EE795: Computer Vision and Intelligent Systems

Spring 2012
TTh 17:30-18:45 FDH 204

Lecture 11
130226

http://www.ee.unlv.edu/~b1morris/ecg795/

Outline

- Review
    - Feature-Based Alignment
    - Image Warping
    - 2D Alignment Using Least Squares
- Mosaics
- Panoramas

Feature-Based Alignment

- After detecting and matching features, may want to verify if the matches are geometrically consistent
  - Can feature displacements be described by 2D and 3D geometric transformations

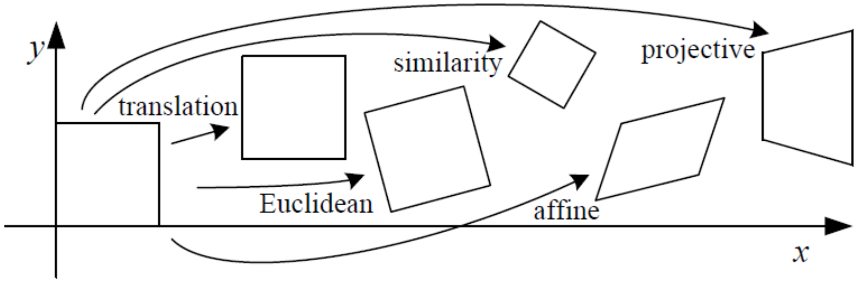
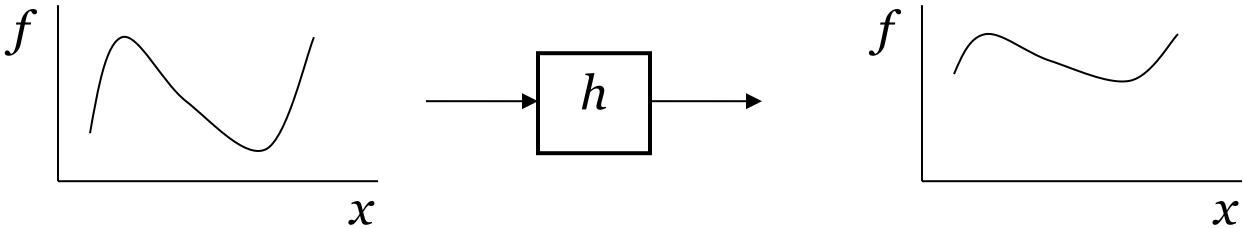


Figure 6.2 Basic set of 2D planar transformations

- Provides
- Geometric registration
  - 2D/3D mapping between images
- Pose estimation
  - Camera position with respect to a known 3D scene/object
- Intrinsic camera calibration
  - Find internal parameters of cameras (e.g. focal length, radial distortion)

Image Warping

- image filtering: change *range* of image
  - $g(x) = h(f(x))$



- image warping: change *domain* of image
  - $g(x) = f(h(x))$

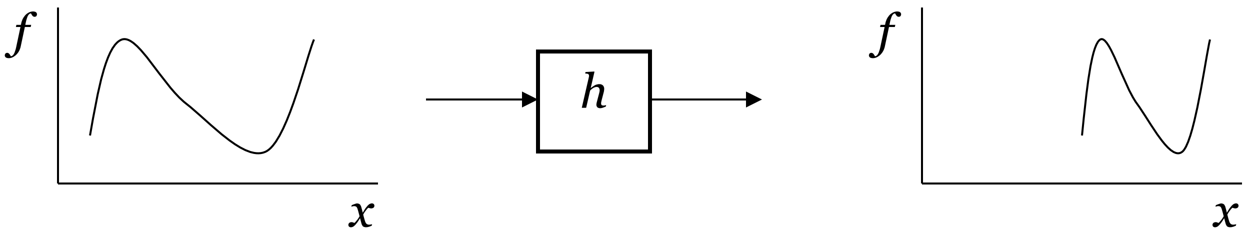
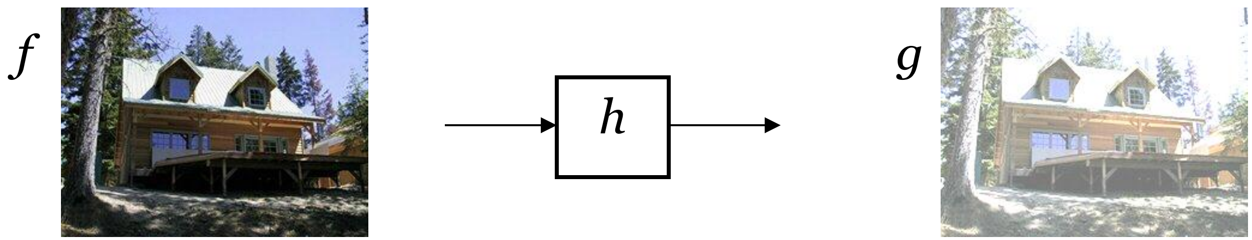


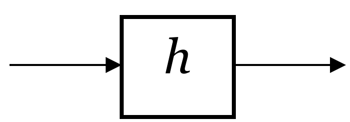
Image Warping

- image filtering: change *range* of image
  - $g(x) = h(f(x))$



- image warping: change *domain* of image
  - $g(x) = f(h(x))$

f



g

Parametric (global) warping

- Examples of parametric warps:


translation


rotation


aspect


affine


perspective

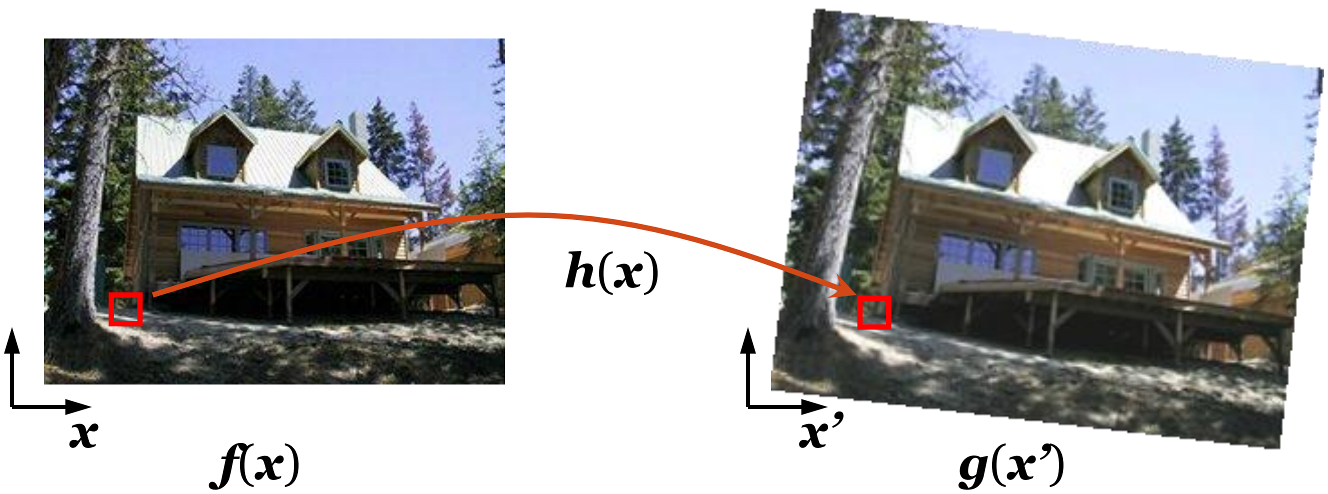

cylindrical

2D coordinate transformations

- translation: $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$ $\boldsymbol{x} = (x,y)$
- rotation: $\boldsymbol{x}' = \boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- similarity: $\boldsymbol{x}' = s\,\boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- affine: $\boldsymbol{x}' = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{t}$
- perspective: $\underline{\boldsymbol{x}}' \cong \boldsymbol{H}\,\underline{\boldsymbol{x}}$ $\underline{\boldsymbol{x}} = (x,y,1)$
  ($\underline{\boldsymbol{x}}$ is a *homogeneous* coordinate)
- These all form a nested *group* (closed w/ inv.)

Image Warping

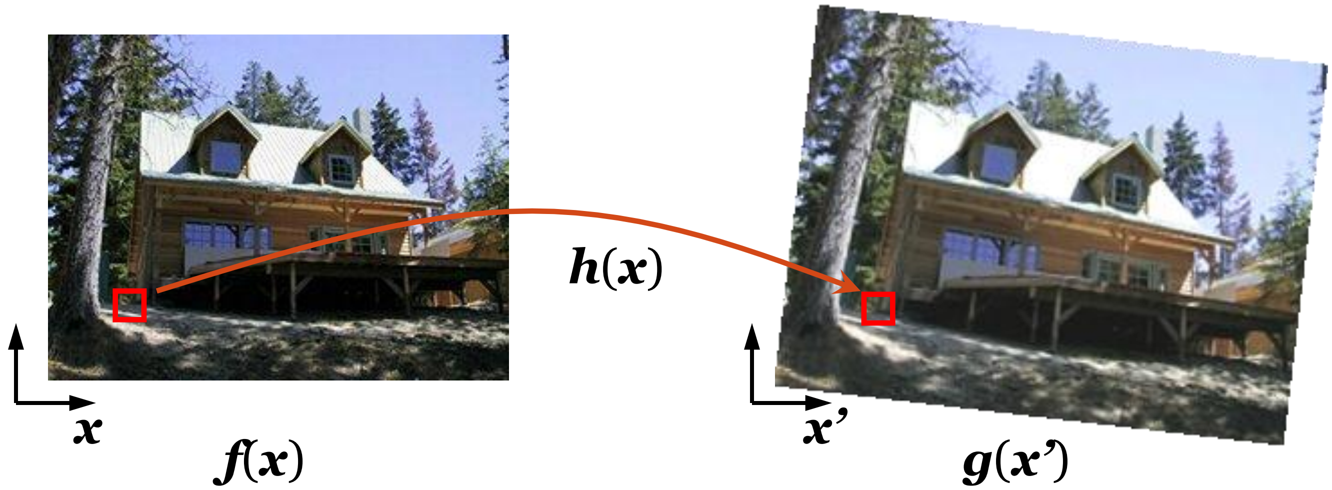
- Given a coordinate transform $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$ and a source image $\boldsymbol{f}(\boldsymbol{x})$, how do we compute a transformed image $\boldsymbol{g}(\boldsymbol{x}') = \boldsymbol{f}(\boldsymbol{h}(\boldsymbol{x}))$?



Forward Warping

- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x}')$

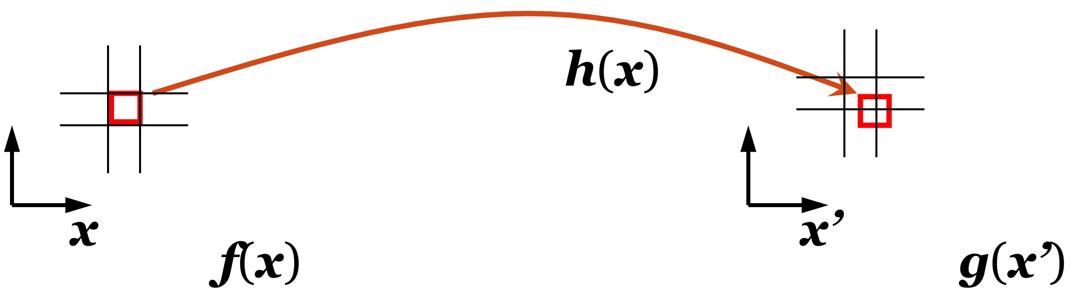
- What if pixel lands "between" two pixels?



Forward Warping

- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x}')$

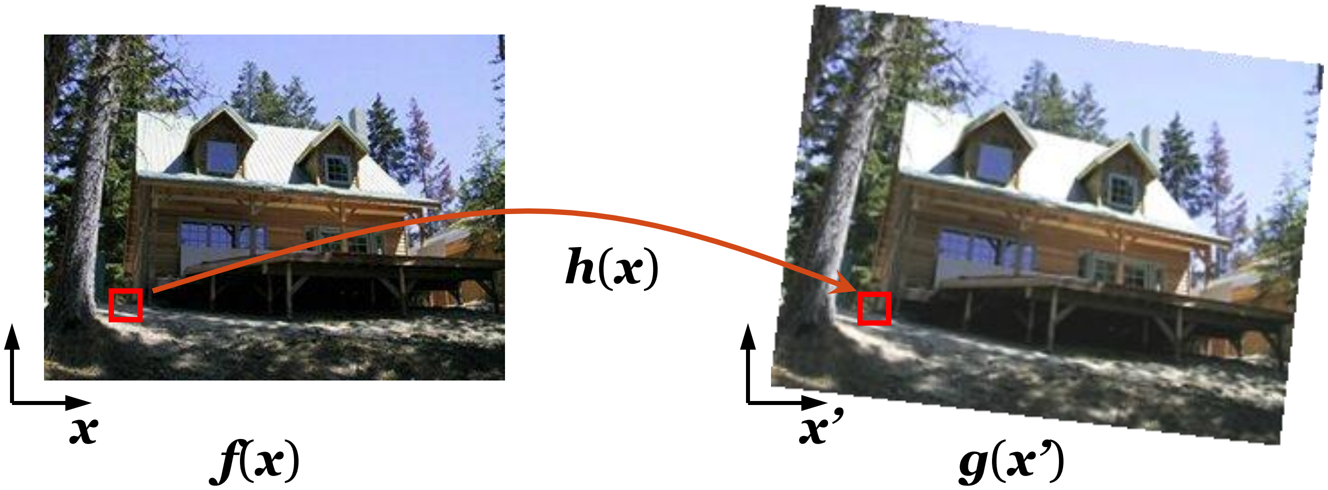
- What if pixel lands “between” two pixels?
- Answer: add “contribution” to several pixels, normalize later (*splatting*)
- See `griddata.m`



Inverse Warping

- Get each pixel $\boldsymbol{g}(\boldsymbol{x}')$ from its corresponding location $\boldsymbol{x} = \boldsymbol{h}^{-1}(\boldsymbol{x}')$ in $\boldsymbol{f}(\boldsymbol{x})$

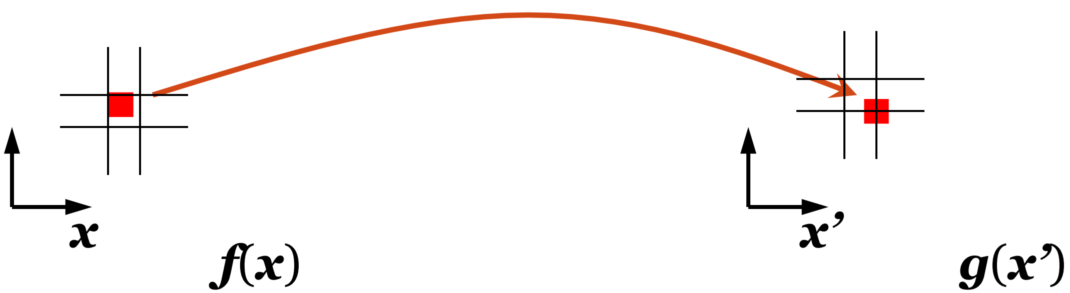
- What if pixel comes from "between" two pixels?



Inverse Warping

- Get each pixel $\boldsymbol{g}(\boldsymbol{x'})$ from its corresponding location $\boldsymbol{x} = \boldsymbol{h}^{-1}(\boldsymbol{x'})$ in $\boldsymbol{f}(\boldsymbol{x})$

- What if pixel comes from "between" two pixels?
- Answer: *resample* color value from *interpolated* (*prefiltered*) source image
- See `interp2.m`



Forward vs. Inverse Warping

- Which type of warping is better?

- Usually inverse warping is preferred
  - It eliminates holes
  - However, it requires an invertible warp function
    - Not always possible

Least Squares Alignment

- Given a set of matched features $\{(x_i, x_i')\}$, minimize sum of squared residual error
  - $E_{LS} = \sum_i \|r_i\|^2 = \sum_i \|f(x_i; p) - x_i'\|^2$
    - $f(x_i; p)$ - is the predicted location based on the transformation p

- The unknowns are the parameters p
  - Need to have a model for transformation
  - Estimate the parameters based on matched features

Linear Least Squares Alignment

- Many useful motion models have a linear relationship between motion and parameters p
  - $\Delta x = x' - x = J(x)p$
    - $J = \frac{\partial f}{\partial p}$ - the Jacobian of the transform f with respect to the motion parameters p
- Linear least squares
  - $E_{LLS} = \sum_i \|J(x_i)p - \Delta x_i\|^2 = p^T A p - 2p^T b + c$
    - Quadratic form
- The minimum is found by solving the normal equations
  - $Ap = b$
    - $A = \sum_i J^T(x_i)J(x_i)$ – Hessian matrix
    - $b = \sum_i J^T(x_i)\Delta x_i$
  - Gives the LLS estimate for the motion parameters

Jacobians of 2D Transformations

Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$
projective	$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$	$(h_{00}, h_{01}, \ldots, h_{21})$	(see Section 6.1.3)

Table 6.1 Jacobians of the 2D coordinate transformations $\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$ shown in Table 2.1, where we have re-parameterized the motions so that they are identity for $\boldsymbol{p} = 0$.

Improving Motion Estimates

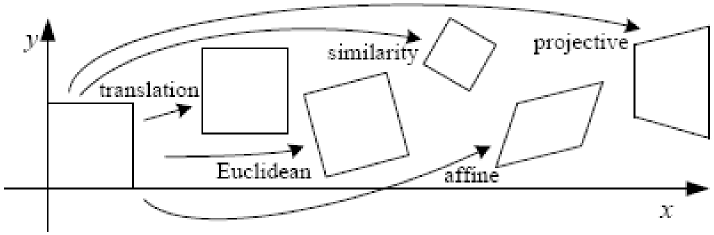
- A number of techniques can improve upon linear least squares
- Uncertainty weighting
  - Weight the matches based certainty of the match – texture in the match region
- Non-linear least squares
  - Iterative algorithm to guess parameters and iteratively improve guess
- Robust least squares
  - Explicitly handle outliers (bad matches) – don't use L2 norm
- RANSAC
  - Randomly select subset of corresponding points, compute initial estimate of p, count the inliers from all the other correspondences, good match has many inliers

Image Mosaics

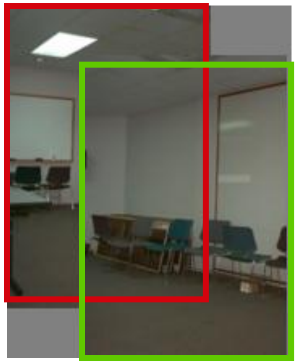
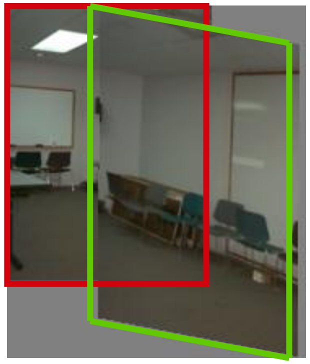
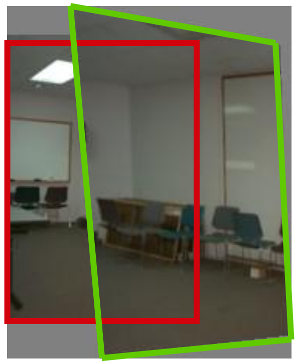
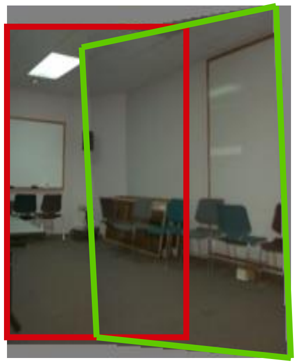
-  +  + … +  =

Goal: Stitch together several images into a seamless composite

Motion models



Translation | **Affine** | **Perspective** | **3D rotation**

2 unknowns | **6 unknowns** | **8 unknowns** | **3 unknowns**

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

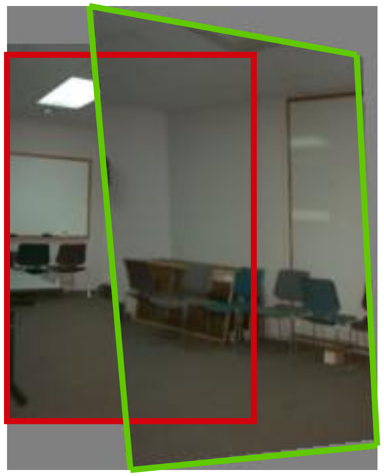


Image warping with homographies

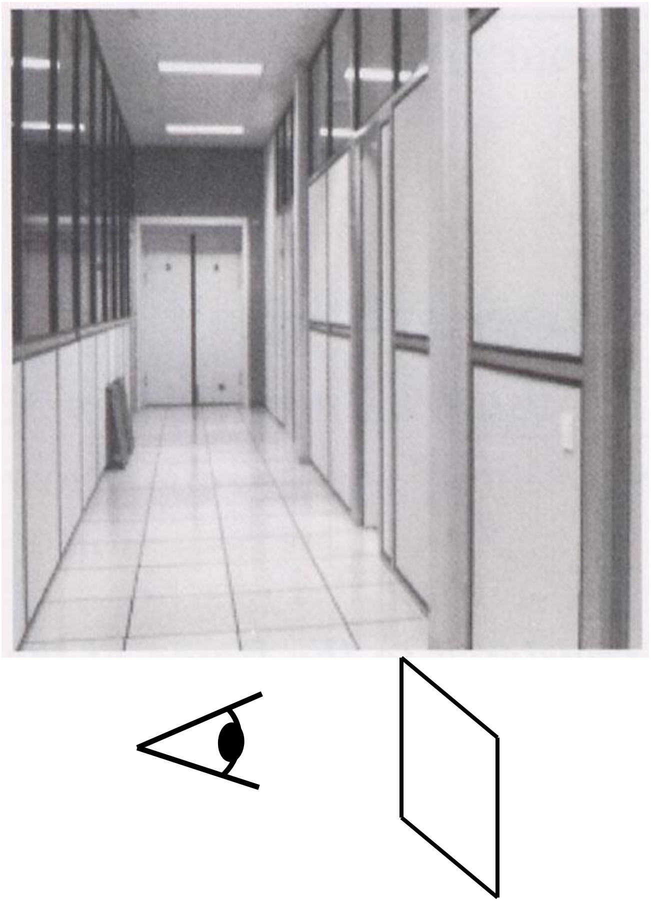


image plane in front

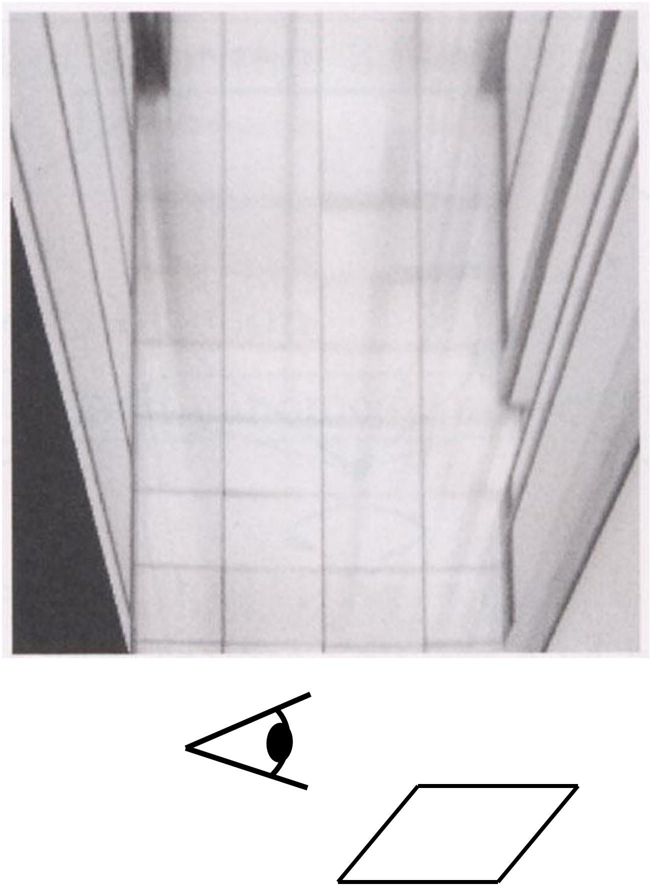
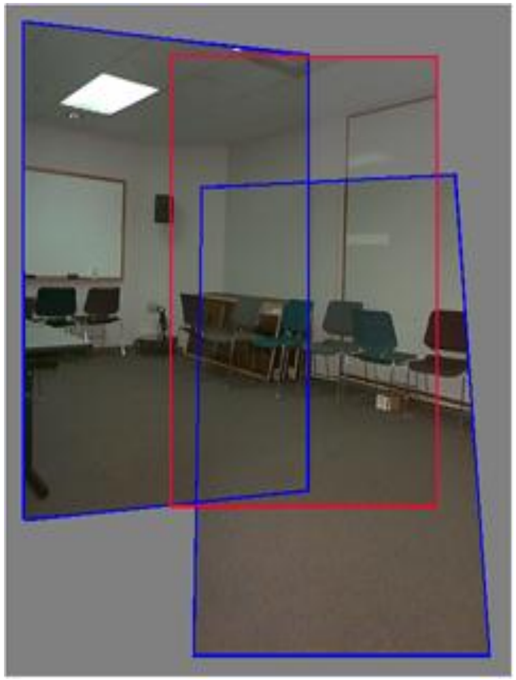


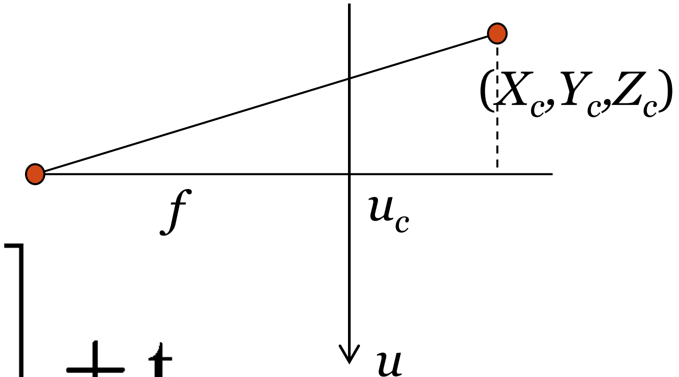
image plane below

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 → 2D Perspective Projection



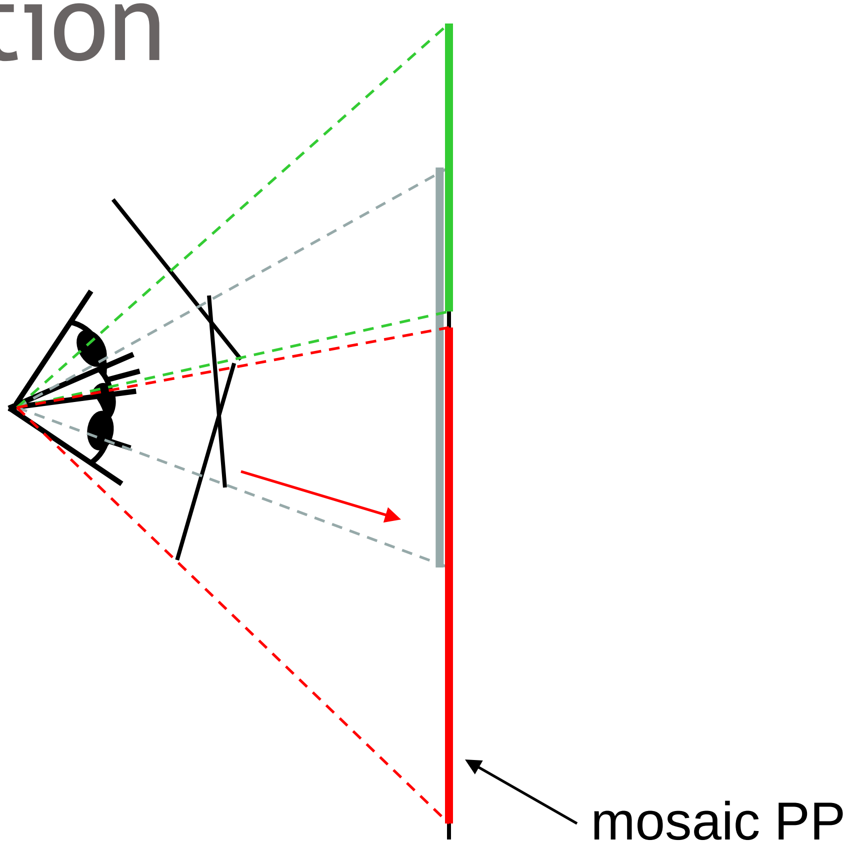
$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

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

Rotational mosaic

- Projection equations
1. Project from image to 3D ray
- $(x_0,y_0,z_0) \quad = (u_0-u_c,v_0-v_c,f)$
2. Rotate the ray by camera motion
- $(x_1,y_1,z_1) \quad = \boldsymbol{R}_{01}\ (x_0,y_0,z_0)$
3. Project back into new (source) image
- $(u_1,v_1)= (fx_1/z_1+u_c,fy_1/z_1+v_c)$

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 Mosaics (stitching)

- Blend together several overlapping images into one seamless *mosaic* (composite)
  - [Szeliski & Shum, SIGGRAPH'97]
  - [Szeliski, FnT CVCG, 2006]
-  +  + … +  =

Mosaics for Video Coding

- Convert masked images into a background sprite for content-based coding

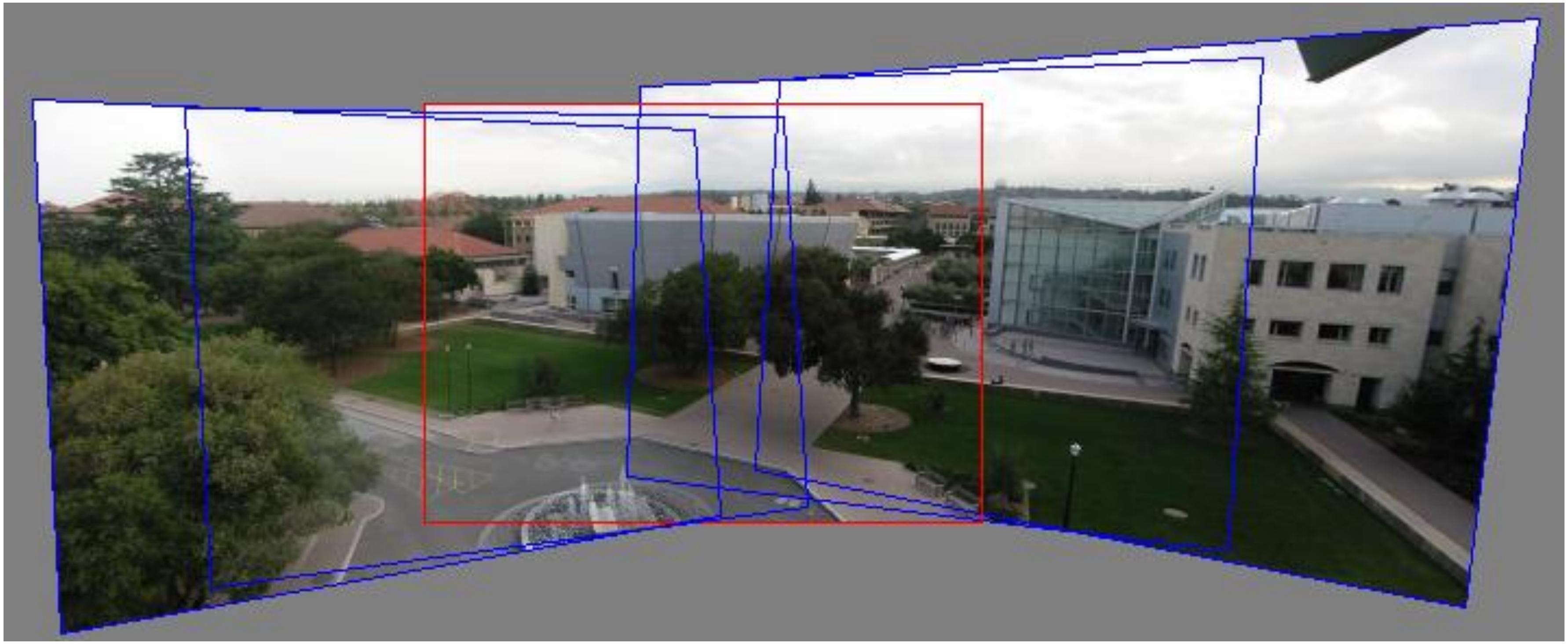
-

=

Establishing correspondences

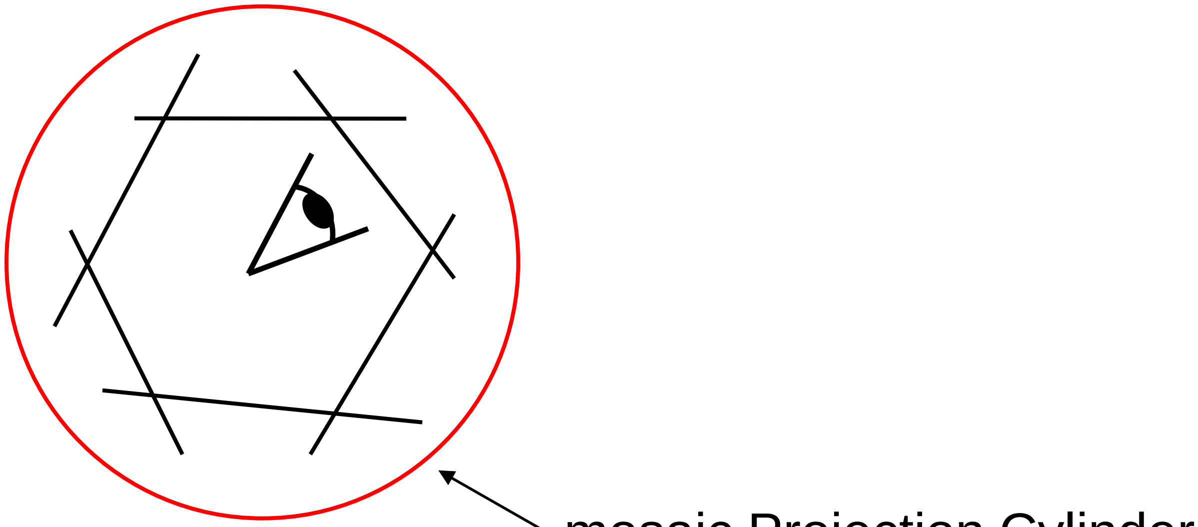
1. Direct method:
   - Use generalization of affine motion model [Szeliski & Shum ’97]
2. Feature-based method
   - Extract features, match, find consisten *inliers* [Lowe ICCV’99; Schmid ICCV’98, Brown&Lowe ICCV’2003]
   - Compute ***R*** from correspondences (absolute orientation)

Stitching demo

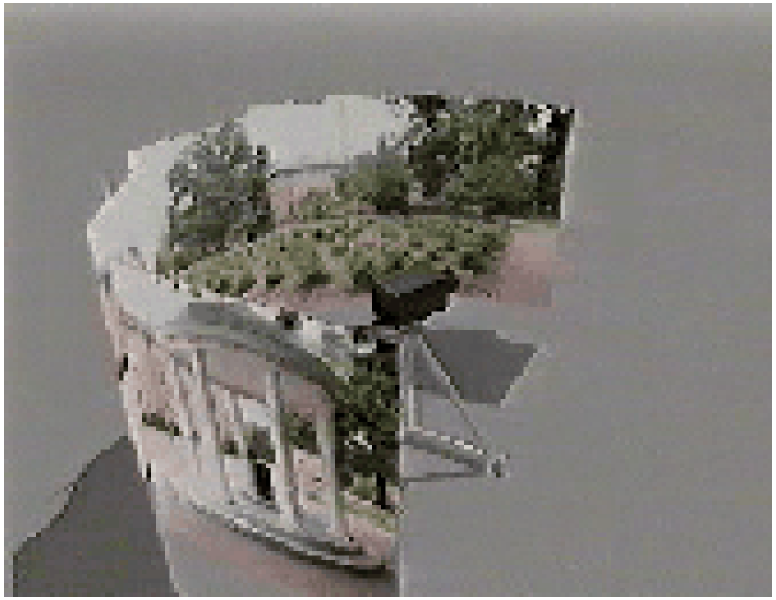


Panoramas

- 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

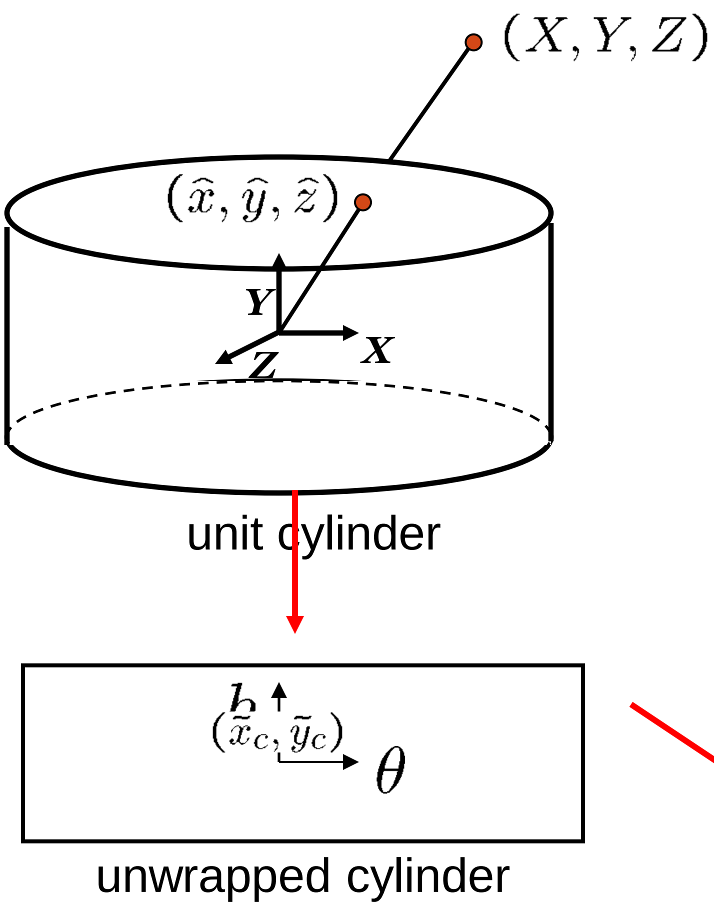
Image 384x300

f = 180 (pixels)

f = 280

f = 380

Cylindrical projection



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

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

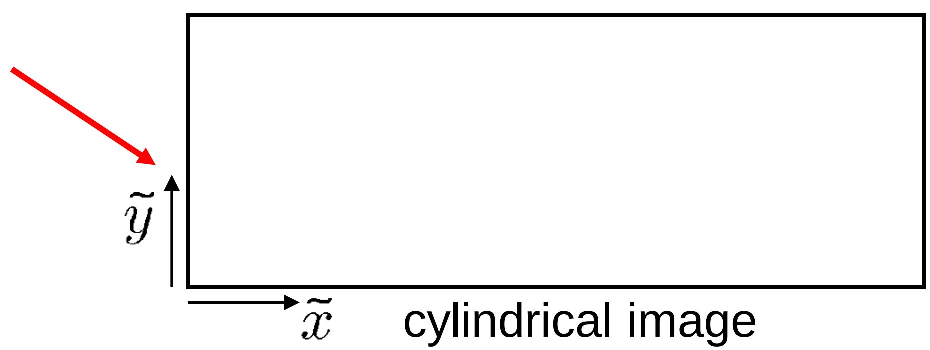
- Convert to cylindrical coordinates

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

- Convert to cylindrical image coordinates

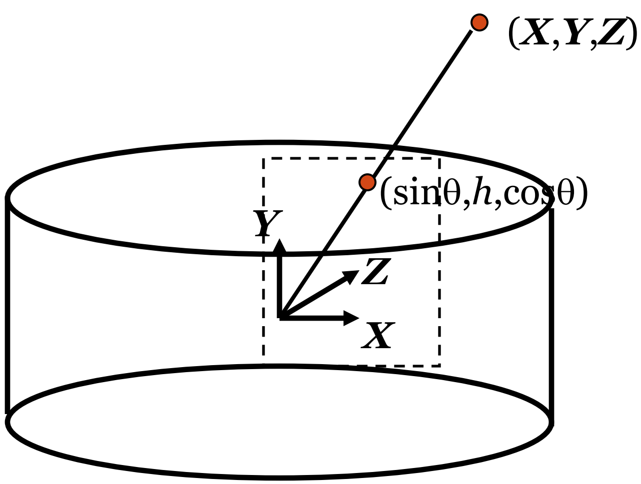
$$(\tilde{x}, \tilde{y}) = (s\theta, sh) + (\tilde{x}_c, \tilde{y}_c)$$

  – s defines size of the final image



Cylindrical warping

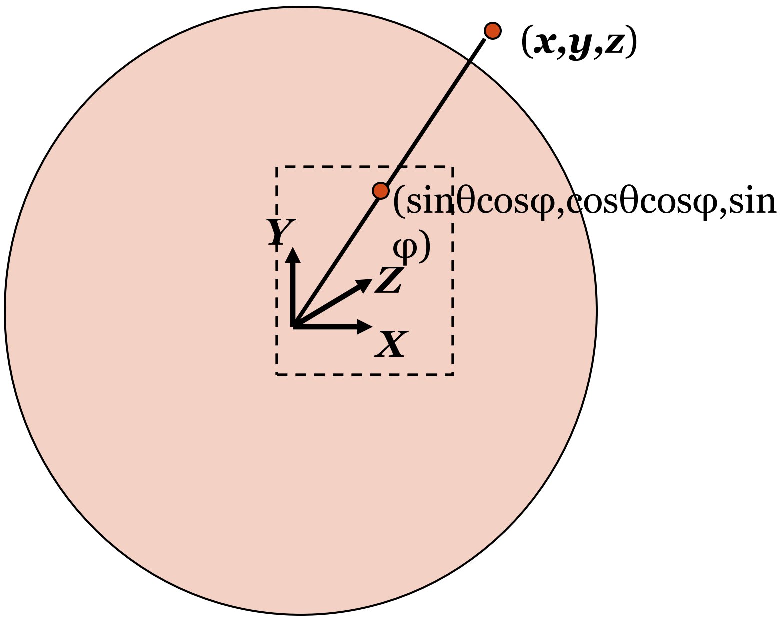
- Given focal length *f* and image center (x_c, y_c)



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

Spherical warping

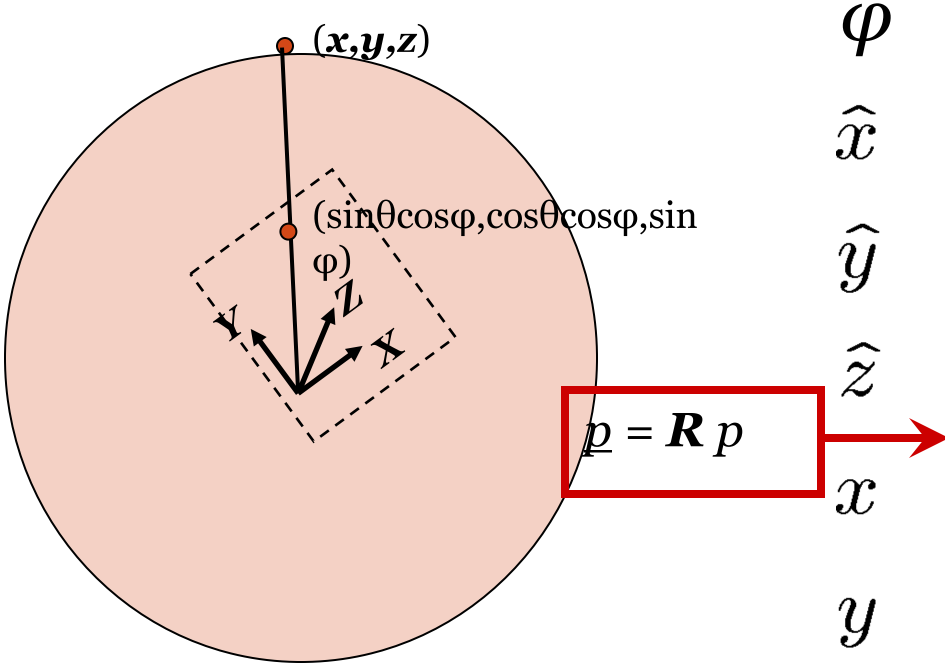
- Given focal length f and image center (x_c, y_c)



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

3D rotation

- Rotate image before placing on unrolled sphere



$$\theta = (x_{cyl} - x_c)/f$$

$$\varphi = (y_{cyl} - y_c)/f$$

$$\hat{x} = \sin\theta\cos\varphi$$

$$\hat{y} = \sin\varphi$$

$$\hat{z} = \cos\theta\cos\varphi$$

$$x = f\underline{\hat{x}}/\underline{\hat{z}} + x_c$$

$$y = f\underline{\hat{y}}/\underline{\hat{z}} + y_c$$

Distortion Correction

- Radial distortion
  - Correct for “bending” in wide field of view lenses

- Fisheye lens
  - Extreme “bending” in ultra-wide fields of view

Adapted from R. Szeliski

Image Stitching

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

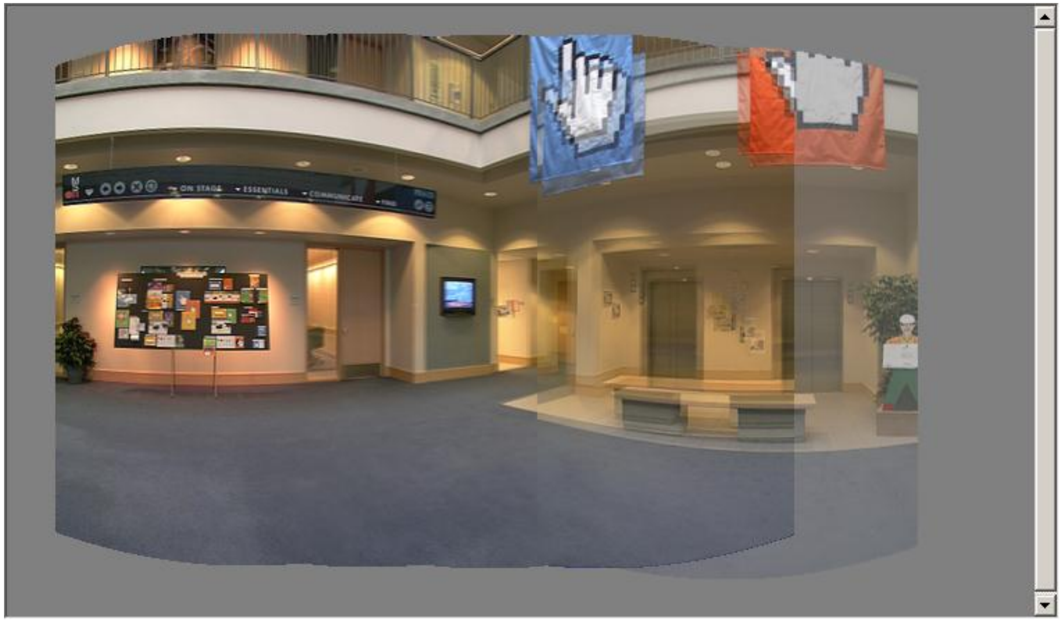
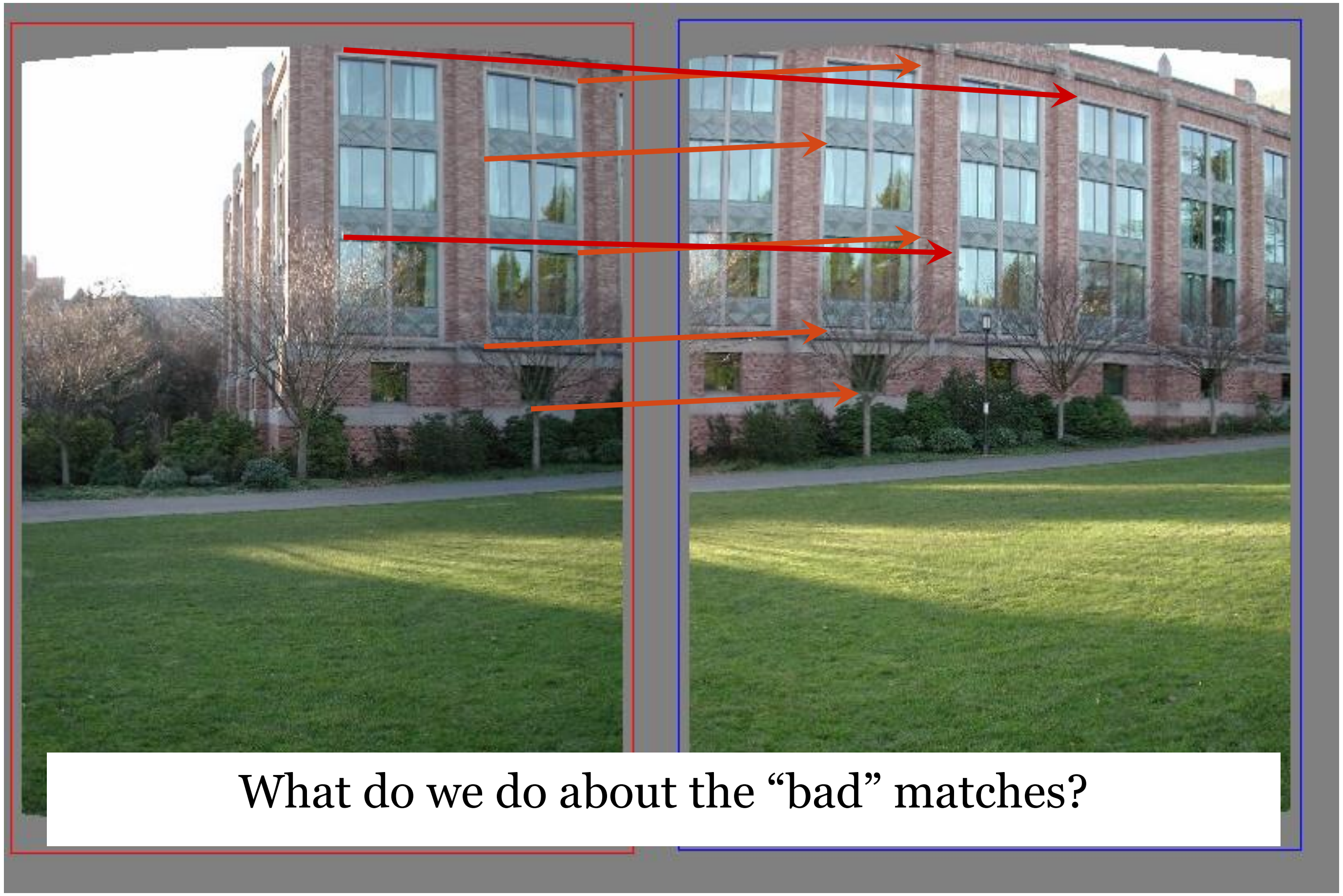


Image stitching steps

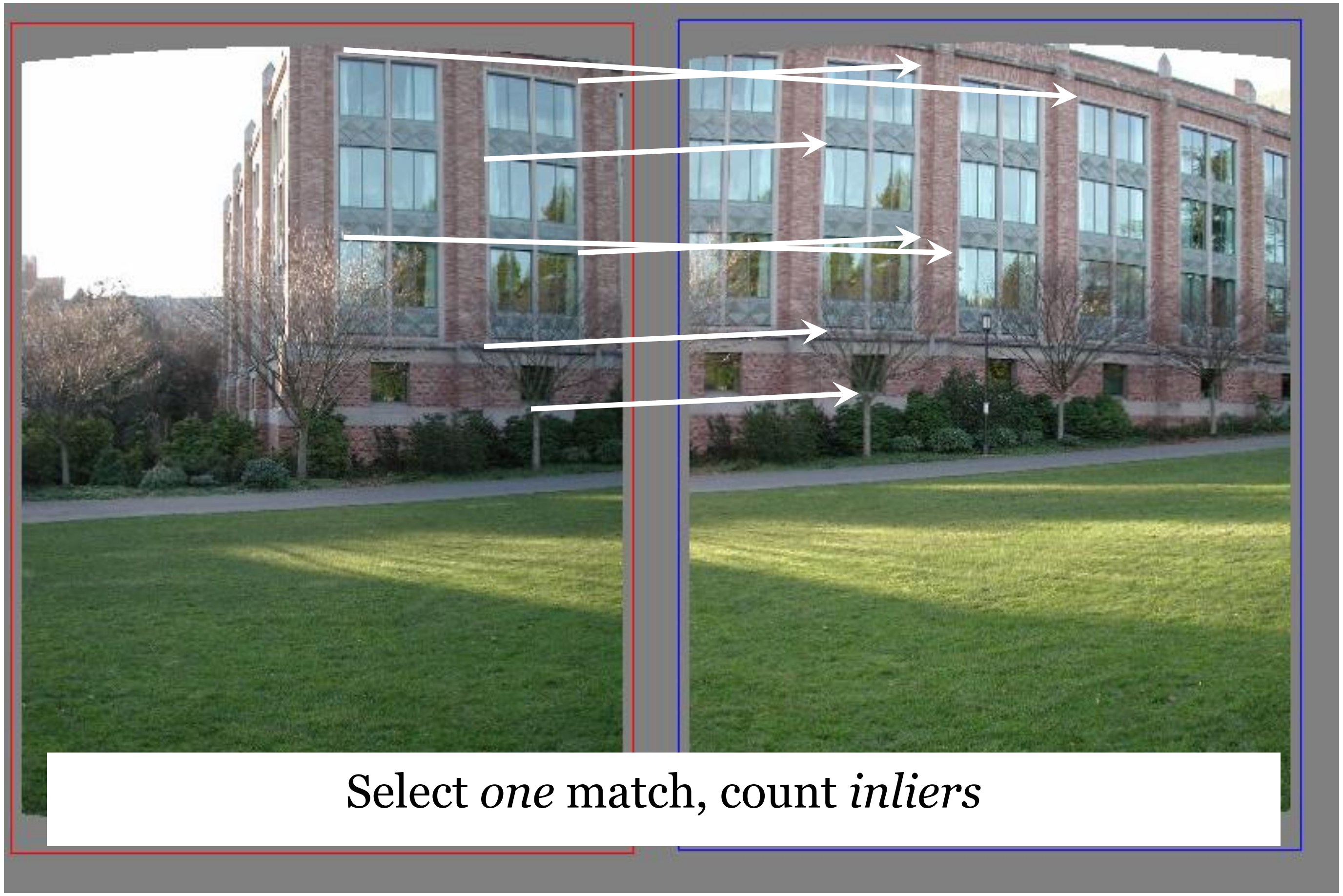
1. Take pictures on a tripod (or handheld)
2. Warp images to spherical coordinates
3. Extract features
4. Align neighboring pairs using RANSAC
5. Write out list of neighboring translations
6. Correct for drift
7. Read in warped images and blend them
8. Crop the result and import into a viewer

Matching features

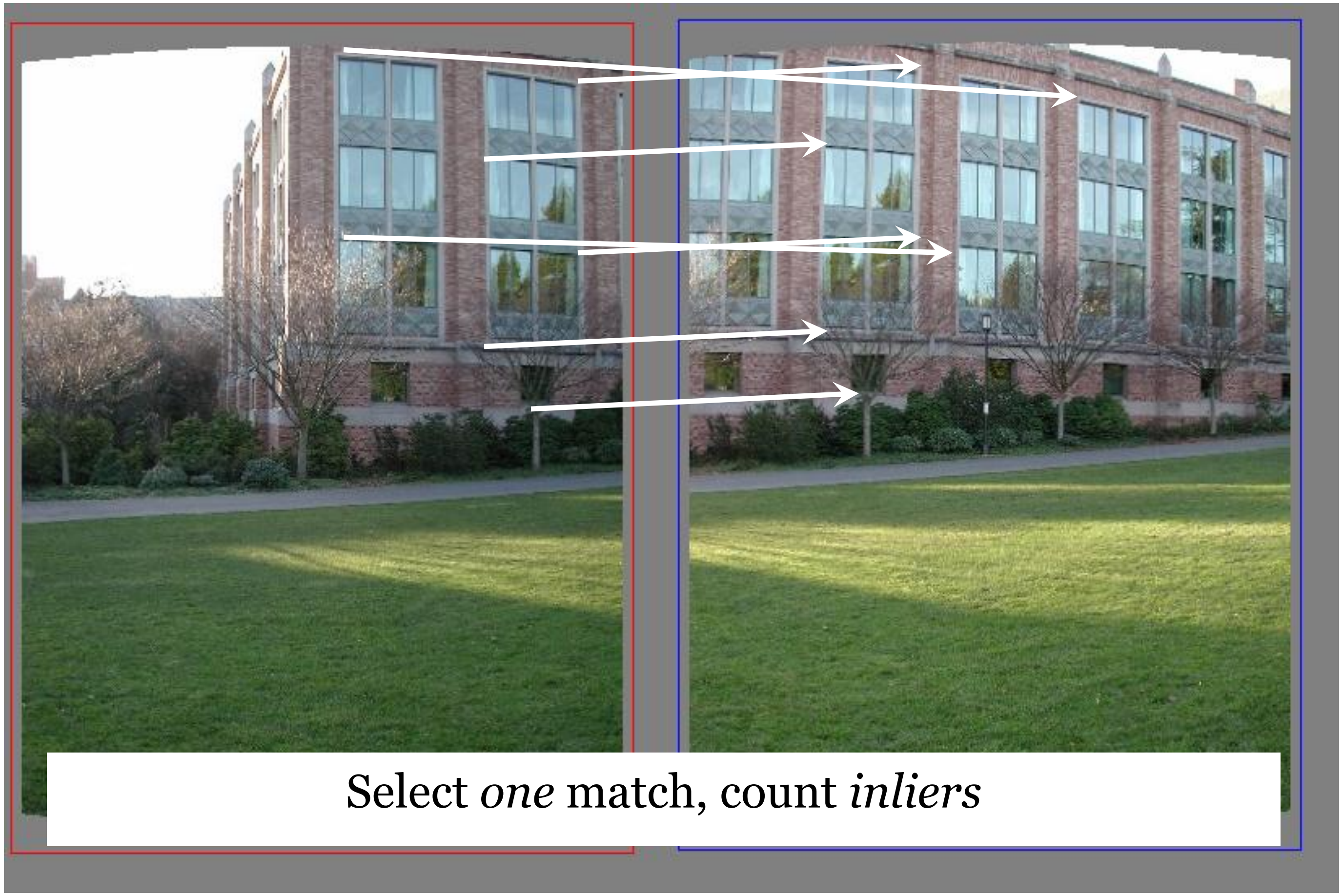


What do we do about the “bad” matches?

RAndom SAmple Consensus

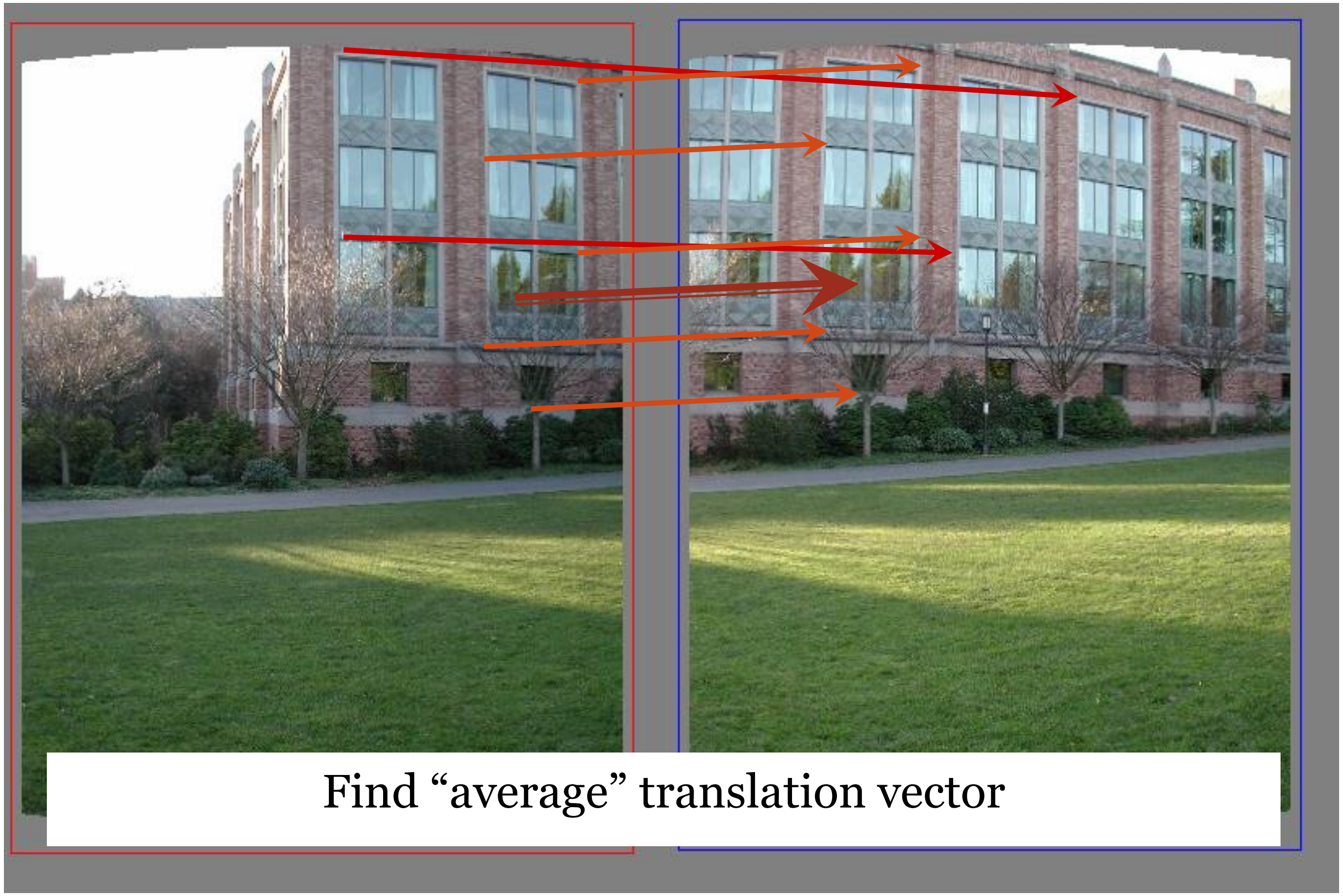


RAndom SAmple Consensus



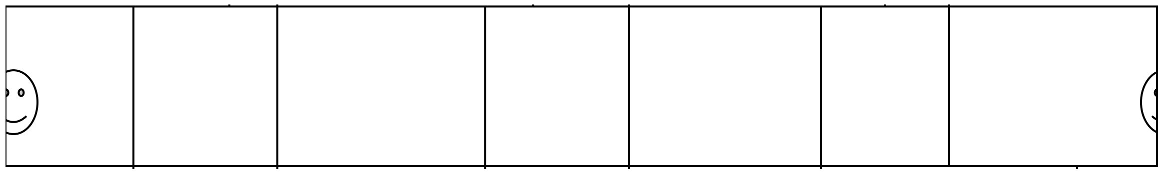
Select *one* match, count *inliers*

Least squares fit



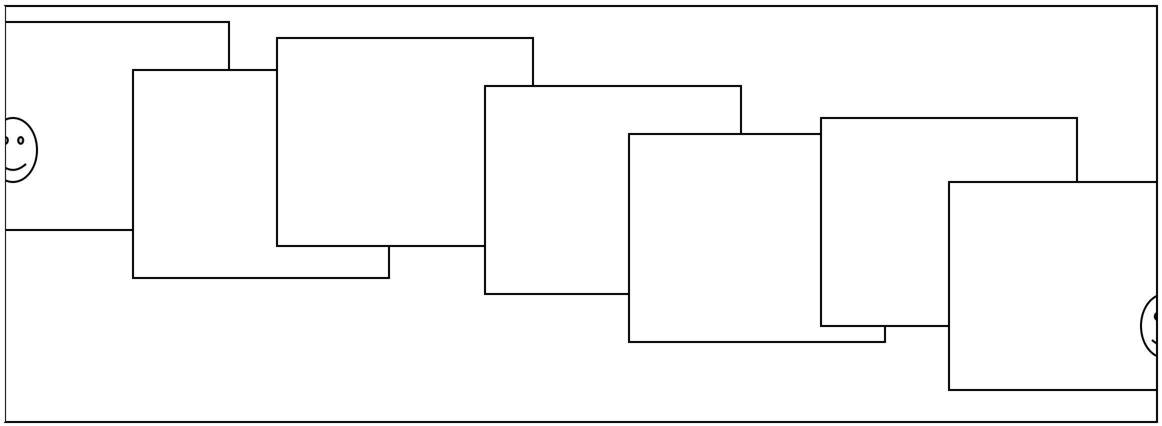
Find “average” translation vector

Assembling the panorama



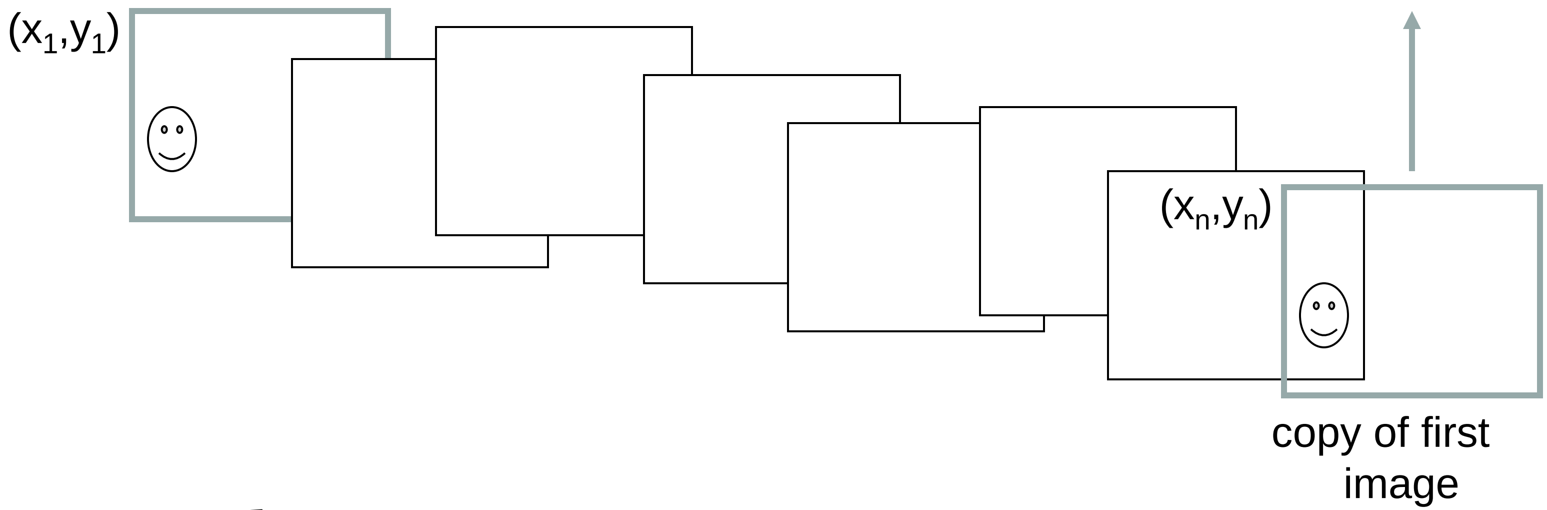
- Stitch pairs together, blend, then crop

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"

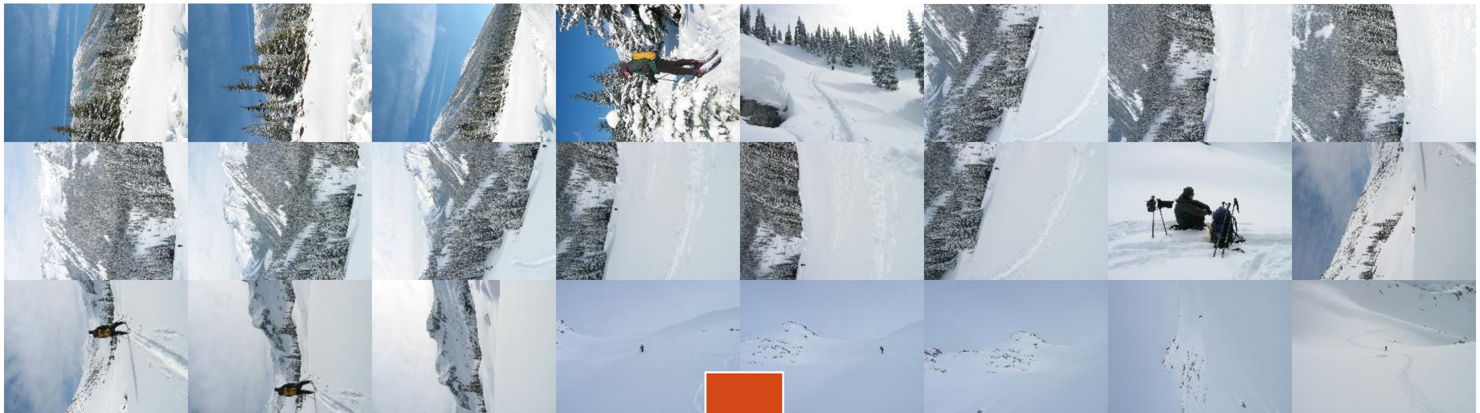
Full-view Panorama

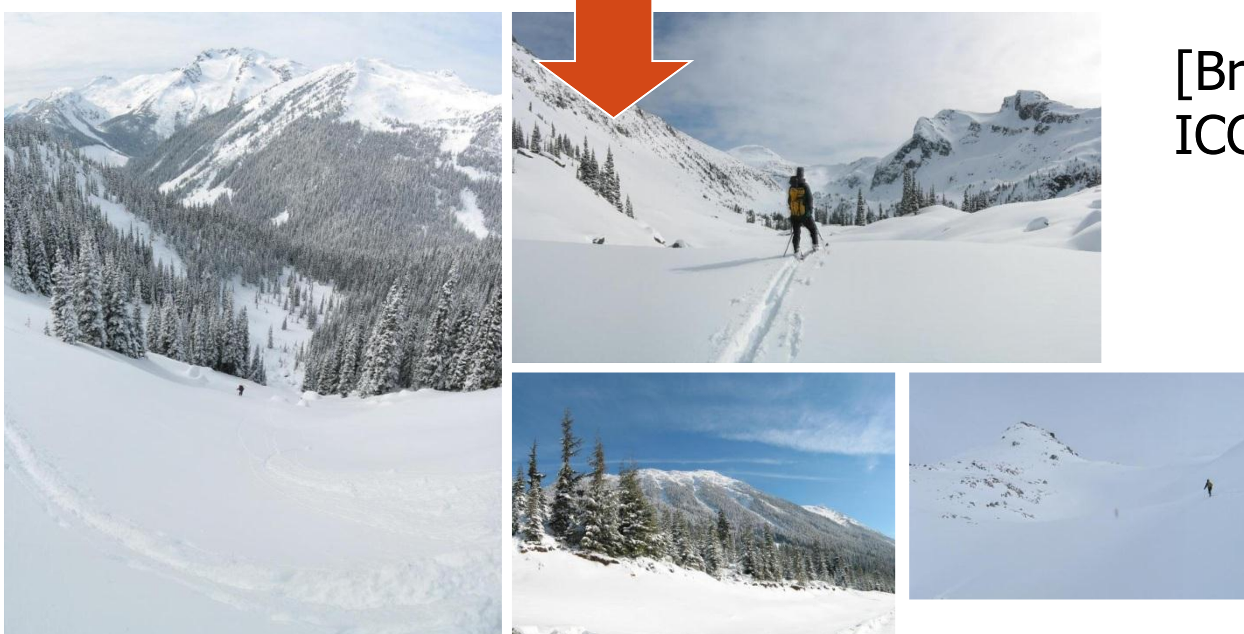
Texture Mapped Model

Global alignment

- Register *all* pairwise overlapping images
- Use a 3D rotation model (one R per image)
- Use direct alignment (patch centers) or feature based
- *Infer* overlaps based on previous matches (incremental)
- Optionally *discover* which images overlap other images using feature selection (RANSAC)

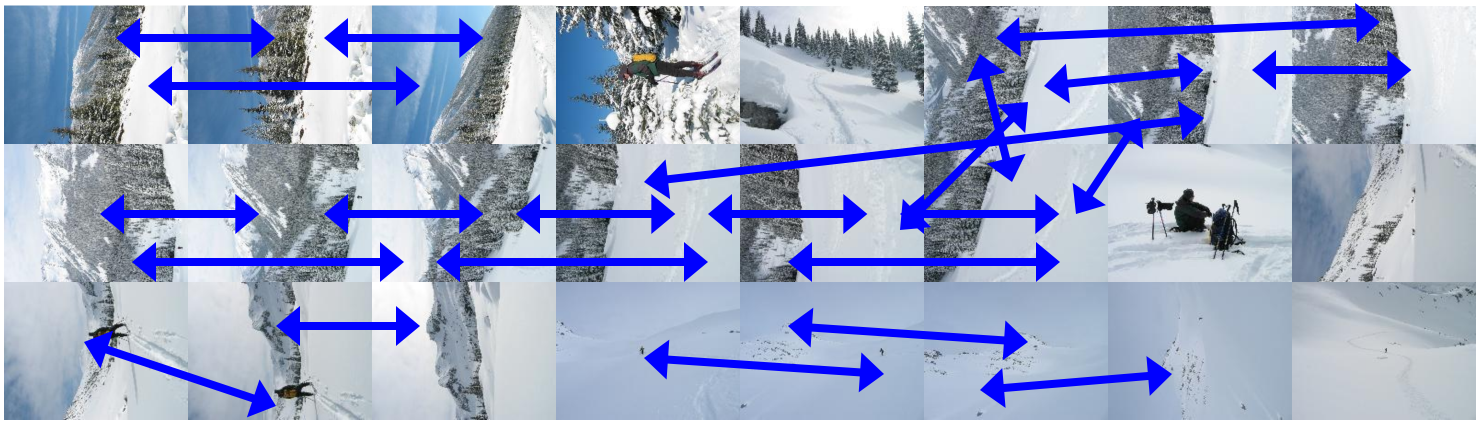
Recognizing Panoramas



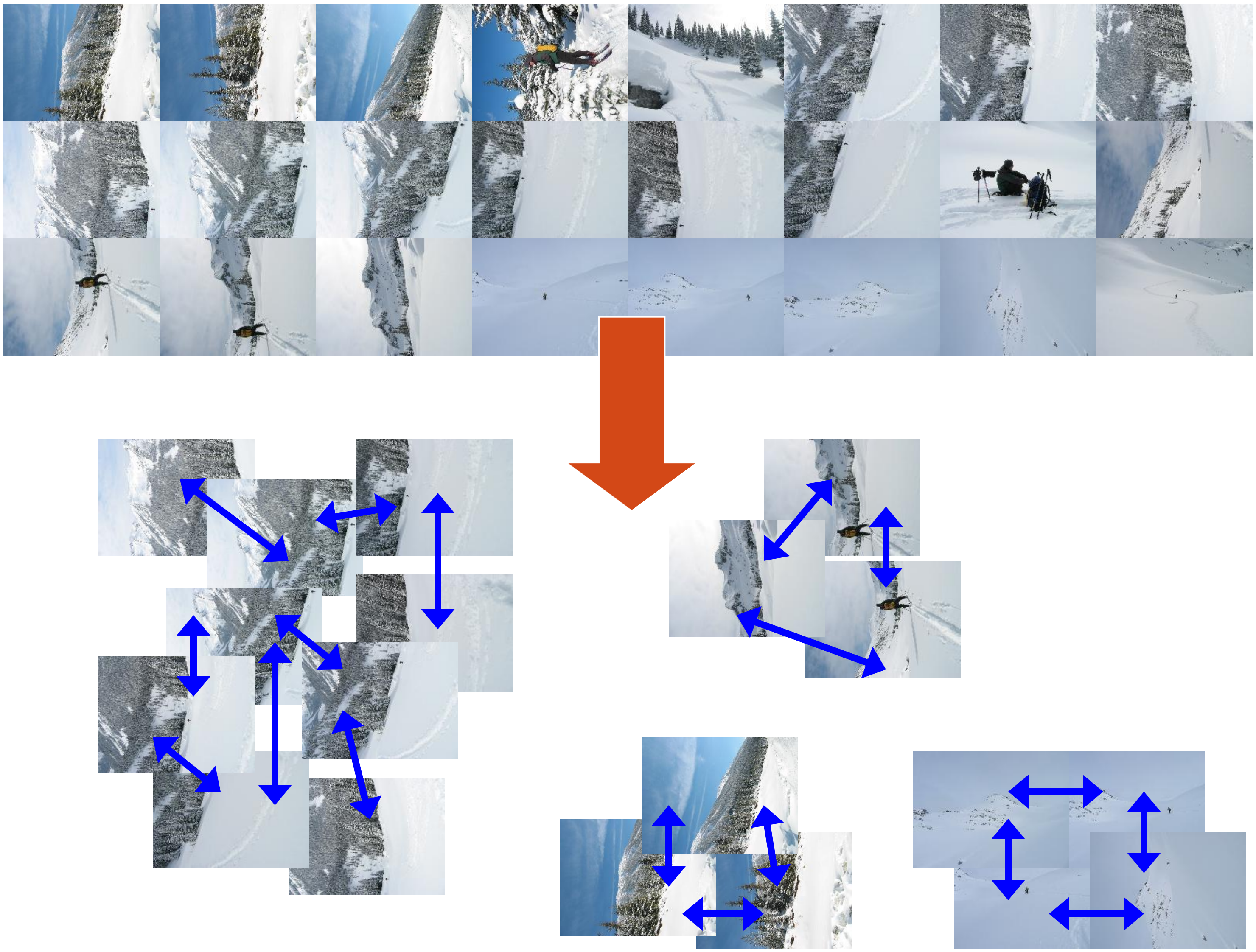


[Brown & Lowe, ICCV'03]

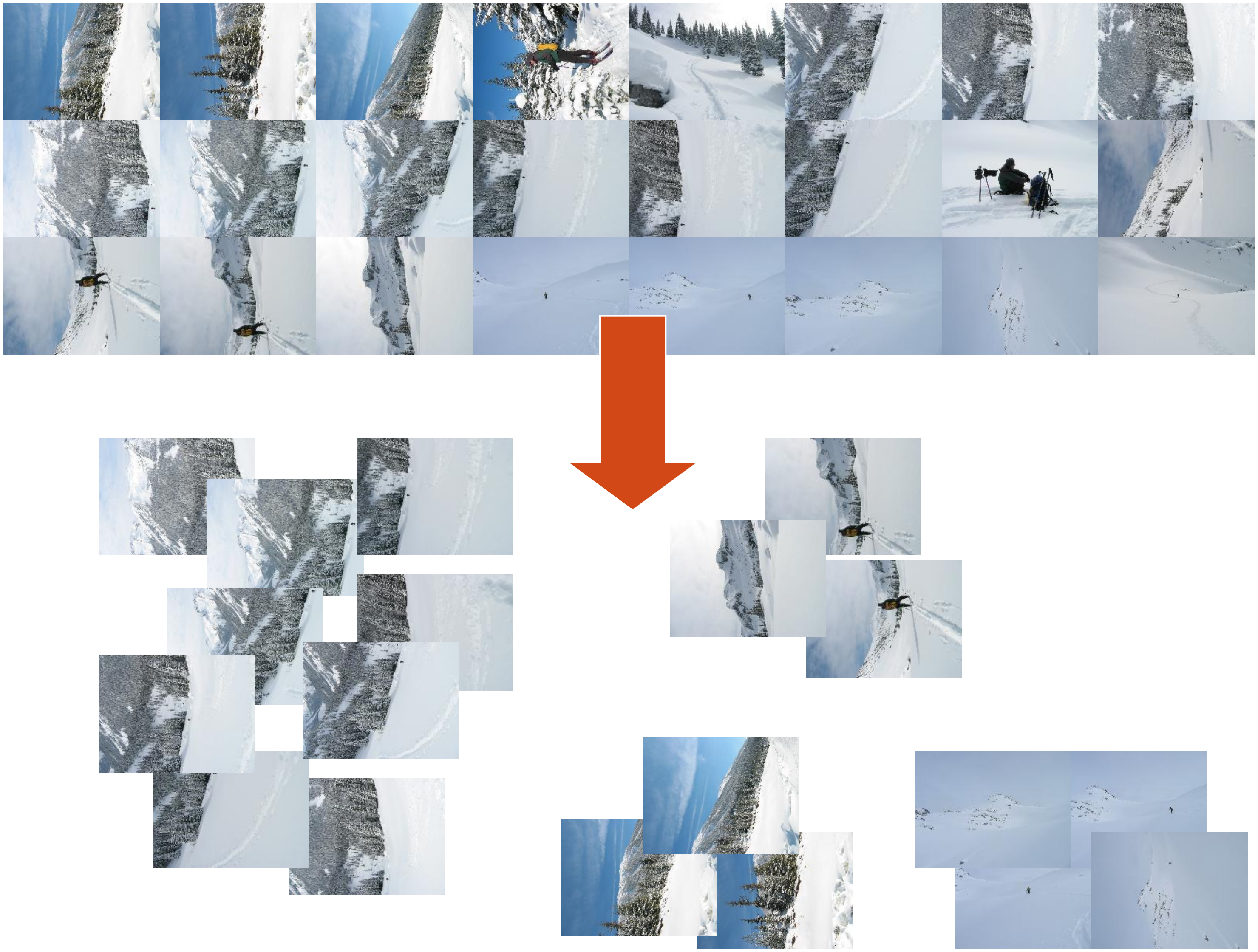
Finding the panoramas



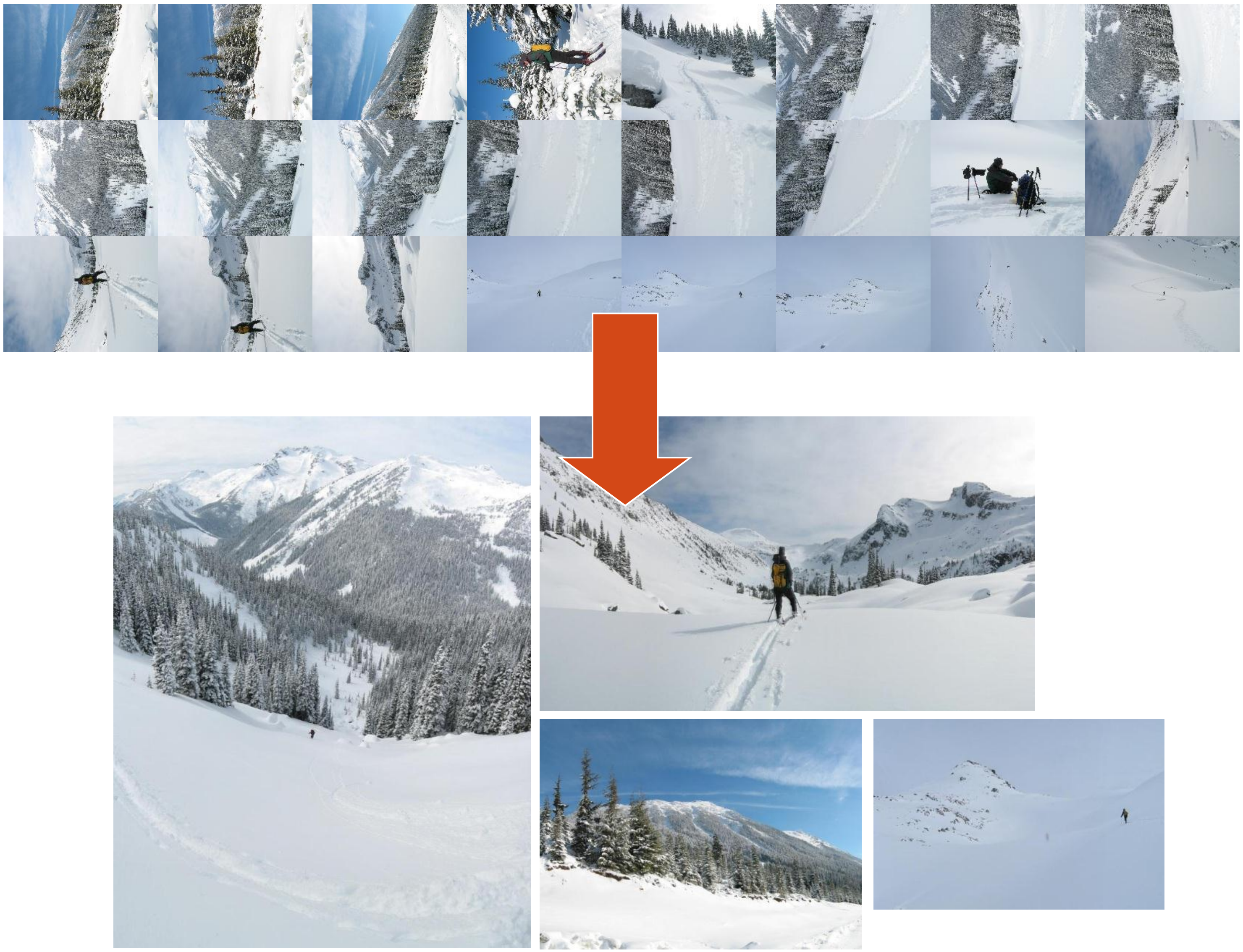
Finding the panoramas



Finding the panoramas



Finding the panoramas



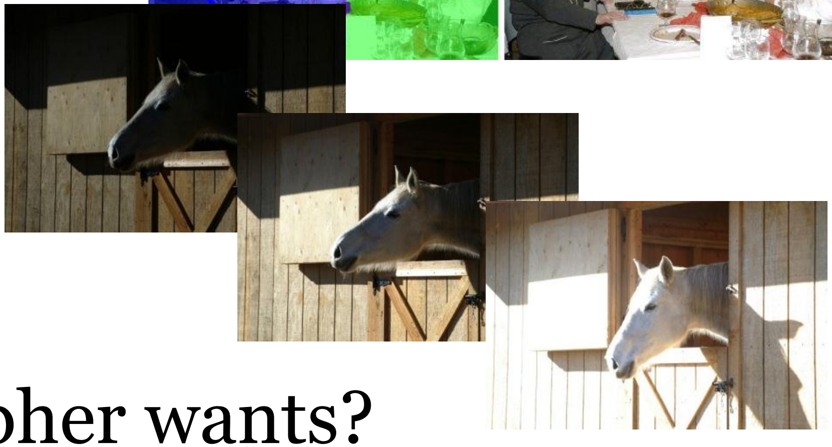
Fully automated 2D stitching

- Free copy from Microsoft Essentials
  - http://windows.microsoft.com/en-us/windows-live/photo-gallery-get-started

Final thought:
What is a "panorama"?

- Tracking a subject
  - Panorama
- Repeated (best) shots
  - Photo Fuse

- Multiple exposures
  - Photo Fuse



- "Infer" what photographer wants?