Alignment

Lihi Zelnik-Manor, Computer Vision

Today

- Matching local features
- Parametric transformations
- Computing parametric transformations
- Panoramas

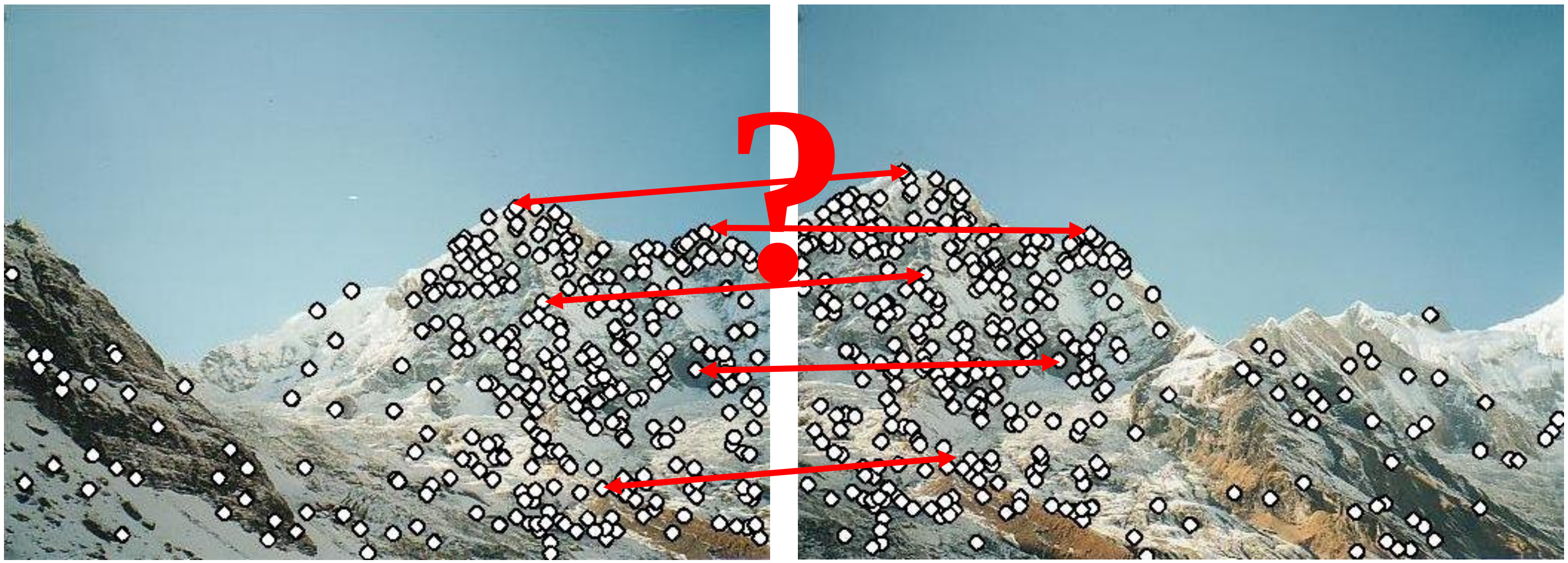
Today

- Matching local features
- Parametric transformations
- Computing parametric transformations
- Panoramas

Feature matching

We know how to detect **and describe** good points

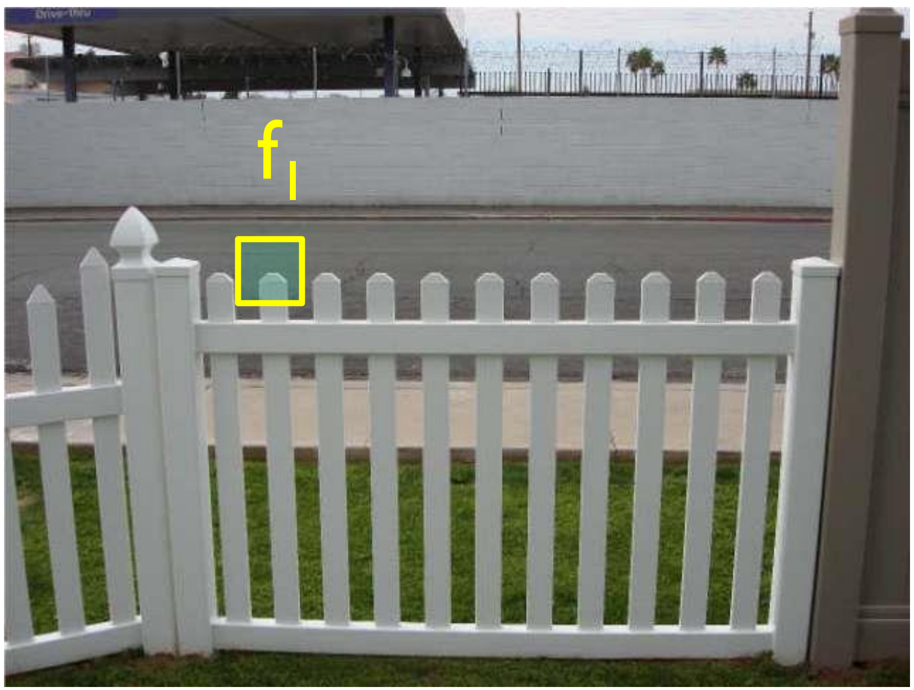
Next question: **How to match them?**

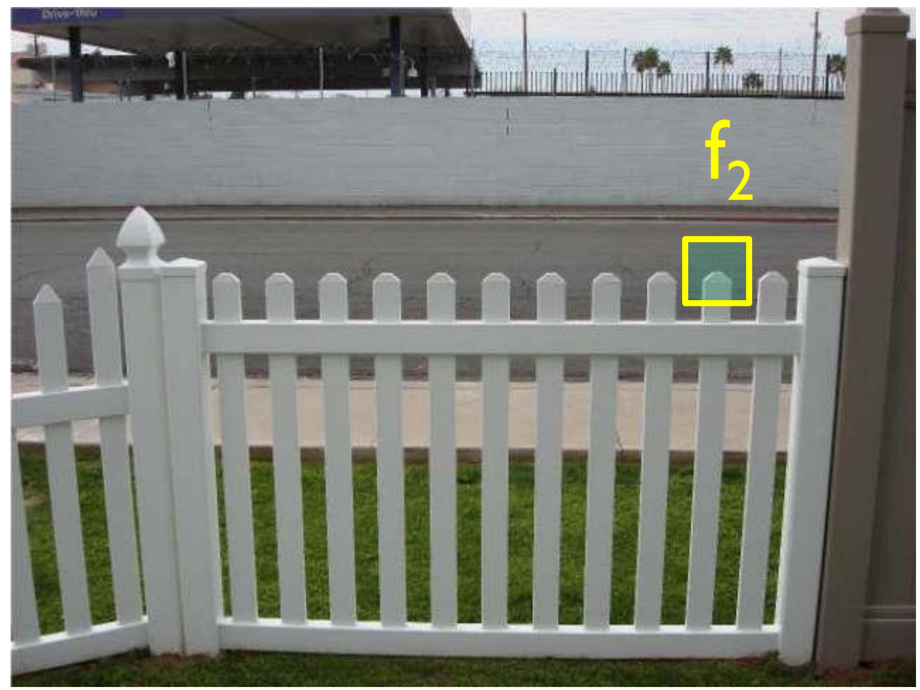


Feature matching

Given a feature in *Image1*, how to find the best match in *Image2*

1. Define distance function that compares two descriptors
2. Compute distances between all pairs features and find the ones with min distance





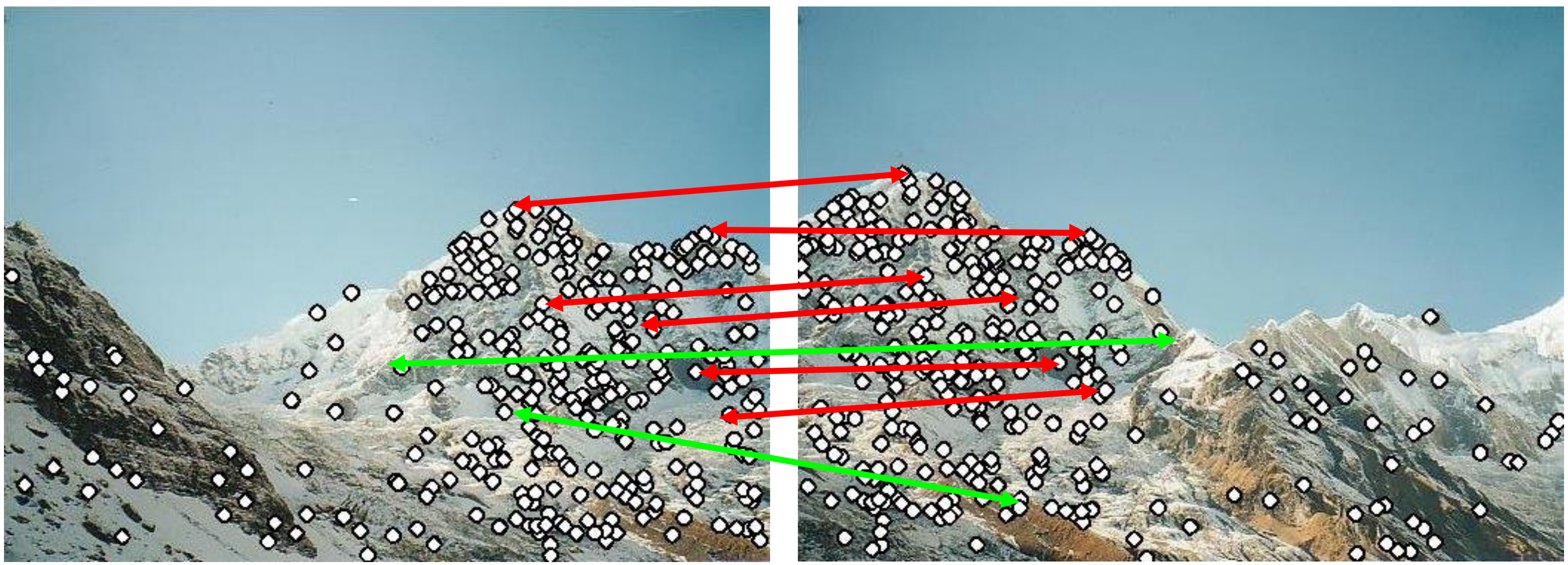
Feature matching - better

Given a feature in *Image1*, how to find the best match in *Image2*

1. Define distance function that compares two descriptors
2. Compute distances between all pairs features and find matches that *dist(f1,best match) / dist(f1,second-best match) > threshold*

Typical feature matching results

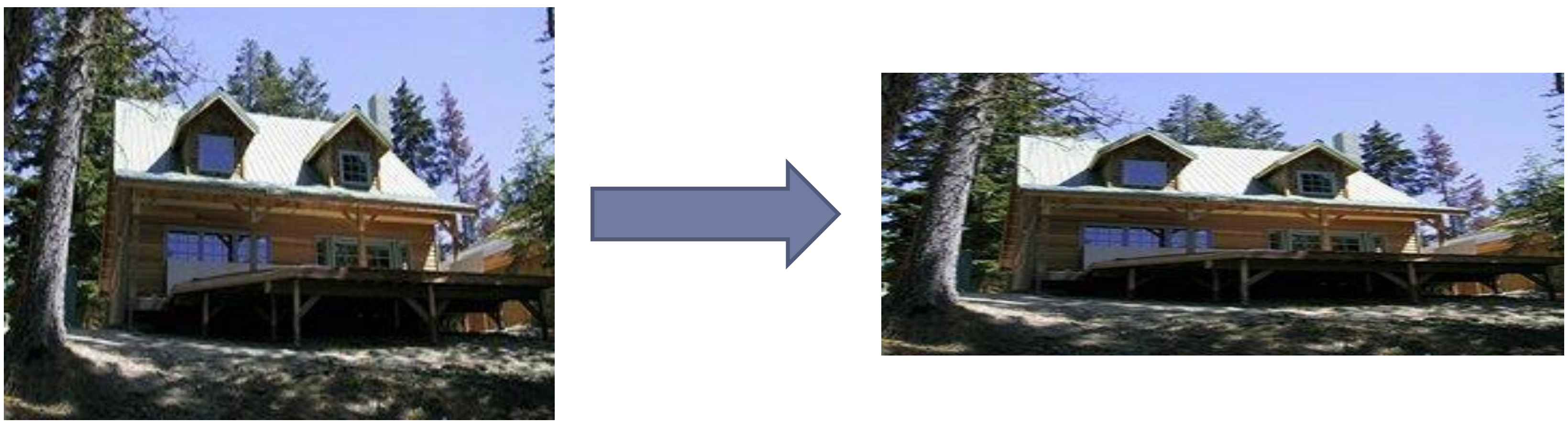
- Some matches are correct
- Some matches are incorrect



- <u>Solution</u>:   search for a set of geometrically consistent matches

What we need to solve

- Given source and target images, how do we compute the transformation between them?



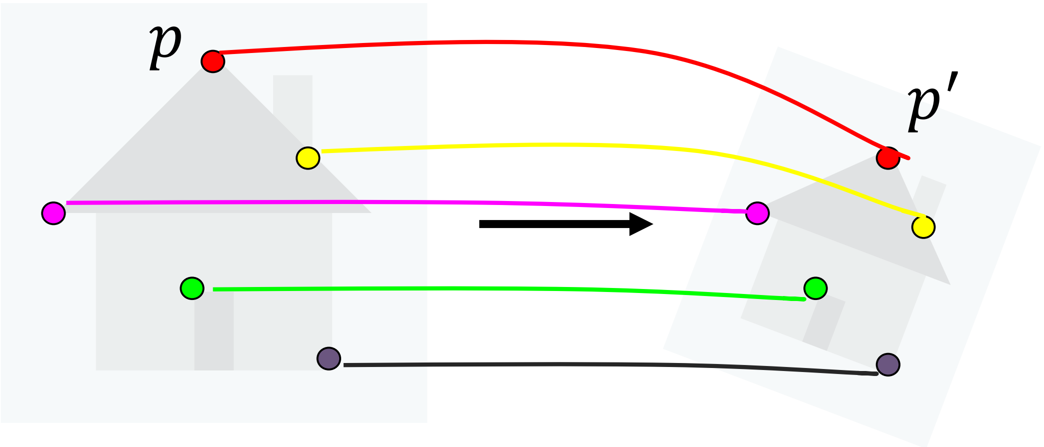
Today

- Matching local features
- Parametric transformations
- Computing parametric transformations
- Panoramas

Image alignment

- Turns out that in many cases there's a global transformation relating points in two images:

$$p' = Hp \qquad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Parametric (global) warping

original

- Examples of parametric transformations:

translation

In-plane rotation

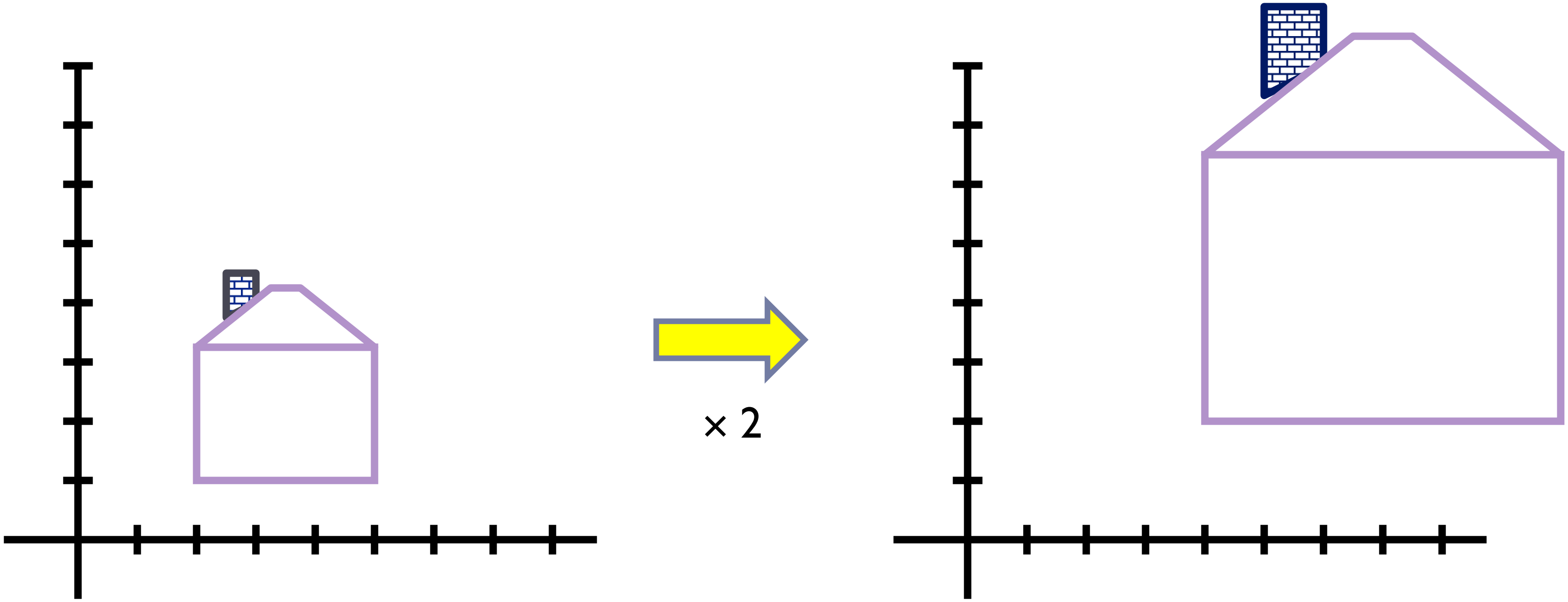
Aspect ratio

affine

perspective

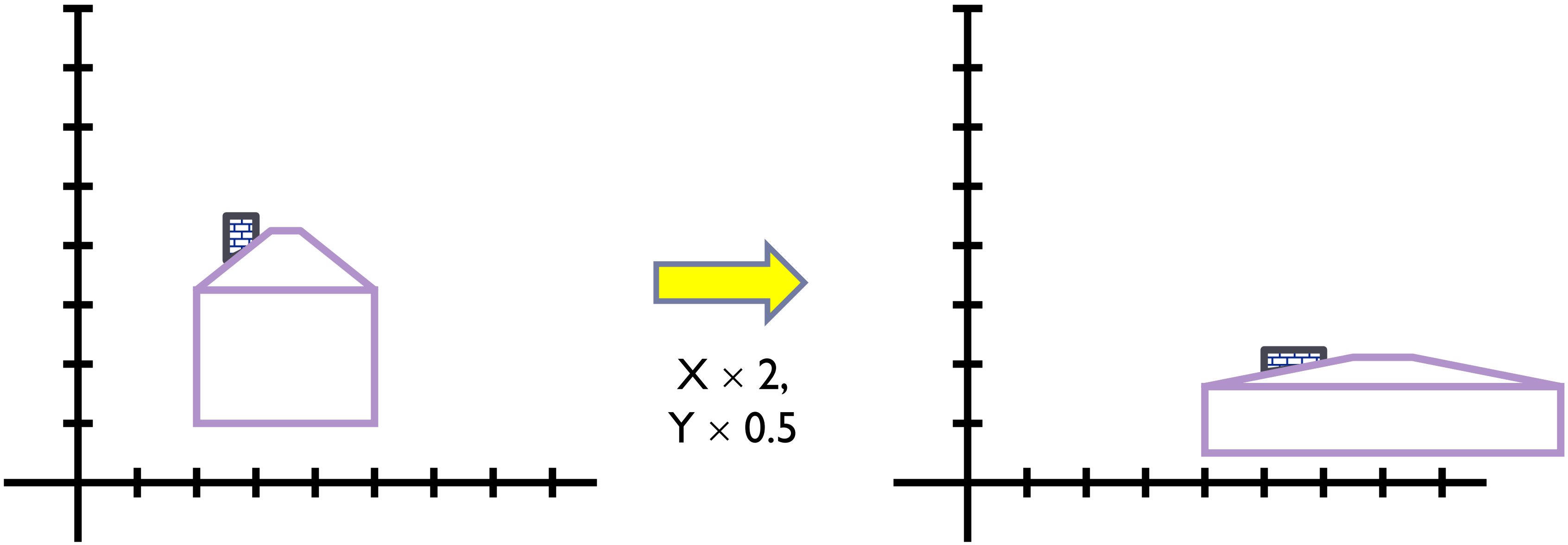
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



Scaling

- *Non-uniform scaling*: different scalars per component:



Scaling

- Scaling operation:

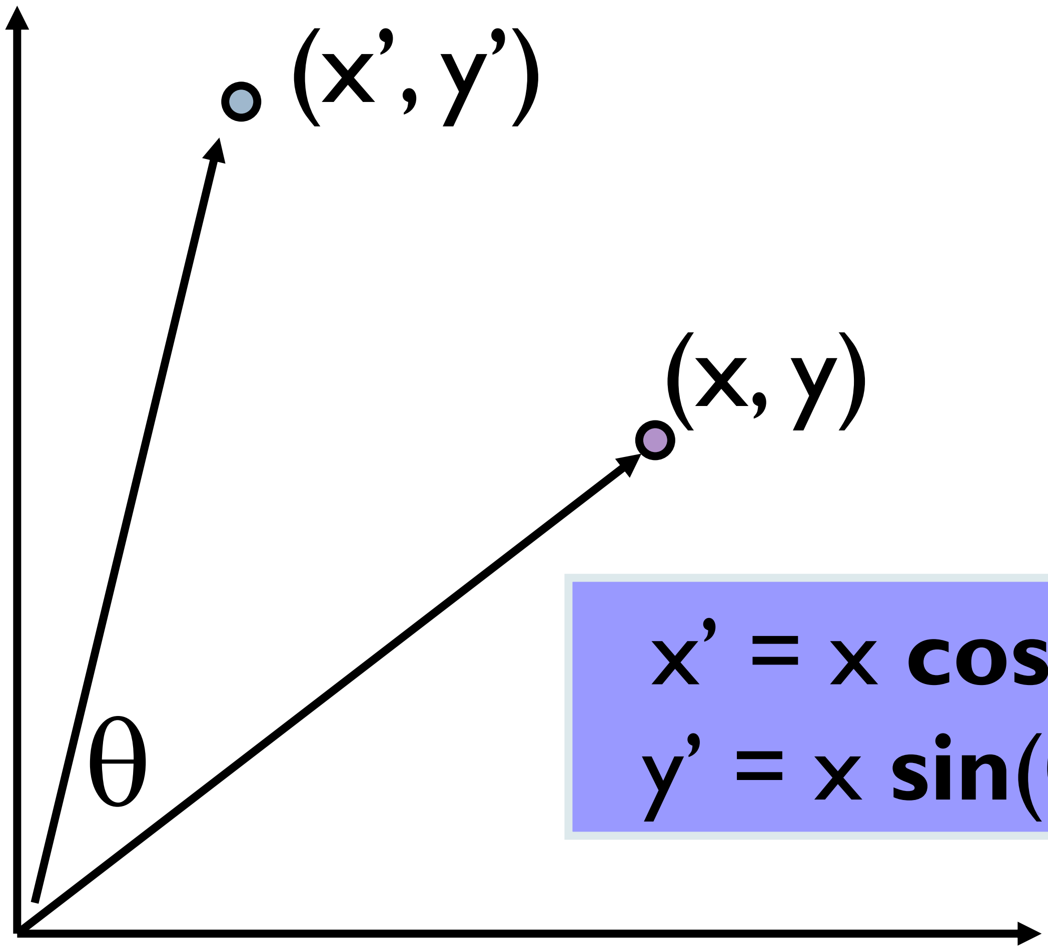
$$x' = ax$$

$$y' = by$$

- Or, in matrix form:

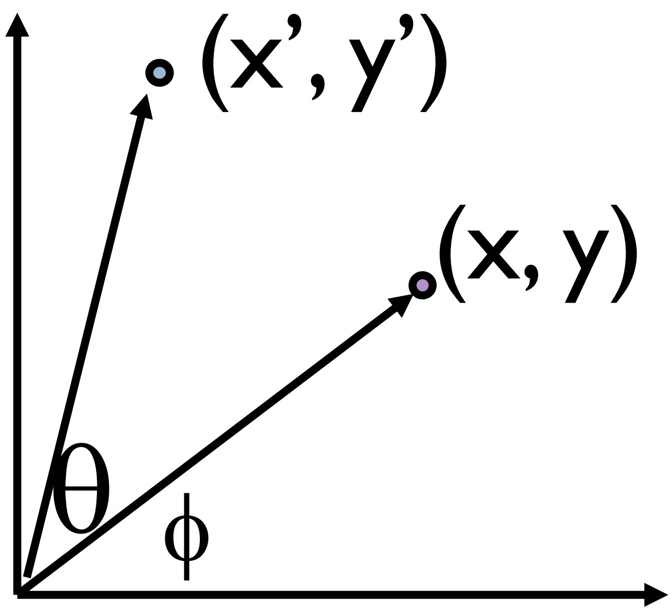
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\textit{scaling matrix } S} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2-D Rotation



$$x' = x \cos(\theta) - y \sin(\theta)$$
$$y' = x \sin(\theta) + y \cos(\theta)$$

2-D Rotation



Polar coordinates…

$x = r \cos(\phi)$

$y = r \sin(\phi)$

$x' = r \cos(\phi + \theta)$

$y' = r \sin(\phi + \theta)$

Trig Identity…

$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$

$y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$

Substitute…

$x' = x\, \mathbf{cos}(\theta) - y\, \mathbf{sin}(\theta)$

$y' = x\, \mathbf{sin}(\theta) + y\, \mathbf{cos}(\theta)$

2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Even though sin(θ) and cos(θ) are nonlinear functions of θ,

- ***x' is a linear combination of x and y***
- ***y' is a linear combination of x and y***

What is the inverse transformation?

- Rotation by –θ
- For rotation matrices $R^{-1} = R^T$

Basic 2D→2D Transformations

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

Translation

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

Rotate in-plane

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

Scale/Aspect ratio

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

Affine

Combination of translation, scale, rotation, shear

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

Shear

Affine Transformations

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations

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

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

Projective Transformations

- Projective transformations …
  - Affine transformations, and
  - Projective warps

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

- Properties of projective transformations:
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition
  - Projective matrix is defined up to a scale (8 DOF)

Basic 2D→2D Transformations

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

Translation

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

Rotate in-plane

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

Scale/Aspect ratio

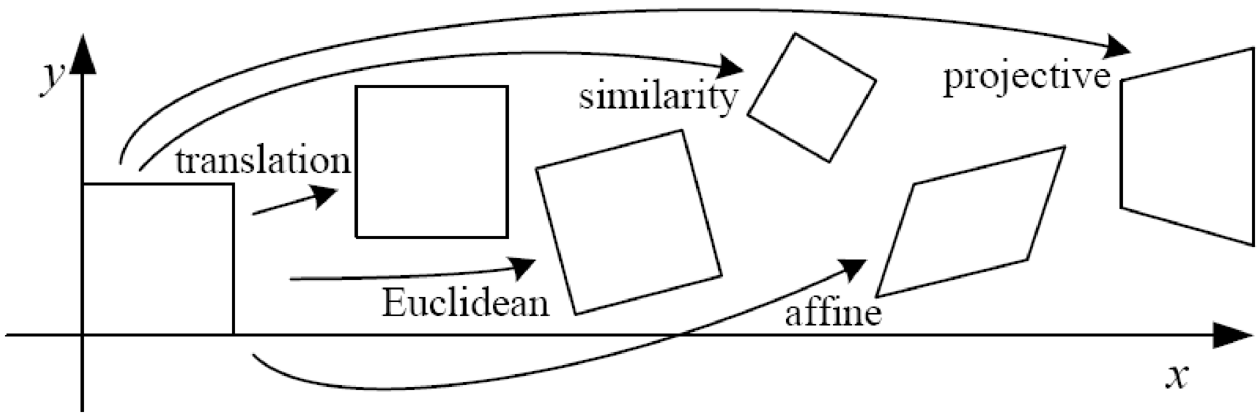
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

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

Projective
(Homography)

2D image transformations (reference table)



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	

When do we get affine or homography?

- Camera does not translate (only rotation and scale)

Transformation between coordinates in 3D:

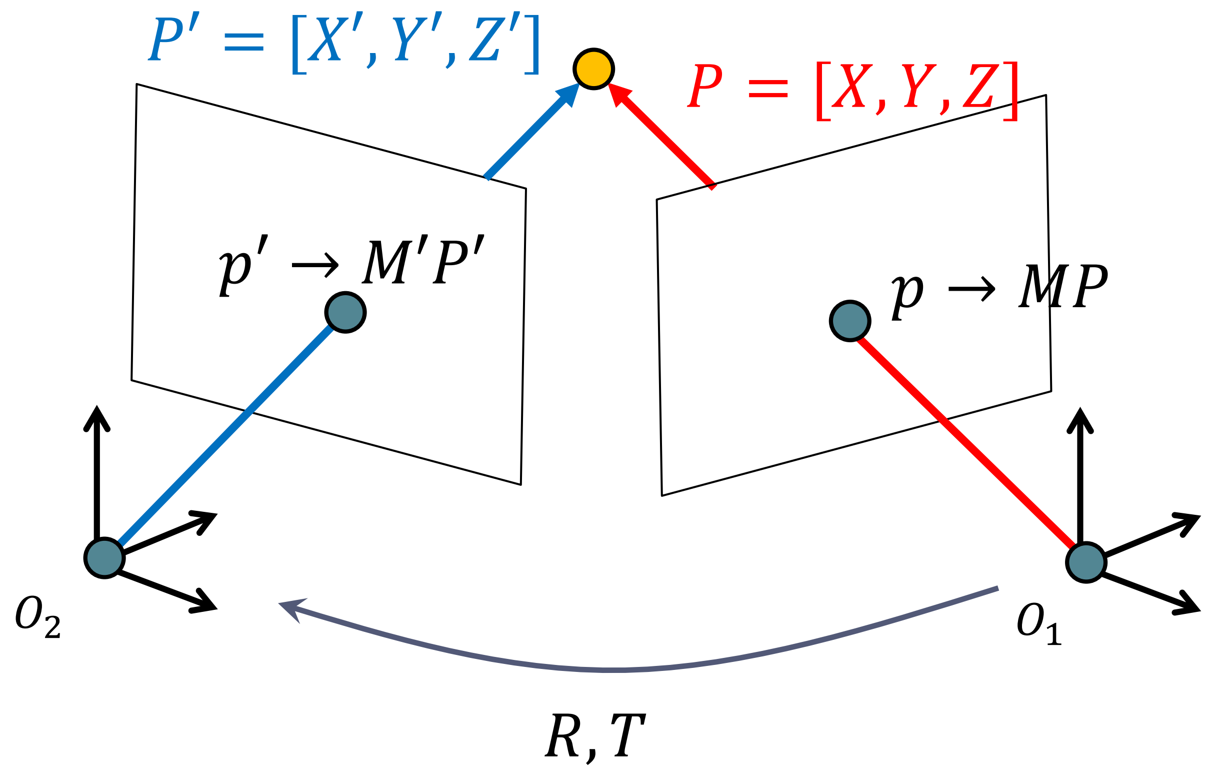
$$P' = RP + T$$

Without translation:

$$P' = RP$$

And the projections:

$$p' = Hp$$



When do we get affine or homography?

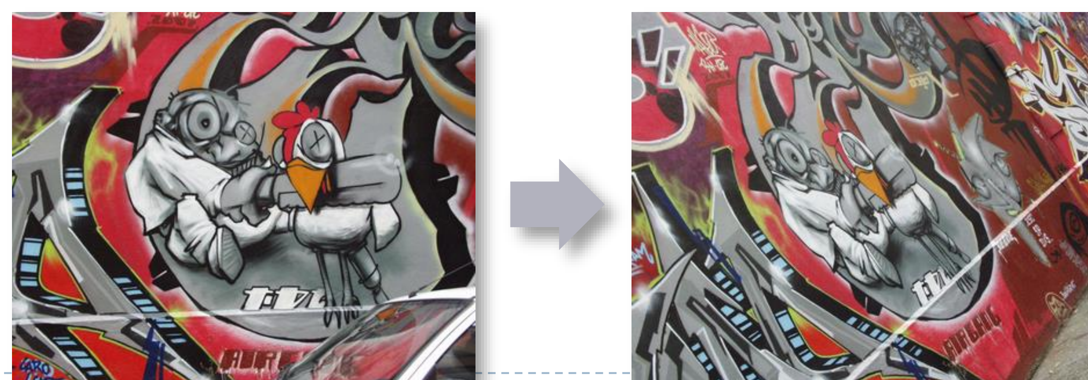
- Camera does not translate (only rotation and scale)

- Object is planar
- Works fine for small viewpoint changes

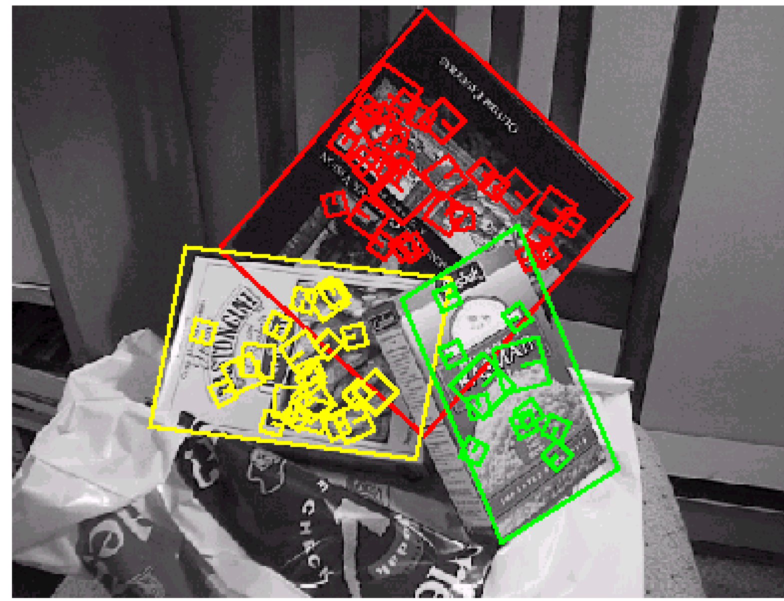
Homographies == Planar Perspective Maps

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)



Fitting an affine transformation



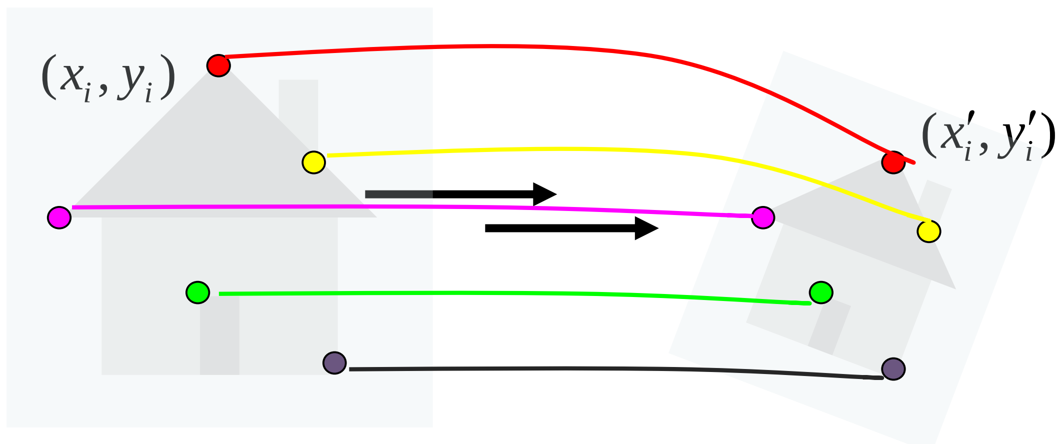
Affine model approximates perspective projection of planar objects.

Today

- Matching local features
- Parametric transformations
- Computing parametric transformations
  - Least-squares
  - RANSAC
- Panoramas

Fitting an affine transformation

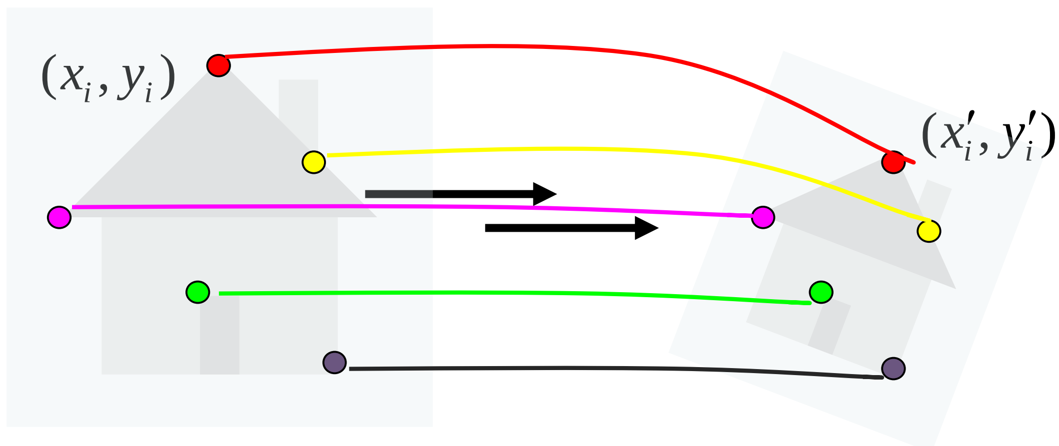
- Assuming we know the correspondences, how do we get the transformation?



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

Fitting an affine transformation

- Assuming we know the correspondences, how do we get the transformation?



$$
\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}
$$

Fitting an affine transformation

- Solve with Least-squares

$$\begin{bmatrix} \\ \\ \\ \end{bmatrix}\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

- How many matches (correspondence pairs) do we need to solve for the transformation parameters?

- Once we have solved for the parameters, how do we compute the coordinates of the corresponding point for any pixel (x_{new}, y_{new}) ?

Fitting a projective transformation

- Recall working with homogenous coordinates

$$\begin{bmatrix} x_i' \\ y_i' \\ w_i' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad x_i' \rightarrow \frac{x_i'}{w_i'} \qquad y_i' \rightarrow \frac{y_i'}{w_i'}$$

- The equations we get are

$$x_i' = \frac{ax_i + by_i + c}{gx_i + hy_i + a}$$

$$y_i' = \frac{dx_i + ey_i + f}{gx_i + hy_i + a}$$

$$\begin{bmatrix} & \cdots & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \end{bmatrix}$$

- Solve with SVD

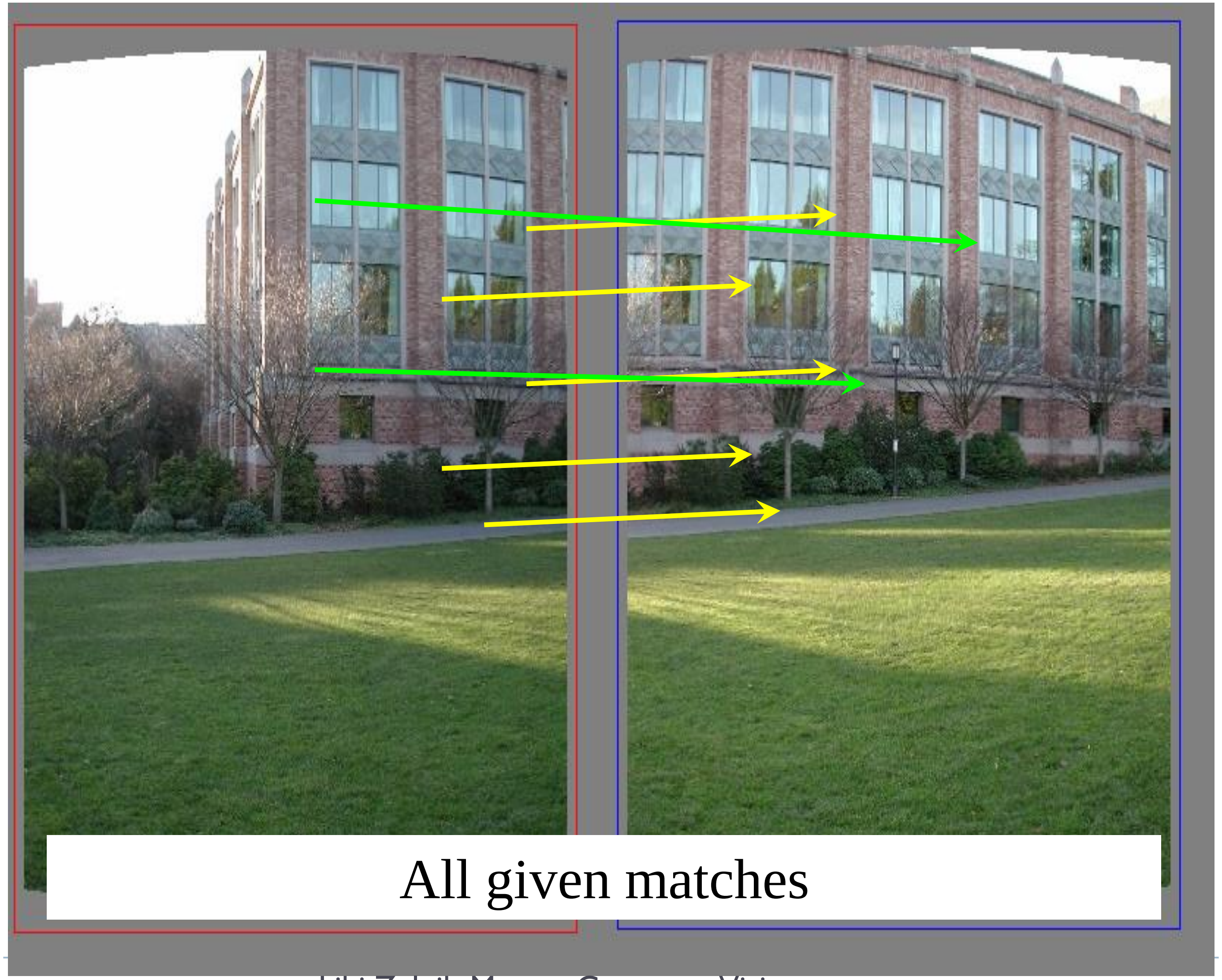
Today

- Matching local features
- Parametric transformations
- Computing parametric transformations
  - Least-squares
  - RANSAC
- Panoramas

RANSAC

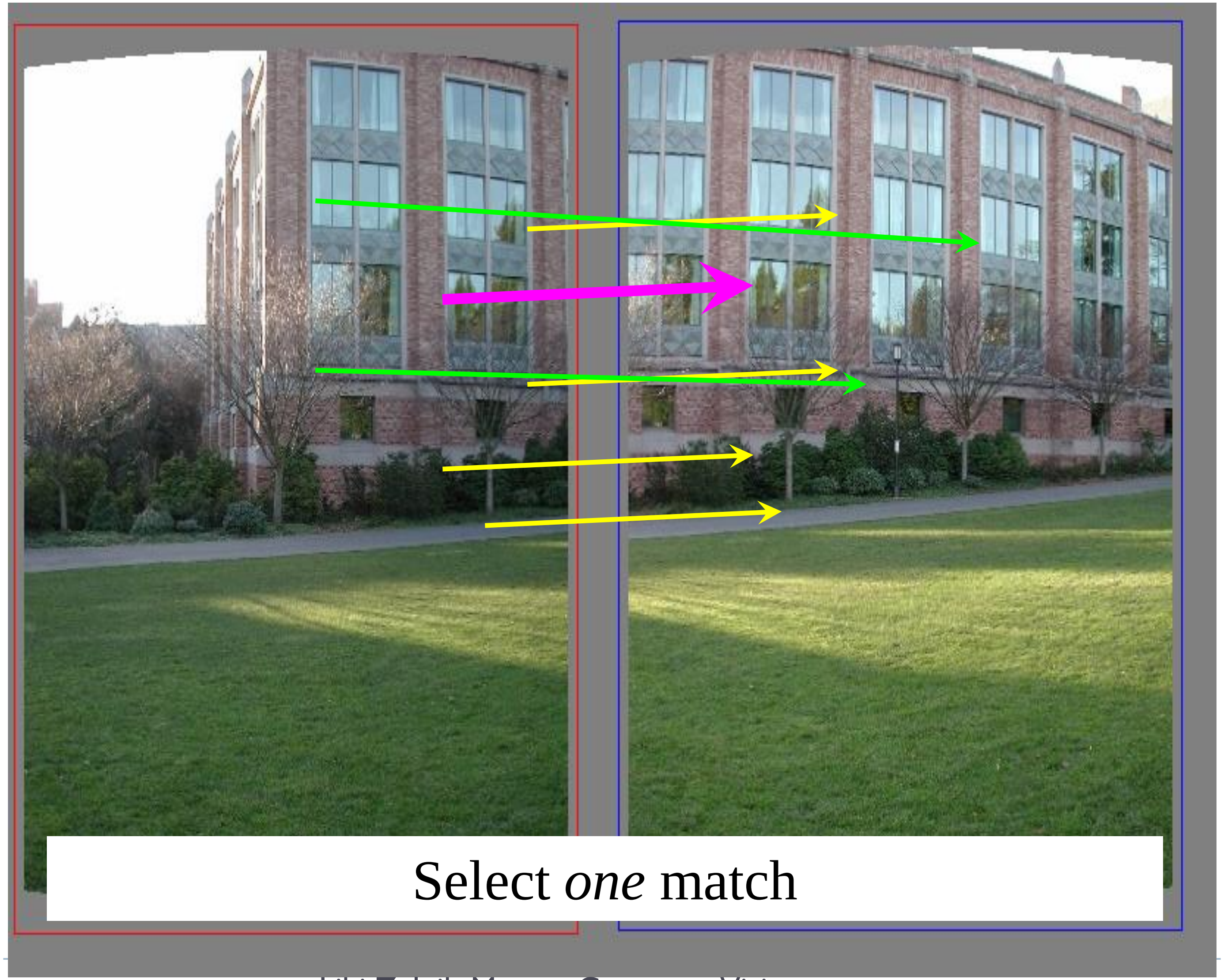
1. Randomly select a *seed group* of matches
2. Compute transformation (using Least-squares) from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers
5. Keep the transformation with the largest number of inliers

RANSAC example: Translation

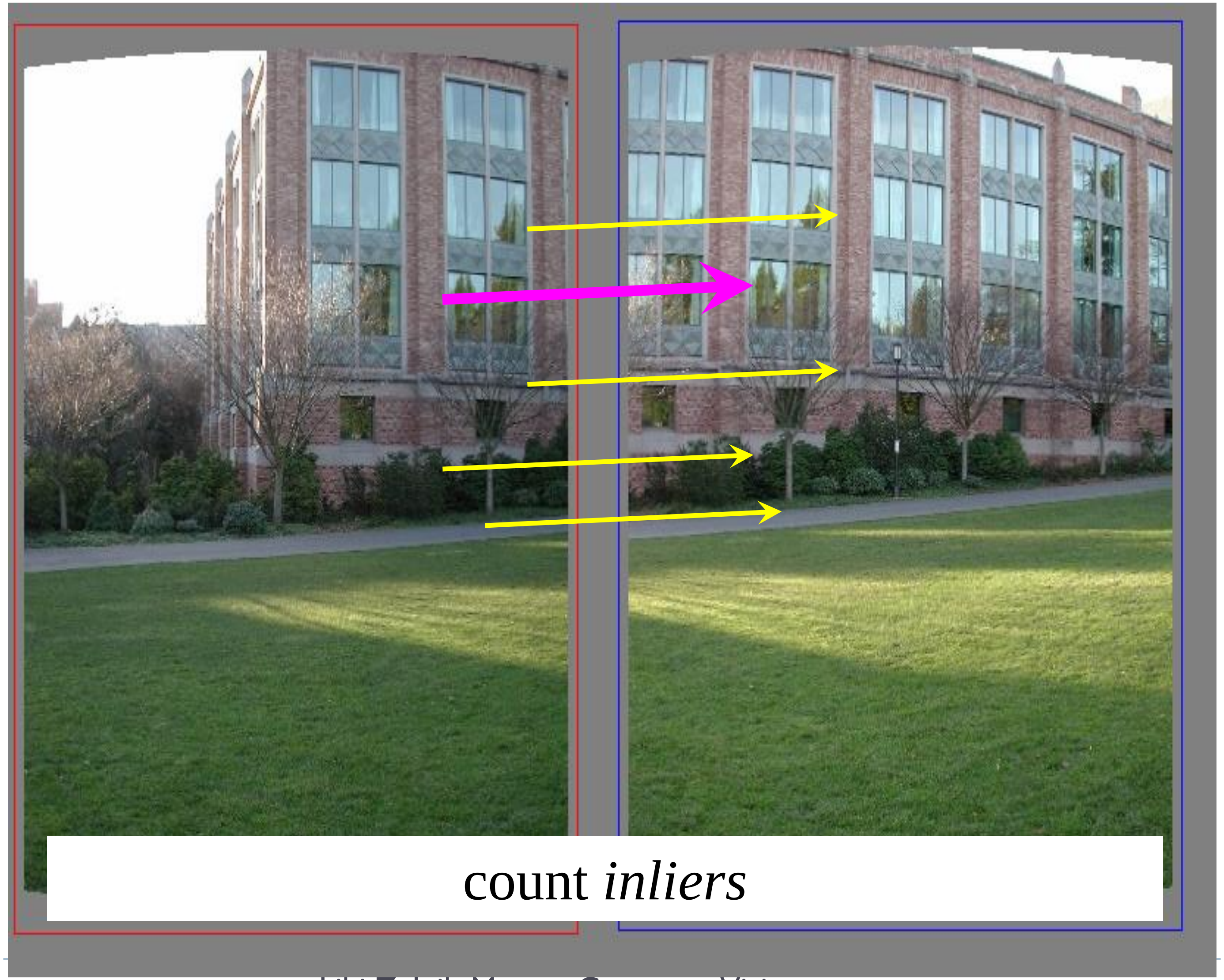


All given matches

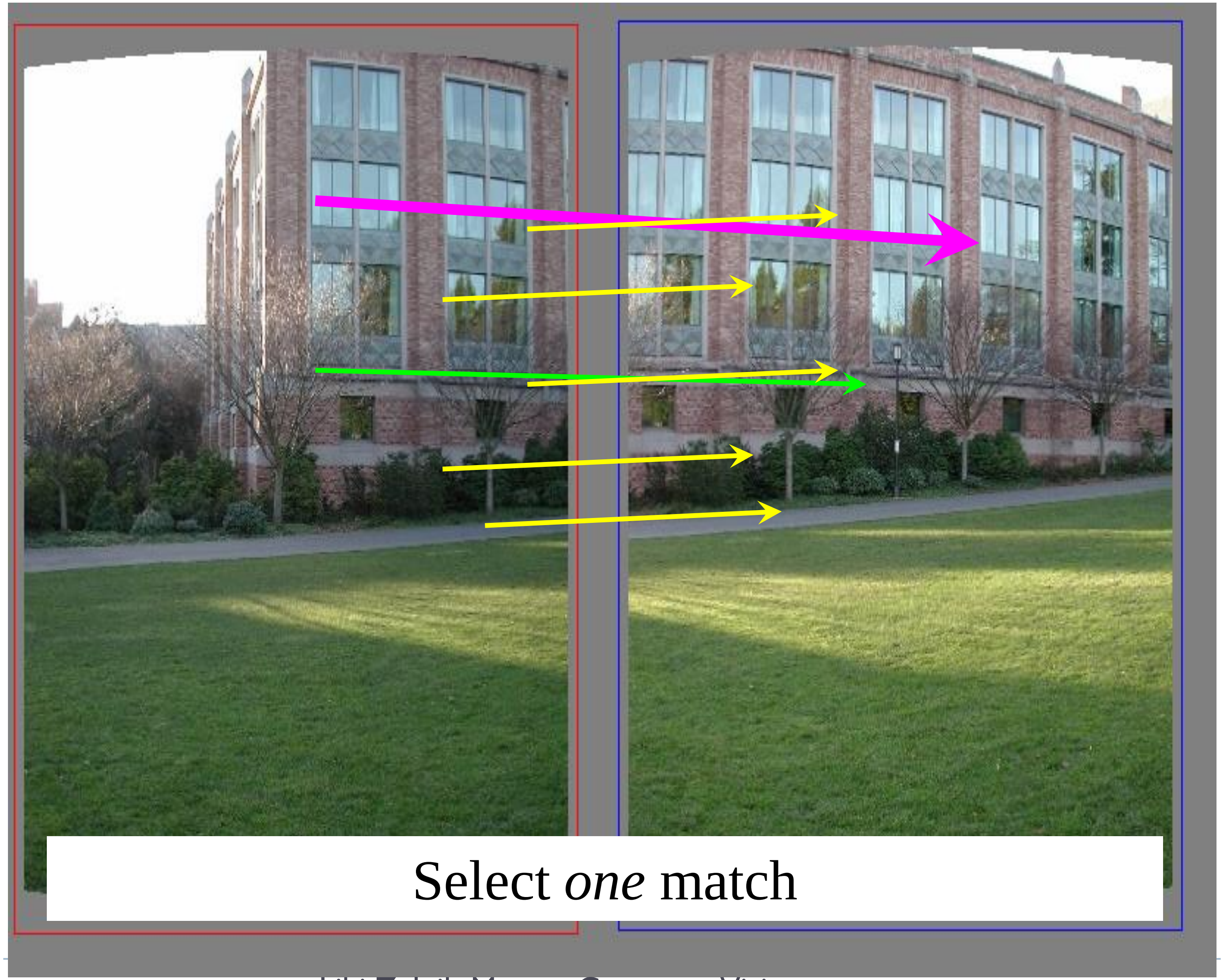
RANSAC example: Translation



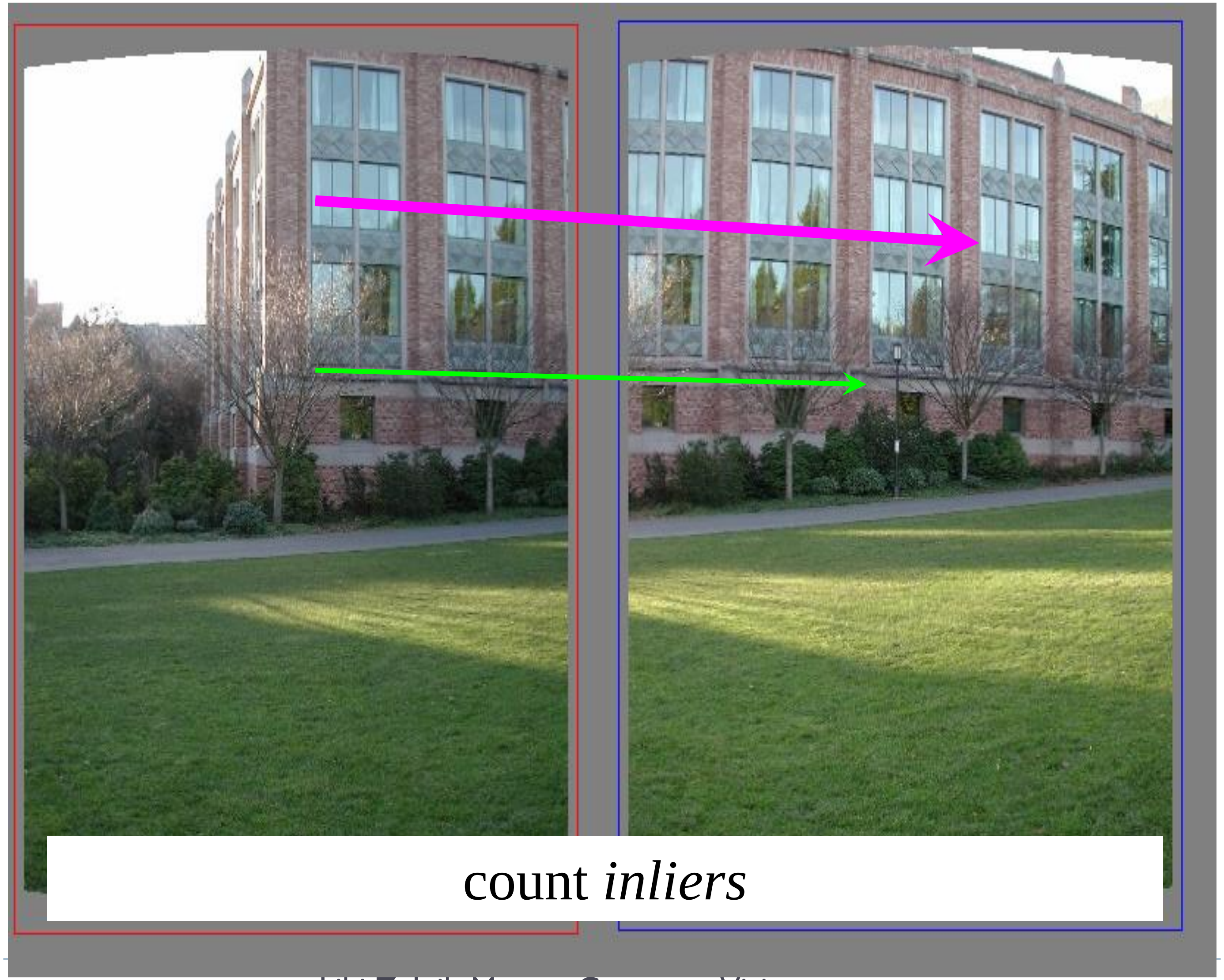
RANSAC example: Translation



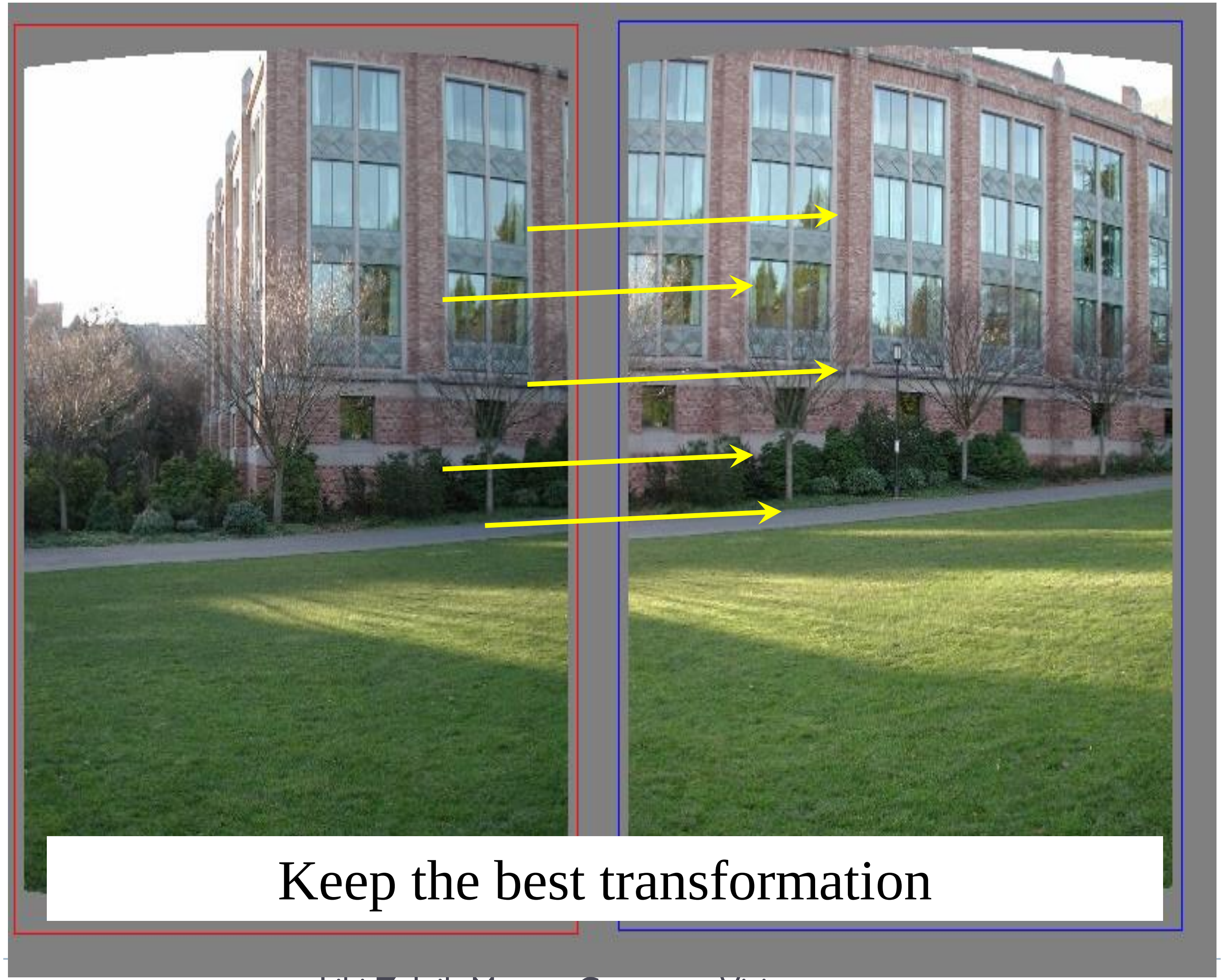
RANSAC example: Translation



RANSAC example: Translation

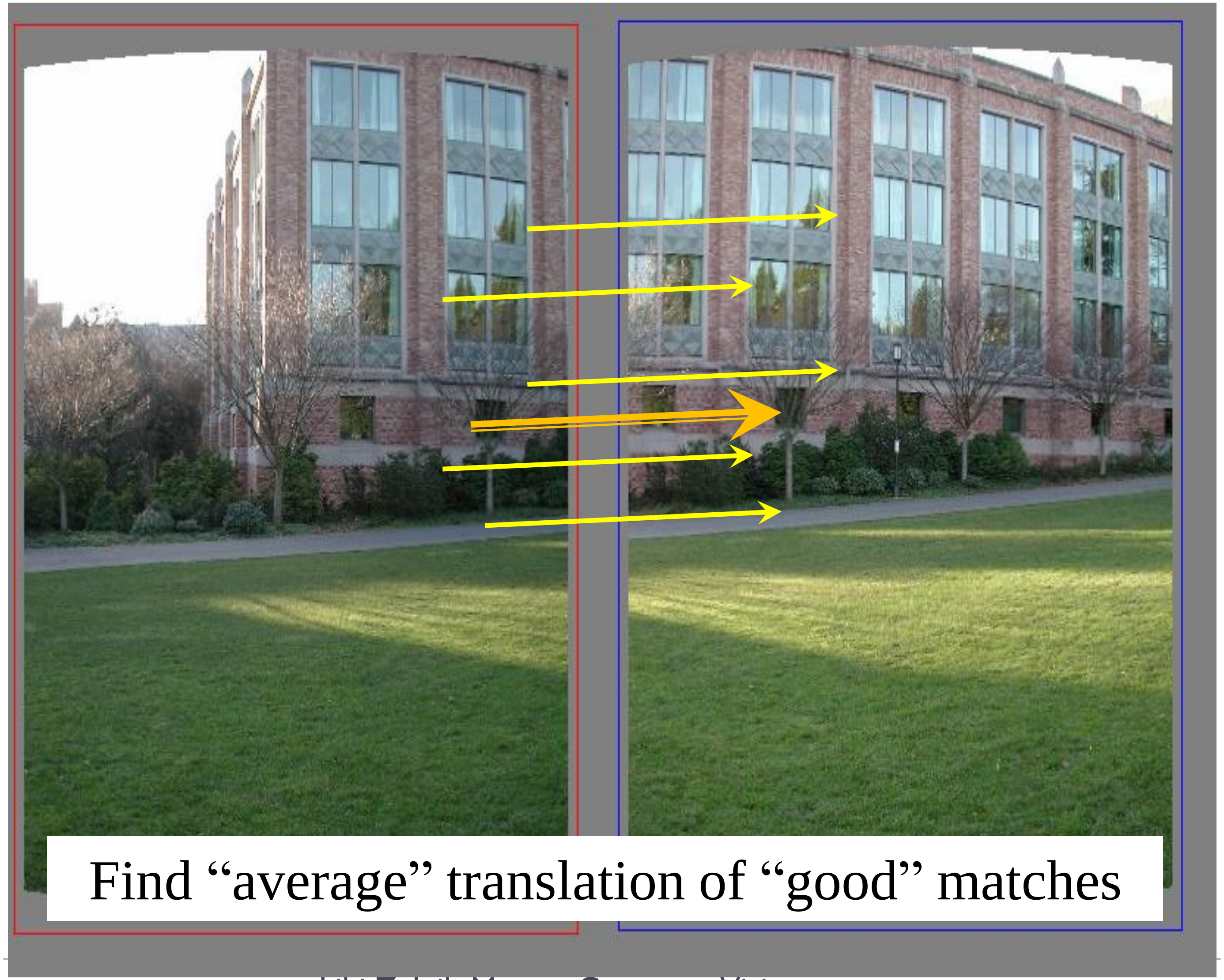


RANSAC example: Translation



Keep the best transformation

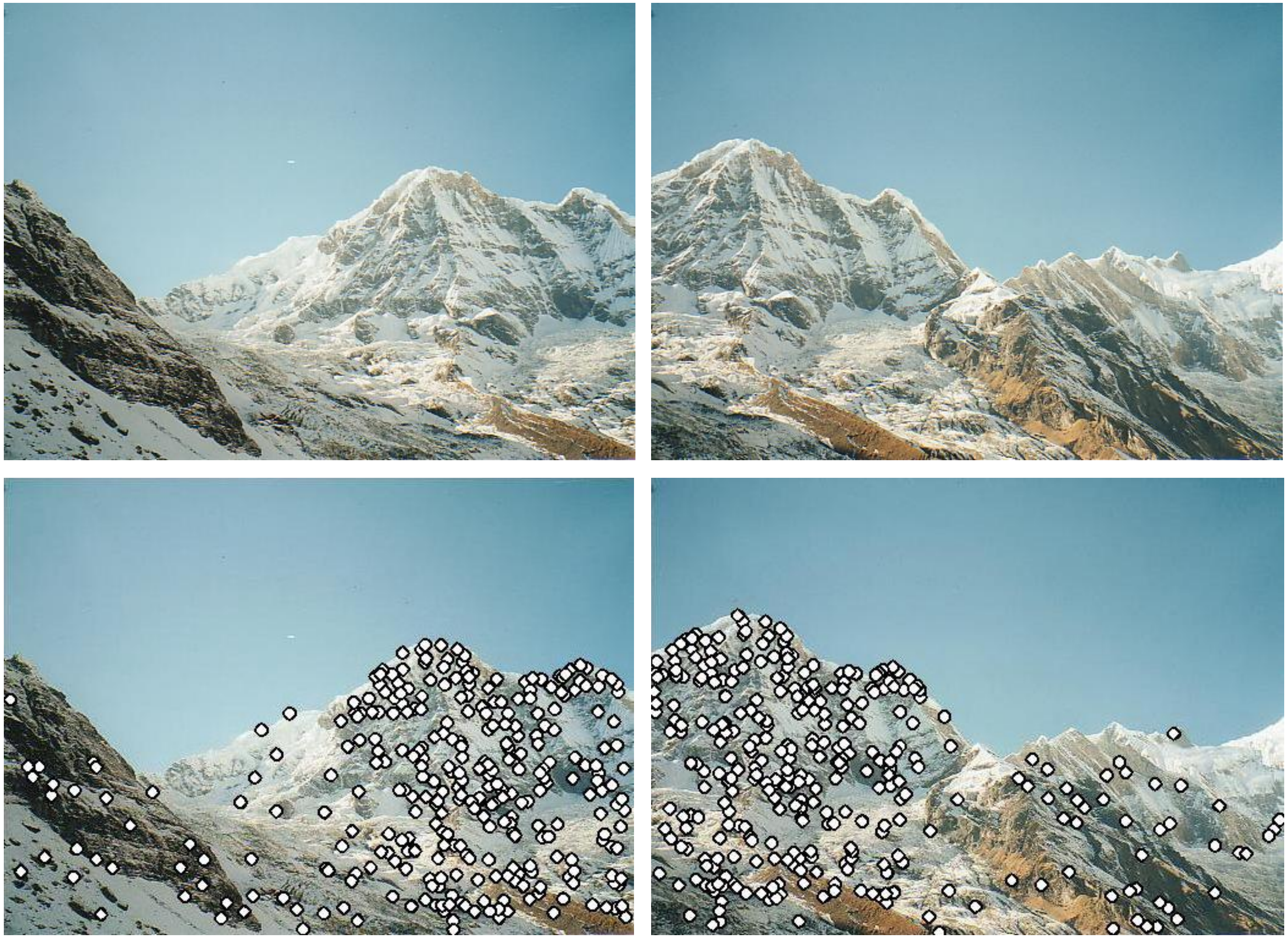
RANSAC example: Translation



Find “average” translation of “good” matches

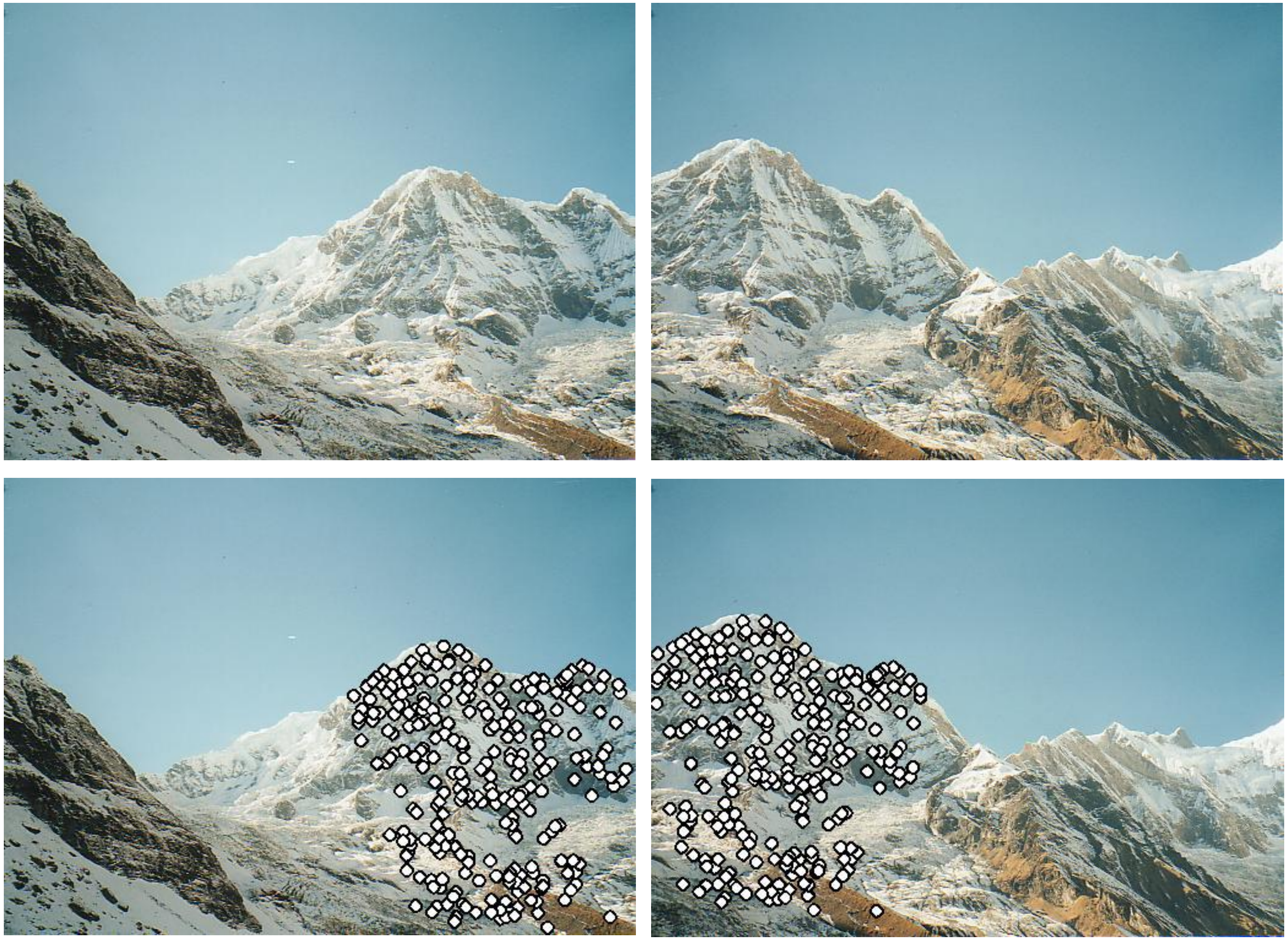
RANSAC example: homography

All matches



RANSAC example: homography

After RANSAC



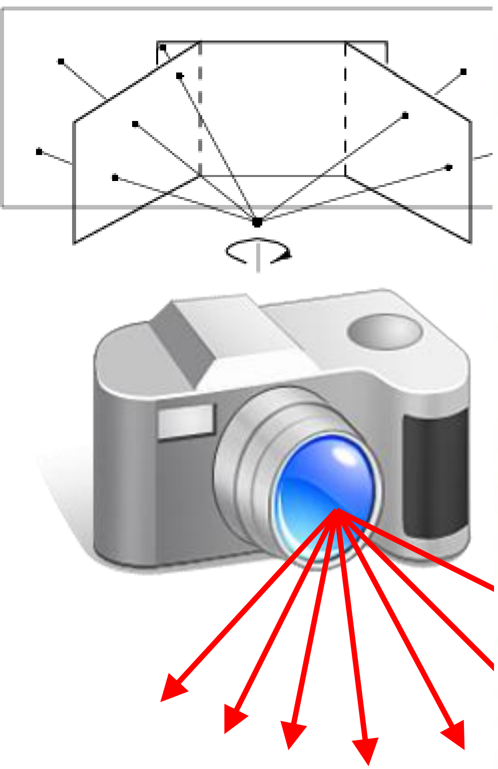
RANSAC example: homography

Applying the homography

Today

- Matching local features
- Parametric transformations
- Computing parametric transformations
  - Least-squares
  - RANSAC
- Panoramas
  - Multi-frame estimation
  - Warping
  - Blending

Panoramas



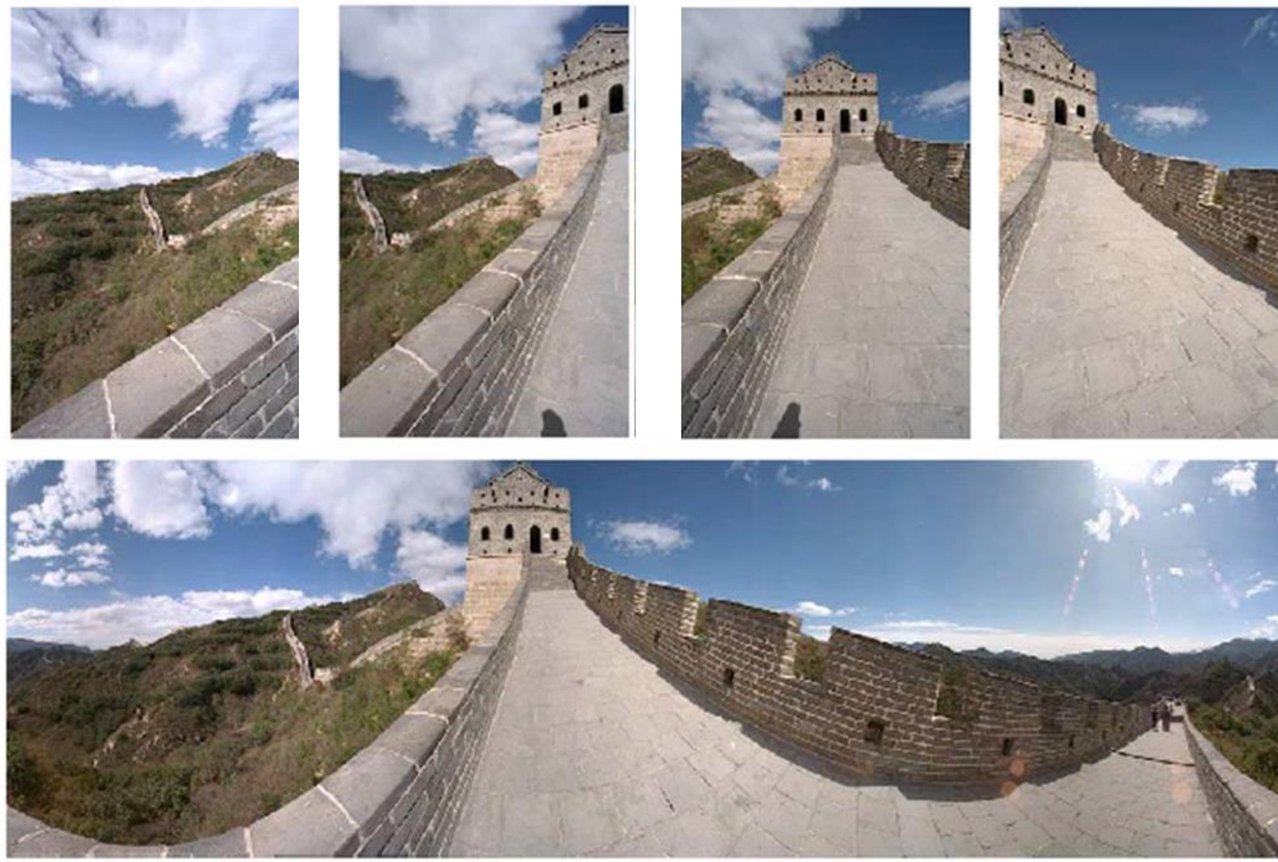
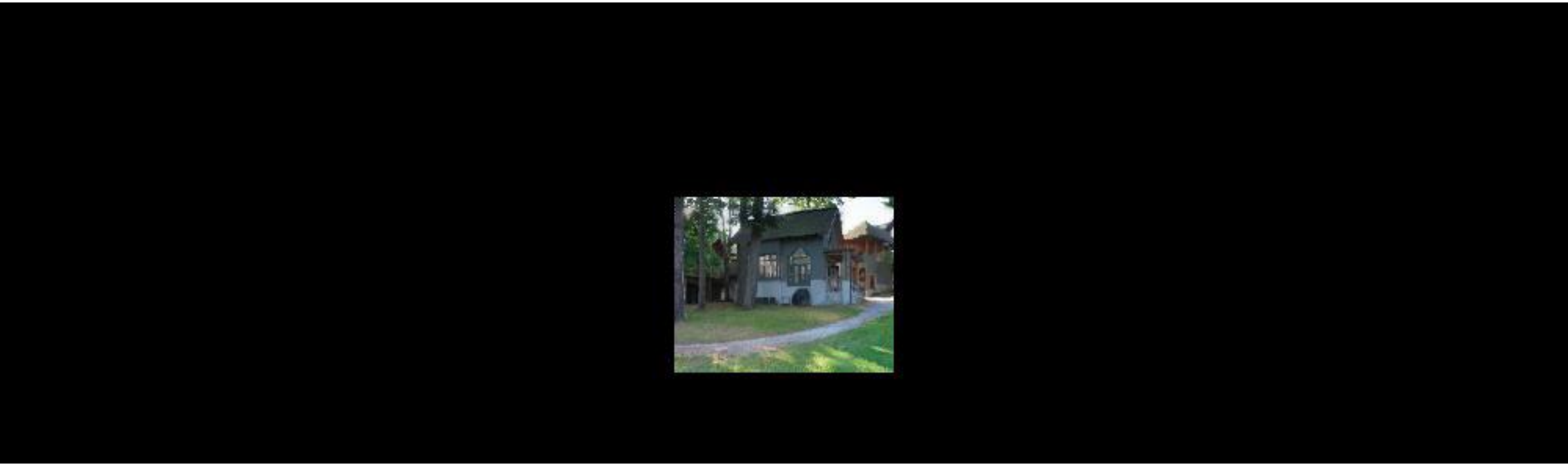


image from S. Seitz

Obtain a wider angle view by combining multiple images.

Panoramas

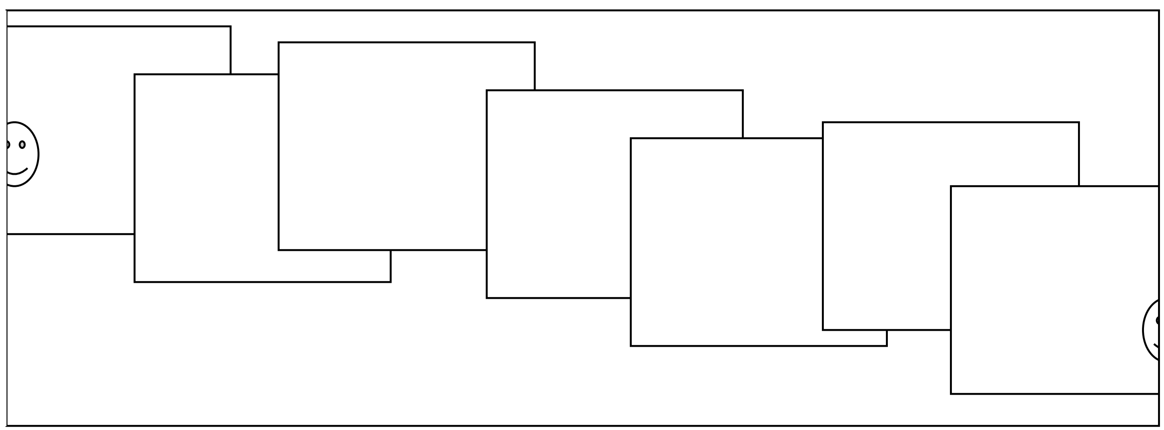


Panoramas

Panoramas

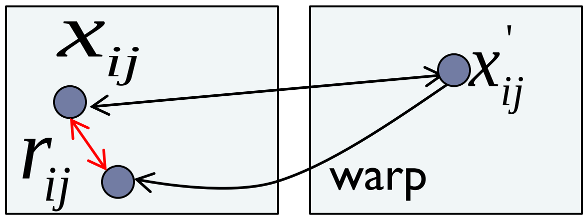
Problem: Drift

- Error accumulation
  - small (vertical) errors accumulate over time
  - apply correction so that sum = 0 (for 360° pan.)



Aligning multiple images

- Bundle adjustment



i'th image pair

$$E = \sum_{i=1}^{all\ pairs} \sum_{j=1}^{matches\ in\ i} f\left(r_{ij}\right)$$

$$f(\mathbf{x}) = \begin{cases} |\mathbf{x}|, & \text{if } |\mathbf{x}| < x_{max} \\ x_{max}, & \text{if } |\mathbf{x}| \geq x_{max} \end{cases}$$

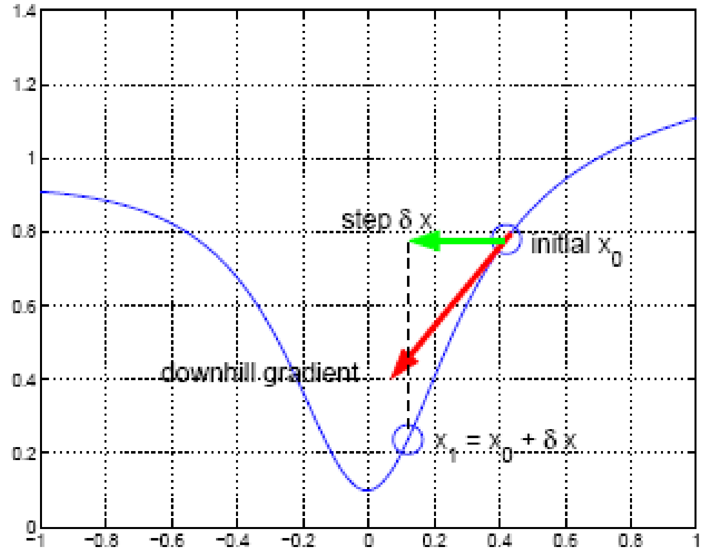
f, x_{max} let us bound the effect of outliers

Gradient descent

$$\mathbf{x}_{min} = \underset{\mathbf{x}}{\operatorname{argmin}} \; f(\mathbf{x}) : \Re^n \to \Re$$

$$x_1 = x_0 + \delta x$$

Given a starting location x_0, we can look at the $\frac{df}{dx}$ and move in the *downhill* direction to generate a new estimate, $x_1 = x_0 + \delta x$.



Newton's method

Fit a quadratic approximation to $f(x)$ using both gradient and curvature information at x.

- Expand $f(x)$ locally using a Taylor series.

$$f(x+\delta x) = f(x) + \delta x f'(x) + \frac{\delta x^2}{2} f''(x) + \text{h.o.t}$$

- Find the δx which minimizes this local quadratic approximation.

$$\delta x = -\frac{f'(x)}{f''(x)}$$

- Update x

$$\boxed{x_{n+1} = x_n - \frac{f'(x)}{f''(x)}}$$

Newton in N dimensions

$$f(\mathbf{x}+\mathbf{h}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \mathbf{h} + \frac{1}{2}\mathbf{h}^\top \mathtt{H}(\mathbf{x})\mathbf{h}$$

$$\boxed{\mathbf{x}_{n+1} = \mathbf{x}_n - \lambda_n \mathtt{H}(\mathbf{x}_n)^{-1}\nabla f(\mathbf{x}_n)}$$

The gradient $\nabla f(\mathbf{x})$ of $f(\mathbf{x})$ is the vector

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1} \dots \frac{\partial f}{\partial x_N}\right]$$

The Hessian $\mathtt{H}(\mathbf{x})$ of $f(\mathbf{x})$ is the symmetric matrix

$$\mathtt{H}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \vdots & \ddots & \\ \frac{\partial^2 f}{\partial x_1 \partial x_N} & & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$

Minimizing sum of residuals

$$f(\mathbf{x}) = \sum_{i=1}^{M} r_i^2$$

$$\mathtt{J}(\mathbf{x}) = \begin{pmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_N} \\ \vdots & \ddots & \\ \frac{\partial r_M}{\partial x_1} & & \frac{\partial r_M}{\partial x_N} \end{pmatrix}$$

$$\nabla f(\mathbf{x}) = 2\mathtt{J}^\top \mathbf{r}$$

$$\mathtt{H}(\mathbf{x}) = 2\mathtt{J}^\top \mathtt{J} + \sum_{i=1}^{M} r_i \frac{\mathrm{d}^2 r_i}{\mathrm{d}\mathbf{x}^2}$$

Levenberg-Marquardt

$$\mathtt{H}(\mathbf{x}, \lambda) = 2\mathtt{J}^{\top}\mathtt{J} + \lambda\mathtt{I}$$

Levenberg-Marquardt

$$\mathtt{H}(\mathbf{x}, \lambda) = 2\mathtt{J}^\top \mathtt{J} + \lambda \mathtt{I}$$

All unknowns $\qquad \boldsymbol{\Theta} = (\mathbf{J}^T\mathbf{J} + \sigma^2 \mathbf{C}_p^{-1})^{-1}\mathbf{J}^T\mathbf{r}$

Compute analytically $\qquad J = \dfrac{\partial r}{\partial \Theta}$ $\qquad$ Jacobian

Prior parameter
Covariance matrix $\qquad C_p \qquad \sigma_\theta = \pi/16 \qquad \sigma_f = \bar{f}/10$

Step size $\qquad \sigma$

Multi-frame optimization

Rotation matrix

$$\mathbf{R}_i = e^{[\boldsymbol{\theta}_i]_\times}, \quad [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}$$

Calibration matrix

$$\mathbf{K}_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Unknown parameters for image *i*

$$\theta_i = \begin{bmatrix} \theta_{i1} & \theta_{i2} & \theta_{i3} & f_i \end{bmatrix}^T$$

Bundle adjustment formulations

All pairs optimization:

Confidence / uncertainty of point i in image j

$$E_{\text{all-pairs-2D}} = \sum_i \sum_{jk} c_{ij} c_{ik} \|\tilde{\boldsymbol{x}}_{ik}(\hat{\boldsymbol{x}}_{ij}; \boldsymbol{R}_j, f_j, \boldsymbol{R}_k, f_k) - \hat{\boldsymbol{x}}_{ik}\|^2,$$

Map 2D point i in image j to 2D point in image k

Full bundle adjustment, using 3-D point positions $\{\boldsymbol{x}_i\}$

$$E_{\text{BA-2D}} = \sum_i \sum_j c_{ij} \|\tilde{\boldsymbol{x}}_{ij}(\boldsymbol{x}_i; \boldsymbol{R}_j, f_j) - \hat{\boldsymbol{x}}_{ij}\|^2,$$

Map 3D point i to 2D point in image i

Bundle adjustment using 3-D ray:

$$E_{\text{BA-3D}} = \sum_i \sum_j c_{ij} \|\tilde{\boldsymbol{x}}_i(\hat{\boldsymbol{x}}_{ij}; \boldsymbol{R}_j, f_j) - \boldsymbol{x}_i\|^2,$$

3-D ray from point i

All-pairs 3-D ray formulation:

$$E_{\text{all-pairs-3D}} = \sum_i \sum_{jk} c_{ij} c_{ik} \|\tilde{\boldsymbol{x}}_i(\hat{\boldsymbol{x}}_{ij}; \boldsymbol{R}_j, f_j) - \tilde{\boldsymbol{x}}_i(\hat{\boldsymbol{x}}_{ik}; \boldsymbol{R}_k, f_k)\|^2.$$

3-D ray from points i and j

Projected point

$$\boxed{\tilde{\boldsymbol{x}}_{ij} \sim \boldsymbol{K}_j \boldsymbol{R}_j \boldsymbol{x}_i \;\text{ and }\; \boldsymbol{x}_i \sim \boldsymbol{R}_j^{-1} \boldsymbol{K}_j^{-1} \tilde{\boldsymbol{x}}_{ij},}$$

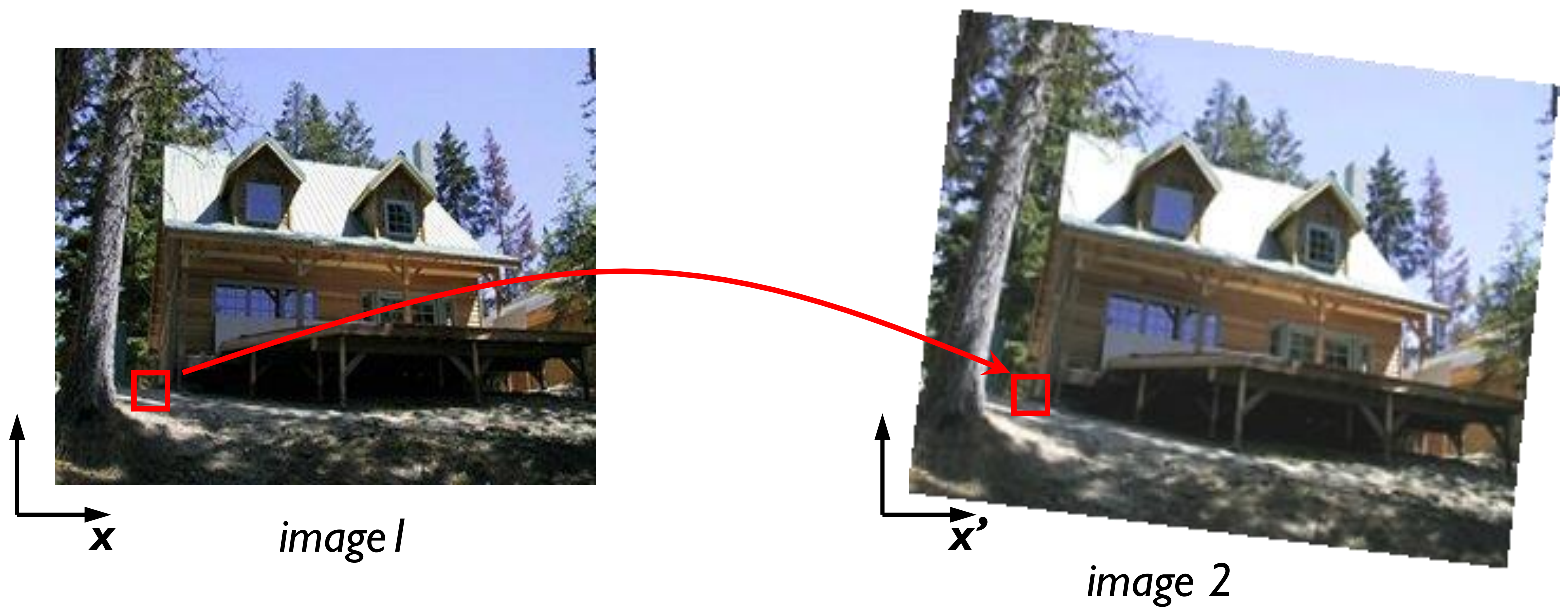
3-D ray from point

Today

- Matching local features
- Parametric transformations
- Computing parametric transformations
  - Least-squares
  - RANSAC
- Panoramas
  - Multi-frame estimation
  - Warping
  - Blending

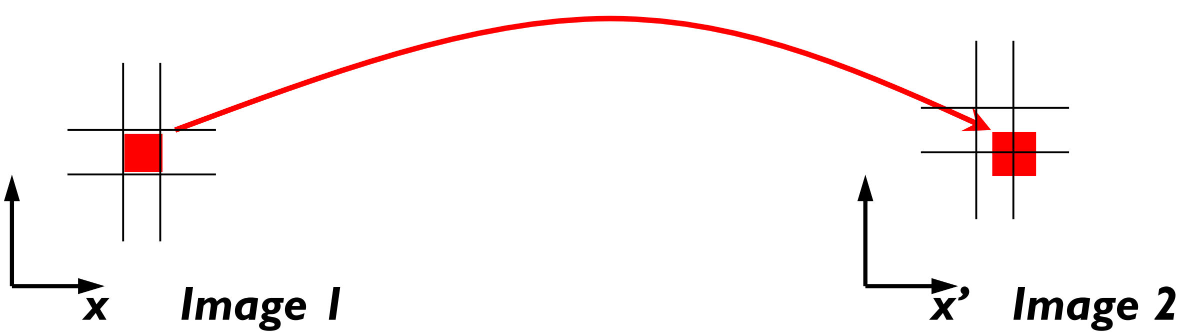
What we need to solve

- Given source and target images, and the transformation between them, how do we align them?
- Send each pixel ***x*** in *image1* to its corresponding location ***x'*** in *image 2*



Forward Warping

- What if pixel lands "between" two pixels?
- Answer: add "contribution" to several pixels and normalize (*splatting*)



Forward Warping

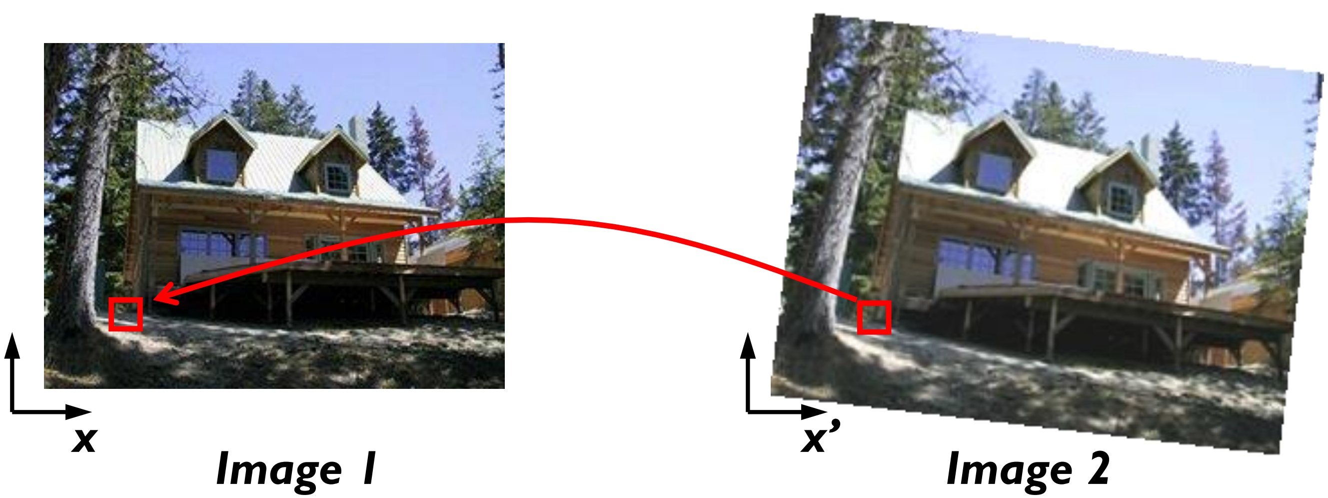
- What if pixel lands "between" two pixels?

- <u>Answer</u>: add "contribution" to several pixels and normalize (*splatting*)
- <u>Limitation</u>: Holes (some pixels are never visited)

Inverse Warping

- For each pixel ***x'*** in image 2 find its origin ***x*** in image 1

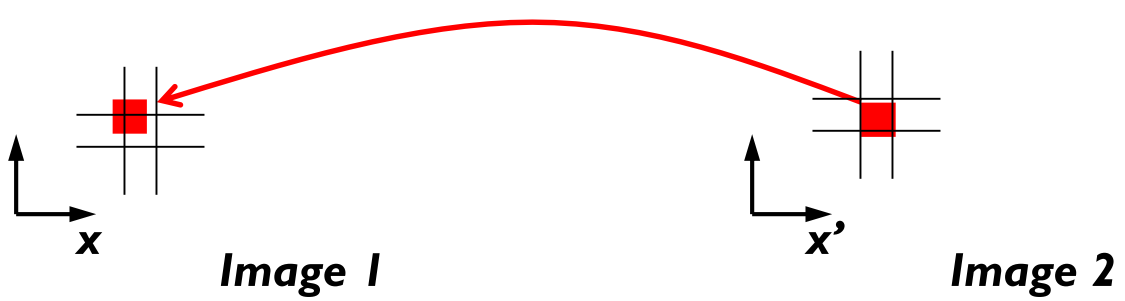
- <u>Problem</u>: What if pixel comes from "between" two pixels?



Inverse Warping

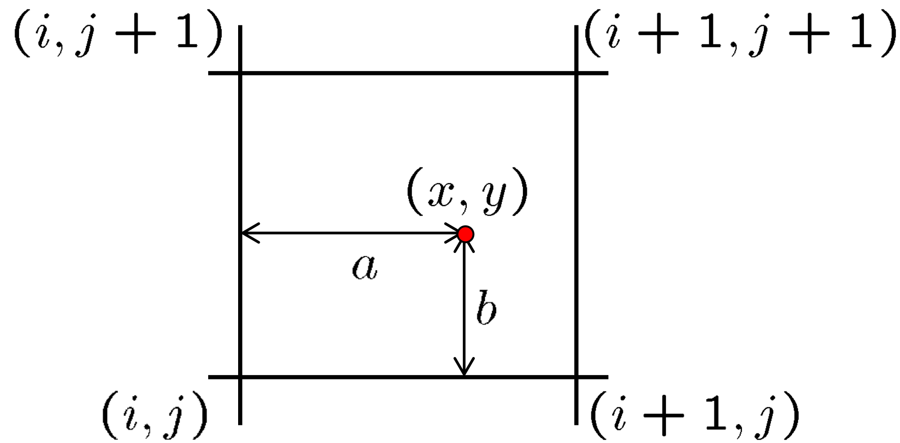
- For each pixel ***x'*** in image 2 find its origin ***x*** in image 1

- Problem: What if pixel comes from "between" two pixels?
- Answer: *interpolate* color value from *neighbors*



Bilinear interpolation

Sampling at *f(x,y)*:



$$
\begin{aligned}
f(x,y) = \; & (1-a)(1-b) && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - Bilinear interpolation
  - bicubic interpolation
  - sinc / FIR

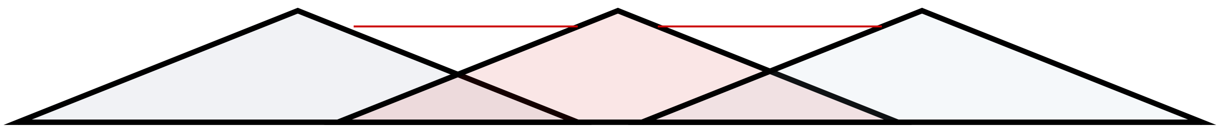
- Needed to prevent "jaggies" and "texture crawl"

Today

- Matching local features
- Parametric transformations
- Computing parametric transformations
  - Least-squares
  - RANSAC
- Panoramas
  - Multi-frame estimation
  - Warping
  - Blending

Image feathering

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



- 1. Generate *weight map* for each image
- 2. Sum up all of the weights and divide by sum: weights sum up to 1: $w_i' = w_i / (\sum_i w_i)$

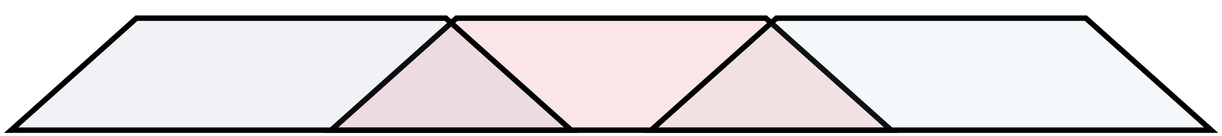


Image feathering

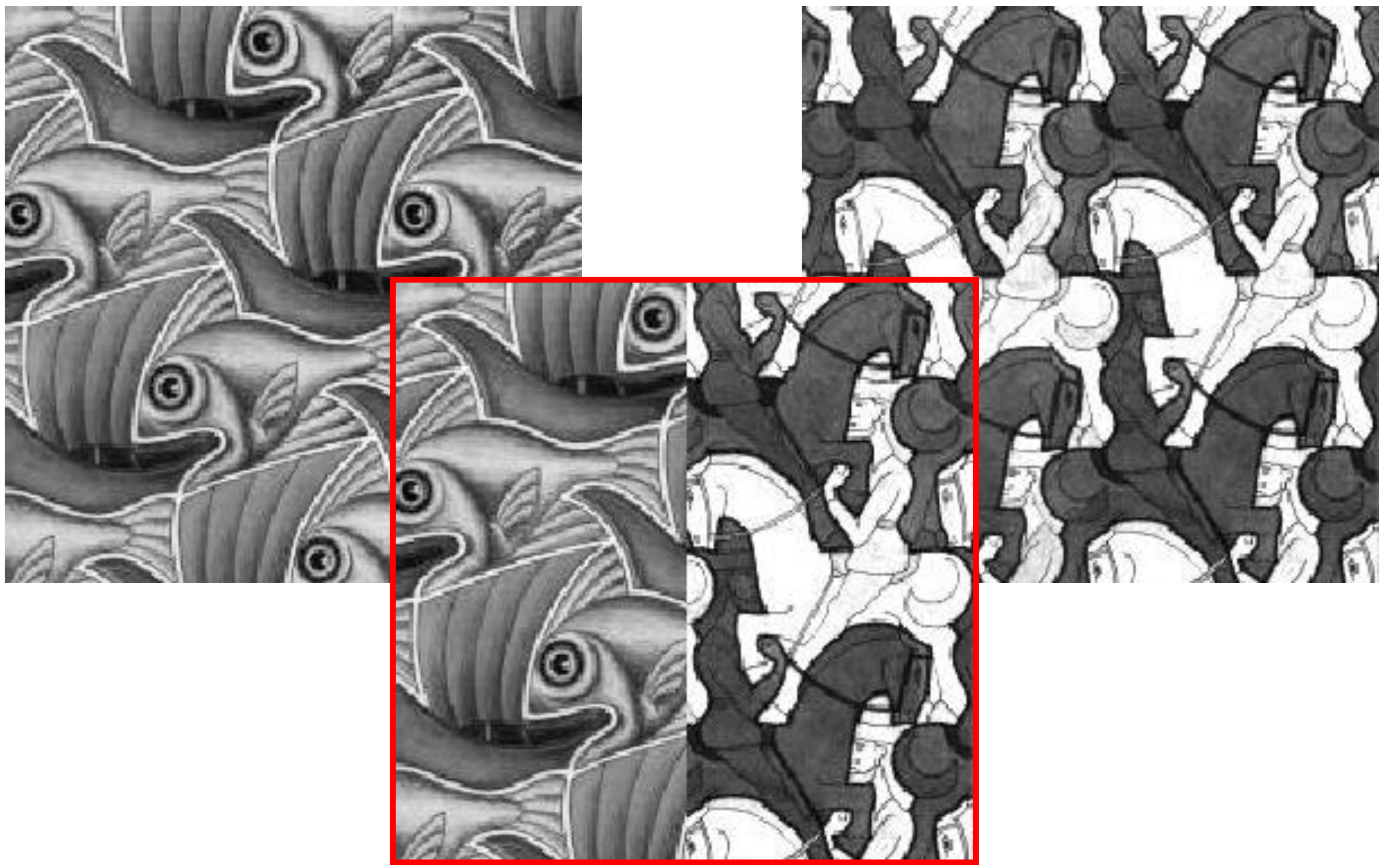
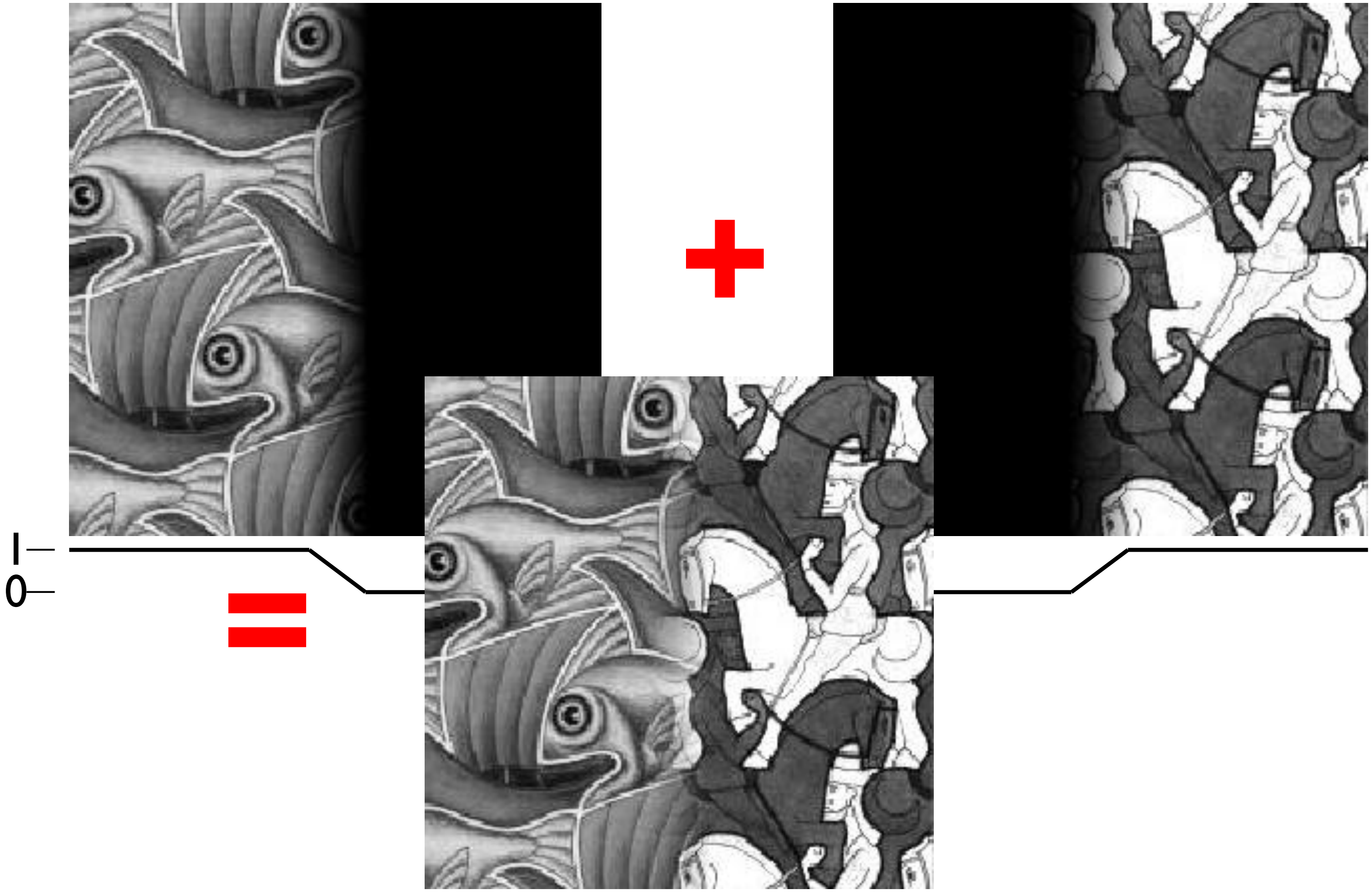


Image feathering



Panoramas – summary

- Detect features
- Compute transformations between pairs of frames
- Refine transformations using bundle-adjustment
- Warp all images onto a single coordinate system
- Blend

End – Alignment

Now you know how it works