Panoramas

Based on slides from
Richard Szeliski

(*Back in 2010*)

Overview

- Panoramas
- Motion Models
- Image Stitching
- De-ghosting

Panoramas

Panoramas

360 Cities: https://www.360cities.net/

2003 New Years Eve: http://www.panoramas.dk/fullscreen3/f1.html

Creating a Panorama (Image Mosaic)

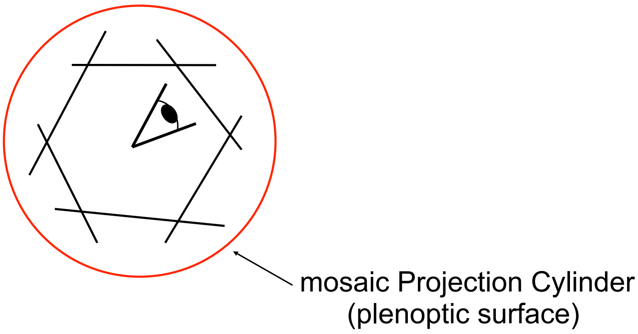
Blend together several overlapping images into one seamless *mosaic* (composite)

Establishing correspondences

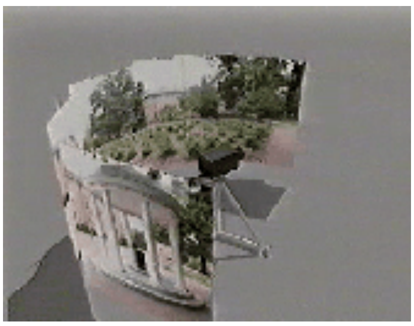
1. Direct method:
   - Use generalization of affine motion model [Szeliski & Shum ’97]
2. Feature-based method
   - Compute feature-based correspondence [Lowe ICCV’99; Schmid ICCV’98, Brown&Lowe ICCV’2003]
   - Compute R from correspondences (absolute orientation)

Panoramas

What if you want a 360° field of view?



Cylindrical panoramas



Steps

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

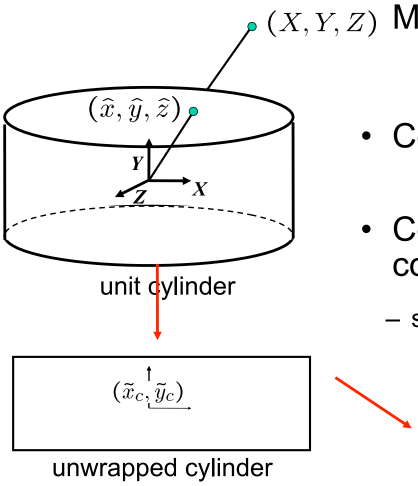
Cylindrical Panoramas

Map image to cylindrical or spherical coordinates

- need *known* focal length

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

Cylindrical projection



Map 3D point (X,Y,Z) onto cylinder

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

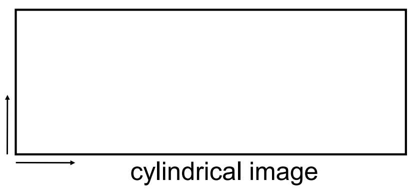
- Convert to cylindrical coordinates

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

- Convert to cylindrical image coordinates

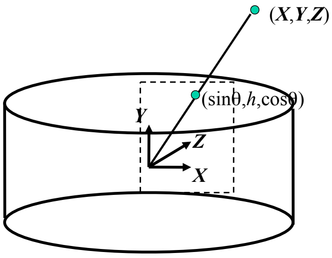
$$(\tilde{x}, \tilde{y}) = (s\theta, sh) + (\tilde{x}_c, \tilde{y}_c)$$

  – s defines size of the final image



Cylindrical warping

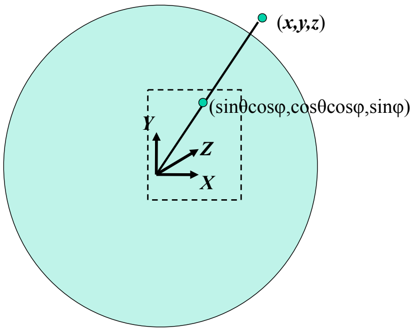
Given focal length f and image center (x_c, y_c)



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

Spherical warping

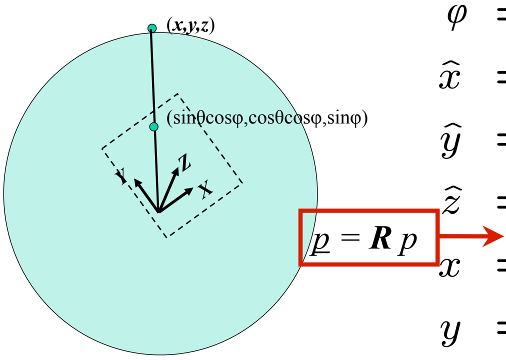
Given focal length f and image center (x_c, y_c)



$$\begin{aligned}
\theta &= (x_{sph} - x_c)/f \\
\varphi &= (y_{sph} - y_c)/f \\
\hat{x} &= \sin\theta\ \cos\varphi \\
\hat{y} &= \sin\varphi \\
\hat{z} &= \cos\theta\cos\varphi \\
x &= f\hat{x}/\hat{z} + x_c \\
y &= f\hat{y}/\hat{z} + y_c
\end{aligned}$$

3D rotation

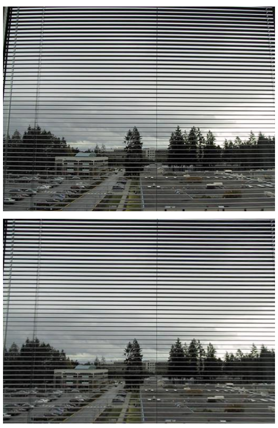
Rotate image before placing on unrolled sphere



$$
\begin{aligned}
\theta &= (x_{sph} - x_c)/f \\
\varphi &= (y_{sph} - y_c)/f \\
\hat{x} &= \sin\theta \, \cos\varphi \\
\hat{y} &= \sin\varphi \\
\hat{z} &= \cos\theta \cos\varphi \\
x &= f\underline{\hat{x}}/\underline{\hat{z}} + x_c \\
y &= f\underline{\hat{y}}/\underline{\hat{z}} + y_c
\end{aligned}
$$

Radial distortion

Correct for "bending" in wide field of view lenses



$$
\begin{aligned}
\hat{r}^2 &= \hat{x}^2 + \hat{y}^2 \\
\hat{x}' &= \hat{x}/(1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
\hat{y}' &= \hat{y}/(1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
x &= f\hat{x}'/\hat{z} + x_c \\
y &= f\hat{y}'/\hat{z} + y_c
\end{aligned}
$$

Fisheye lens

Extreme "bending" in ultra-wide fields of view

$$\hat{r}^2 = \hat{x}^2 + \hat{y}^2$$

$$(\cos\theta\sin\phi, \sin\theta\sin\phi, \cos\phi) = s\,(x, y, z)$$

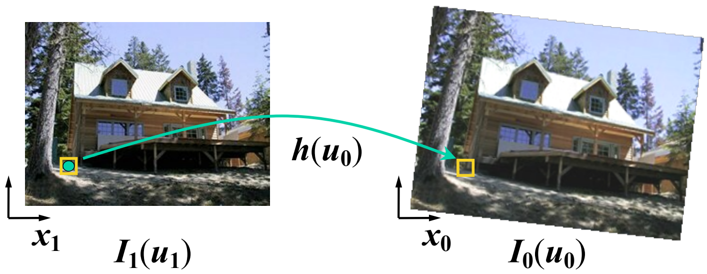
Equations become

$$\begin{aligned} x' &= s\phi\cos\theta = s\frac{x}{r}\tan^{-1}\frac{r}{z}, \\ y' &= s\phi\sin\theta = s\frac{y}{r}\tan^{-1}\frac{r}{z}, \end{aligned}$$

Inverse Warping

Get each pixel $I_0(u_0)$ from its corresponding location $u_1 = h(u_0)$ in $I_1(u_1)$

- What if pixel comes from "between" two pixels?



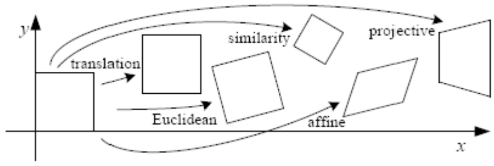
Motion models

Motion models

What happens when we take two images with a camera and try to align them?

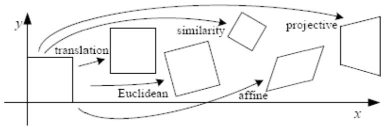
- translation?
- rotation?
- scale?
- affine?
- Perspective?

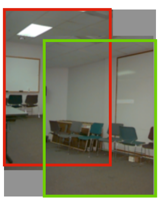
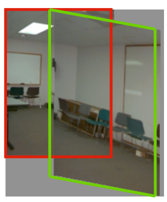
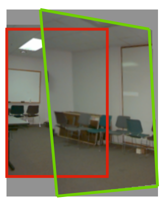
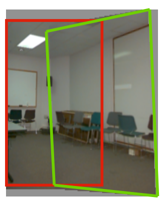
Motion models



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\, I \mid t \,\right]_{2\times 3}$	2	orientation $+ \cdots$	
rigid (Euclidean)	$\left[\, R \mid t \,\right]_{2\times 3}$	3	lengths $+ \cdots$	
similarity	$\left[\, sR \mid t \,\right]_{2\times 3}$	4	angles $+ \cdots$	
affine	$\left[\, A \,\right]_{2\times 3}$	6	parallelism $+ \cdots$	
projective	$\left[\, \tilde{H} \,\right]_{3\times 3}$	8	straight lines	

Motion models



Translation	Affine	Perspective	3D rotation
			
2 unknowns	6 unknowns	8 unknowns	3 unknowns

Homographies

Perspective projection of a plane

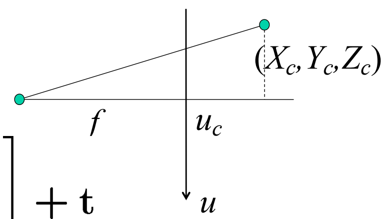
- Lots of names for this:
  - **homography**, texture-map, colineation, planar projective map
- Modeled as a 2D warp using homogeneous coordinates

$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p'}} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$

To apply a homography **H**

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

3D → 2D Perspective Projection

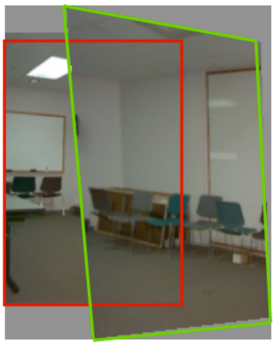


$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

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

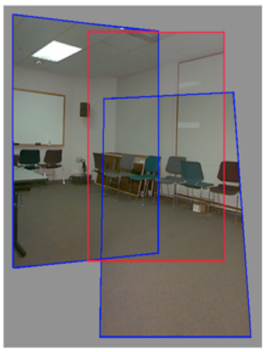
Plane perspective mosaics

- 8-parameter generalization of affine motion
  - works for pure rotation or planar surfaces
- Limitations:
  - local minima
  - slow convergence
  - difficult to control interactively



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 Rotation Model

Projection equations

1. Project from image to 3D ray

   $(x_0,y_0,z_0) \quad = (u_0-u_c,v_0-v_c,f)$

- Rotate the ray by camera motion

  $(x_1,y_1,z_1) \quad = \boldsymbol{R}_{01} \,(x_0,y_0,z_0)$

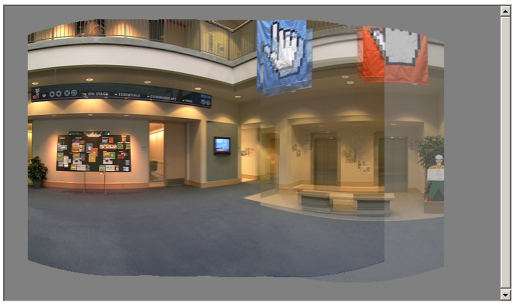
- Project back into new (source) image

  $(u_1,v_1) \quad = (fx_1/z_1+u_c,fy_1/z_1+v_c)$

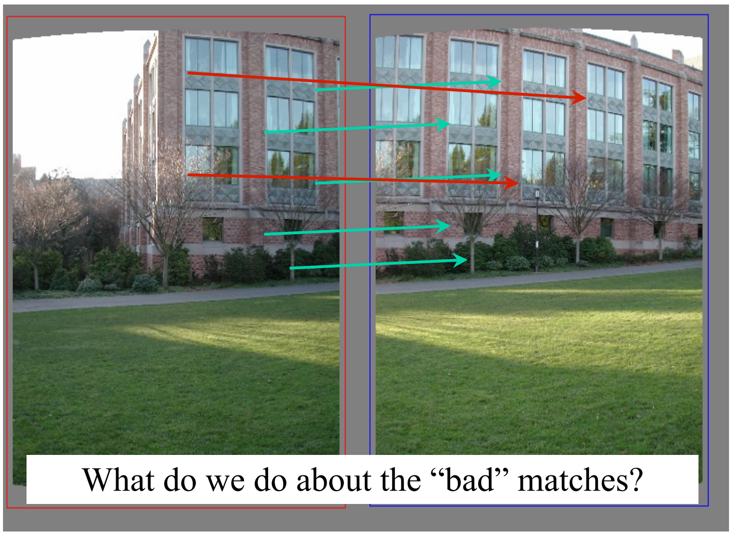
Image Stitching

Image Stitching

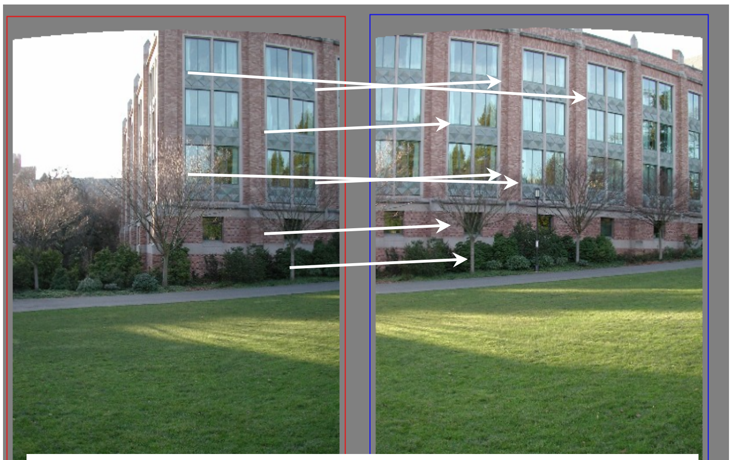
1. Align the images over each other
   - camera pan ↔ translation on cylinder!
- Blend the images together



Matching features

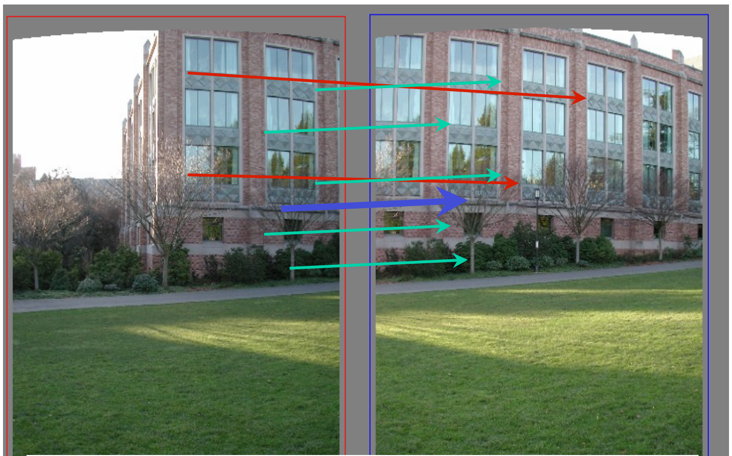


RAndom SAmple Consensus



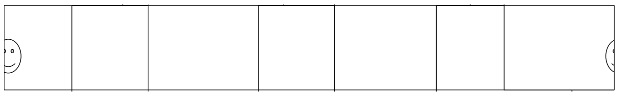
Select *one* match, count *inliers*

Least squares fit



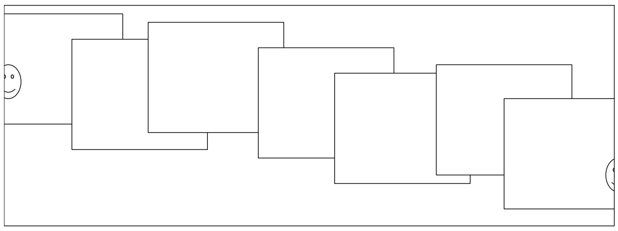
Find “average” translation vector

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.)

Full-view Panoramas

Global alignment

- Register *all* pairwise overlapping images
- Use a 3D rotation model (one R per image)
- Use feature based registration of *unwarped* images
- *Discover* which images overlap other images using feature selection (RANSAC)
- *Chain* together inter-frame rotations
- *Optimize* all R estimates together (bundle adjustment)

3D Rotation Model

Projection equations

1. Project from image to 3D ray

   $(x_0,y_0,z_0) \quad = (u_0\text{-}u_c, v_0\text{-}v_c, f)$

- Rotate the ray by camera motion

  $(x_1,y_1,z_1) \quad = \boldsymbol{R}_{01}\ (x_0,y_0,z_0)$

- Project back into new (source) image

  $(u_1,v_1) \quad = (fx_1/z_1+u_c, fy_1/z_1+v_c)$

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

$p_i' = \boldsymbol{R}\ p_i$ with 3D rays

$p_i = (x_i,y_i,z_i) = (u_i\text{-}u_c, v_i\text{-}v_c, f)$

$\boldsymbol{A} = \Sigma_\mathbf{i}\, p_i\, p_i'^T = \Sigma_\mathbf{i}\, p_i\, p_i^T\, \boldsymbol{R}^T = \boldsymbol{U}\, \boldsymbol{S}\, \boldsymbol{V}^T = (\boldsymbol{U}\, \boldsymbol{S}\, \boldsymbol{U}^T)\, \boldsymbol{R}^T$

$\boldsymbol{V}^T = \boldsymbol{U}^T \boldsymbol{R}^T$

$\boldsymbol{R} = \boldsymbol{V}\, \boldsymbol{U}^T$

Deghosting and Blending

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]

Figure 4: Deghosting a mosaic with optical distortion: (a) with distortion; (b) after multiple steps.

Image feathering

Weight each image proportional to its distance from the edge
(distance map [Danielsson, CVGIP 1980]

Cut out the appropriate region from each image and then blend together

Region-based de-ghosting

Select <u>only one image</u> in *regions-of-difference* using weighted vertex cover [Uyttendaele *et al.*, CVPR'01]

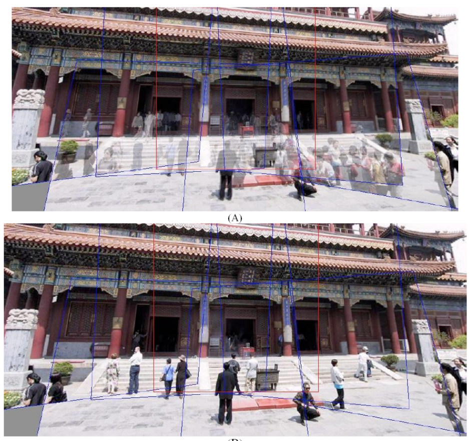
(A) (B)

Figure 5 – (A) Ghosted mosaic. (B) Result of de-ghosting algorithm.

Region-based de-ghosting

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

Striking Example



(A) (B)

Figure 6 – (A) Ghosted mosaic. (B) Result of de-ghosting algorithm.

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]