Announcements

- Project 1
  - Grading session this afternoon
  - Artifacts due Friday (voting TBA)
- Project 2 out (online)
  - Signup for panorama kits ASAP (weekend slots go quickly…)
  - help session at end of class

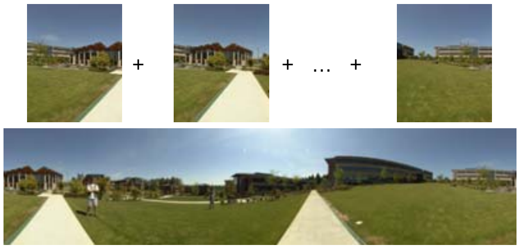
Mosaics

VR Seattle: http://www.vrseattle.com/
Full screen panoramas (cubic): http://www.panoramas.dk/
Mars: http://www.panoramas.dk/fullscreen3/f2_mars97.html

Today's Readings

- Szeliski and Shum paper (sections 1 and 2, skim the rest)
  - http://citeseer.ist.psu.edu/rd/0,282987,1,0.25,Download/http://citeseer.ist.psu.edu/cache/papers/cs/12745/http:zSzzSzcdserver.icemt.iastate.eduzSzcdzSzs97cpzSzcontentszSzpaperszSzszeliskizSzszeliski.pdf/szeliski97creating.pdf

Image Mosaics



Goal

- Stitch together several images into a seamless composite

How to do it?

Basic Procedure

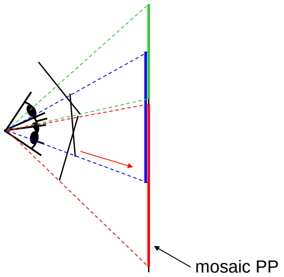
- Take a sequence of images from the same position
  - Rotate the camera about its optical center
- Compute transformation between second image and first
  - Lucas-Kanade registration
- Shift the second image to overlap with the first
- Blend the two together to create a mosaic
- If there are more images,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

Aligning images

How to account for warping?

- Translations are not enough to align the images
- Photoshop demo

Image reprojection



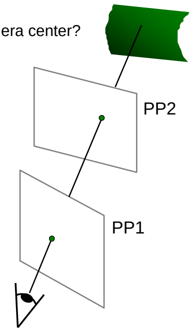
The mosaic has a natural interpretation in 3D

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

Image reprojection

Basic question

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

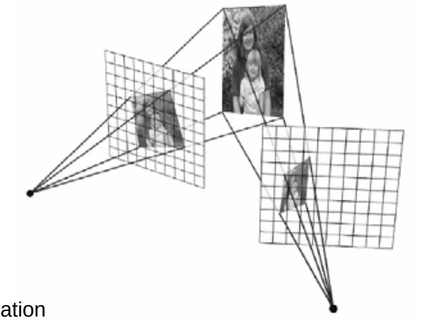


Answer

- 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!

Image reprojection



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

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

$$\mathbf{p'} \qquad \mathbf{H} \qquad \mathbf{p}$$

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

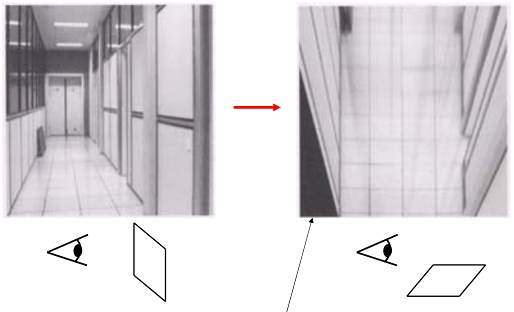


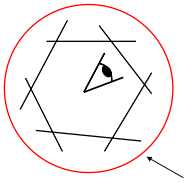
image plane in front

image plane below

black area where no pixel maps to

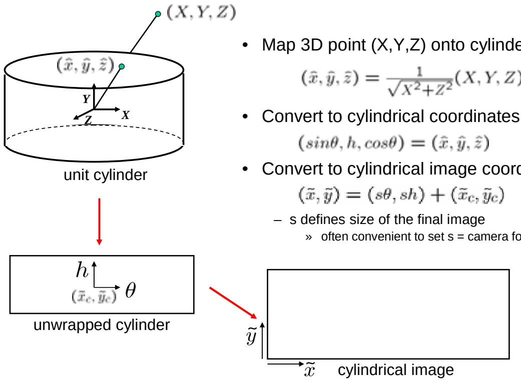
Panoramas

What if you want a 360° field of view?



mosaic Projection Cylinder

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

$$(sin\theta, h, cos\theta) = (\hat{x}, \hat{y}, \hat{z})$$

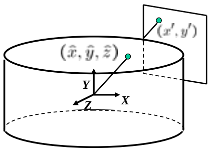
- Convert to cylindrical image coordinates

$$(\tilde{x}, \tilde{y}) = (s\theta, sh) + (\tilde{x}_c, \tilde{y}_c)$$

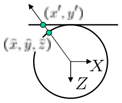
  - s defines size of the final image
    - » often convenient to set s = camera focal length

Cylindrical reprojection

How to map from a cylinder to a planar image?



side view



top-down view

- Apply camera 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.

Cylindrical reprojection

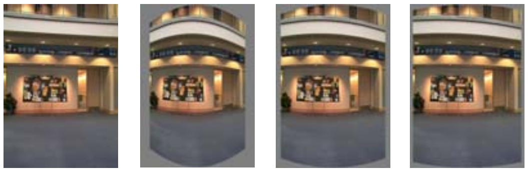
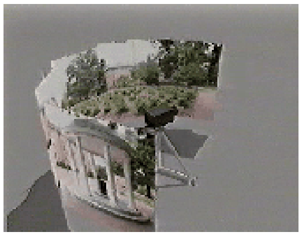


Image 384x300 **f = 180 (pixels)** **f = 280** **f = 380**

Map image to cylindrical coordinates

- need to know the focal length

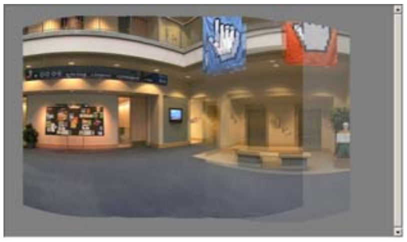
Cylindrical panoramas



Steps

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

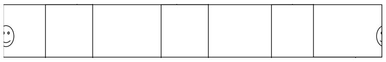
Cylindrical image stitching



What if you don't know the camera rotation?

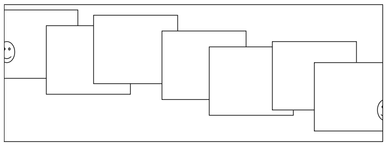
- Solve for the camera rotations
  - Note that a pan (rotation) of the camera is a **translation** of the cylinder!
  - Use Lucas-Kanade to solve for translations of cylindrically-warped images

Assembling the panorama



Stitch pairs together, blend, then crop

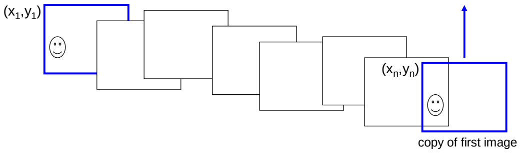
Problem: Drift



Error accumulation

- small errors accumulate over time

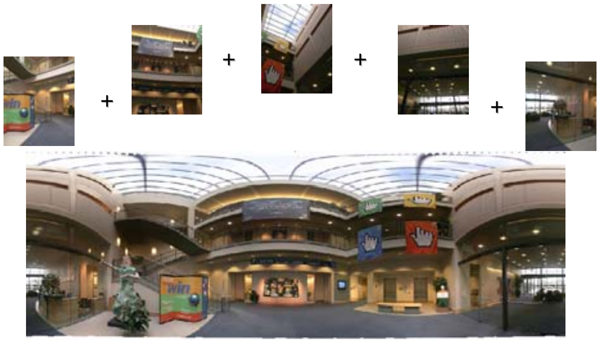
Problem: Drift



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 Panorama

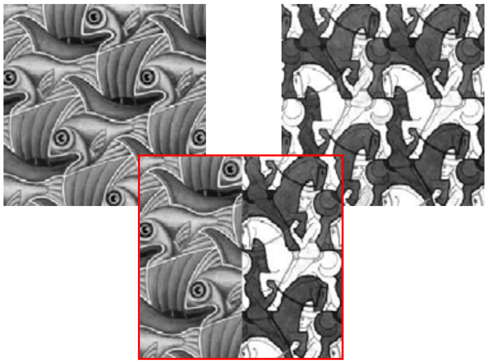


Different projections are possible

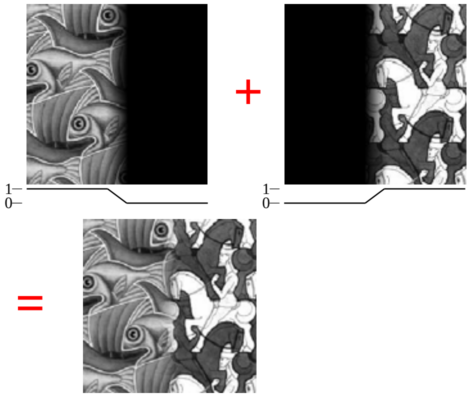
Project 2 (out today)

1. Take pictures on a tripod (or handheld)
2. Warp to cylindrical coordinates
3. Automatically compute pair-wise alignments
4. Correct for drift
5. Blend the images together
6. Crop the result and import into a viewer

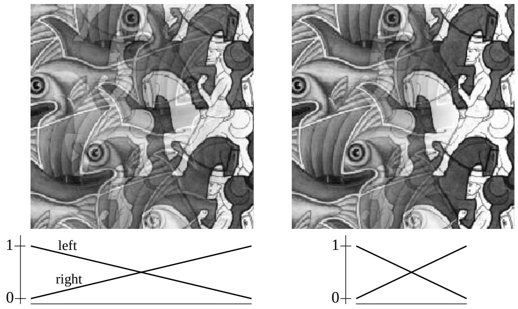
Image Blending



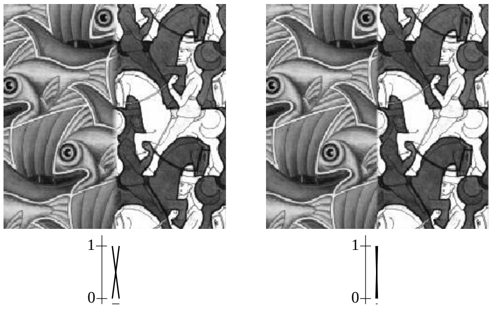
Feathering



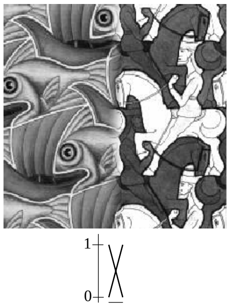
Effect of window size



Effect of window size



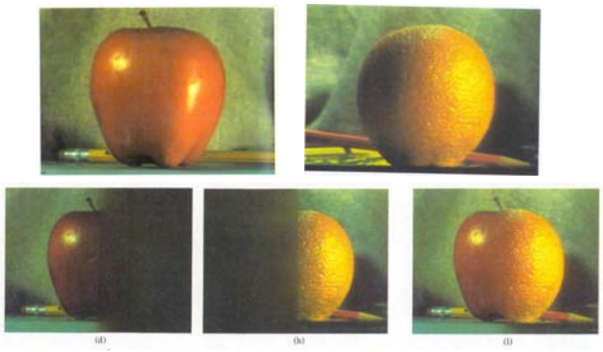
Good window size



“Optimal” window: smooth but not ghosted

- Doesn’t always work...

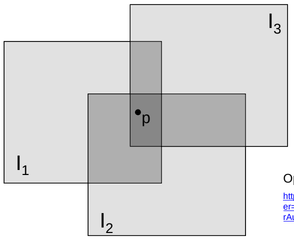
Pyramid blending



Create a Laplacian pyramid, blend each level

- 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.

Alpha Blending



Optional: see Blinn (CGA, 1994) for details:
http://ieeexplore.ieee.org/iel1/38/7531/00310740.pdf?isNumber=7531&prod=JNL&arnumber=310740&arSt=83&ared=87&arAuthor=Blinn%2C+J.F.

Encoding blend weights: I(x,y) = (αR, αG, αB, α)

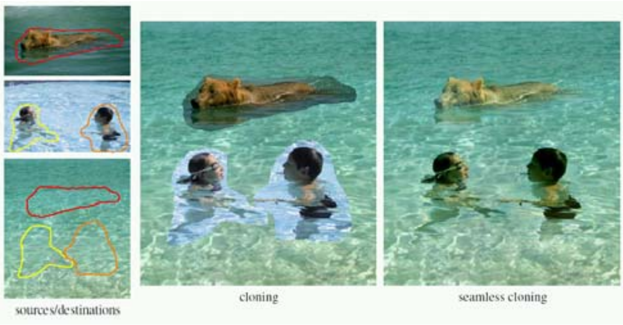
color at p = $\frac{(\alpha_1 R_1,\ \alpha_1 G_1,\ \alpha_1 B_1) + (\alpha_2 R_2,\ \alpha_2 G_2,\ \alpha_2 B_2) + (\alpha_3 R_3,\ \alpha_3 G_3,\ \alpha_3 B_3)}{\alpha_1 + \alpha_2 + \alpha_3}$

Implement this in two steps:

1. accumulate: add up the (α premultiplied) RGBα values at each pixel
2. normalize: divide each pixel's accumulated RGB by its α value

   Q: what if α = 0?

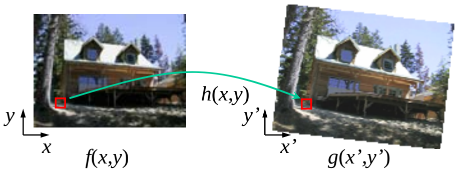
Poisson Image Editing



For more info: Perez et al, SIGGRAPH 2003

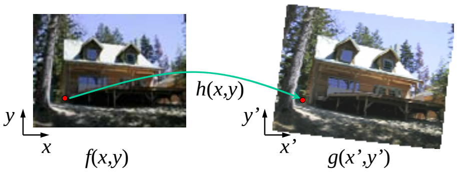
- http://research.microsoft.com/vision/cambridge/papers/perez_siggraph03.pdf

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*))?

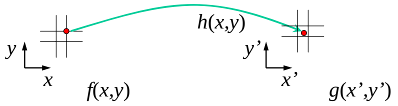
Forward warping



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



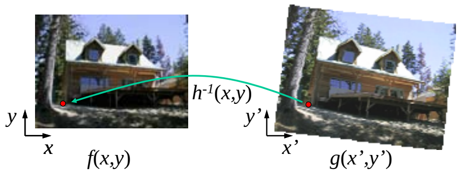
Send each pixel $f(x,y)$ to its corresponding location $(x',y') = h(x,y)$ in the second image

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

A: distribute color among neighboring pixels (x’,y’)

- Known as “splatting”

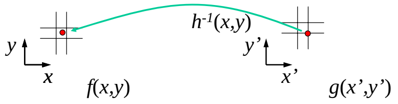
Inverse warping



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?

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/

AutoStitch

Method so far is not completely automatic

- need to know which pairs fit together
- need to initialize Lucas-Kanade to get good results

Newer methods are fully automatic

- AutoStitch, by Matthew Brown and David Lowe:
  - http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html
- Based on feature matching techniques