Image Alignment and Stitching

COS 429: Computer Vision

PRINCETON UNIVERSITY

Credits: S. Rusinkiewicz, D. Fouhey, R. Szeliski, S. Lazebnik, H. Sawhney

Motivation: panorama stitching

- We have two images — how do we combine them?

Motivation: panorama stitching

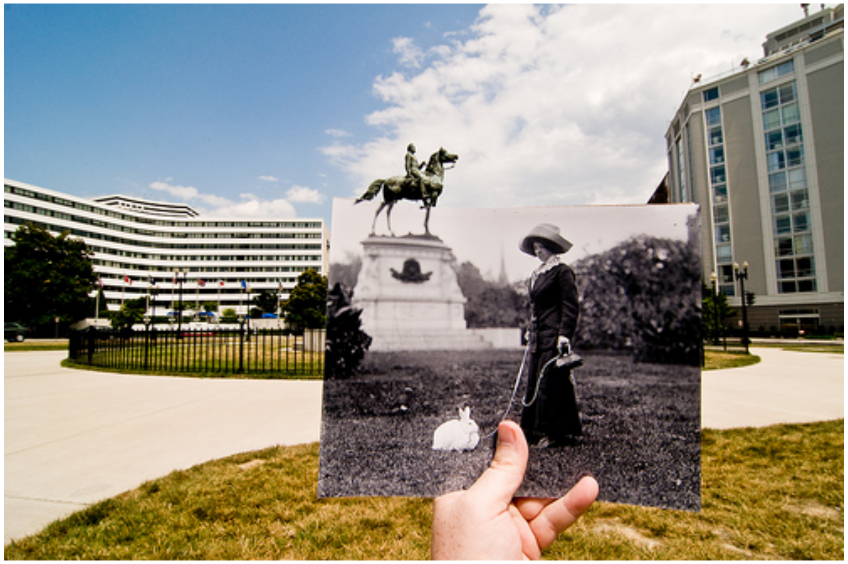
- We have two images — how do we combine them?

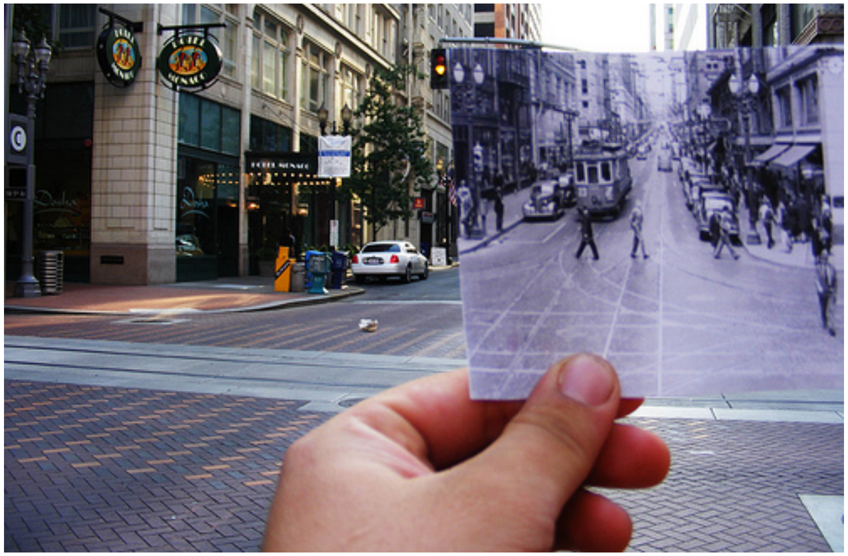
Panoramic Mosaics

Gigapixel Images

danielhartz.com

Applications – Look into the Past





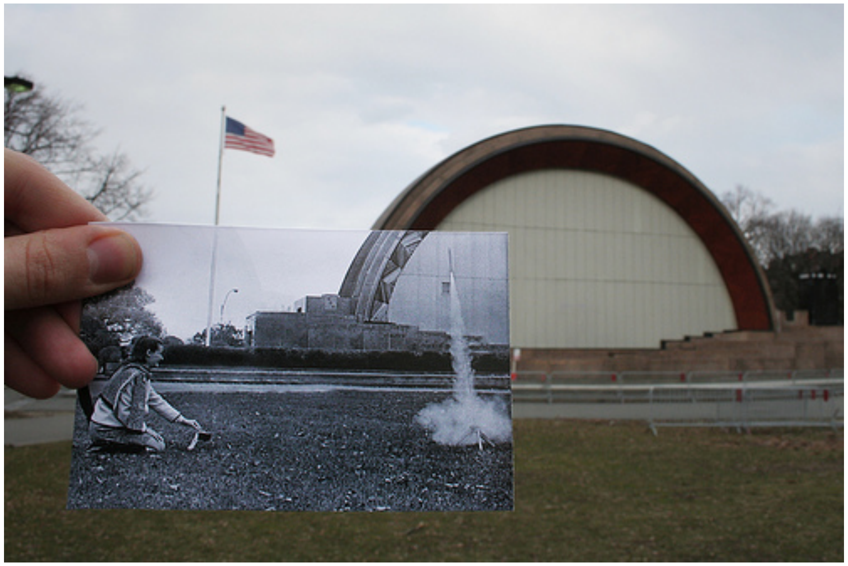


Image Alignment Applications

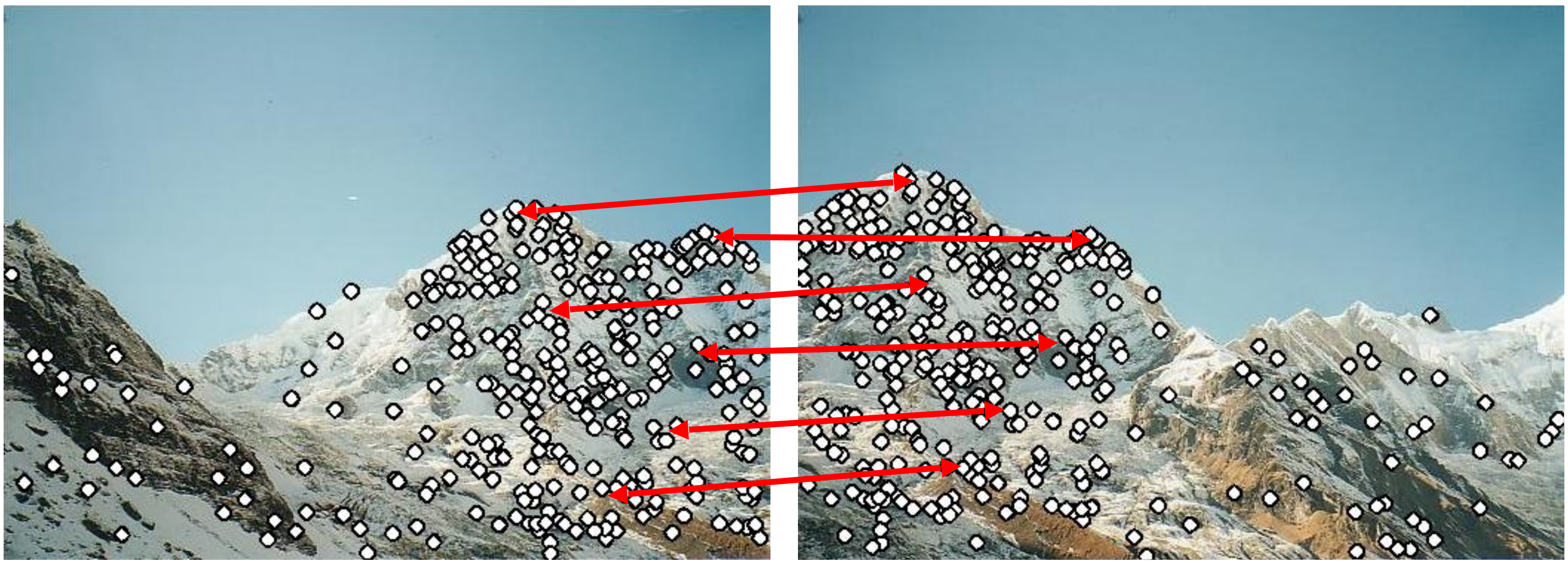
- Local alignment:
  - Tracking
  - Stereo
- Global alignment:
  - Camera jitter elimination
  - Image enhancement
  - Panoramic mosaicing

Image Alignment Approaches

- Direct alignment: see which image transformation maximizes similarity in overlap region
  - Often performed coarse-to-fine
- Feature-based alignment: find image transformation that matches keypoint locations

Panorama stitching

- We have two images — how do we combine them?



Step 1: extract keypoints
Step 2: match keypoint features

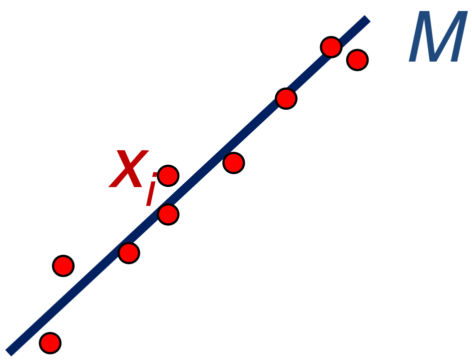
Panorama stitching

- We have two images — how do we combine them?

Step 1: extract keypoints
Step 2: match keypoint features
Step 3: align images

Alignment as Fitting

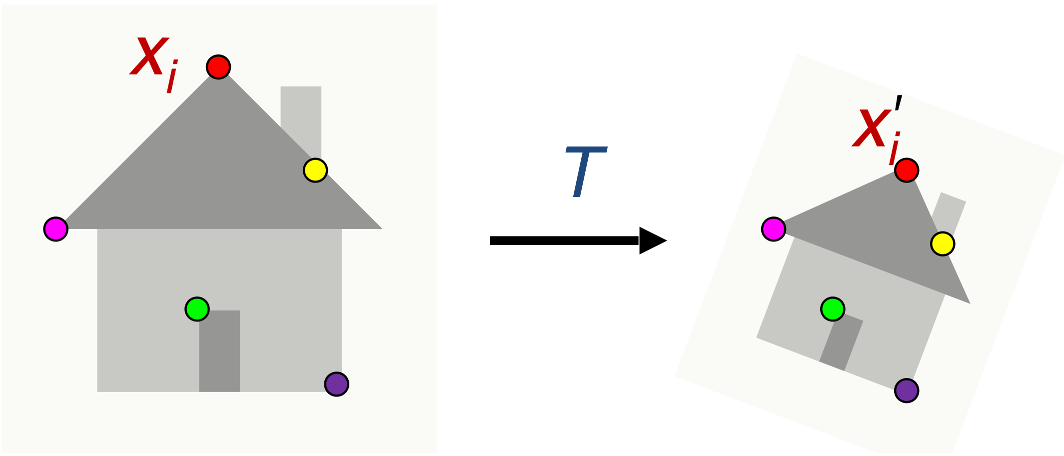
- **Previously:** fitting a model to features in one image



Find model M that minimizes

$$\sum_i \mathrm{L}(x_i; M)$$

- **Alignment:** fitting a model to a transformation between pairs of features (matches) in two images



Find transformation T that minimizes

$$\sum_i \mathrm{L}(T(x_i); x_i')$$

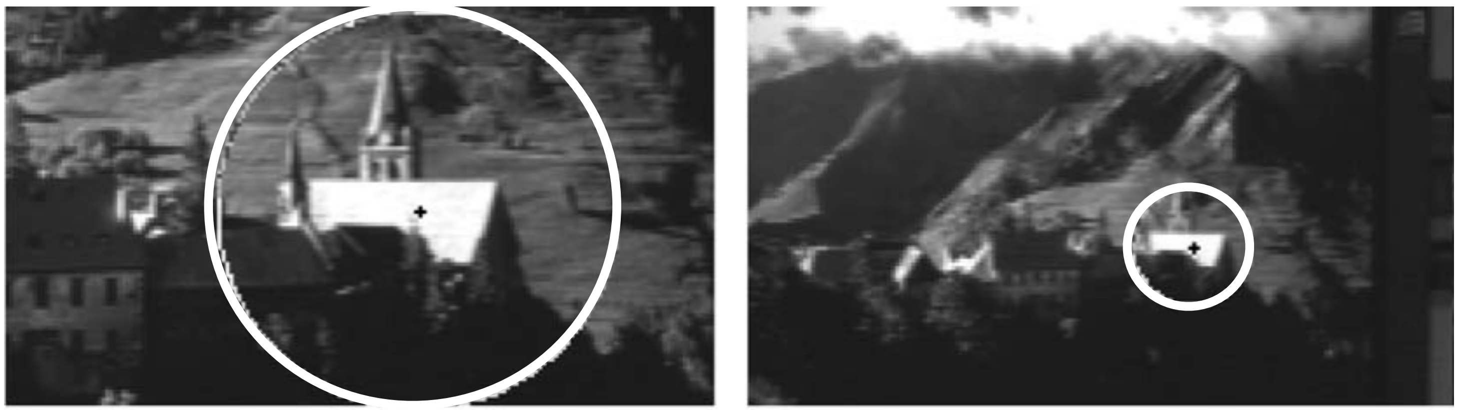
Feature-Based Alignment

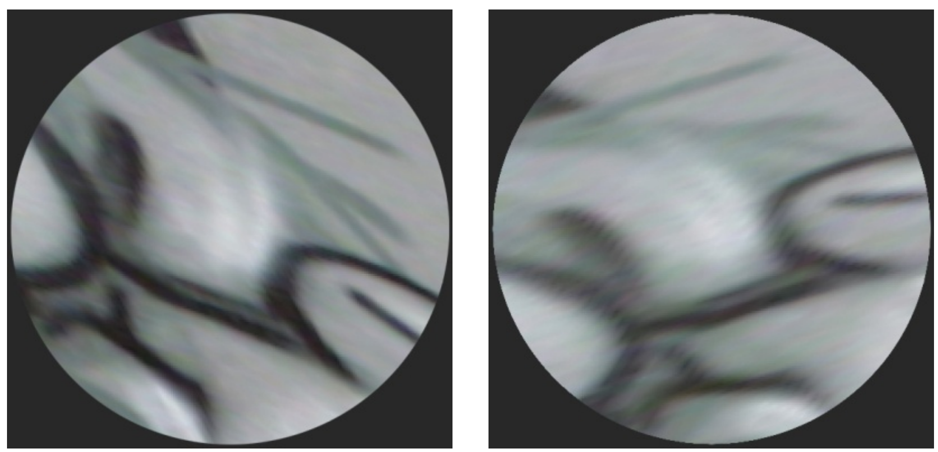
- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- **Fit to find best image transformation**
- Warp images according to transformation
- Blend images in overlapping regions

Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Recall: after blob detection and scale normalization

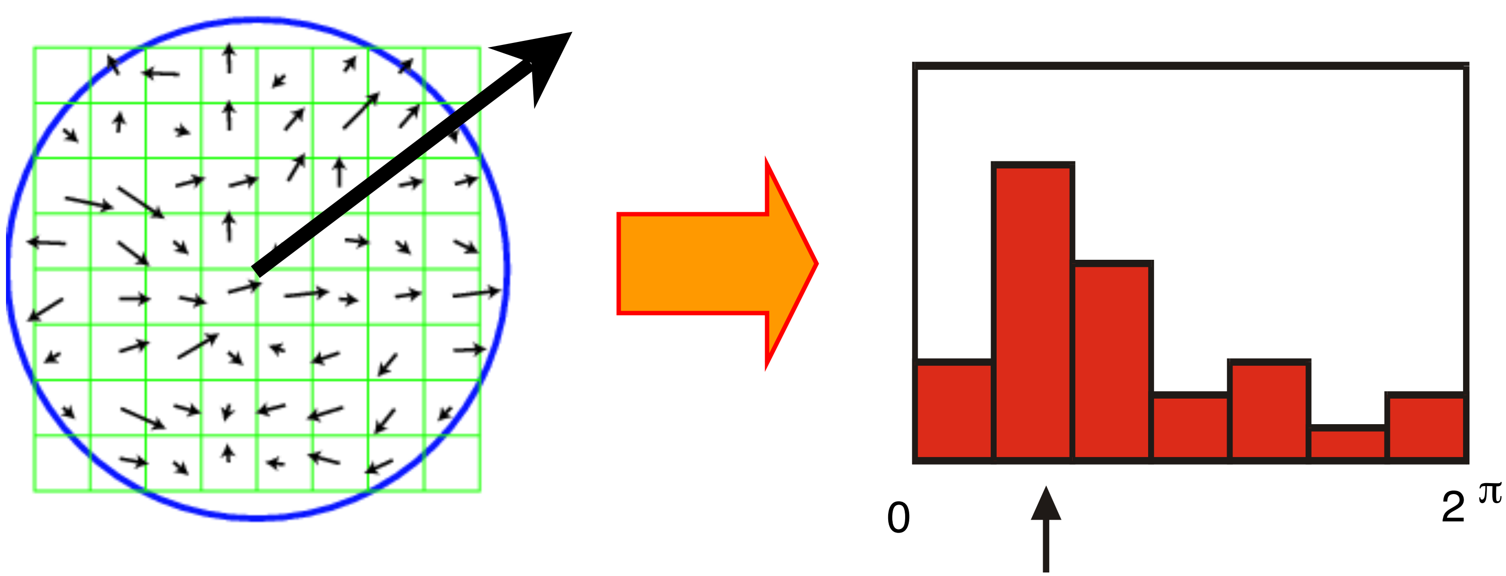




Slide: S. Lazebnik

Recall: eliminating rotation ambiguity

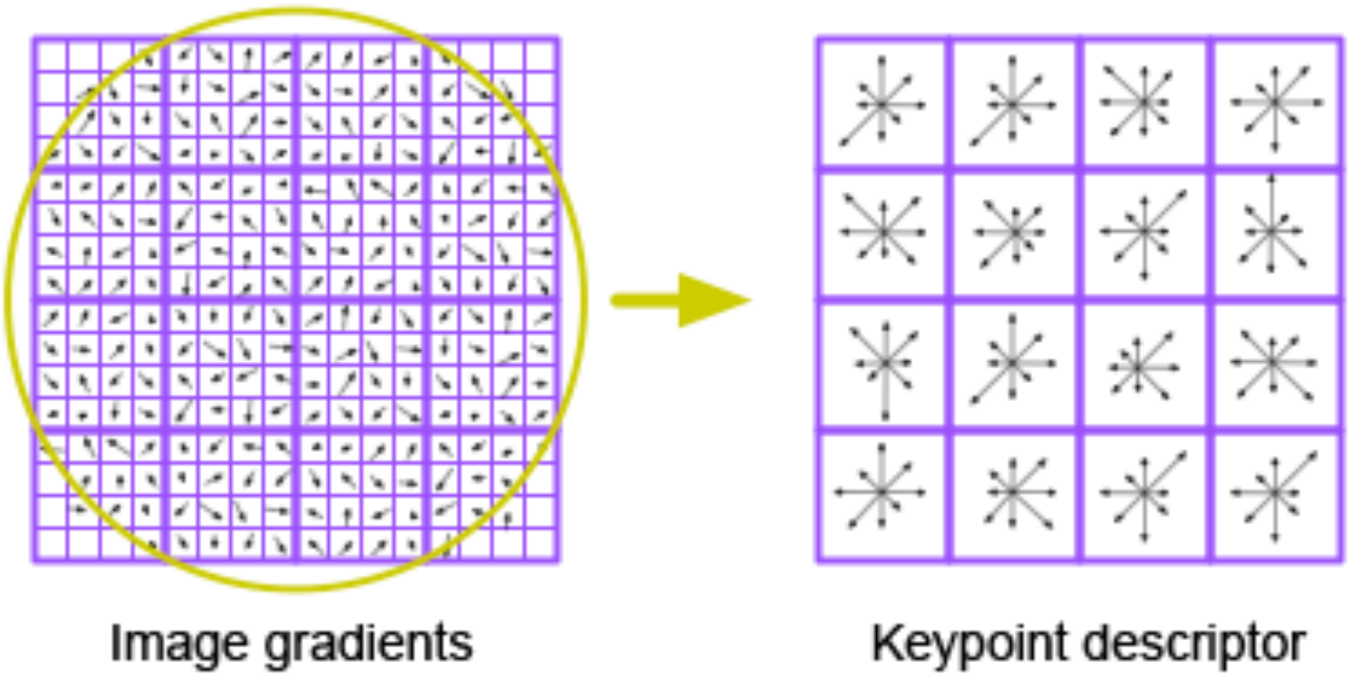
- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram



Slide: S. Lazebnik

Recall: SIFT Descriptor

- Divide 16×16 window into 4×4 grid of cells
- Compute an orientation histogram for each cell
  - 16 cells * 8 orientations = 128-dimensional descriptor



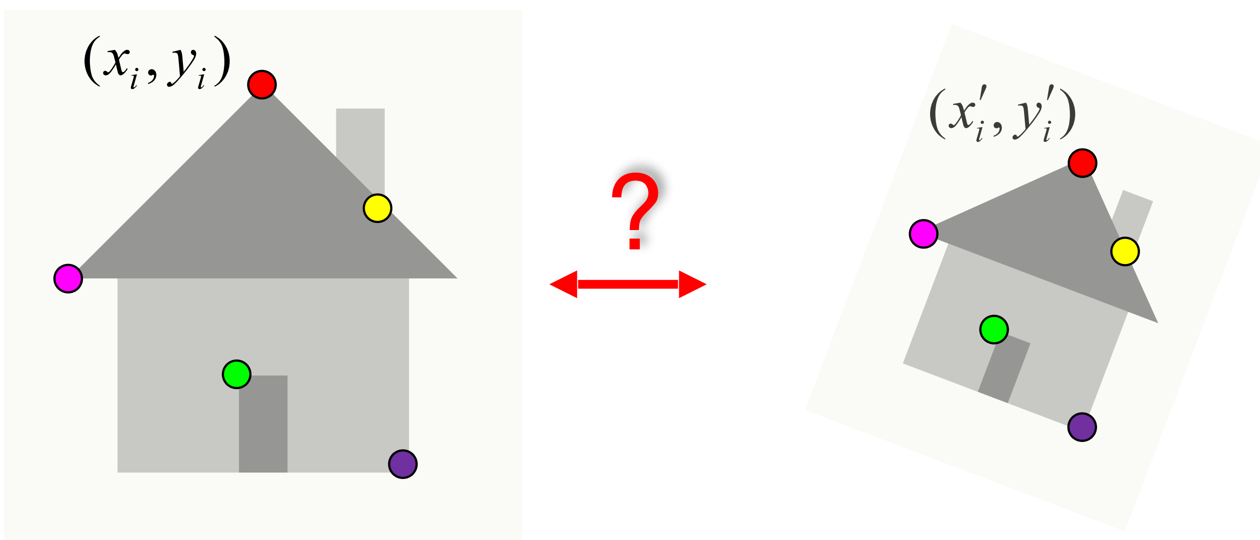
David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Candidate Matches

- For a given keypoint in image A, how to find candidate match in image B?

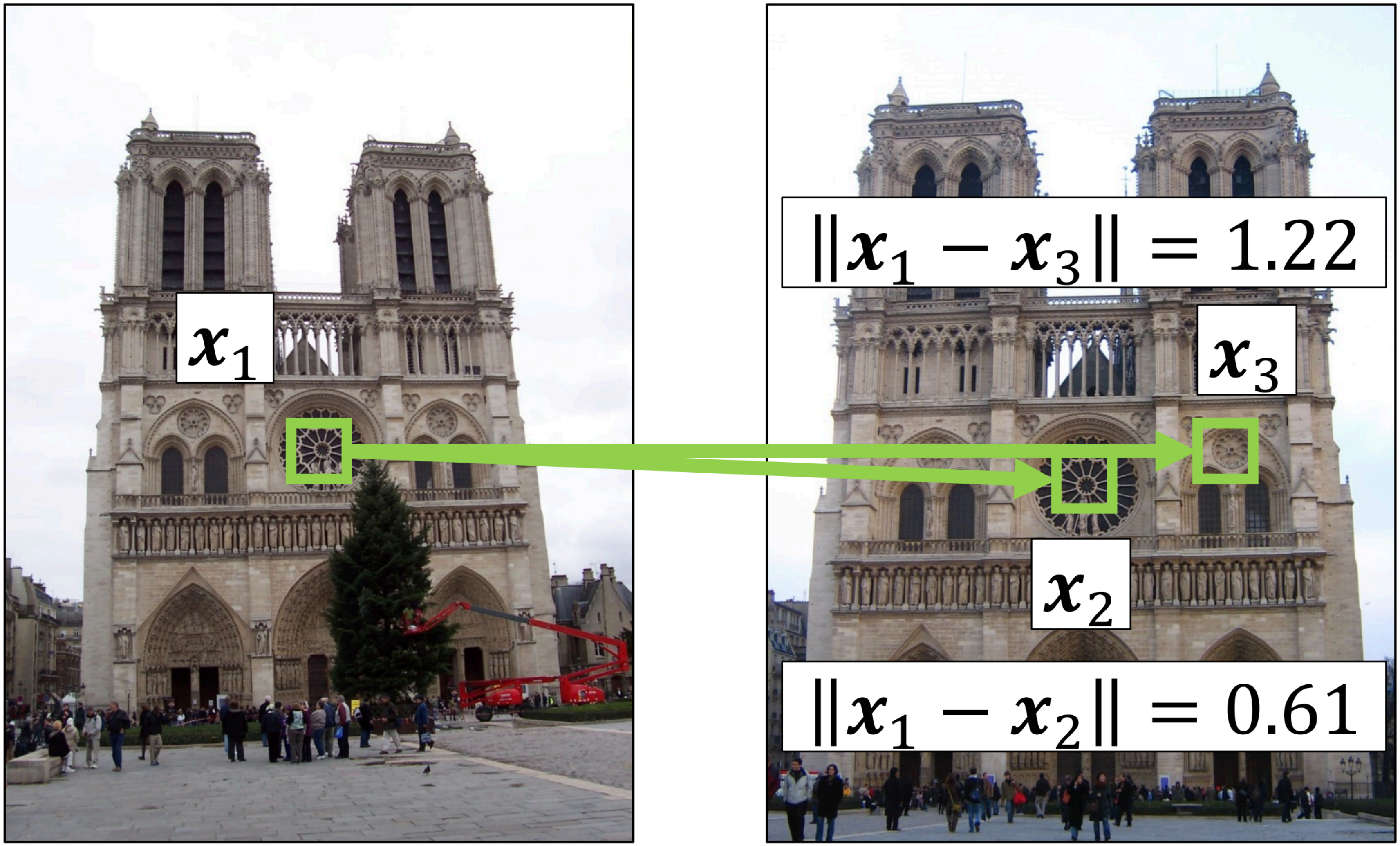


Candidate Matches

- For each SIFT descriptor in image A, find closest (according to Euclidean distance) in image B

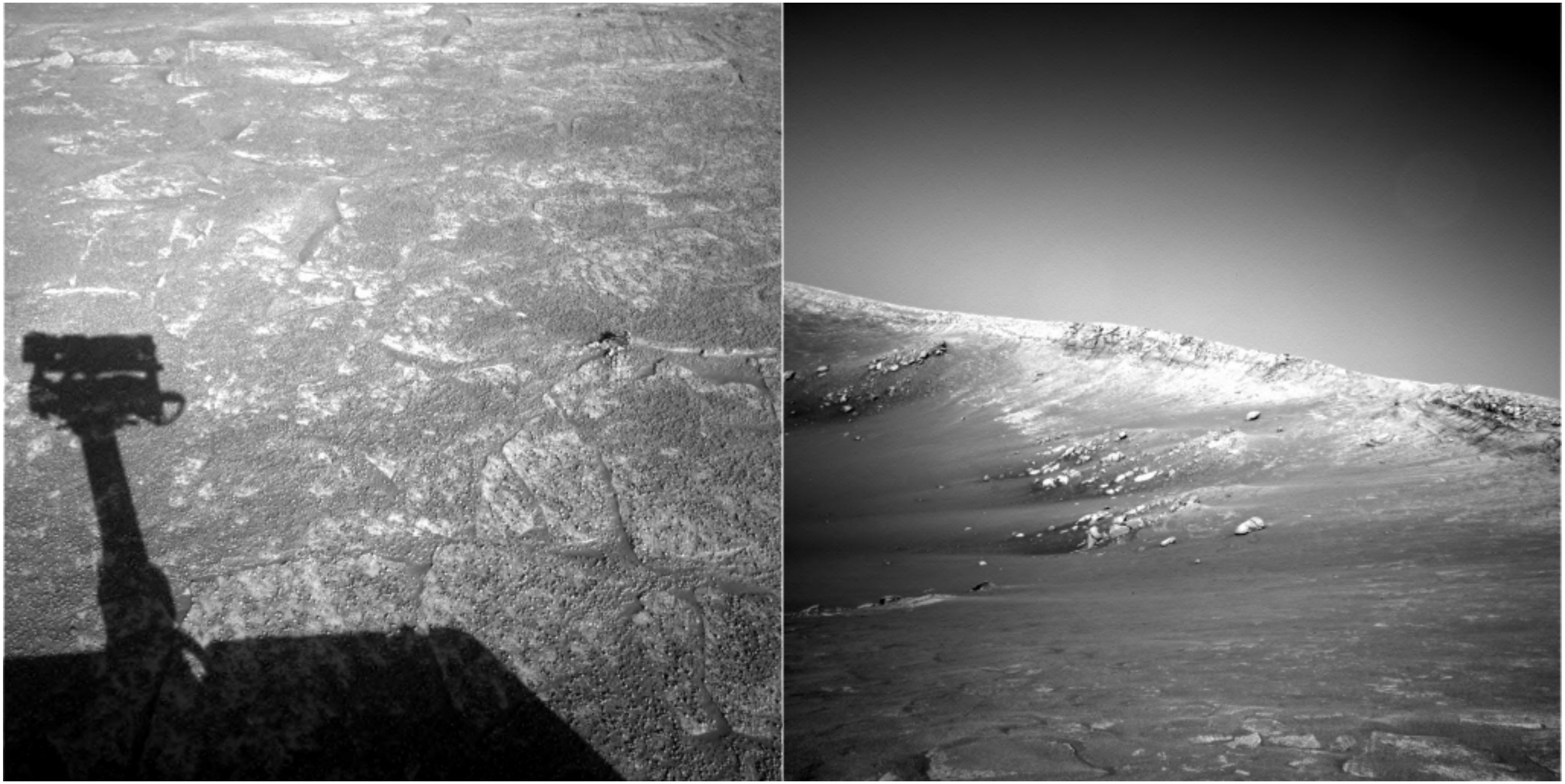
$$best\_match(x) = \underset{x_i'}{\operatorname{argmin}} \left\| x - x_i' \right\|^2$$

Instance matching



Credit: James Hays

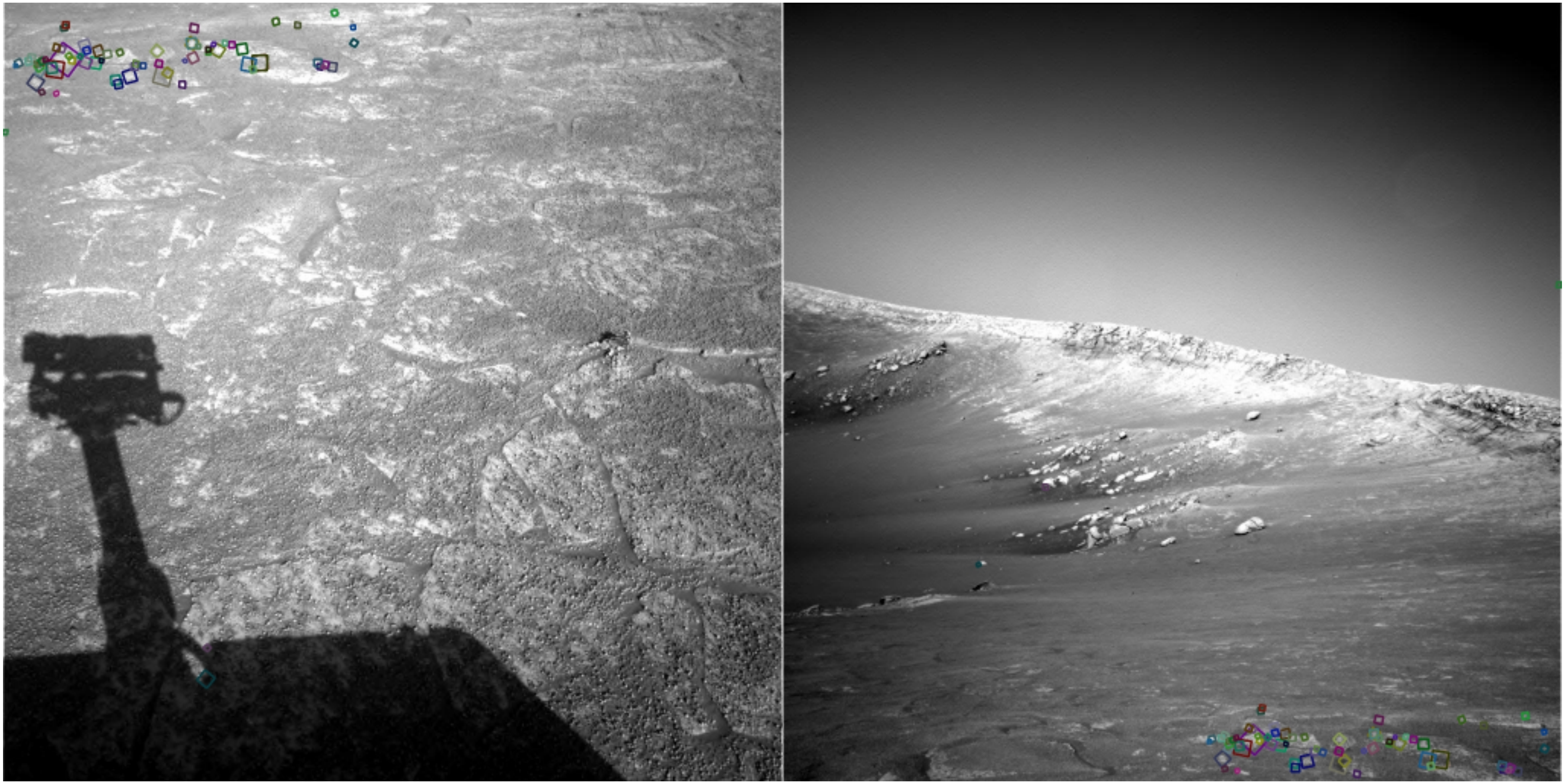
Example: instance matching



NASA Mars Rover images

Slide credit: S. Lazebnik

Example: instance matching (look for tiny colored squares)

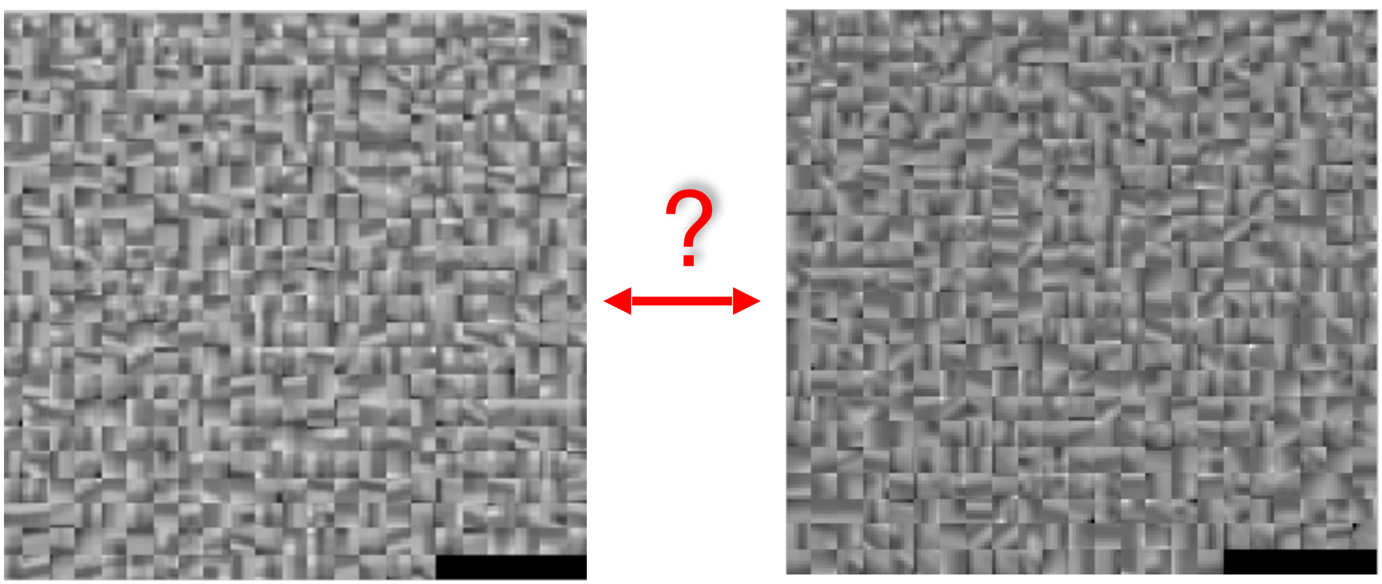


NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

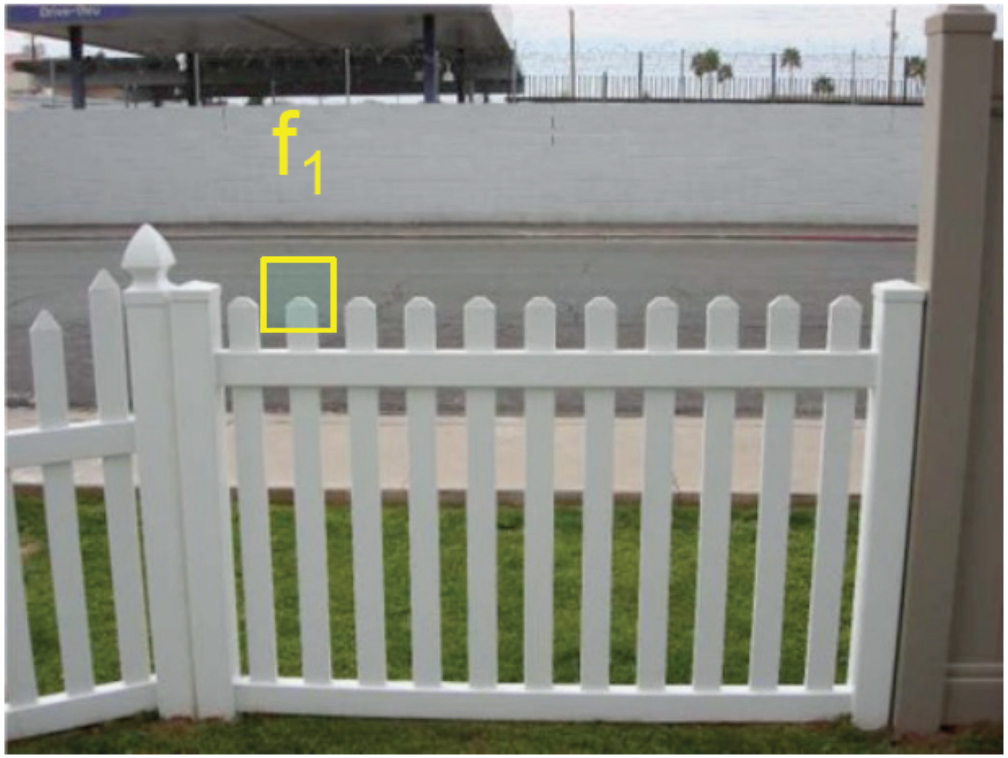
Slide credit: S. Lazebnik

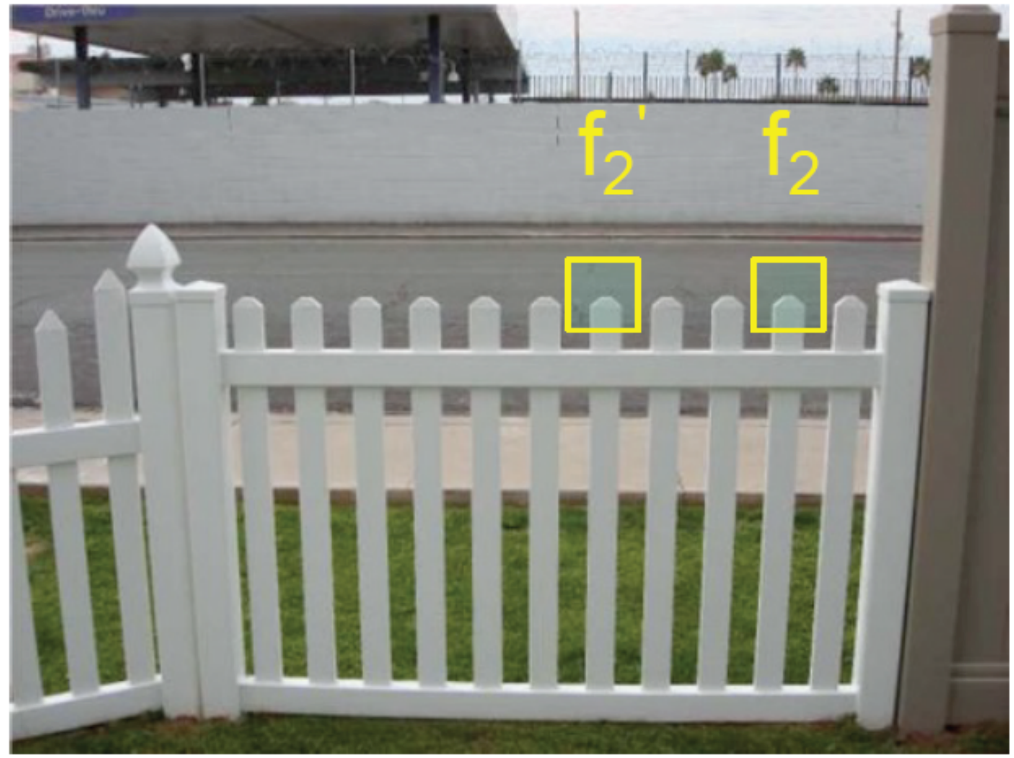
Candidate Matches

- For a given keypoint in image A, how to find candidate match in image B?
  - What if there are a lot of keypoints?



Problem: Ambiguous Correspondences





Source: Y. Furukawa

Candidate Matches

- For each SIFT descriptor in image A, find closest (according to Euclidean distance) in image B

$$best\_match(x) = \operatorname*{argmin}_{x_i'} \left\| x - x_i' \right\|^2$$

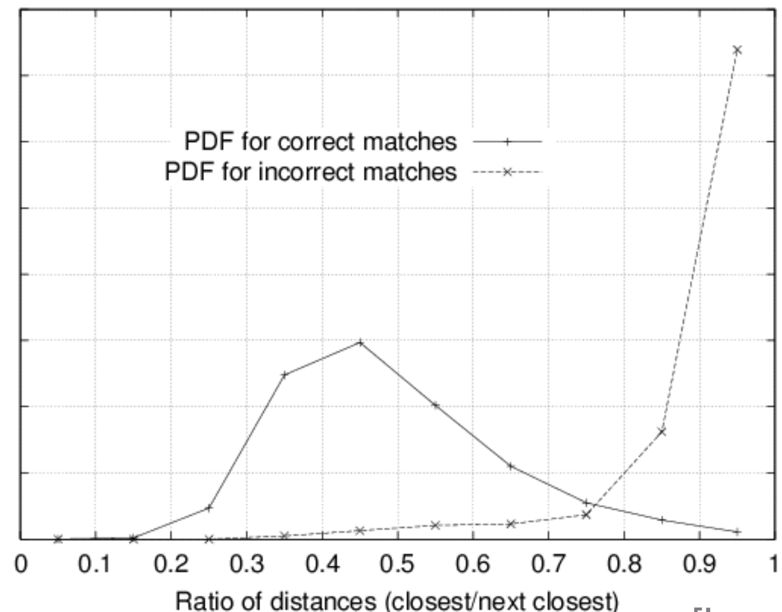
- Refinement: mutual best match
  - *x'* is most similar to *x* and *x* is most similar to *x'*

Candidate Matches

- For each SIFT descriptor in image A, find closest (according to Euclidean distance) in image B

$$best\_match(x) = \underset{x_i'}{\operatorname{argmin}} \left\| x - x_i' \right\|^2$$

- Refinement: mutual best match
  - *x'* is most similar to *x* and *x* is most similar to *x'*
- Refinement: best match is much better than second-best
  - Ratio of closest to second-closest distance is high for non-distinctive features
  - Threshold ratio of e.g. 0.8



[Lowe]

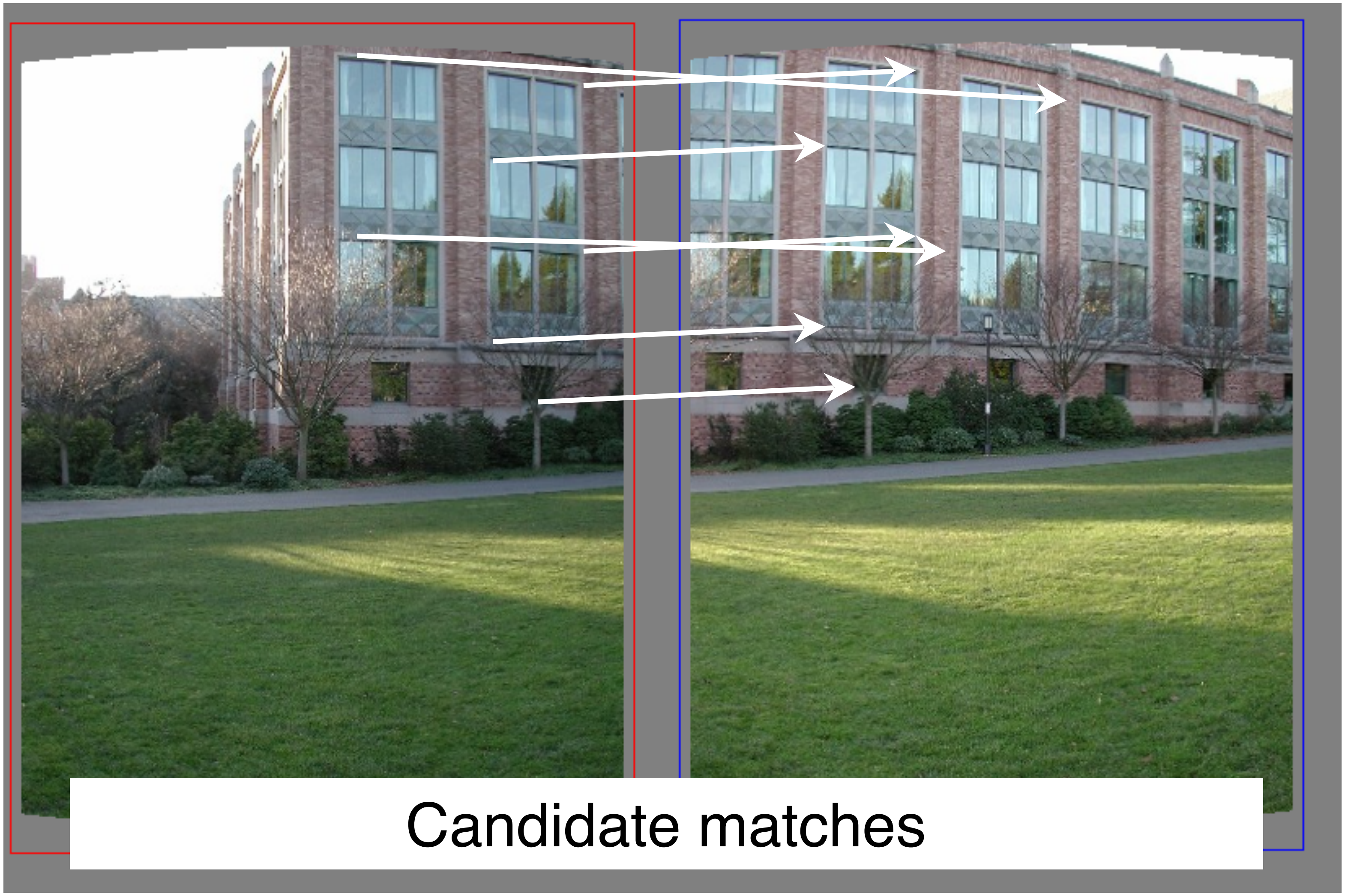
Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Review: RANSAC

- Set of candidate matches contains many outliers
- RANSAC loop:
  - Randomly select a minimal set of matches
  - Compute transformation from seed group
  - Find inliers to this transformation
  - Keep the transformation with the largest number of inliers
- At end, re-estimate best transform using all inliers

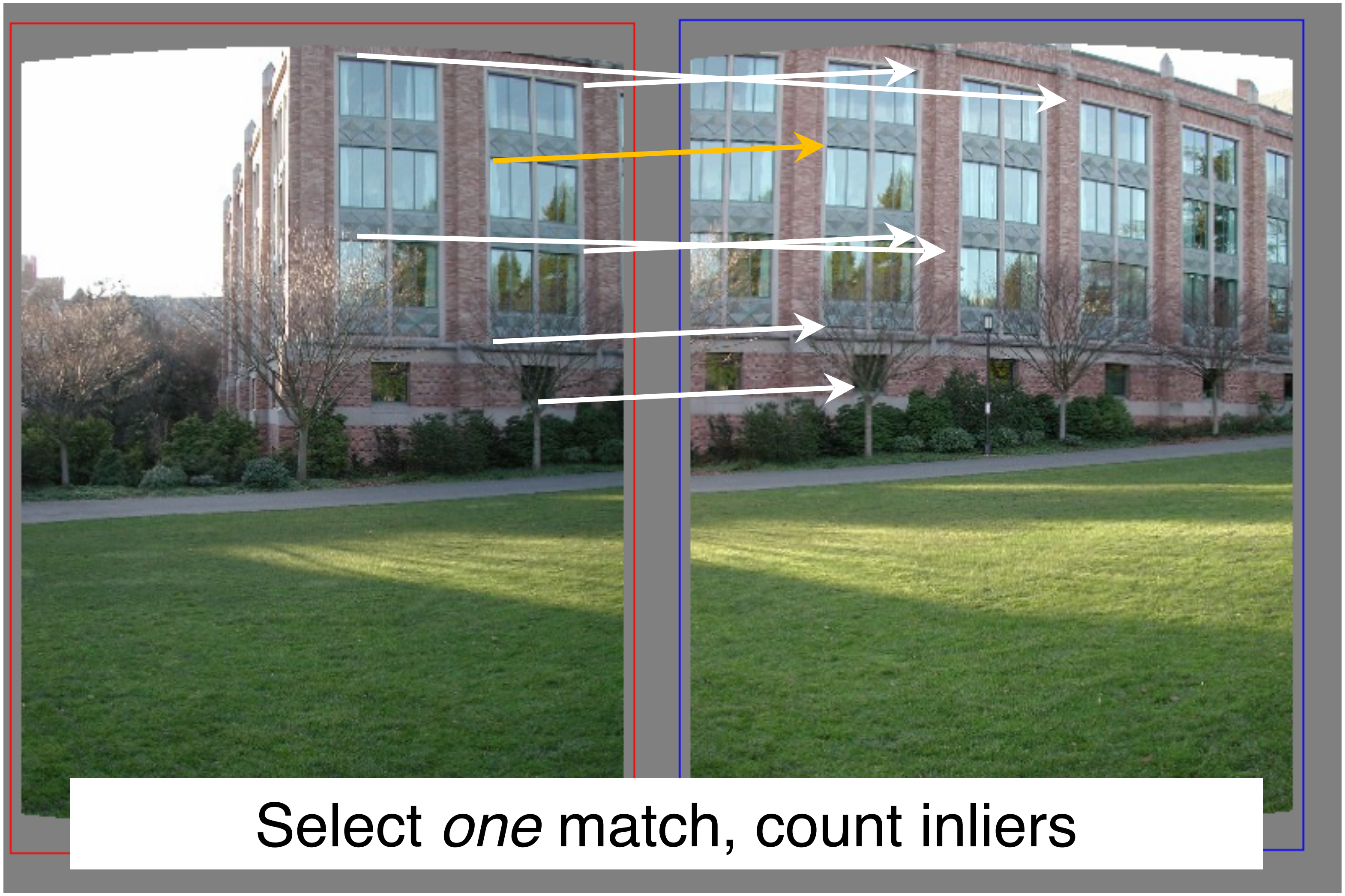
RANSAC: Translation Only



Candidate matches

[Szeliski]

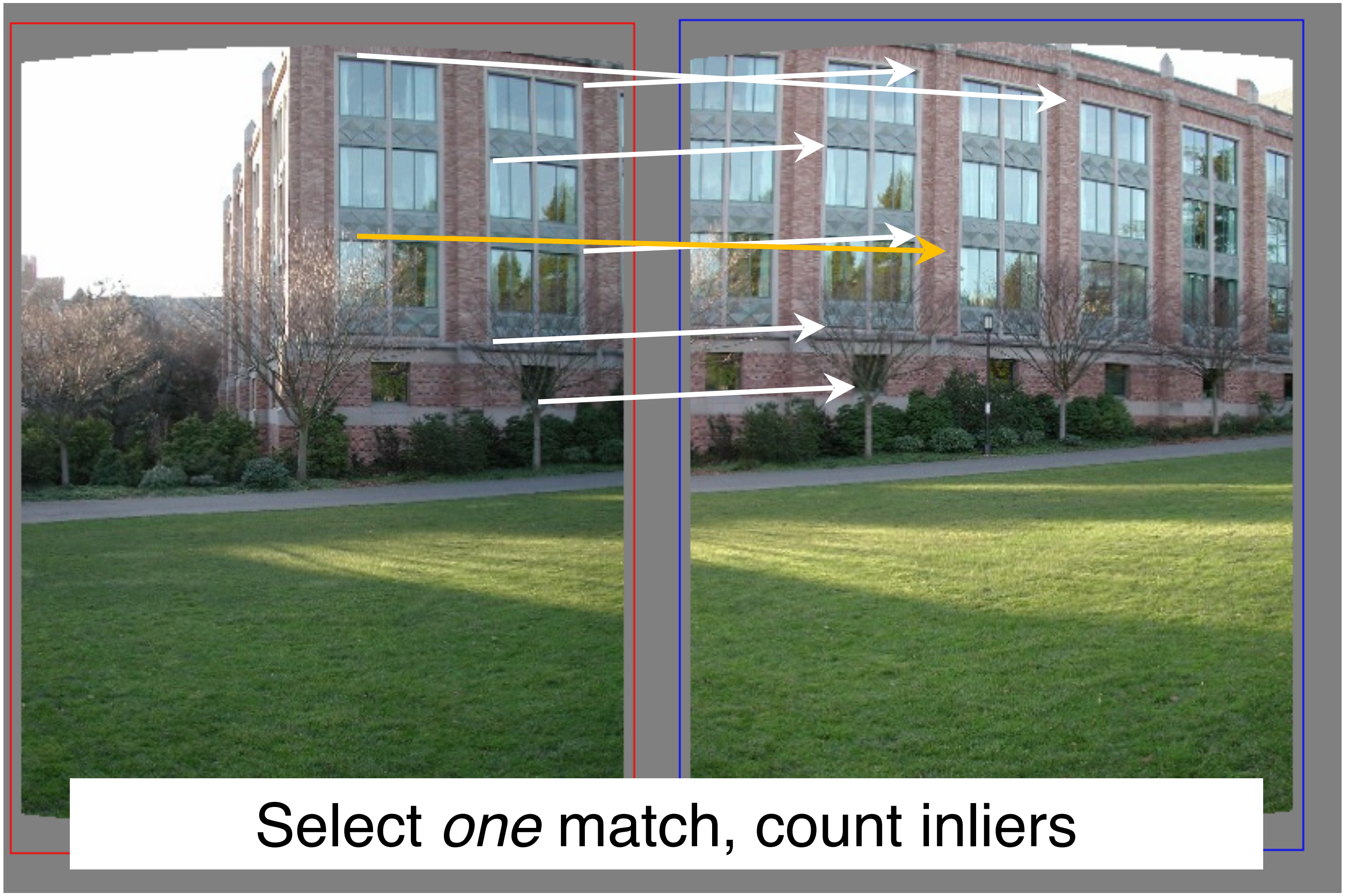
RANSAC: Translation Only



Select *one* match, count inliers

[Szeliski]

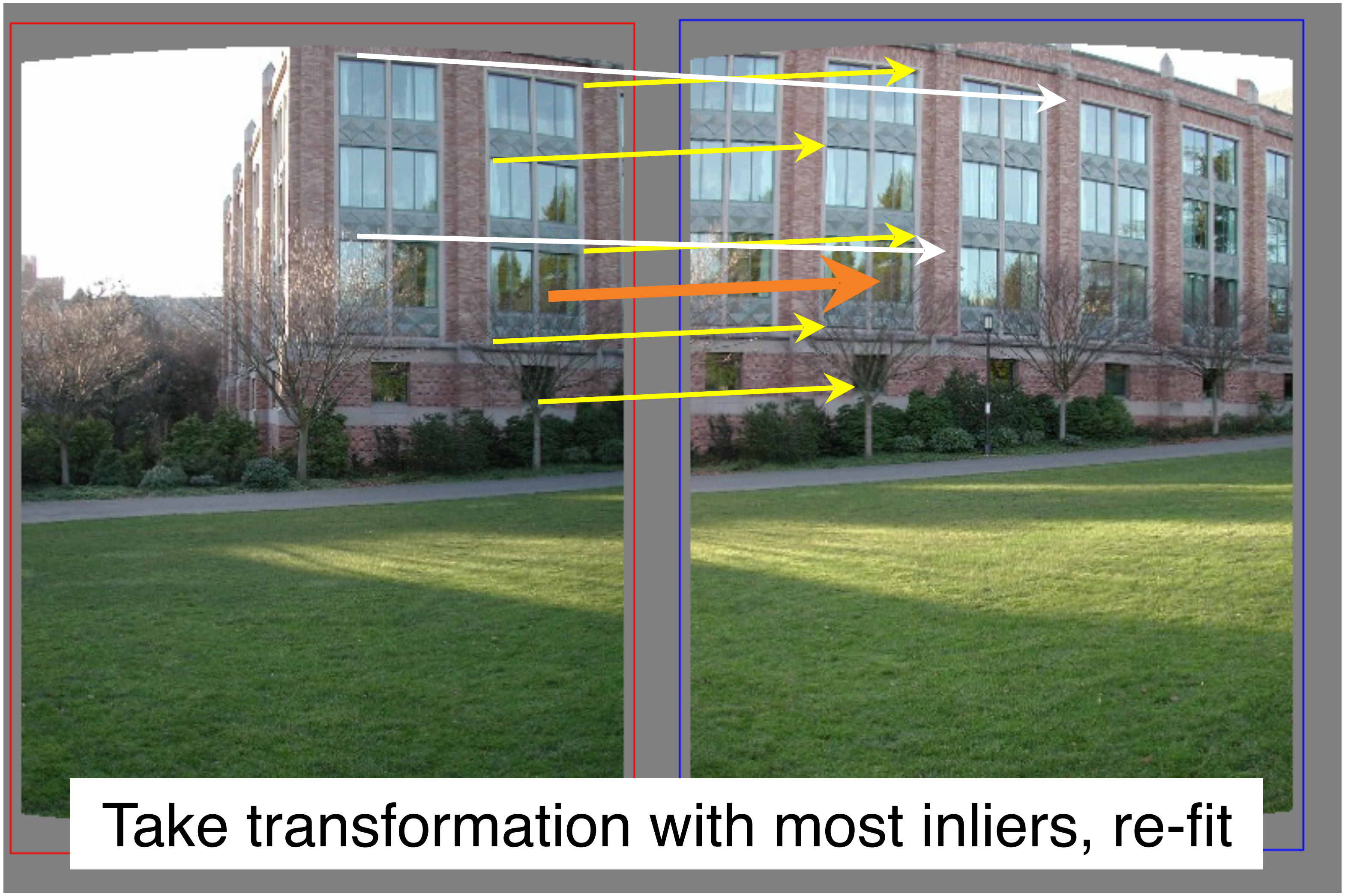
RANSAC: Translation Only



Select *one* match, count inliers

[Szeliski]

RANSAC: Translation Only



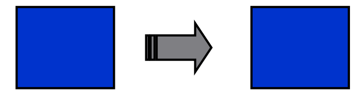
[Szeliski]

Feature-Based Alignment

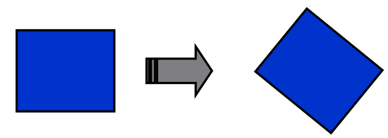
- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- **Fit to find best image transformation**
- Warp images according to transformation
- Blend images in overlapping regions

2D Transformation Models

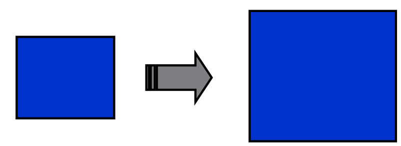
- Translation only



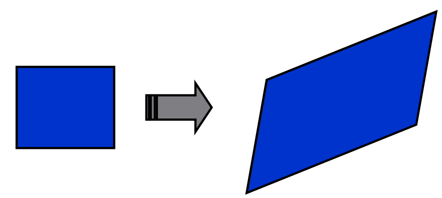
- Rigid body (translation+rotation)



- Similarity (translation+rotation+scale)



- Affine



- Homography (projective)

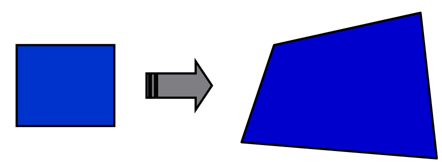
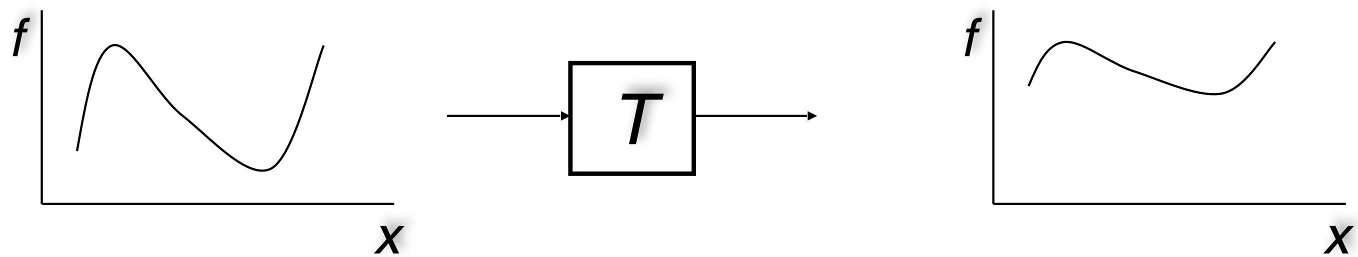


Image Warping

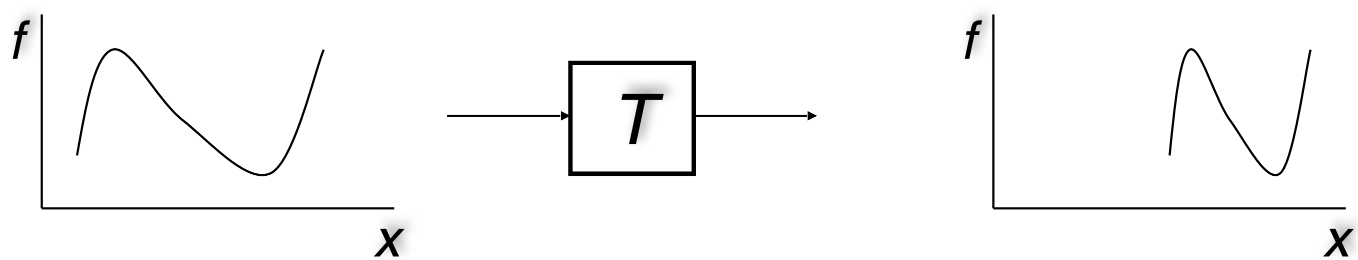
- Image filtering: change *range* of image

$$g(x) = T(f(x))$$



- Image warping: change *domain* of image

$$g(x) = f(T(x))$$

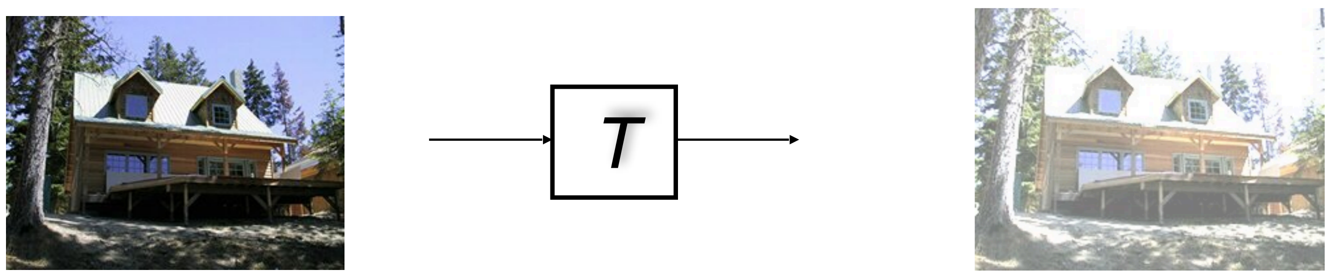


Richard Szeliski

Image Warping

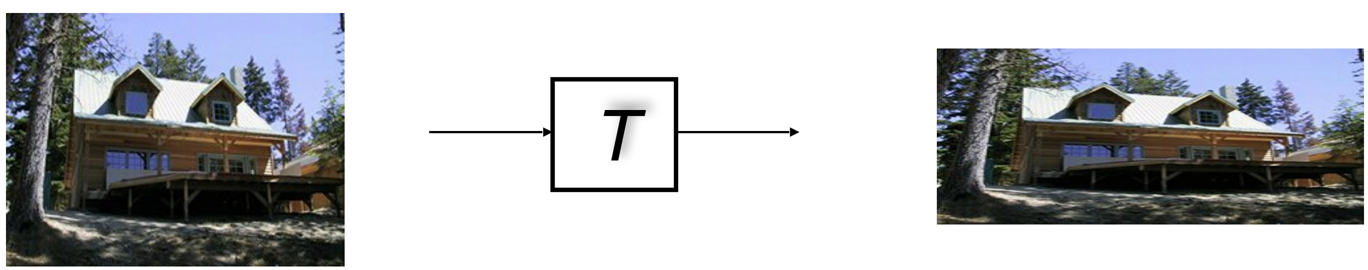
- Image filtering: change *range* of image

$$g(x) = T(f(x))$$



- Image warping: change *domain* of image

$$g(x) = f(T(x))$$



Richard Szeliski

Parametric (Global) Warping

- Examples of parametric warps:

translation

rotation

aspect

affine

perspective

cylindrical

Richard Szeliski

Parametric (Global) Warping

T is a coordinate changing machine

$$p' = T(p)$$

Note: T is the same for all points, has relatively few parameters, and does **not** depend on image content

$$\longrightarrow \boxed{T} \longrightarrow$$

p = (x,y)

p' = (x',y')

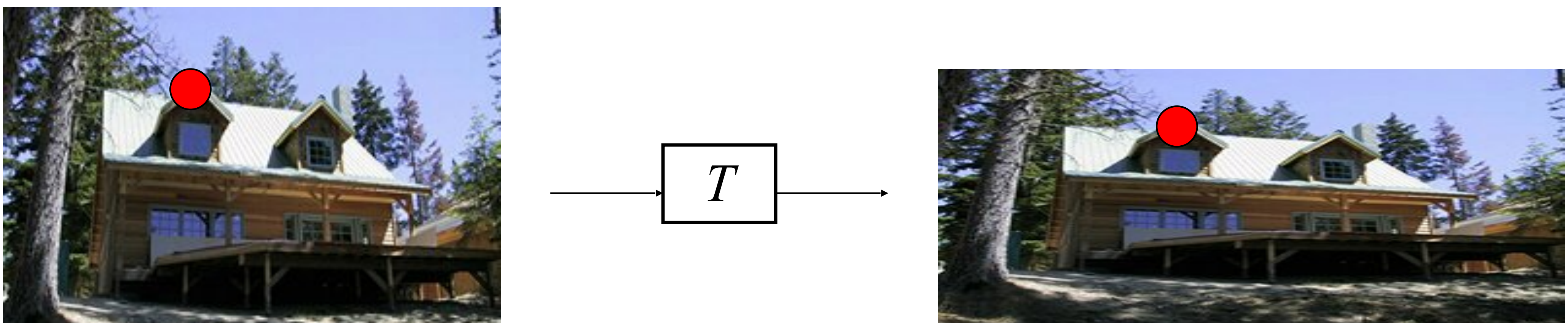
Slide credit: A. Efros

Parametric (Global) Warping

Today we'll deal with linear warps

$$p' \equiv Tp$$

T: matrix; p, p': 2D points. Start with normal points and =, then do homogeneous cords and ≡



p = (x,y)

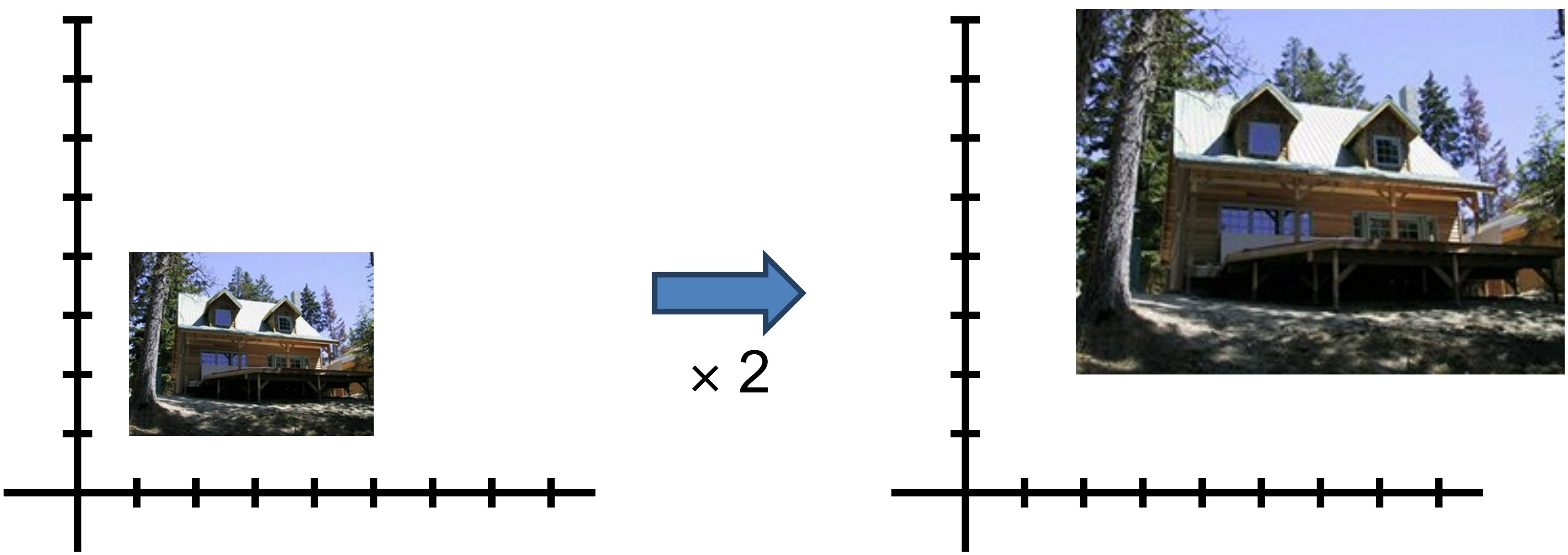
p' = (x',y')

Slide credit: A. Efros

Scaling

Scaling multiplies each component (x,y) by a scalar.
Uniform scaling is the same for all components.

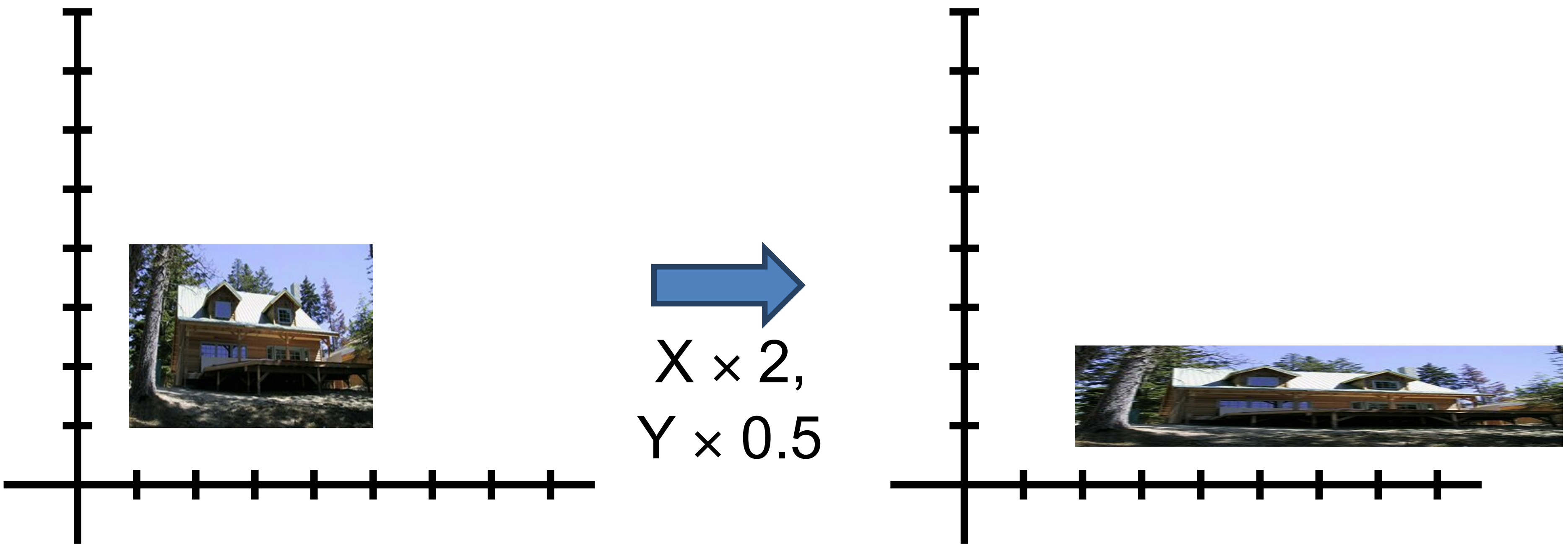
Note the corner goes from (1,1) to (2,2)



Slide credit: A. Efros

Scaling

Non-uniform scaling multiplies each component by a different scalar.



Slide credit: A. Efros

Scaling

What does T look like?

$$x' = ax$$
$$y' = by$$

Let's convert to a matrix:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}}_{\textit{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

What's the inverse of S?

Slide credit: A. Efros

2D Rotation

Rotation Matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

But wait! Aren’t sin/cos non-linear?

x’ *is* a linear combination/function of *x*, *y*
x’ *is not* a linear function of θ

What’s the inverse of R_θ? $\boldsymbol{I} = \boldsymbol{R}_\theta^T \boldsymbol{R}_\theta$

Slide credit: A. Efros

Things You Can Do With 2x2

Identity / No Transformation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

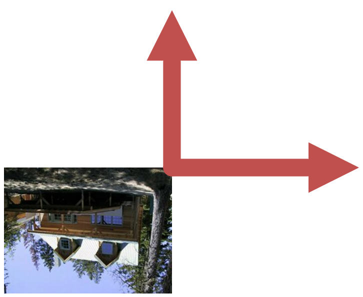
Things You Can Do With 2x2

Before

After

Before

After



2D Mirror About Y-Axis

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Mirror About X,Y

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

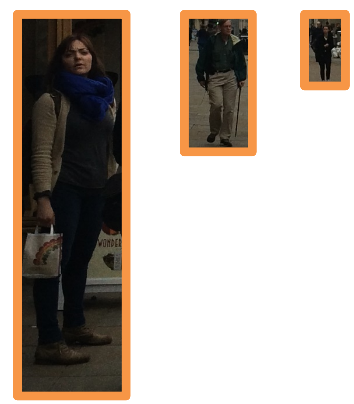
Slide credit: A. Efros

Recall: what's preserved in images?

3D lines project to 2D lines so lines are preserved

Projections of parallel 3D lines are not necessarily parallel, so not parallelism

Distant objects are smaller so size is not preserved



Slide credit: D. Fouhey

What's Preserved With a 2x2

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$$

After multiplication by T (irrespective of T)

- Origin is origin: **0 = T0**
- Lines are lines $\begin{bmatrix} (ax+by)+\lambda(adir_x+bdir_y) \\ (cx+dy)+\lambda(cdir_x+ddir_y) \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x+\lambda dir_x \\ y+\lambda dir_y \end{bmatrix}$
- Parallel lines are parallel
- Ratios between distances the same *if scaling is uniform (otherwise no)*

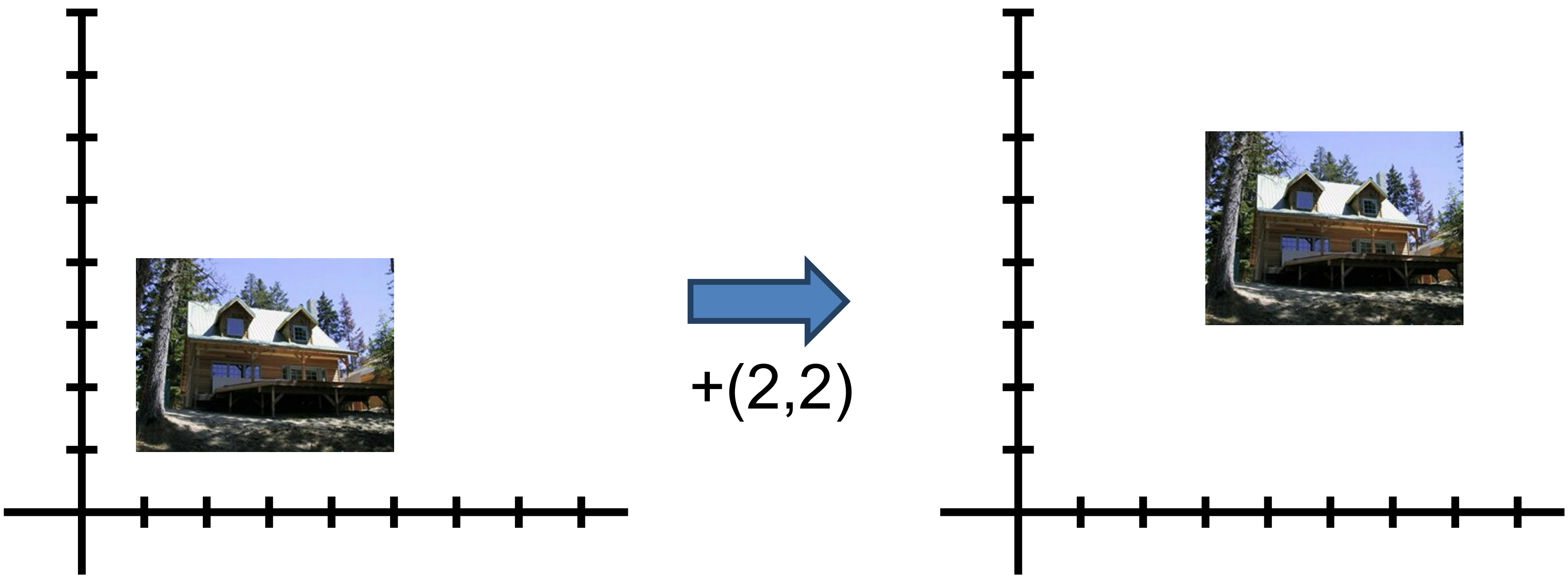
Slide credit: D. Fouhey

Things You Can’t Do With 2x2

What about translation?

$x' = x + t_x,\ y' = y + t_y$

How do we fix it?

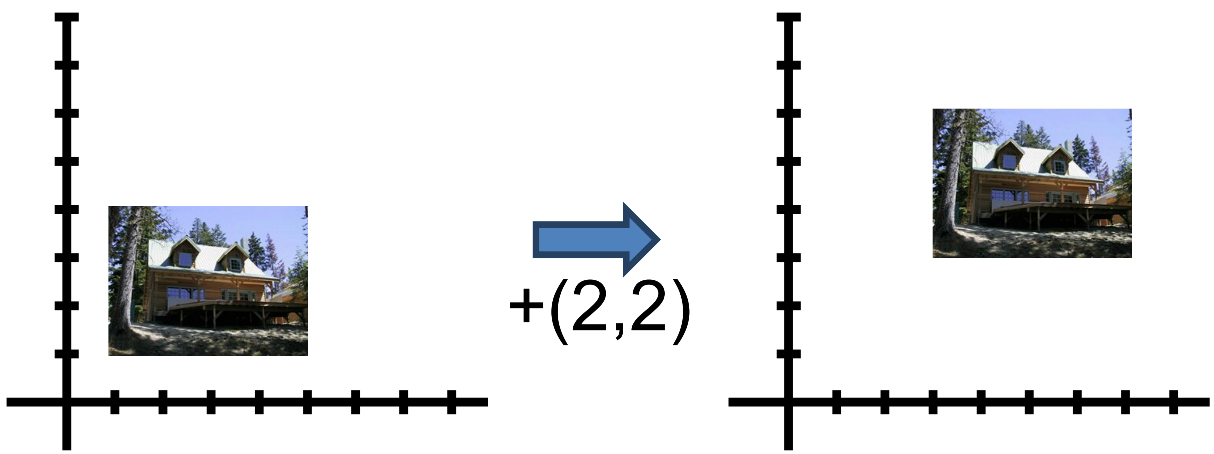


Homogeneous Coordinates

What about translation?

x' = x + t_x, y' = y+t_y

$$\begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Slide credit: A. Efros

Representing 2D Transformations

How do we represent a 2D transformation?
Let's pick scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} s_x & 0 & a \\ 0 & s_y & b \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

What's	a	b	d	e	f
	0	0	0	0	1

Affine Transformations

Affine: *linear transformation* plus *translation*

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

Will the last coordinate always be 1?

In general (without homogeneous coordinates)

$$\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$$

Matrix Composition

We can combine transformations via matrix multiplication.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \underbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{T\left(t_x,\, t_y\right)} \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R(\theta)} \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{S\left(s_x,\, s_y\right)} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Does order matter?

Slide credit: A. Efros

What's Preserved With Affine

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \boldsymbol{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

After multiplication by T (irrespective of T)

- ~~Origin is origin: **0 = T0**~~
- Lines are lines $\begin{bmatrix} (ax+by+c)+\lambda(adir_x+bdir_y) \\ (dx+ey+f)+\lambda(ddir_x+edir_y) \\ 1 \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x+\lambda dir_x \\ y+\lambda dir_y \\ 1 \end{bmatrix}$
- Parallel lines are parallel
- Ratios between distances? if scaling is uniform yes, otherwise no

Slide credit: D. Fouhey

Perspective Transformations

Set bottom row to not [0,0,1]
Called a perspective/projective transformation or a ***homography***

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

How many degrees of freedom?

How Many Degrees of Freedom?

Recall: can always scale by non-zero value

Perspective $$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \frac{1}{i} \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \frac{1}{i} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \equiv \begin{bmatrix} a/i & b/i & c/i \\ d/i & e/i & f/i \\ g/i & h/i & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Homography can always be re-scaled by $\lambda \neq 0$

What's Preserved With Perspective

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \boldsymbol{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

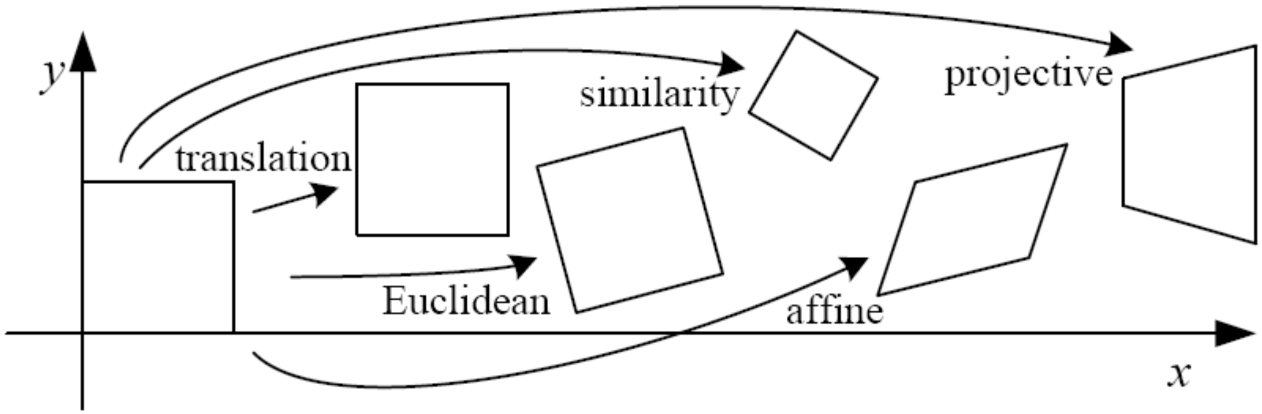
After multiplication by T (irrespective of T)

- ~~Origin is origin: **0 = T0**~~
- Lines are lines
- ~~Parallel lines are parallel~~
- ~~Ratios between distances~~

Slide credit: D. Fouhey

Transformation Families

In general: transformations are a nested set of groups



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

Diagram credit: R. Szeliski

What Can Homographies Do?

Homography example 1: any two views of a *planar* surface

Figure Credit: S. Lazebnik

What Can Homographies Do?

Homography example 2: any images from two cameras sharing a camera center

Figure Credit: S. Lazebnik

What Can Homographies Do?

Homography sort of example “3”: far away scene that can be approximated by a plane

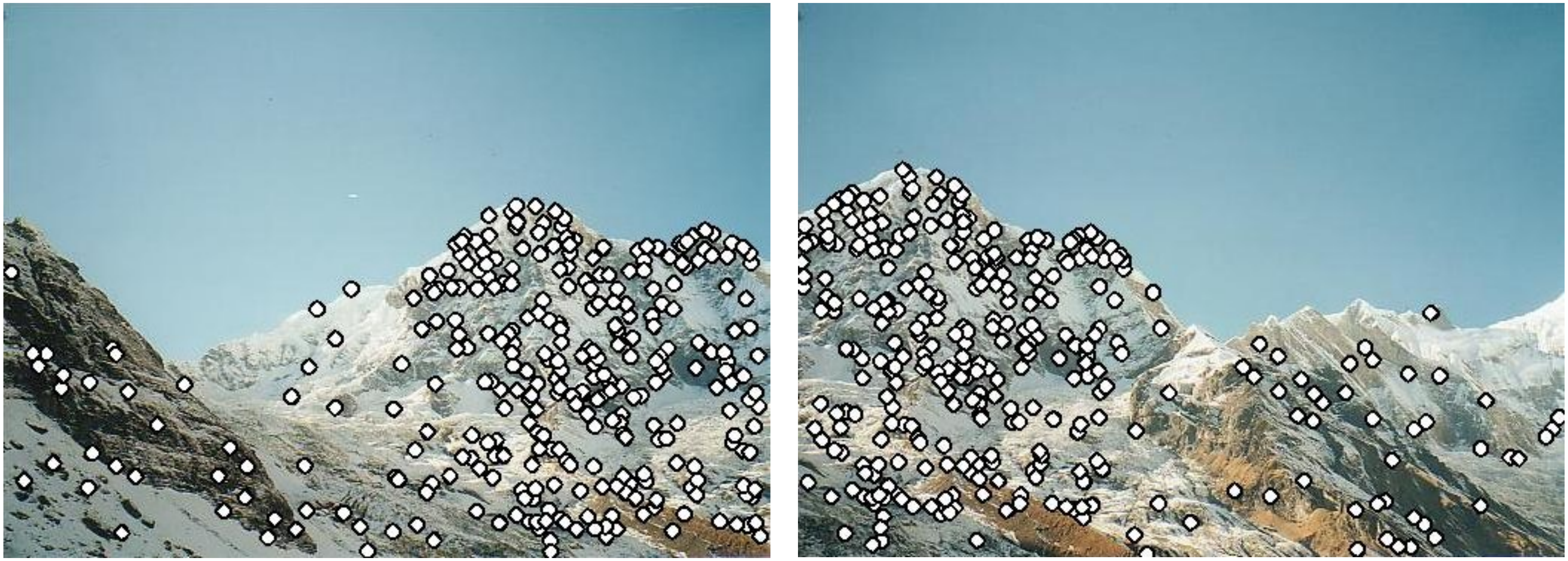
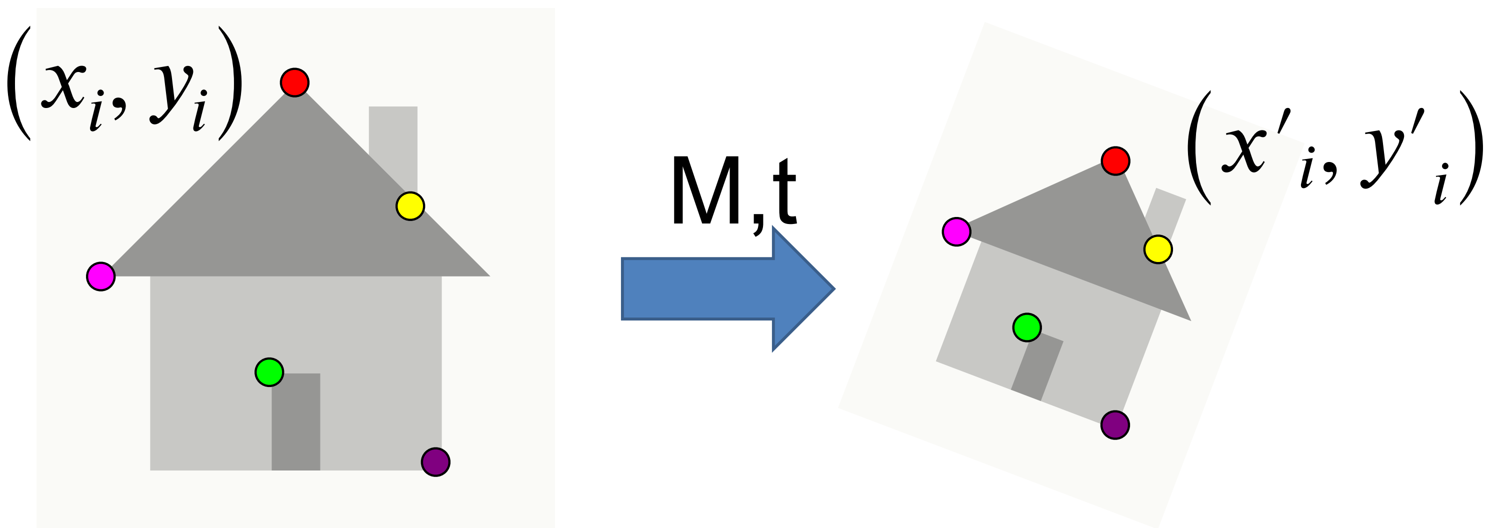


Figure credit: Brown & Lowe

Fitting Transformations

Setup: have pairs of correspondences



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \boldsymbol{t}$$

Slide Credit: S. Lazebnik

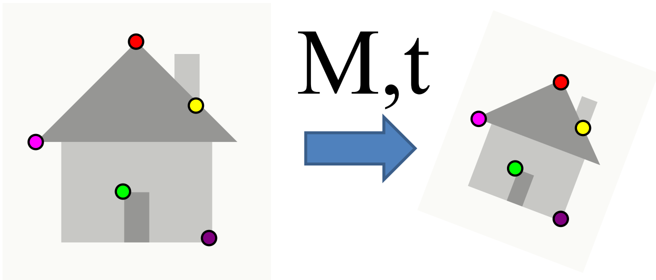
Fitting Transformation

Affine Transformation: M,t

Data: (x_i, y_i, x'_i, y'_i) for $i=1,\ldots,k$

Model:
$[x'_i, y'_i] = \mathbf{M}[x_i, y_i] + \mathbf{t}$

Objective function:
$||[x'_i, y'_i] - \mathbf{M}[x_i, y_i] + \mathbf{t}||^2$



Slide Credit: D. Fouhey

Fitting Transformations

Given correspondences: $\mathbf{p}' = [x'_i, y'_i]$, $\mathbf{p} = [x_i, y_i]$

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Set up two equations per point

$$\begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix}$$

Slide Credit: D. Fouhey

Fitting Transformations

$$\begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix}$$

2 equations per point, 6 unknowns

How many points do we need?

Slide Credit: D. Fouhey

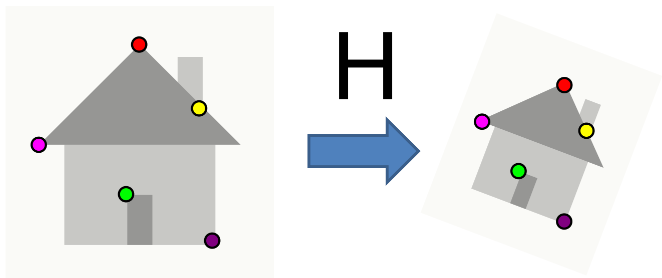
Fitting Transformation

Homography: H

Data: (x_i, y_i, x'_i, y'_i) for $i=1,\ldots,k$

Model: $[x'_i, y'_i, 1] \equiv \mathbf{H}[x_i, y_i, 1]$

Objective function: It's complicated



(Chapters 6.1, 6.2 in the book)

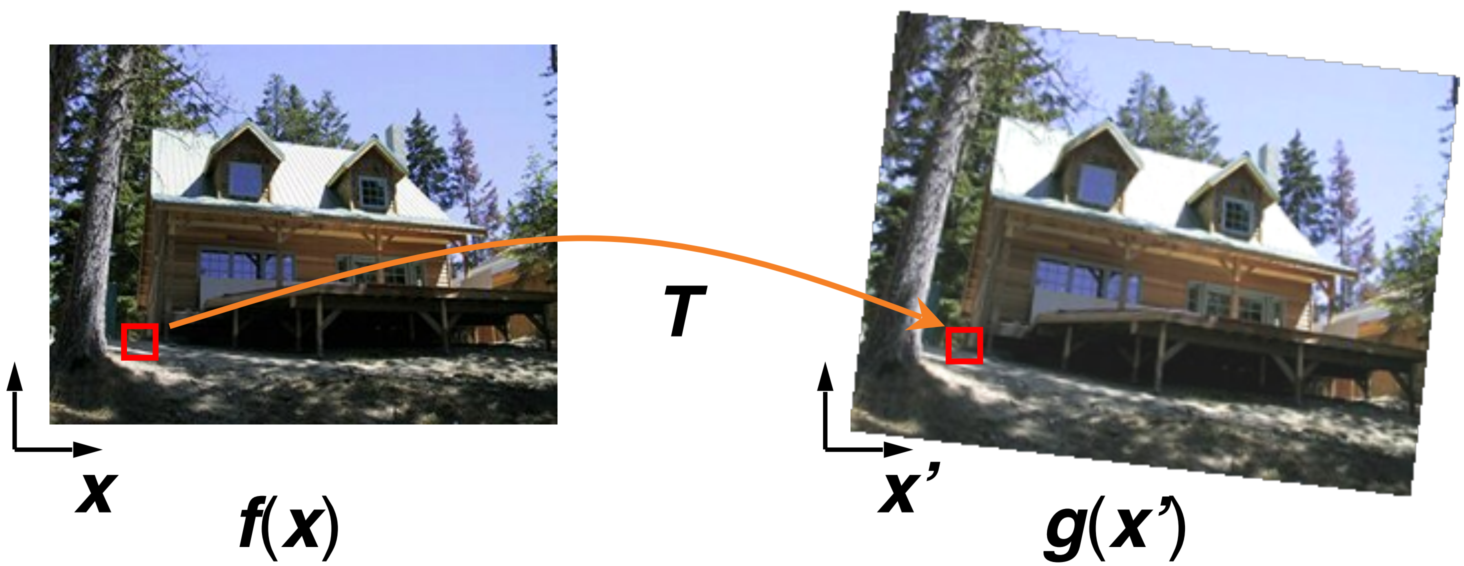
Slide Credit: D. Fouhey

Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Image Warping

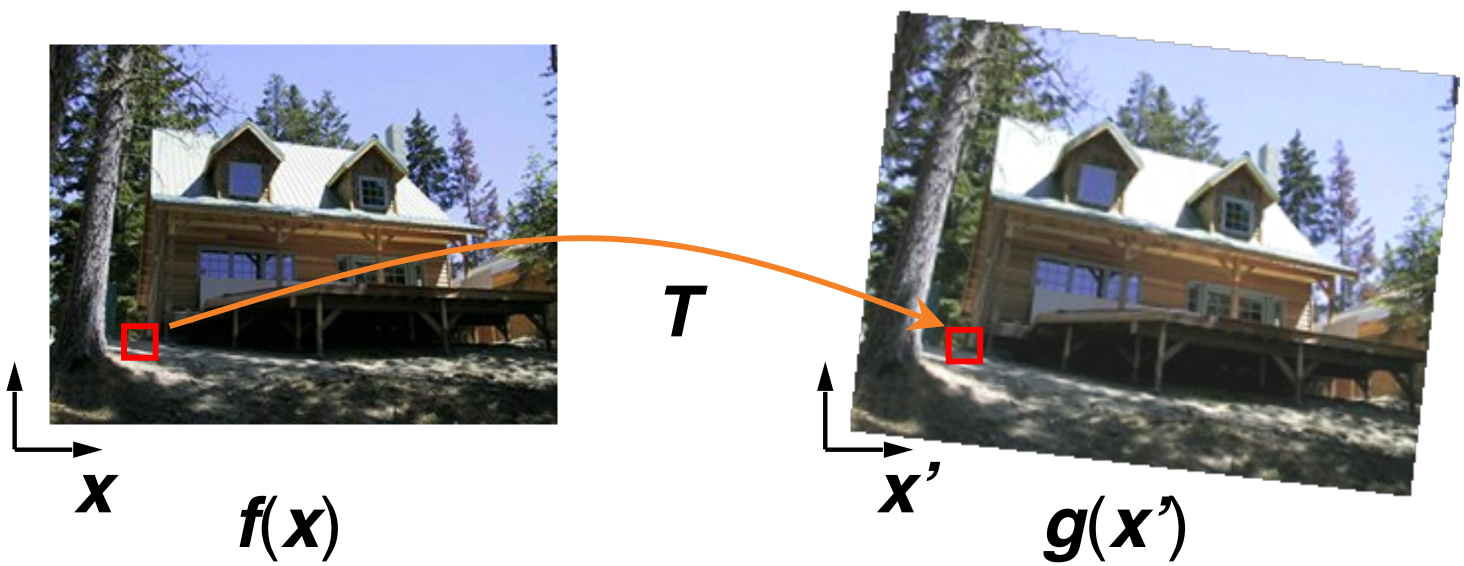
- Given a coordinate transform $\mathbf{x}' = \mathbf{T}(\mathbf{x})$ and a source image $f(\mathbf{x})$, how do we compute a transformed image $g(\mathbf{x}') = f(\mathbf{T}(\mathbf{x}))$?



Richard Szeliski

Forward Warping

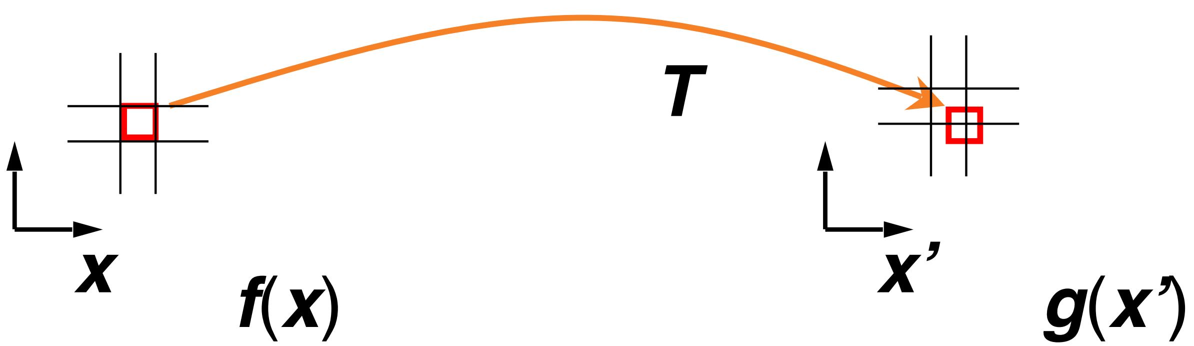
- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x}' = \boldsymbol{T}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x}')$
  - What if pixel lands "between" two pixels?



Richard Szeliski

Forward Warping

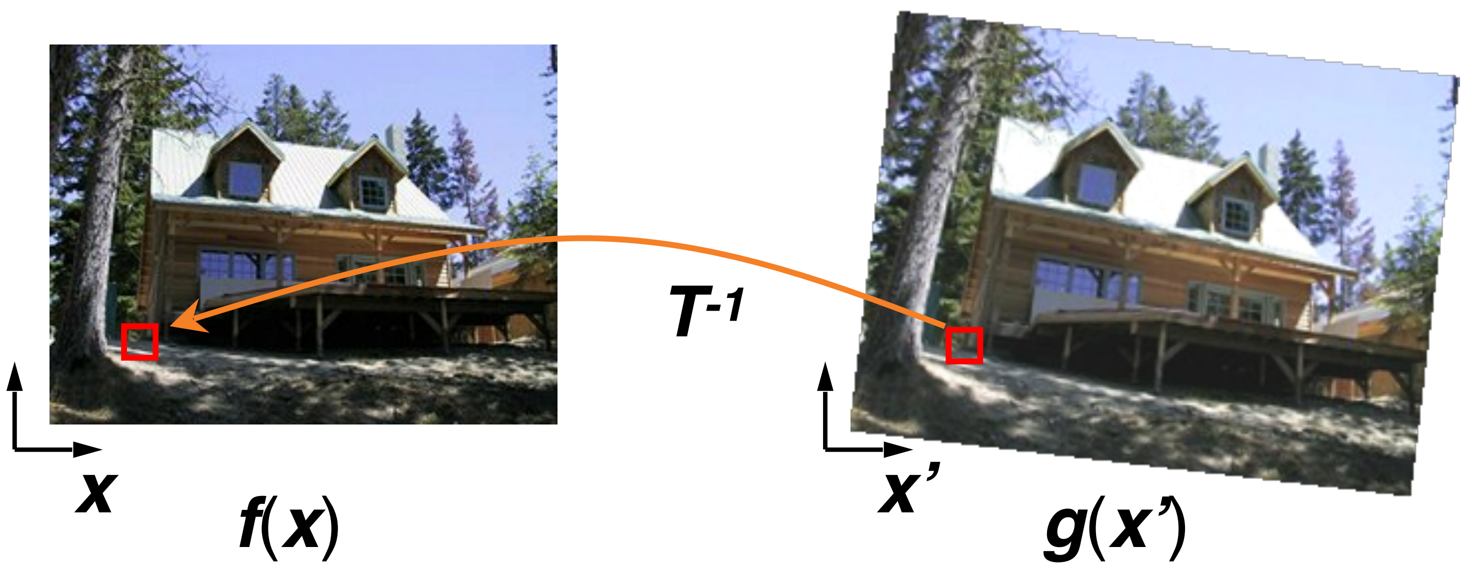
- Send each pixel $f(\mathbf{x})$ to its corresponding location $\mathbf{x}' = T(\mathbf{x})$ in $g(\mathbf{x}')$
  - What if pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)



Richard Szeliski

Inverse Warping

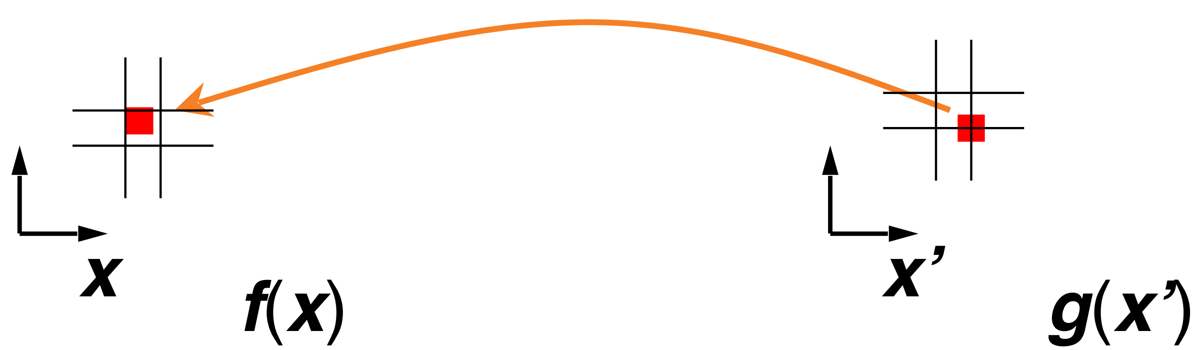
- Get each pixel $\boldsymbol{g}(\boldsymbol{x}')$ from its corresponding location $\boldsymbol{x} = \boldsymbol{T}^{-1}(\boldsymbol{x}')$ in $\boldsymbol{f}(\boldsymbol{x})$
  - What if pixel comes from "between" two pixels?



Richard Szeliski

Inverse Warping

- Get each pixel $\boldsymbol{g}(\boldsymbol{x'})$ from its corresponding location $\boldsymbol{x} = \boldsymbol{T}^{-1}(\boldsymbol{x'})$ in $\boldsymbol{f}(\boldsymbol{x})$
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* (*prefiltered*) source image



Richard Szeliski

Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)
  - sinc / FIR
- See COS 426 for details on how to avoid "jaggies"

Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Blending

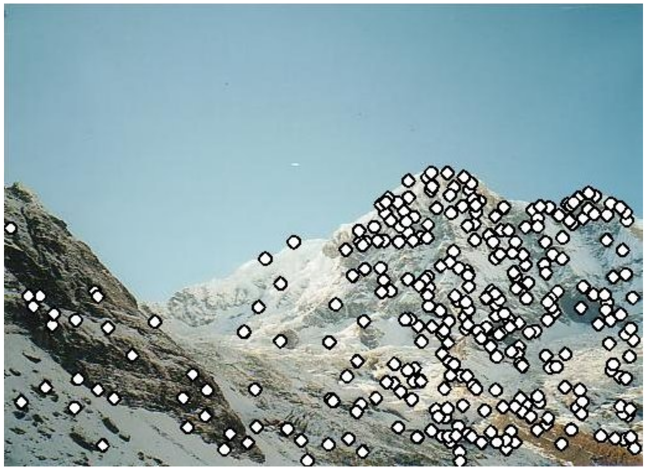
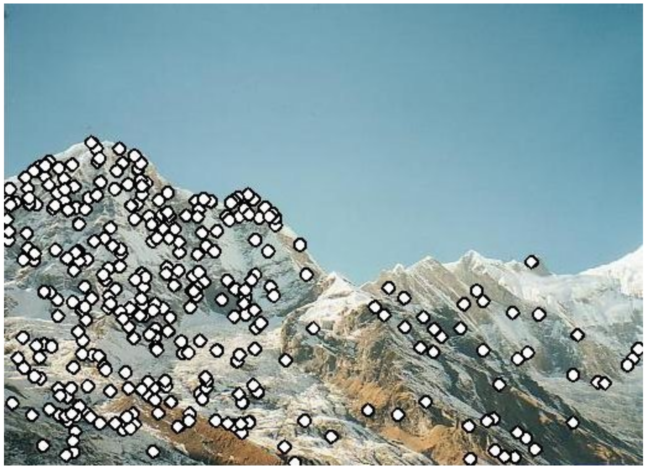
- Blend over too small a region: seams
- Blend over too large a region: ghosting

- COS 426 for details

Putting it all together: making a panorama?

Step 1

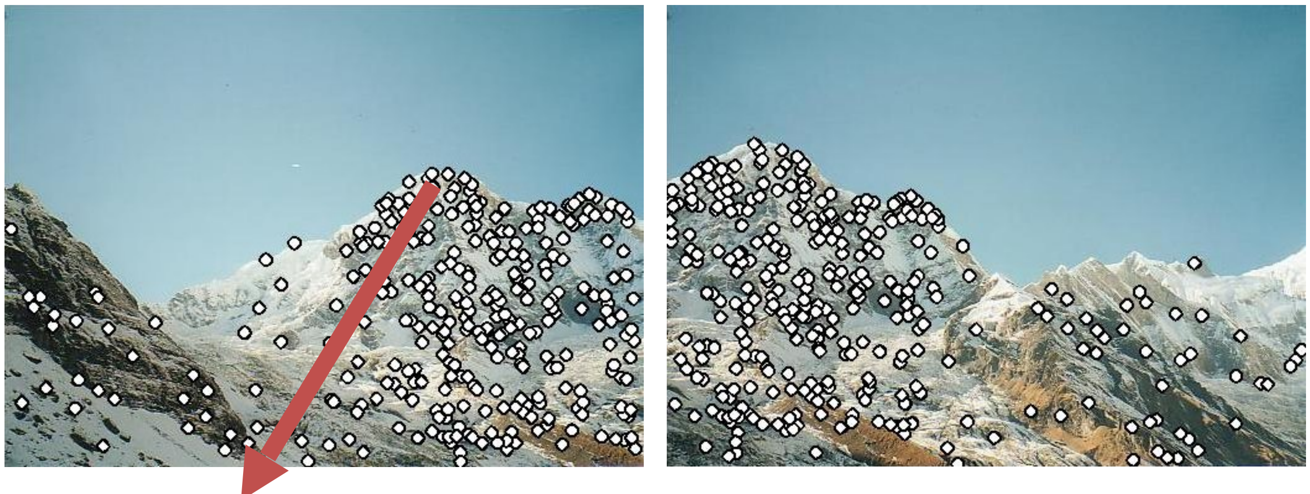
Find corners/blobs

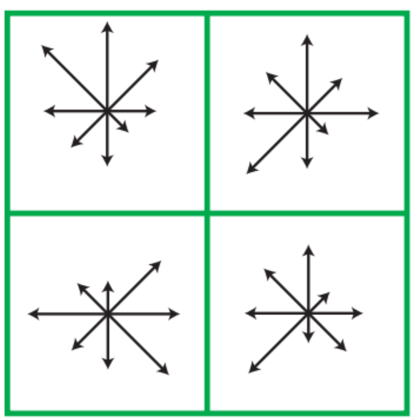
- (Multi-scale) Harris; or
- Laplacian of Gaussian

Step 2

Describe Regions Near Features



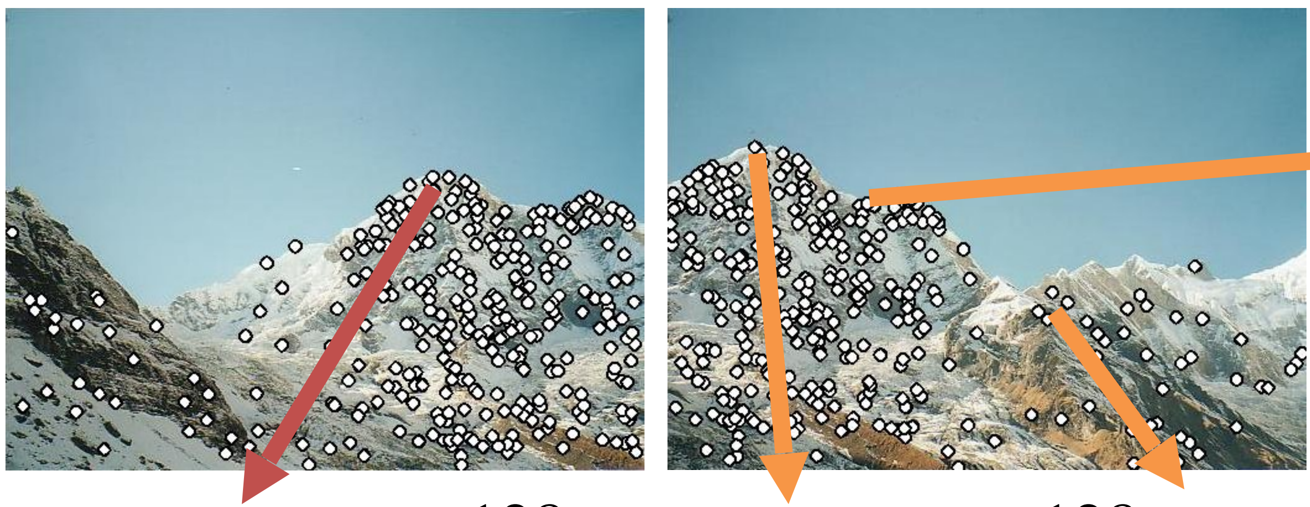
$$x_q \in R^{128}$$



Build histogram of gradient orientations (SIFT)

Step 3

Match Features Based On Region



$x_3 \in R^{128}$

$x_q \in R^{128}$ $\quad x_1 \in R^{128}$ $\quad x_2 \in R^{128}$

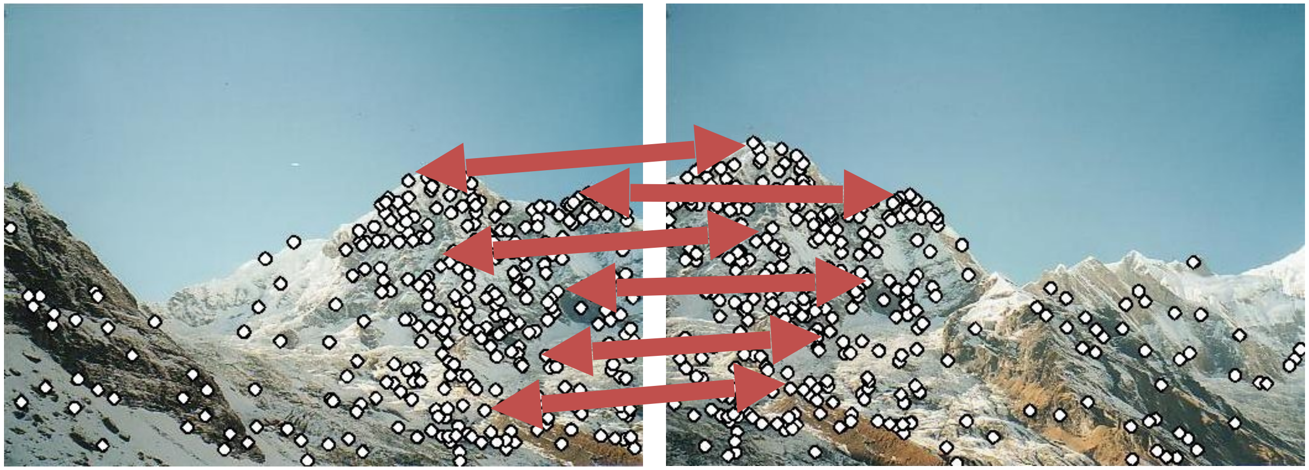
Sort by distance to: x_q $\quad \|x_q - x_1\| < \|x_q - x_2\| < \|x_q - x_3\|$

Accept match if: $\quad \|x_q - x_1\| / \|x_q - x_2\|$

Nearest neighbor is far closer than 2nd nearest neighbor

Step 4

Fit transformation H via RANSAC



for trial in range(Ntrials):
    Pick sample
    Fit model
    Check if more inliers
Re-fit model with most inliers

Step 5

Warp images together

Resample images with inverse warping and blend

Next class: intro to recognition + basics of ML