Announcements

- Project 2
  - Out today
  - Sign up for a panorama kit ASAP!
    - best slots (weekend) go quickly…

Mosaics part 2

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://www.cs.washington.edu/education/courses/455/08wi/readings/szeliski/Shum97.pdf

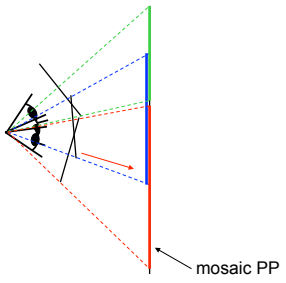
Project 2

1. Take pictures on a tripod (or handheld)
2. Warp 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

Roughly based on **Autostitch**

- By Matthew Brown and David Lowe
- http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Image reprojection



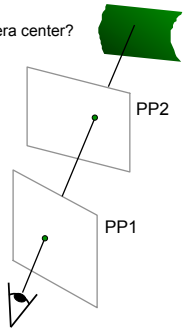
The mosaic has a natural interpretation in 3D

- 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

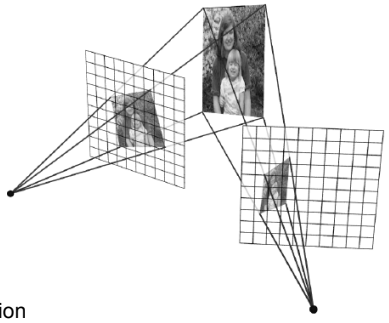


Answer

- 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

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

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

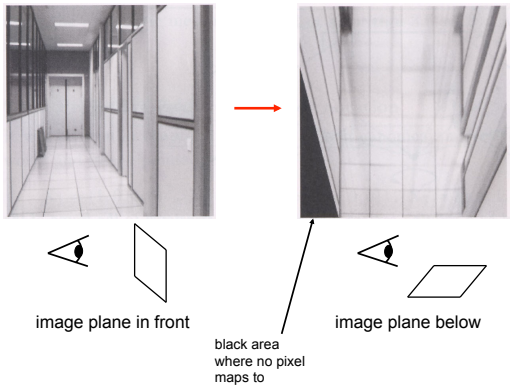
To apply a homography **H**

- Compute **p' = Hp** (regular matrix multiply)
- Convert **p'** from homogeneous to image coordinates
  - divide by w (third) coordinate

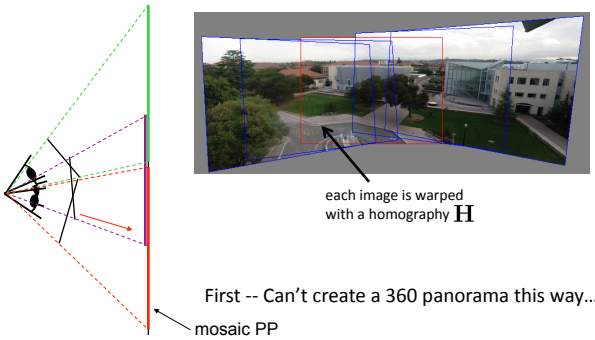
Homography

A few examples on board

Image warping with homographies



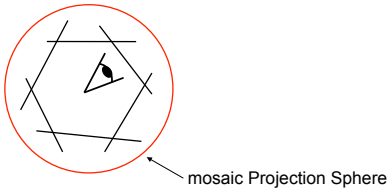
Idea: projecting images onto a common plane



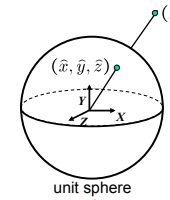
First -- Can't create a 360 panorama this way…

Panoramas

What if you want a 360° field of view?



Spherical projection



- Map 3D point (X,Y,Z) onto sphere

$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2+Y^2+Z^2}}(X, Y, Z)$$

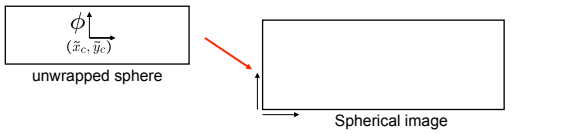
- Convert to spherical coordinates

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

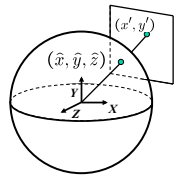
- Convert to spherical image coordinates

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

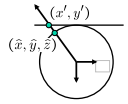
  - s defines size of the final image
    - » often convenient to set s = camera focal length



Spherical reprojection



side view

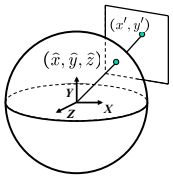


top-down view

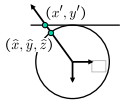
How to map sphere onto a flat image?

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

Spherical reprojection



side view



top-down view

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.

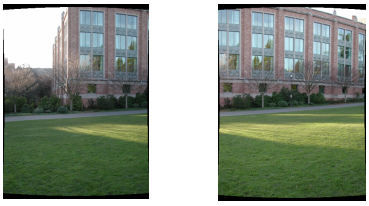
Spherical reprojection

input **f = 200 (pixels)** **f = 400** **f = 800**

Map image to spherical coordinates

- need to know the focal length

Aligning spherical images



Suppose we rotate the camera by θ about the vertical axis

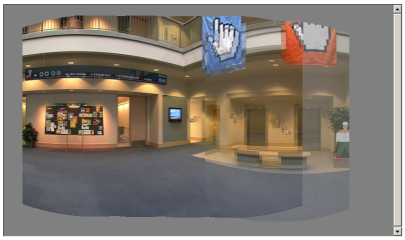
- How does this change the spherical image?

Aligning spherical images

Suppose we rotate the camera by θ about the vertical axis

- How does this change the spherical image?
  - Translation by θ
- This means that we can align spherical images by translation

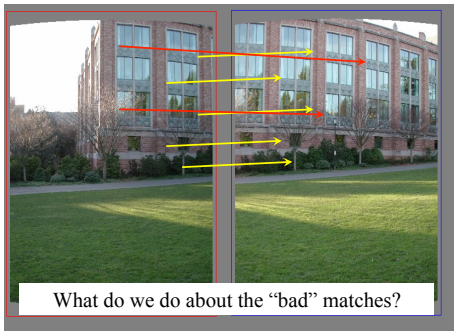
Spherical image stitching



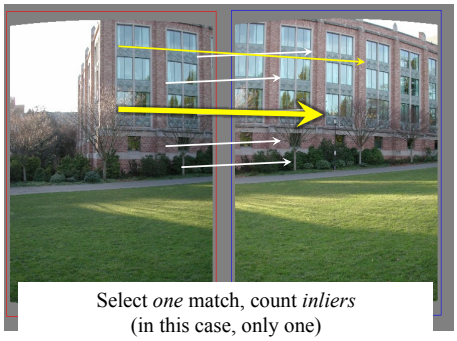
What if you don't know the camera rotation?

- 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

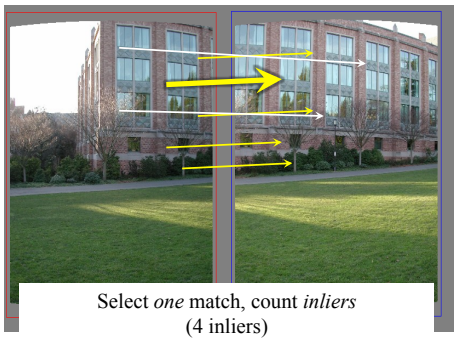
Computing image translations



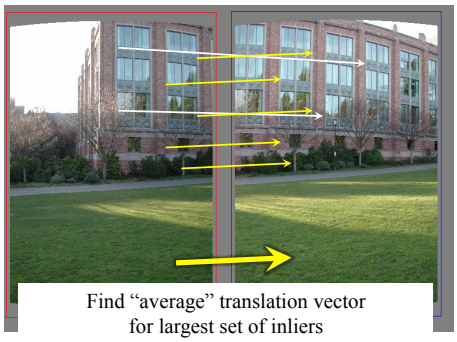
RAndom SAmple Consensus



RAndom SAmple Consensus



Least squares fit



RANSAC

Same basic approach works for any transformation

- Translation, rotation, homographies, etc.
- Very useful tool

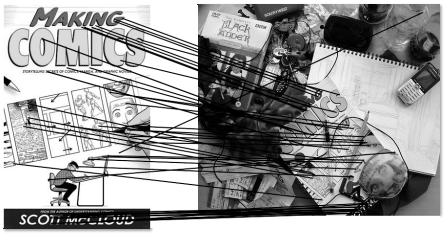
General version

- 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

Computing transformations

Given a set of matches between images A and B

- How can we compute the transform T from A to B?



Solving for translations

Using least squares

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\underset{2n \text{ x } 2}{\mathbf{A}} \quad \underset{2 \text{ x } 1}{\mathbf{t}} = \underset{2n \text{ x } 1}{\mathbf{b}}$$

Affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

How many unknowns?

How many equations per match?

How many matches do we need?

Affine transformations

Residuals:

$$\begin{aligned} r_{x_i}(a,b,c,d,e,f) &= (ax_i + by_i + c) - x'_i \\ r_{y_i}(a,b,c,d,e,f) &= (dx_i + ey_i + f) - y'_i \end{aligned}$$

$$C(a,b,c,d,e,f) = \sum_{i=1}^{n} \left(r_{x_i}(a,b,c,d,e,f)^2 + r_{y_i}(a,b,c,d,e,f)^2 \right)$$

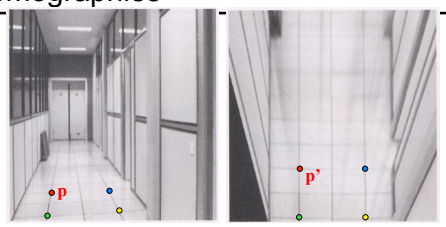
Affine transformations

Matrix form

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

$$\underset{2n \text{ x } 6}{\mathbf{A}} \quad \underset{6 \text{ x } 1}{\mathbf{t}} = \underset{2n \text{ x } 1}{\mathbf{b}}$$

Homographies



To unwarp (rectify) an image

- solve for homography **H** given **p** and **p'**
- solve equations of the form: w**p'** = **Hp**
  - linear in unknowns: w and coefficients of **H**
  - H is defined up to an arbitrary scale factor
  - how many points are necessary to solve for **H**?

Solving for homographies

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$
$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

Not linear!

$$x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$
$$y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

Solving for homographies

$$x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$
$$y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies

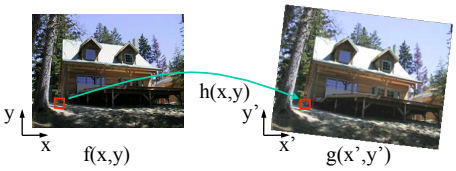
$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{\mathbf{A}} \qquad \underset{9}{\mathbf{h}} \qquad \underset{2n}{\mathbf{0}}$$

Defines a least squares problem: minimize $\|\mathbf{Ah} - 0\|^2$

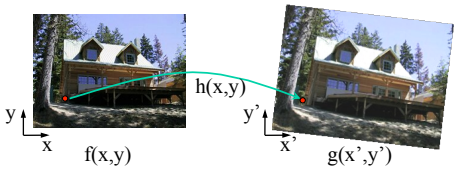
- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

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))?

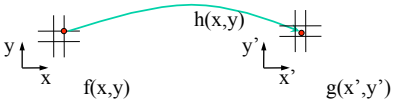
Forward warping



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



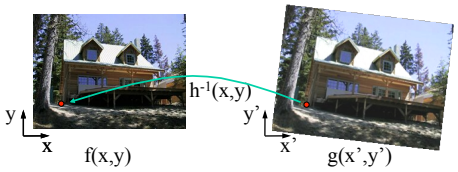
Send each pixel f(x,y) to its corresponding location
(x',y') = h(x,y) in the second image

Q: what if pixel lands "between" two pixels?

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

– Known as "splatting"

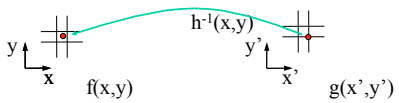
Inverse warping



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...

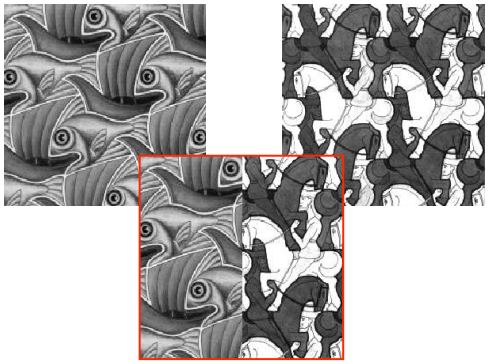
Blending

We've aligned the images – now what?

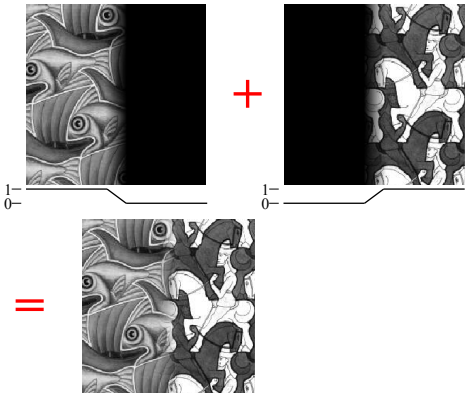
Blending

Want to seamlessly blend them together

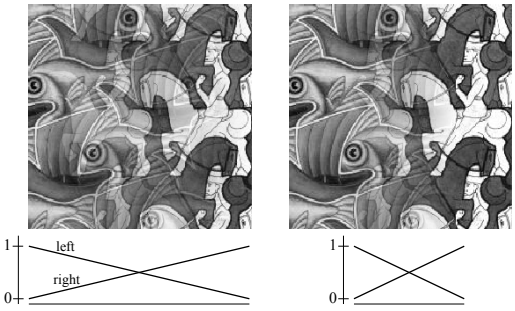
Image Blending



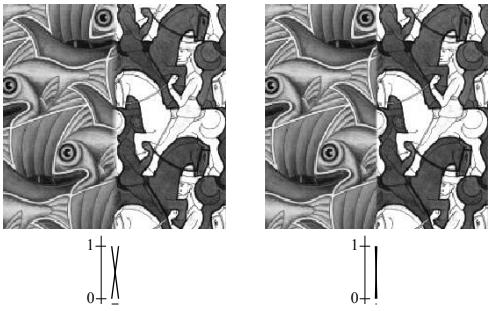
Feathering



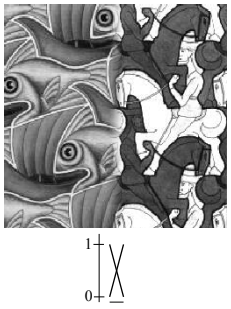
Effect of window size



Effect of window size



Good window size



"Optimal" window: smooth but not ghosted

- Doesn't always work...

Image feathering

What if you're blending more than two images?

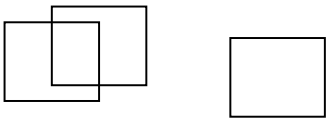
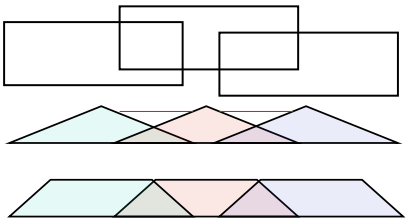


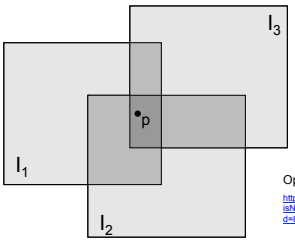
Image feathering

What if you have more than two images?

- Generate weight map for each image
  - typically want large weight at center, small weight at edge
- Each output pixel is a weighted average of inputs
  - be sure to divide by sum of weights at the end



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:

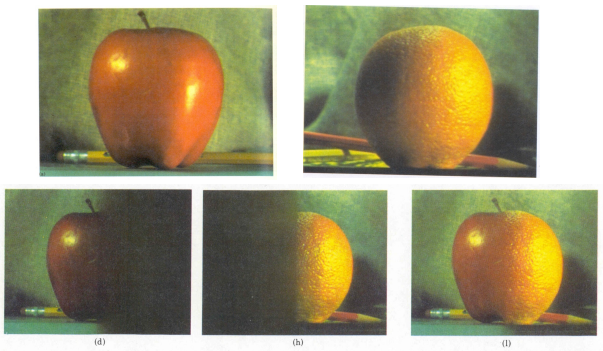
1. accumulate: add up the (α premultiplied) RGBα values at each pixel
2. normalize: divide each pixel's accumulated RGB by its α value

   Q: what if α = 0?

More advanced blending schemes

A quick survey...

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.

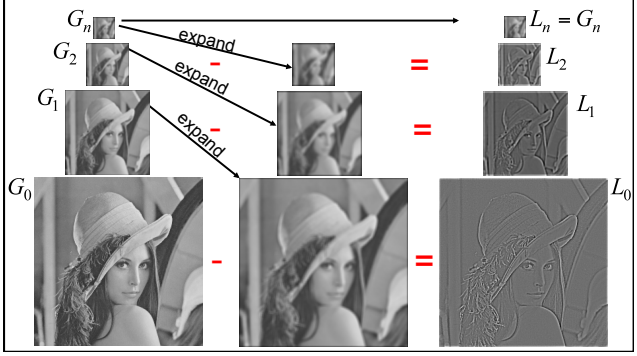
The Laplacian Pyramid

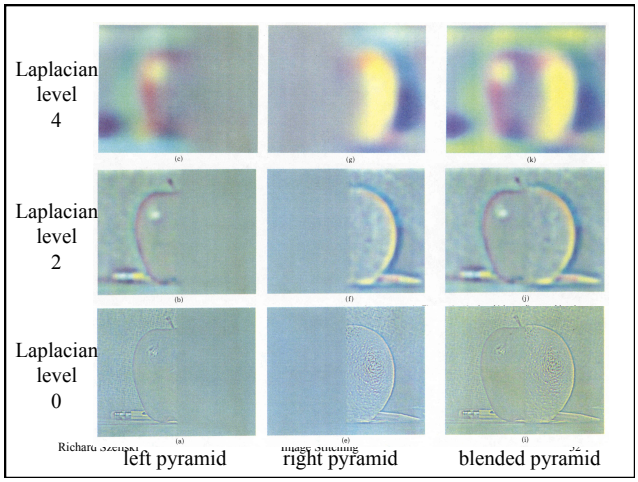
$L_i = G_i - \text{expand}(G_{i+1})$

$G_i = L_i + \text{expand}(G_{i+1})$

Gaussian Pyramid

Laplacian Pyramid





Laplacian image blend

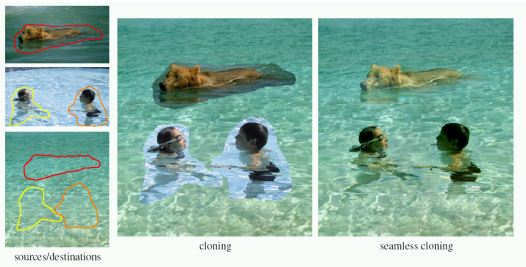
1. Compute Laplacian pyramid
2. Compute Gaussian pyramid on *weight* image (can put this in A channel)
3. Blend Laplacians using Gaussian blurred weights
4. Reconstruct the final image

Q: How do we compute the original weights?

A: For horizontal panorama, use *mid-lines*

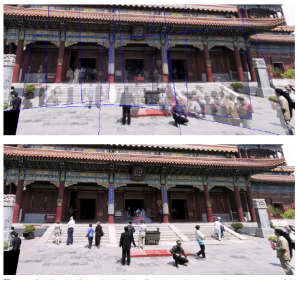
Q: How about for a general "3D" panorama?

Gradient-domain blending



Blend the gradients of the two images, then integrate
For more info: Perez et al, SIGGRAPH 2003
Also called "Poisson" blending

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).

Photomontage [Agarwala et al., SIGGRAPH 2004]

- Each patch of the composite comes from a single image
- Solve for the seams that are hardest to detect (graph cuts)
- Blend across seams using gradient-domain blending

Photomontage [Agarwala et al., SIGGRAPH 2004]

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).

Photomontage [Agarwala et al., SIGGRAPH 2004]

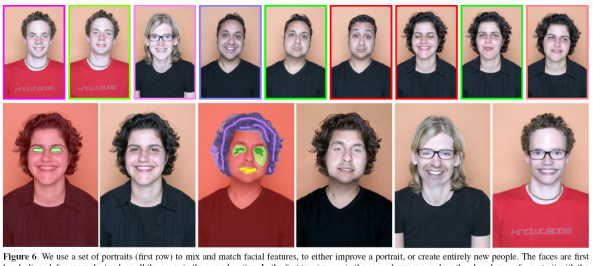


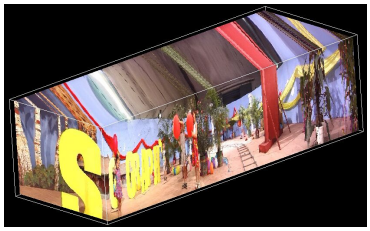
Figure 6 We use a set of portraits (first row) to mix and match facial features, to either improve a portrait, or create entirely new people. The faces are first hand-aligned, for example, to place all the noses in the same location. In the first two images in the second row, we replace the closed eyes of a portrait with the open eyes of another. The user paints strokes with the *designated source* objective to specify desired features. Next, we create a fictional person by combining three source portraits. Gradient-domain fusion is used to smooth out skin tone differences. Finally, we show two additional mixed portraits.

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