ghosting still exist
Feathering and Center-Weighting

• Improve feathering
  – Raise distance map value with larger power \( w_k^p(x) \)
    • Tradeoff between visible exposure difference and blur
  – \( p \to \infty \) \quad C(x) = \tilde{I}_{l(x)}(x), \quad l = \arg \max_k w_k(x)
    • Select pixel with largest weight
Feathering and Center-Weighting

- P=10
  - Have hard edges when exposure vary
- p=∞
Feathering and Center-Weighting

• Seam placement: optimal seam selection
• Moving object produce most visible artifacts: 
  ghost  Uyttendaele, Eden, and Szeliski (2001)
  – Region of difference where overlapping image disagree
  – Construct a graph
  – Weighted vertex cover: find the smallest set of vertexes without disagreement
Seam Selection

Computation of regions of difference (RODs) (Uyttendaele, Eden, and Szeliski 2001) c 2001 IEEE: (a) three overlapping images with a moving face; (b) corresponding RODs; (c) graph of coincident RODs.
ROD Seam Selection

Weighted ROD vertex cover with feathering
Photomontage
Agarwala, Dontcheva, Agrawala et al. 2004

- First, Per-pixel image objective to good composites

\[ C_D = \sum_x D(x, l(x)) \]

Data penalty \hspace{1cm} Chosing image \( l \) at pixel \( x \)
Photomontage

• Second, seam objective that penalizes differences in labeling between adjacent images
  \[ C_S = \sum_{(x,y) \in N} S(x, y, l(x), l(y)) \]

  Seam penalty \[ S(x, y, l(x), l(y)) = \| \tilde{I}_x(x) - \tilde{I}_y(x) \| + \| \tilde{I}_x(y) - \tilde{I}_y(y) \| \]

• Optimization: Markov Random Field (MRF)
• Poisson image blending
Seam Selection

- Graph cuts and Vertex cover both produce similar looking results
  - Graph cuts are slower because of optimization over all pixels
  - Vertex cover is sensitive to thresholds used to determine regions of difference
Blending

• Compensate for exposure difference and other mis-alignments

• Using weighted blending, hard to get balance between smoothing out exposure variations and preventing blurring

• Laplacian pyramid blending

• Gradient domain blending

• Exposure compensation
Image Blending
Laplacian Pyramid

- Result in perfect reconstruction when Q is the identity
Laplacian Pyramid

- Actual computation of high-pass filter

- Low-pass images: Gaussian images
- High-pass (or band-pass) images: Laplacian Images
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
Pyramid Blending

Figure 9.17 h: pyramid blending
Gradient Domain Blending

- Perform blending in gradient domain rather than color domain
Gradient Domain Editing

\[ \min_f \int_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial \Omega} = f^*|_{\partial \Omega} \]

\[ \Delta f = \text{div} \mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial \Omega} = f^*|_{\partial \Omega} \]

\[ \text{div} \mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \]

- Guided Vector Field:
  - Source image
  - Mixture gradient (source + target image)
Exposure Compensation

• Estimate local correction between source image and a blended composite

• Stitch image in the radiance domain
  – Convert image into radiance image
  – Create a stitched, high-dynamic range (HDR) image
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