Mosaics

Today’s Readings
- Szeliski, Ch 5.1, 8.1

How to do it?

Basic Procedure
- Take a sequence of images from the same position
  - Rotate the camera about its optical center
- Compute transformation between second image and first
- Shift the second image to overlap with the first
- Blend the two together to create a mosaic
- If there are more images, repeat

Image Mosaics

Goal
- Stitch together several images into a seamless composite

Aligning images

How to account for warping?
- Translations are not enough to align the images
Alignment Demo

The mosaic has a natural interpretation in 3D

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

Image reprojection

Basic question

• How to relate two images from the same camera center?
  – how to map a pixel from PP1 to PP2

Answer

• Cast a ray through each pixel in PP1
• Draw the pixel where that ray intersects PP2

Don’t need to know what’s in the scene!

Observation

• Rather than thinking of this as a 3D reprojection, think of it as a 2D image warp from one image to another
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

\[
\begin{bmatrix}
wx' \\
y' \\
w
\end{bmatrix} = \begin{bmatrix}
* & * & * \\
* & * & * \\
* & * & *
\end{bmatrix} \begin{bmatrix}
x \\
y \\
f
\end{bmatrix}
\]

To apply a homography \( H \)
  - Compute \( p' = Hp \) (regular matrix multiply)
  - Convert \( p' \) from homogeneous to image coordinates
    - divide by \( w \) (third) coordinate

Image warping with homographies

mosaic PP

each image is warped with a homography \( H \)

black area where no pixel maps to
Panoramas

What if you want a 360° field of view?

Spherical projection systems

Omnimax

CAVE (UI Chicago)

Spherical projection

- Map 3D point $(X, Y, Z)$ onto sphere
  \[ (\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2 + Y^2 + Z^2}} (X, Y, Z) \]

- Convert to spherical coordinates
  \[ (\sin \theta \cos \phi, \sin \phi, \cos \theta \cos \phi) = (\hat{x}, \hat{y}, \hat{z}) \]

- Convert to spherical image coordinates
  \[ (\bar{x}, \bar{y}) = (s \theta, s \phi) + (\bar{x}_c, \bar{y}_c) \]
  - $s$ defines size of the final image
  - Often convenient to set $s = \text{camera focal length}$

Spherical reprojection

How to map sphere onto a flat image?

- $(\hat{x}, \hat{y}, \hat{z}) \to (x', y')$
Spherical reprojection

How to map sphere onto a flat image?
- \((\hat{x}, \hat{y}, \hat{z})\) to \((x', y')\)
- Use image projection matrix!
- or use the version of projection that properly accounts for radial distortion, as discussed in projection slides. This is what you’ll do for project 2.

Aligning spherical images

Suppose we rotate the camera by \(\theta\) about the vertical axis
- How does this change the spherical image?

Suppose we rotate the camera by \(\theta\) about the vertical axis
- How does this change the spherical image?
- Translation by \(\theta\)!
- This means we can align spherical images by translating them
Spherical image stitching

What if you don’t know the camera rotation?

- Solve for the camera rotations
  - Note that a pan (rotation) of the camera is a translation of the sphere!
  - Use feature matching to solve for translations of spherical-warped images

Computing transformations

- Given a set of matches between images A and B
  - How can we compute the transform T from A to B?
    - Find transform T that best “agrees” with the matches

Simple case: translations

How do we solve for \((x_t, y_t)\)?

But not all matches are good

What do we do about the “bad” matches?
RAndom SAmple Consensus

Select one match, count inliers
(in this case, only one)

Least squares fit

Find “average” translation vector
for largest set of inliers

RAndom SAmple Consensus

Select one match, count inliers
(4 inliers)

RANSAC

Same basic approach works for any transformation
- Translation, rotation, homographies, etc.
- Very useful tool

General version
- Randomly choose a set of $K$ correspondences
  - Typically $K$ is the minimum size that lets you fit a model
- Fit a model (e.g., homography) to those correspondences
- Count the number of inliers that “approximately” fit the model
  - Need a threshold on the error
- Repeat as many times as you can
- Choose the model that has the largest set of inliers
- Refine the model by doing a least squares fit using ALL of the inliers
Assembling the panorama

Stitch pairs together, blend, then crop

Problem: Drift

Error accumulation
- small errors accumulate over time

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”
Different projections are possible

Project 2
Take pictures with your phone (or on a tripod)
Warp to spherical coordinates
Extract features
Align neighboring pairs using RANSAC
Write out list of neighboring translations
Correct for drift
Read in warped images and blend them
Crop the result and import into a viewer

Roughly based on Autostitch
  • By Matthew Brown and David Lowe
  • http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Image Blending

Feathering
Effect of window (ramp-width) size

Effect of window size

Good window size

Pyramid blending

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

Create a Laplacian pyramid, blend each level
Poisson Image Editing

Encoding blend weights: \( I(x,y) = (\alpha R, \alpha G, \alpha B, \alpha) \)

Implement this in two steps:
1. accumulate: add up the (\( \alpha \) premultiplied) RGB values at each pixel
2. normalize: divide each pixel’s accumulated RGB by its \( \alpha \) value

Q: what if \( \alpha = 0 \)?

Alpha Blending

Optional: see Blinn (CGA, 1994) for details:

\[ I_1 \quad I_2 \quad I_3 \]

Encoding blend weights: \( I(x,y) = (\alpha R, \alpha G, \alpha B, \alpha) \)

color at \( p = (\alpha_1 \alpha_2 \alpha_3 \alpha_4 \alpha_5 \alpha_6 \alpha_7 \alpha_8) \)

Implement this in two steps:
1. accumulate: add up the (\( \alpha \) premultiplied) RGB values at each pixel
2. normalize: divide each pixel’s accumulated RGB by its \( \alpha \) value

Q: what if \( \alpha = 0 \)?

Image warping

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

Forward warping

Send each pixel \(f(x,y)\) to its corresponding location \((x', y') = h(x,y)\) in the second image

Q: what if pixel lands “between” two pixels?
Forward warping

Send each pixel \( f(x,y) \) to its corresponding location \((x',y') = h(x,y)\) in the second image

Q: what if pixel lands “between” two pixels?
A: distribute color among neighboring pixels \((x',y')\)
   - Known as “splatting”

Inverse warping

Get each pixel \( g(x',y') \) from its corresponding location \( (x,y) = h^{-1}(x',y') \) in the first image

Q: what if pixel comes from “between” two pixels?
A: resample color value
   - We discussed resampling techniques before
     - nearest neighbor, bilinear, Gaussian, bicubic

Forward vs. inverse warping

Q: which is better?
A: usually inverse—eliminates holes
   - however, it requires an invertible warp function—not always possible...
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/
  - Click for images…