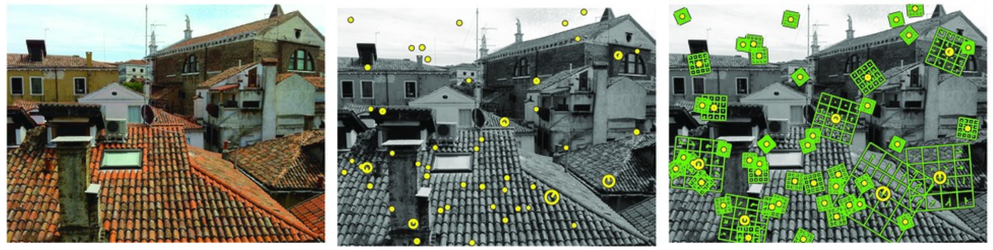
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 13: Planar Geometry and RANSAC

Menu for Today

Topics:

- **Planar** Geometry
- **Image Alignment**, Object Recognition
- **RANSAC**

Readings:

- **Today's** Lecture: Szeliski 2.1, 8.1, Forsyth & Ponce 10.4.2

Reminders:

- **Assignment 4**: RANSAC and Panorama Stitching — **now available**

Image Alignment

- Aim: warp our images together using a 2D transformation

Image Alignment

- Aim: warp our images together using a 2D transformation

Image Alignment

- Find corresponding (matching) points between the images

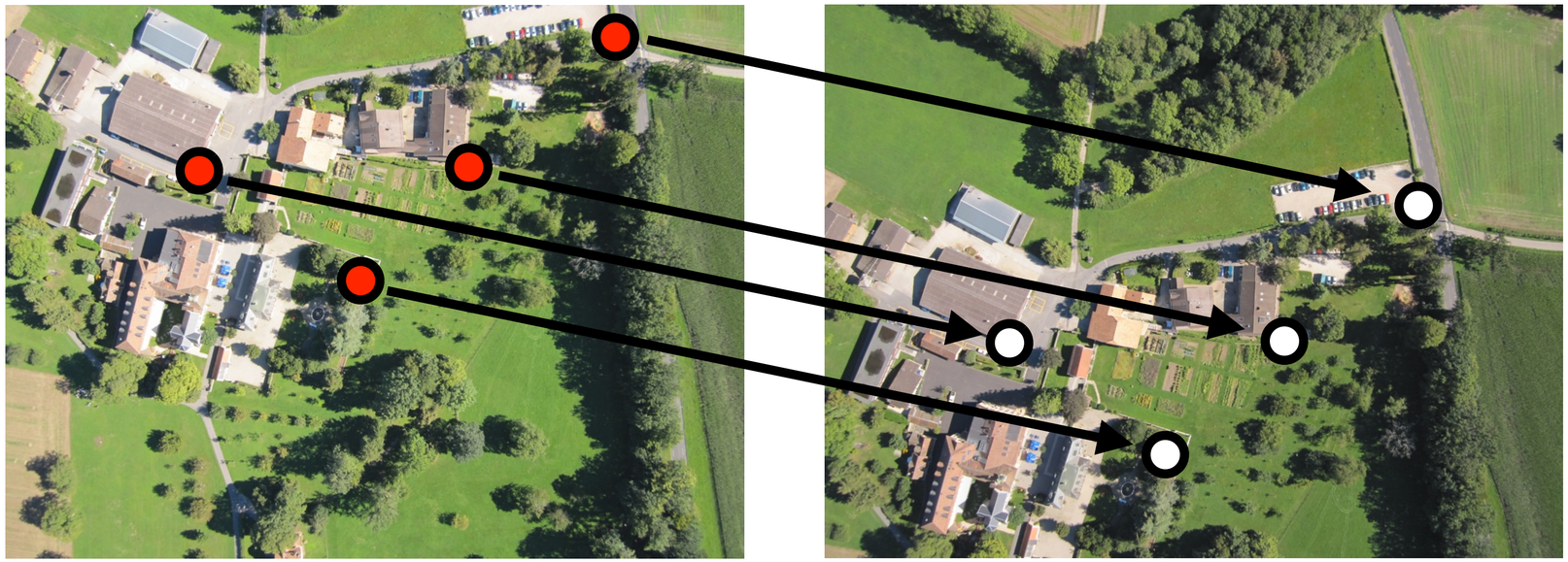


Image Alignment

- Compute the transformation to align the points

Image Alignment

- We can also use this transformation to reject outliers

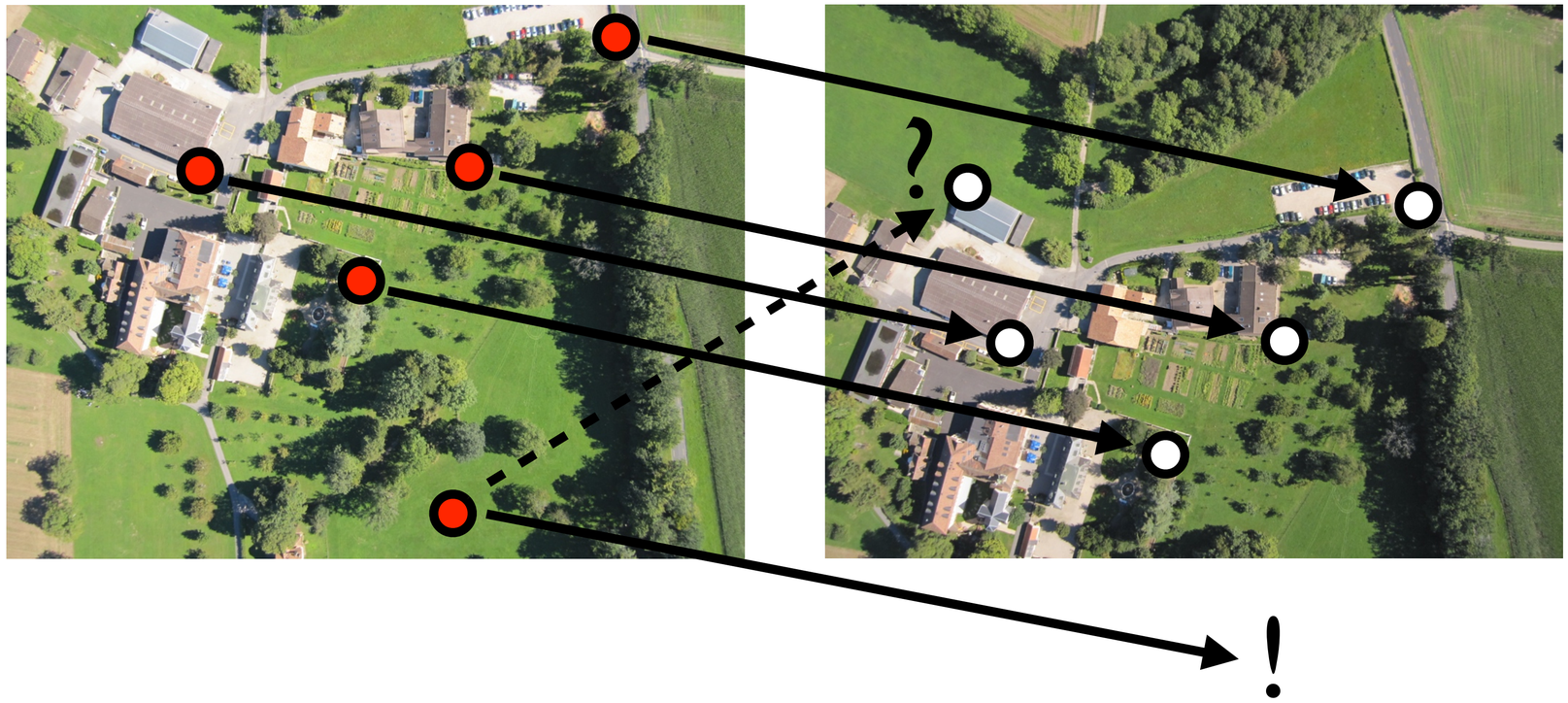


Image Alignment

- We can also use this transformation to reject outliers

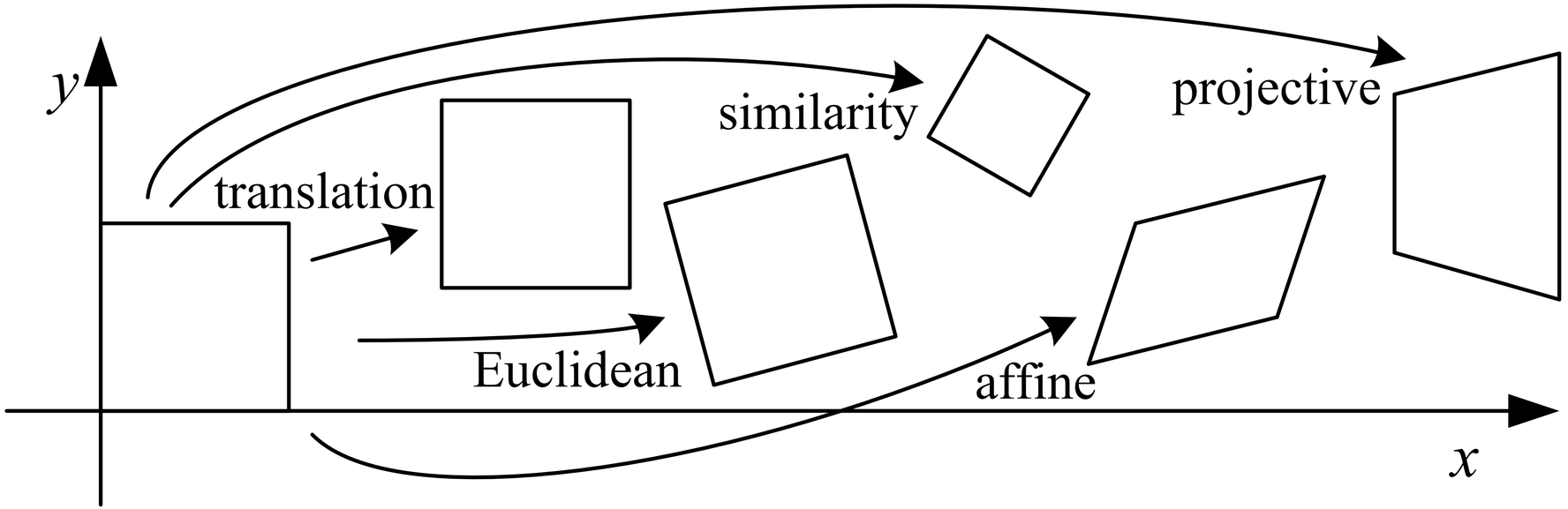
Planar Geometry

- 2D Linear + Projective transformations
  - Euclidean, Similarity, Affine, Homography
- Robust Estimation and RANSAC
  - Estimating 2D transforms with noisy correspondences

[Szeliski 2.1, 8.1.4]

2D Transformations

- We will look at a family that can be represented by 3x3 matrices



This group represents perspective projections of **planar surfaces** in the world

[Szeliski 2.1]

Affine Transformations

- Transformed points are a linear function of the input points

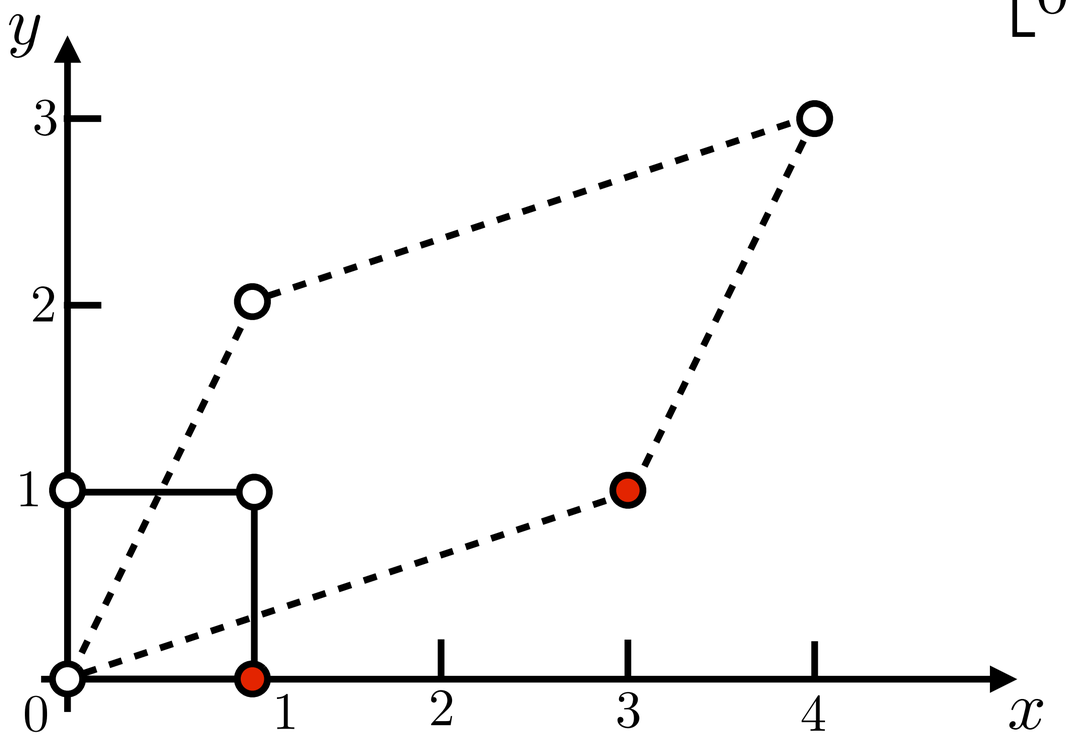
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}$$

- This can be written as a single matrix multiplication

13.1

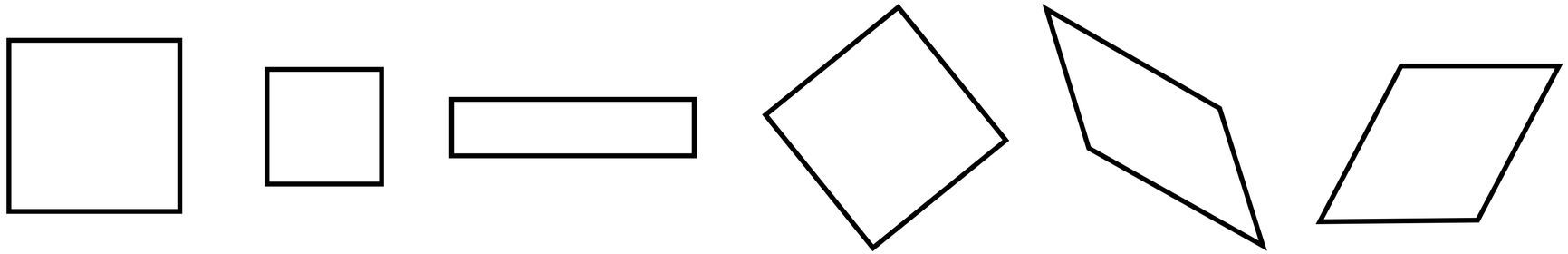
Linear Transformations

- Consider the action of the unit square under $\begin{bmatrix} 3 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

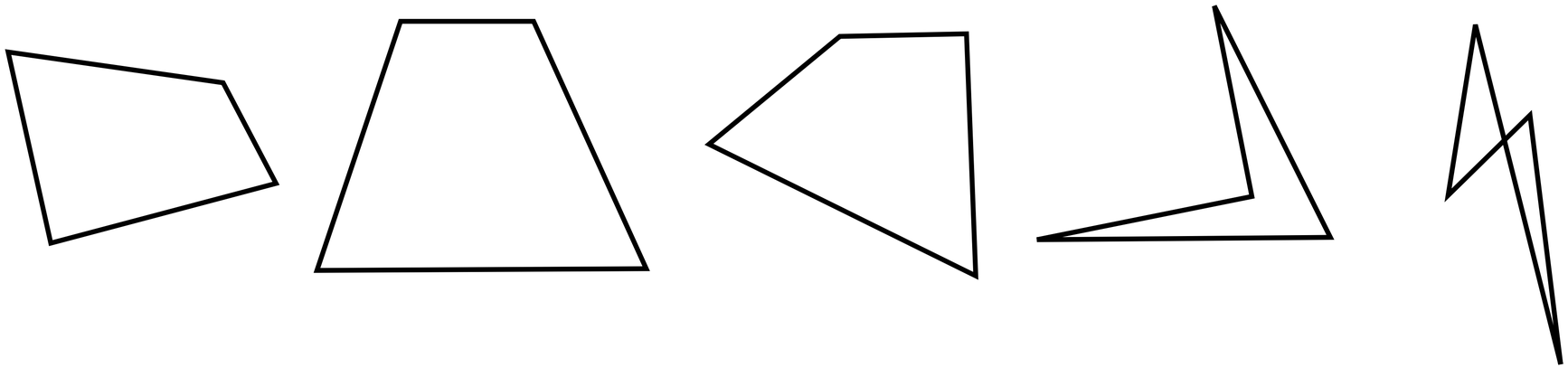


13.2

Linear Transform Examples



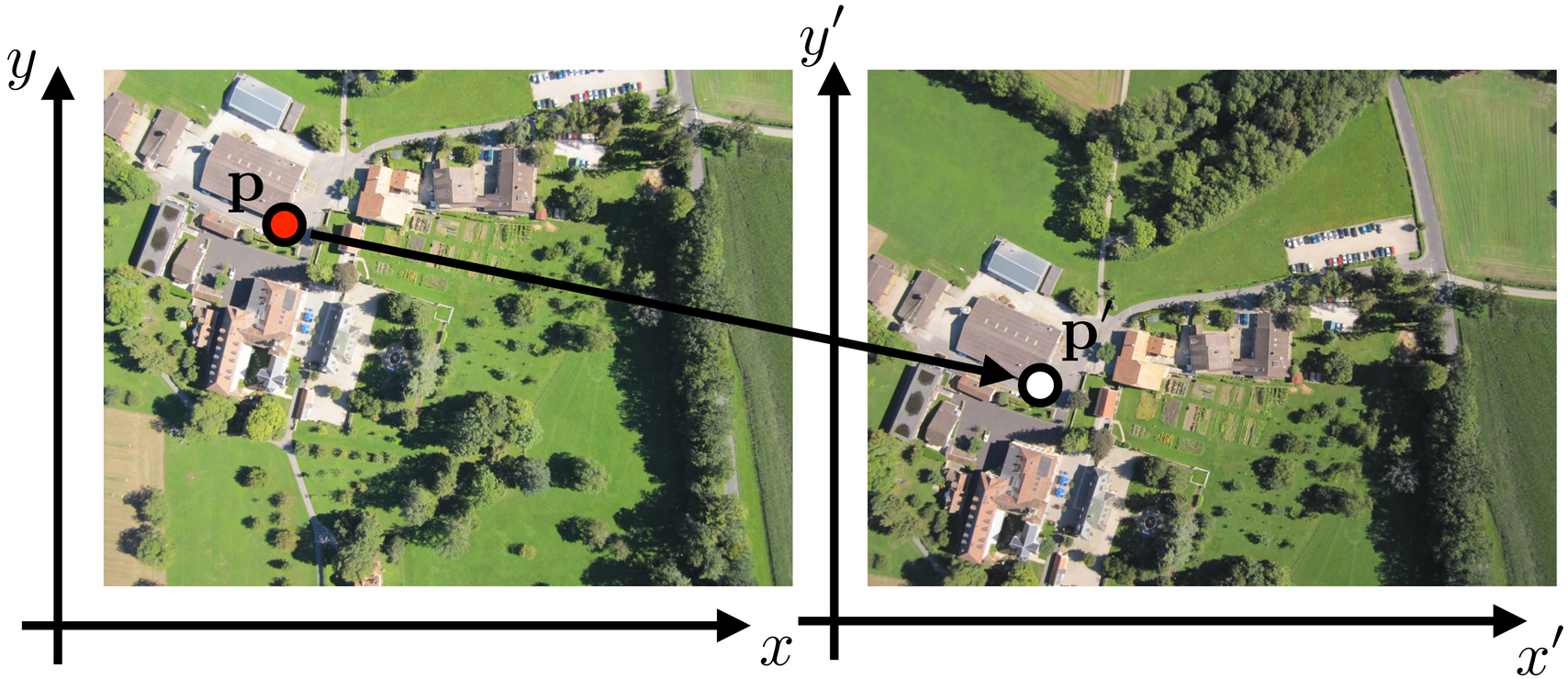
Translation, rotation, scale, shear (parallel lines preserved)



These transforms are not affine (parallel lines not preserved)

Linear Transformations

- Consider a single point correspondence



$$\begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

How many points are needed to solve for **a**?

Computing Affine Transforms

- Lets compute an affine transform from correspondences:

$$\begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

- Re-arrange unknowns into a vector

13.3

Computing Affine Transforms

- Linear system in the unknown parameters $\mathbf{a}$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}$$

- Of the form

$$\mathbf{Ma} = \mathbf{y}$$

Solve for $\mathbf{a}$ using Gaussian Elimination

Computing Affine Transforms

- We can now map any other points between the two images

$$\begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Computing Affine Transforms

- Or resample one image in the coordinate system of the other

This allows us to “stitch” the two images

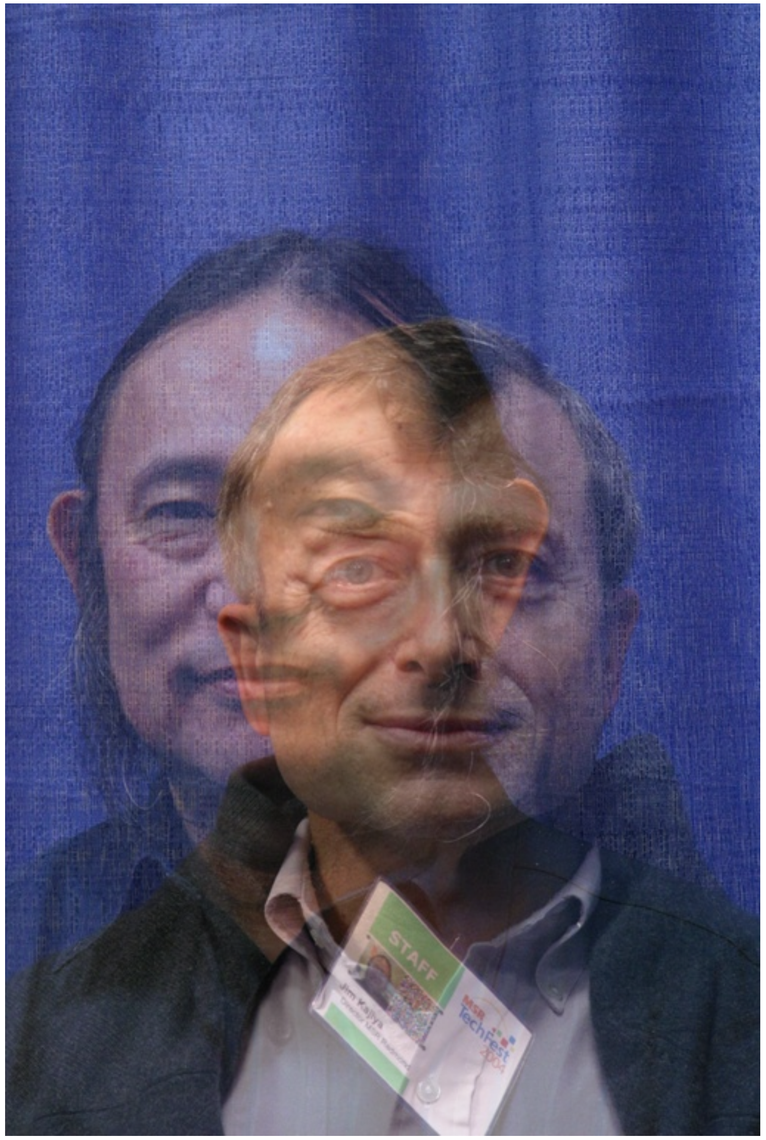
Linear Transformations

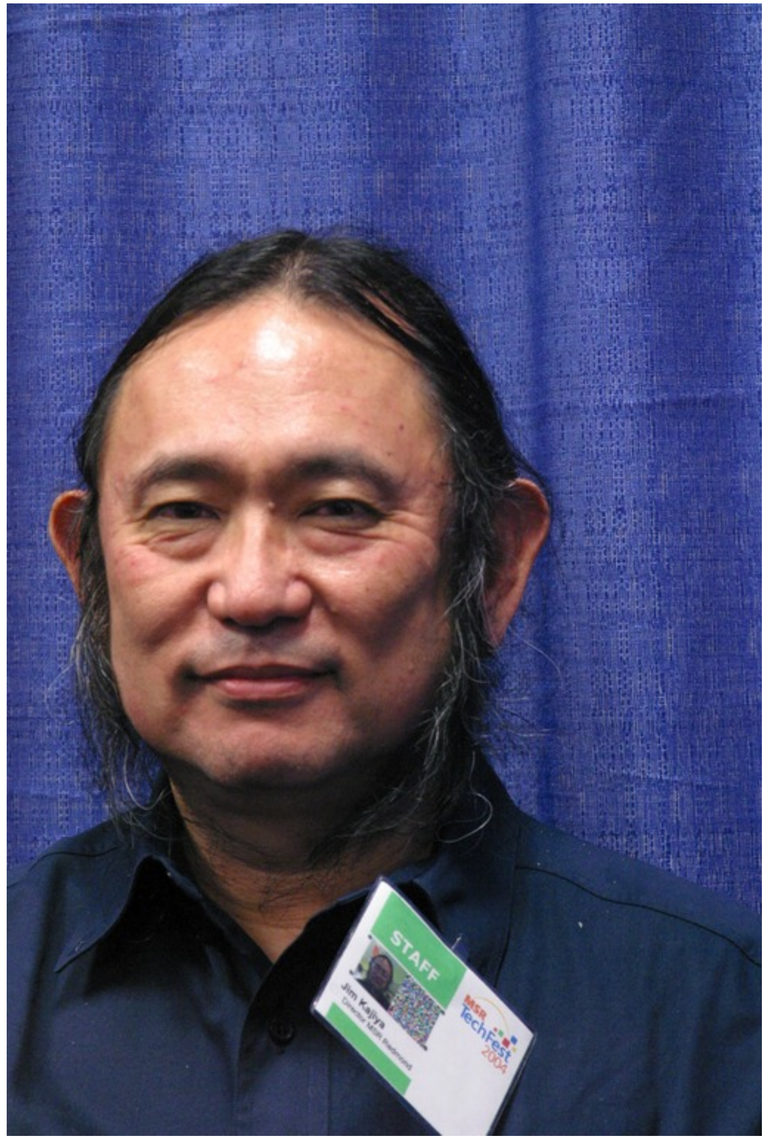
- Other linear transforms are special cases of affine

13.4

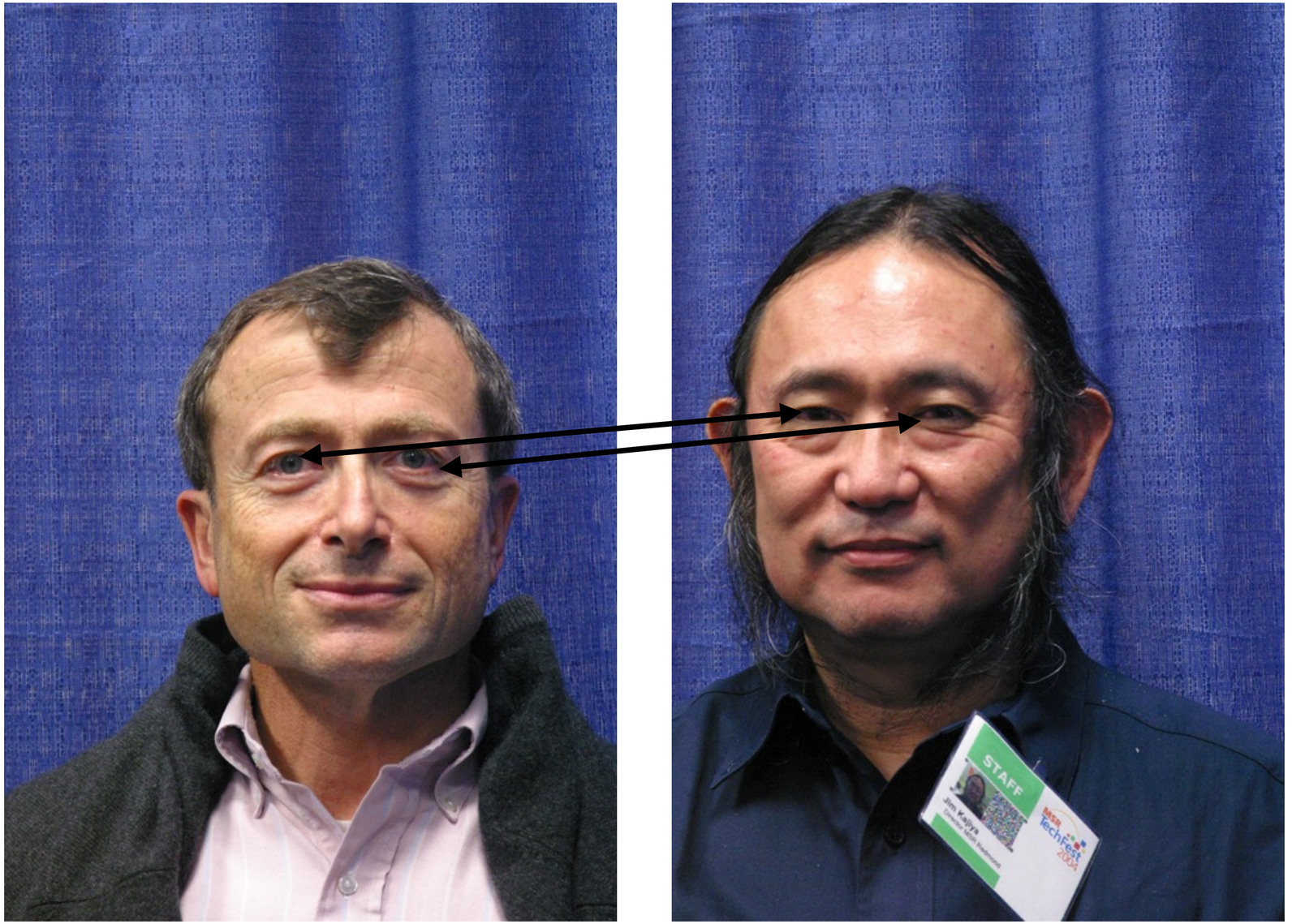
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

Face Alignment

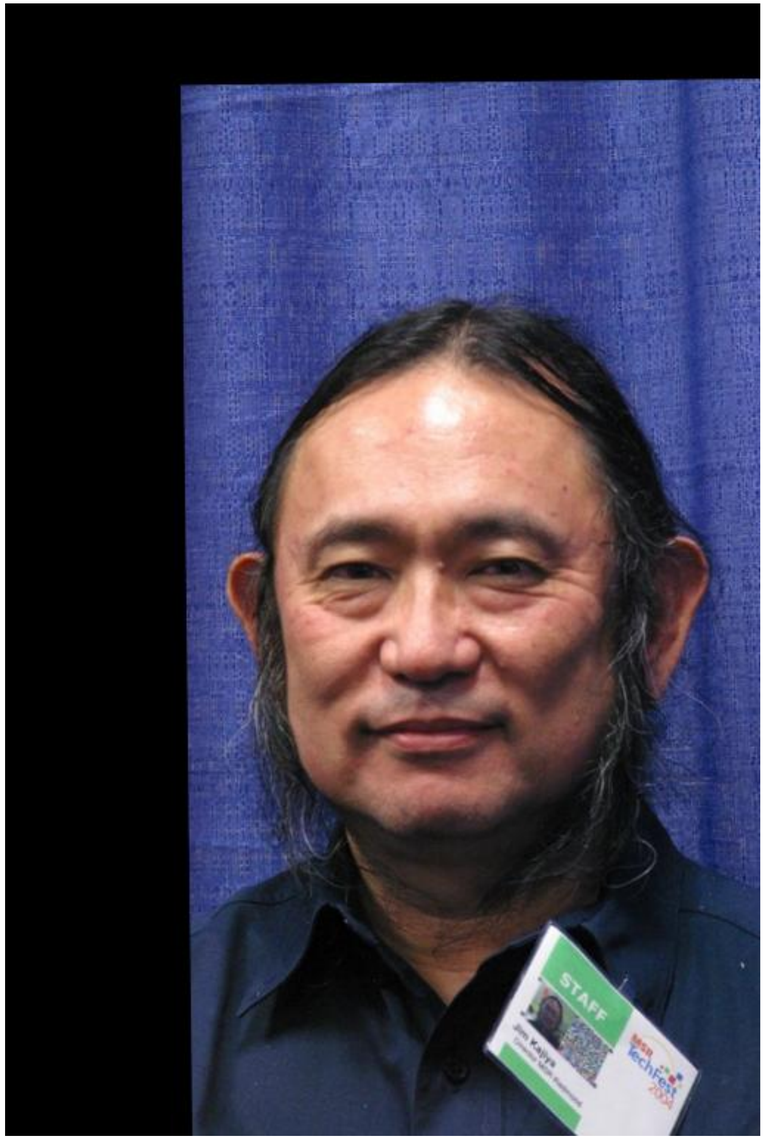


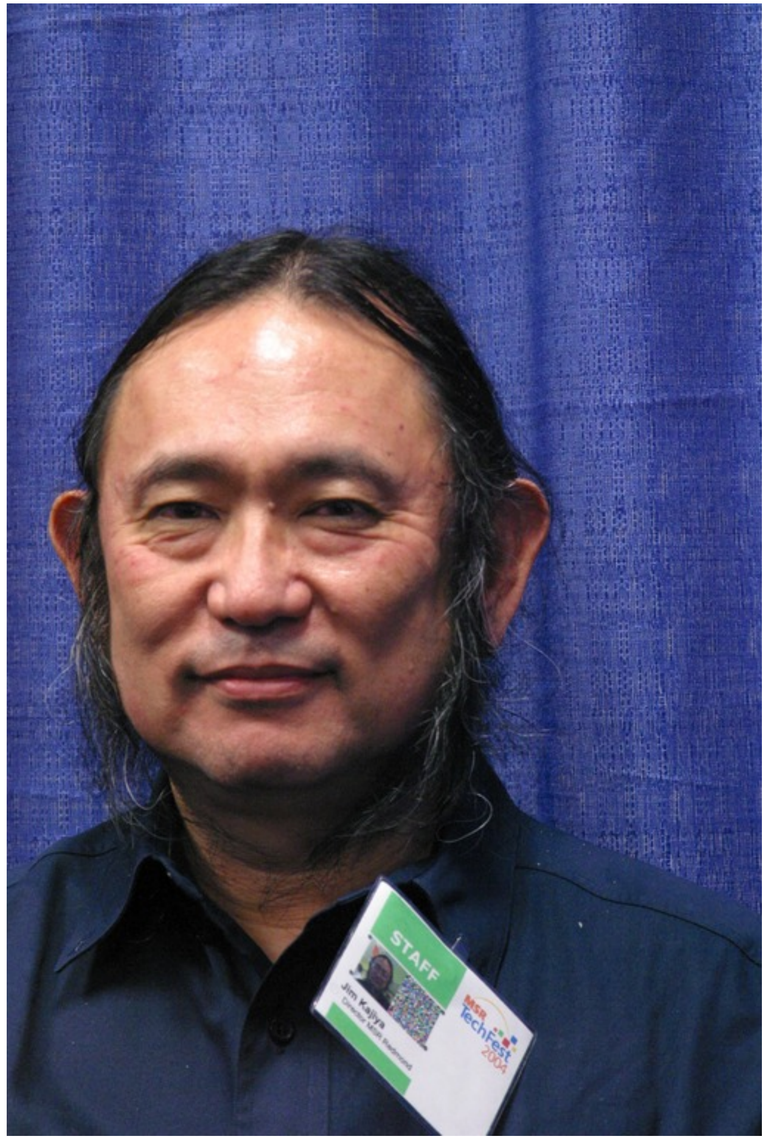


Face Alignment

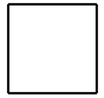
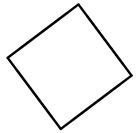
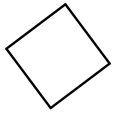
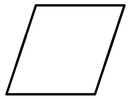
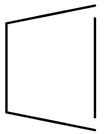


Face Alignment



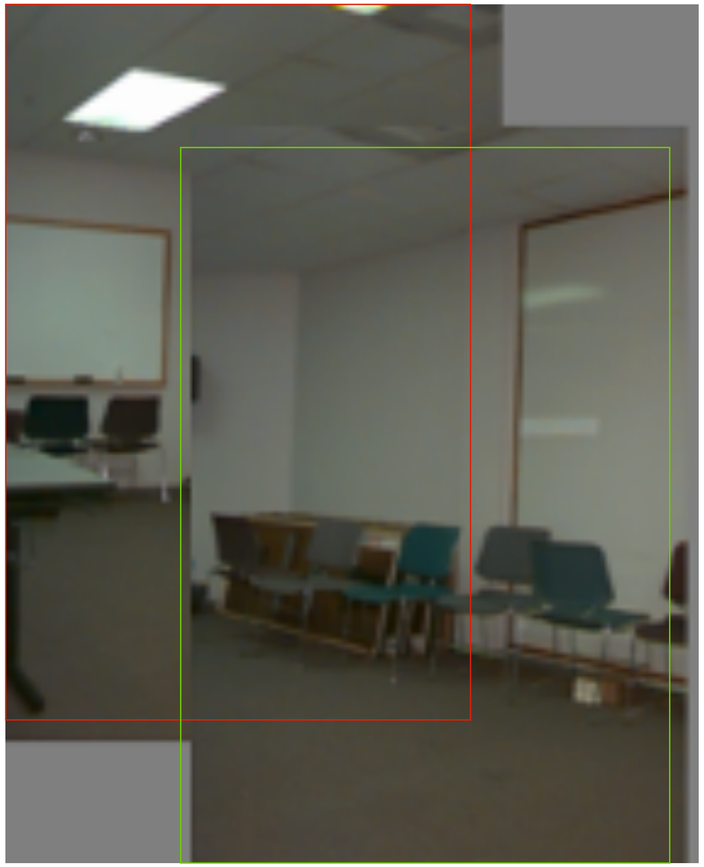


2D Transformations

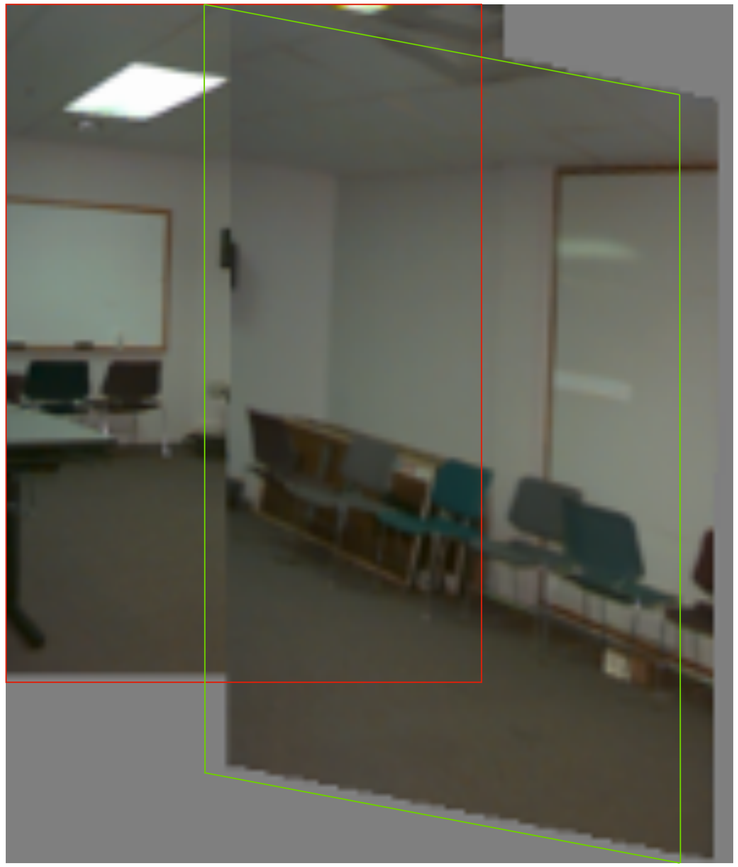
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R}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

Example: Warping with Different Transformations

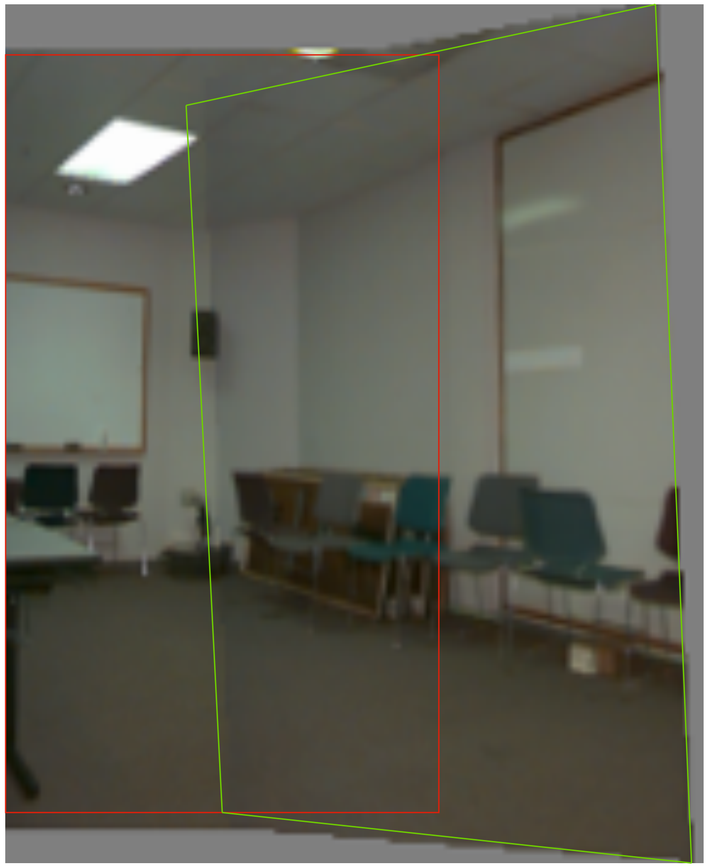
Translation



Affine



Projective (homography)



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Projective Transformation

- General 3x3 matrix transformation (note need scale factor)

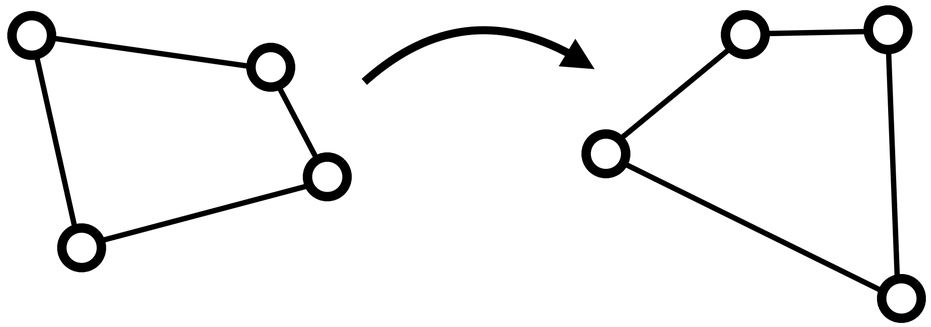
$$s \begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

13.5

Compute H from Correspondences

- Each match gives 2 equations to solve for 8 parameters (arbitrary scale)

$$s \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$



- → 4 correspondences to solve for H matrix
- Solution uses Singular Value Decomposition (SVD)
- In Assignment 4 you can compute this using `cv2.findHomography`

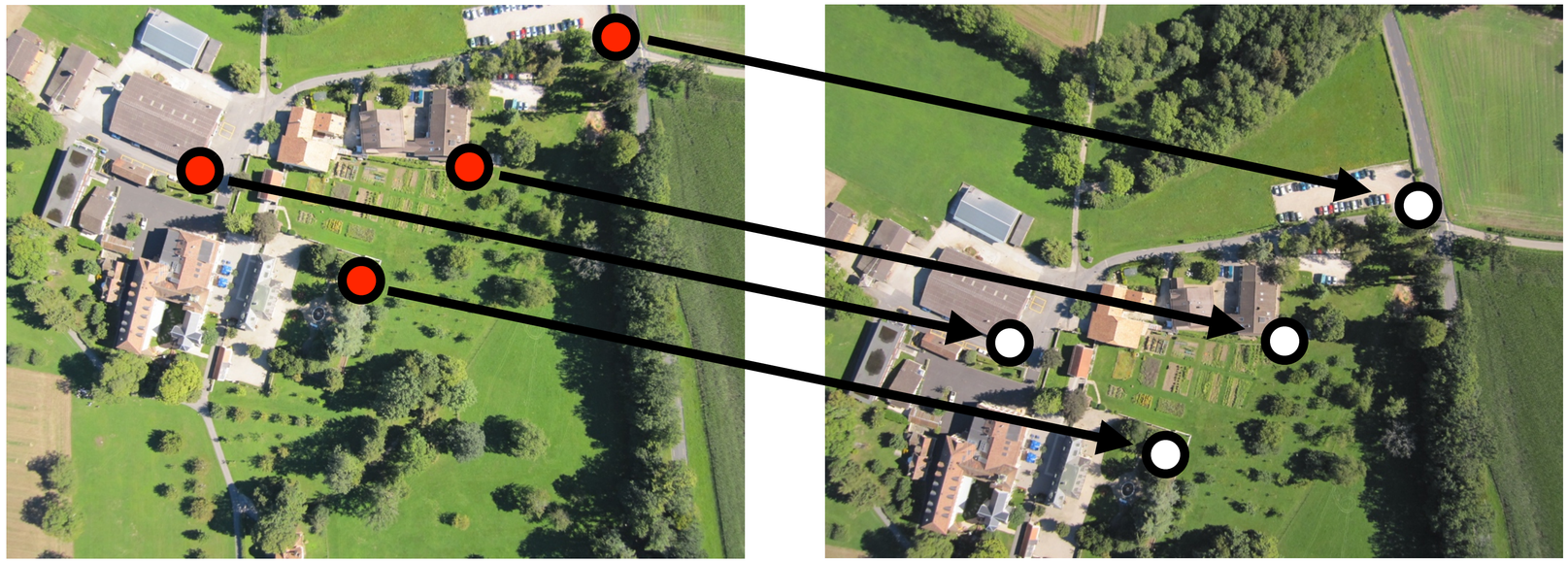
2-view Alignment + RANSAC

- 2-view alignment: linear equations
- Least squares and outliers
- Robust estimation via sampling

[Szeliski 8.1]

Image Alignment

- Find corresponding (matching) points between the images

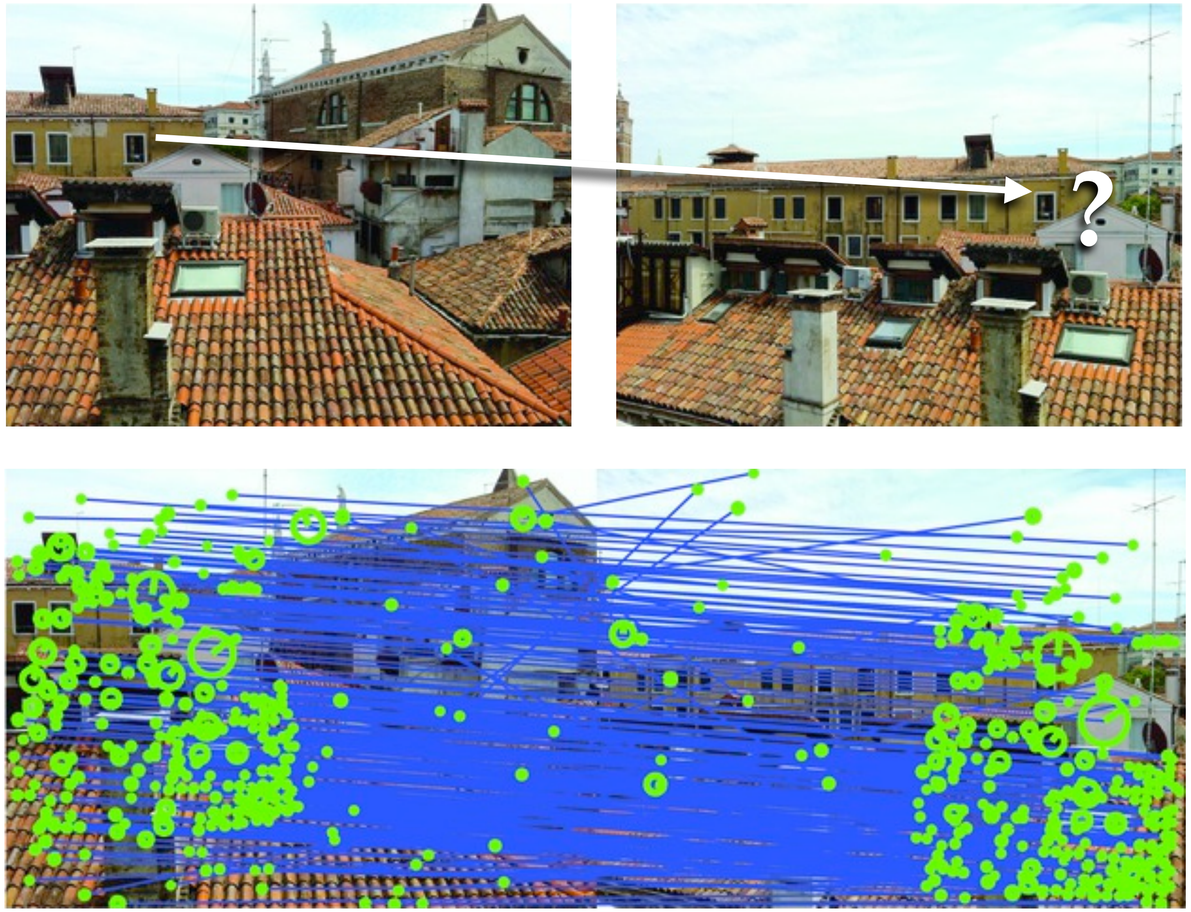


$$\mathbf{u} = \mathbf{H}\mathbf{x}$$

2 points for Similarity
3 for Affine
4 for Homography

Image Alignment

- In practice we have many noisy correspondences + **outliers**



Linear Equations

- e.g., for an affine transform we have a linear system in the unknown parameters **a**:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ \vdots \end{bmatrix}$$

- It is **overconstrained** (more equations than unknowns)
- and subject to **outliers** (some rows are completely wrong)

Let's deal with these problems in a simpler context..

Robust Line Fitting

- Consider fitting a line to noisy points

RANSAC Example

- RANSAC solution for Similarity Transform (2 points)

RANSAC Example

- RANSAC solution for Similarity Transform (2 points)

RANSAC Example

- RANSAC solution for Similarity Transform (2 points)

4 inliers (red, yellow, orange, brown),

RANSAC Example

- RANSAC solution for Similarity Transform (2 points)

4 outliers (blue, light blue, purple, pink)

RANSAC Example

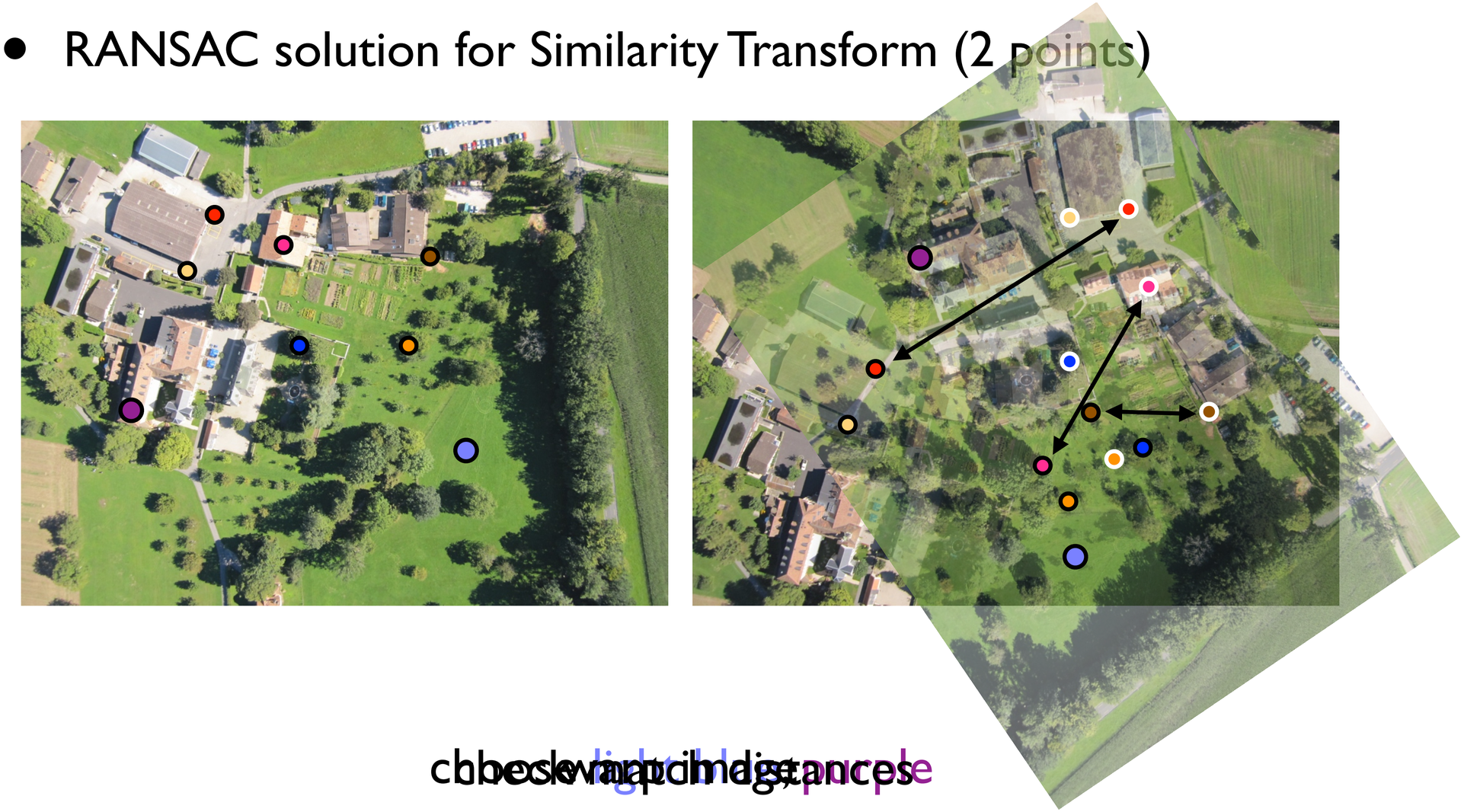
- RANSAC solution for Similarity Transform (2 points)

4 inliers (red, yellow, orange, brown),

4 outliers (blue, light blue, purple, pink)

RANSAC Example

- RANSAC solution for Similarity Transform (2 points)



choose random points

check match distances

light blue, purple

#inliers = 2

RANSAC Example

- RANSAC solution for Similarity Transform (2 points)

RANSAC Example

- RANSAC solution for Similarity Transform (2 points)

check overlap in pink blue

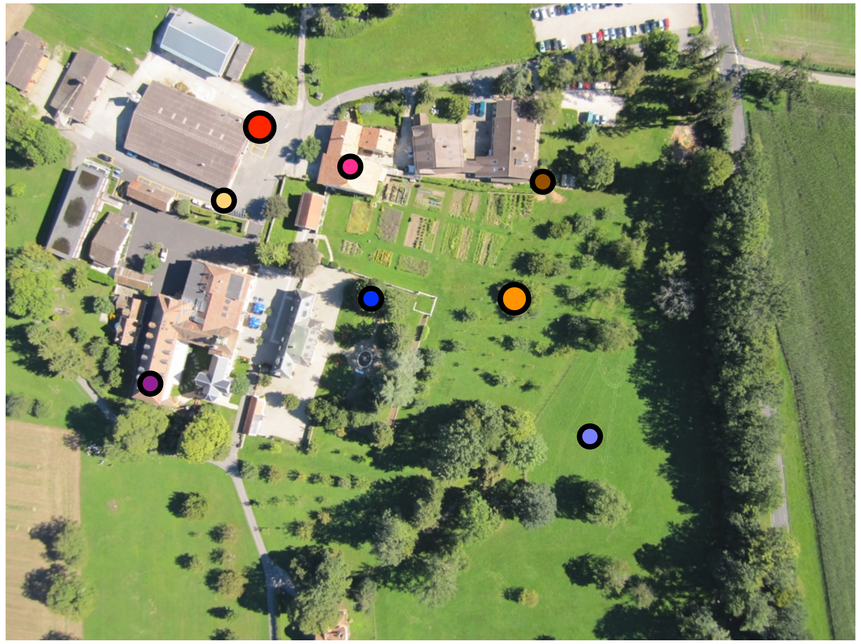
#inliers = 2

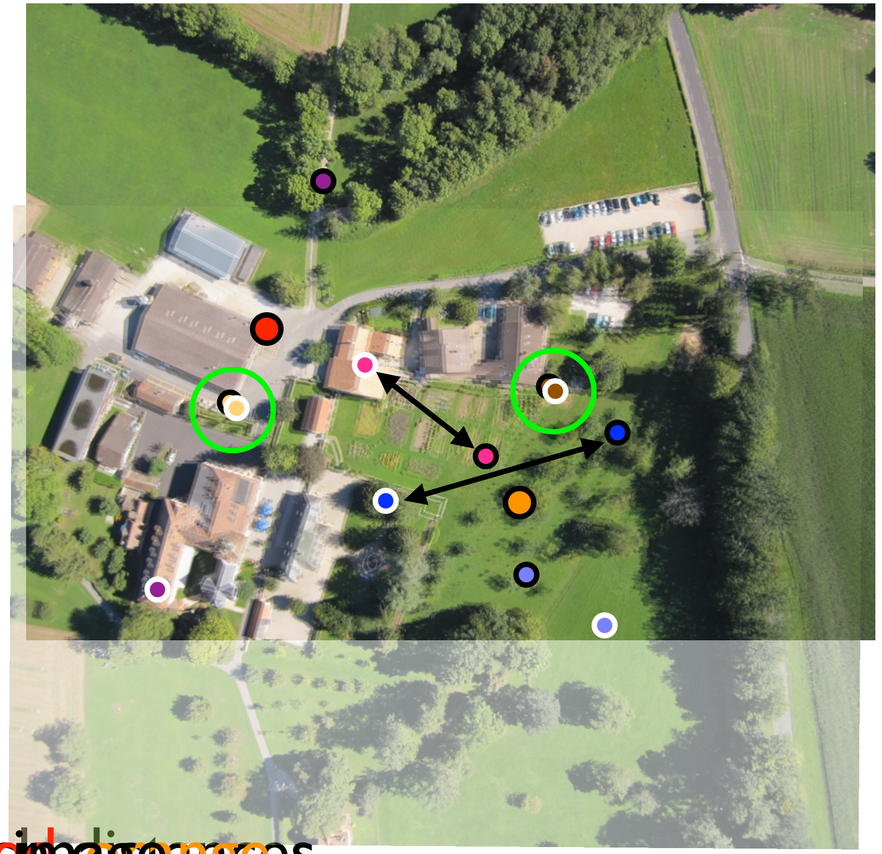
RANSAC Example

- RANSAC solution for Similarity Transform (2 points)

RANSAC Example

- RANSAC solution for Similarity Transform (2 points)





check overlap, edges

#inliers = 4

RANSAC Example

- RANSAC solution for Similarity Transform (2 points)

RANSAC algorithm

1. Match feature points between 2 views
2. Select minimal subset of matches*
3. Compute transformation T using minimal subset
4. Check consistency of all points with T — compute projected position and count #inliers with distance < threshold
5. Repeat steps 2-4 to maximise #inliers

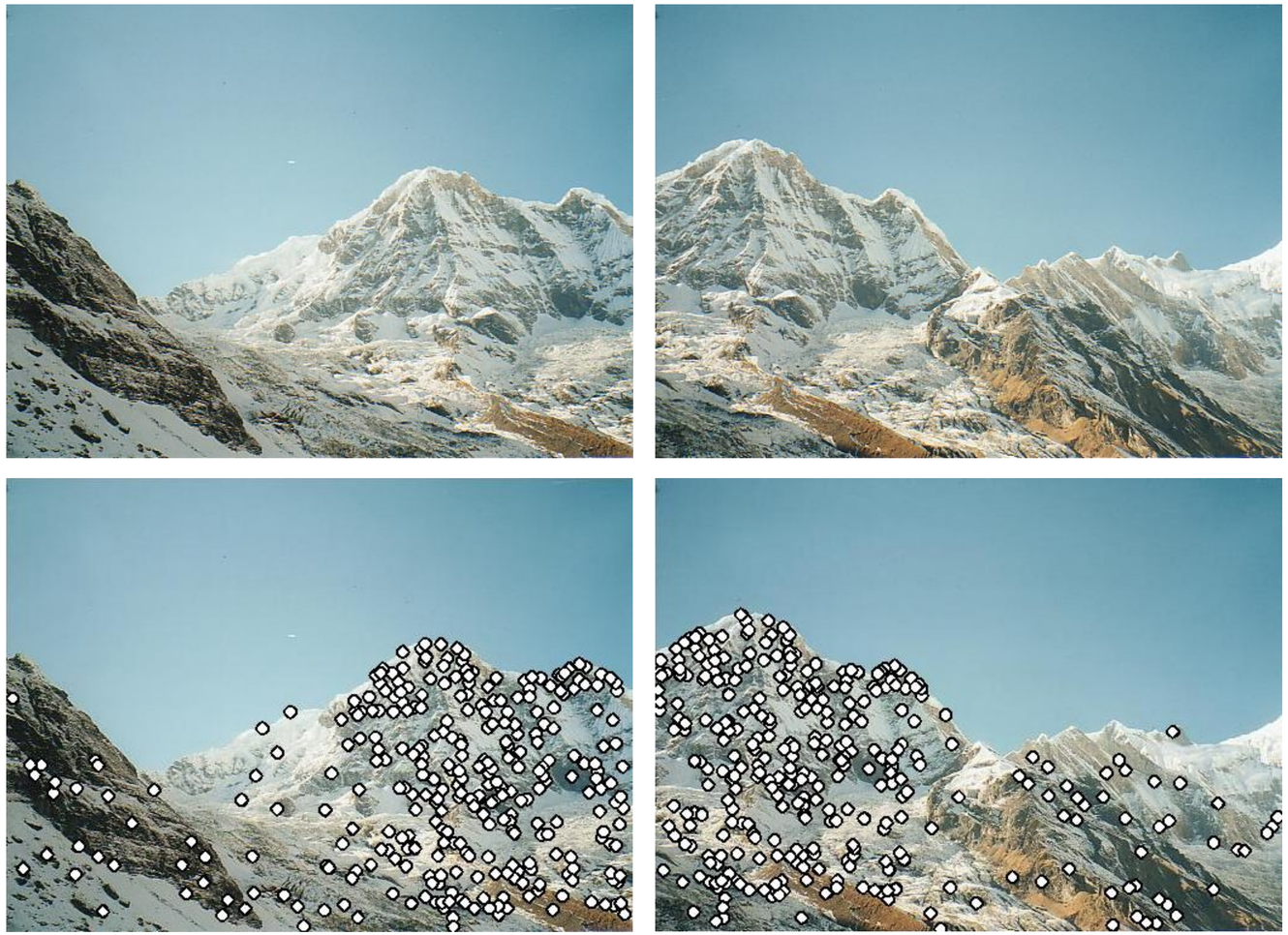
* Similarity transform = 2 points, Affine = 3, Homography = 4

Assignment 4

- Try out the **RANSAC Implementation** section in Assignment 4

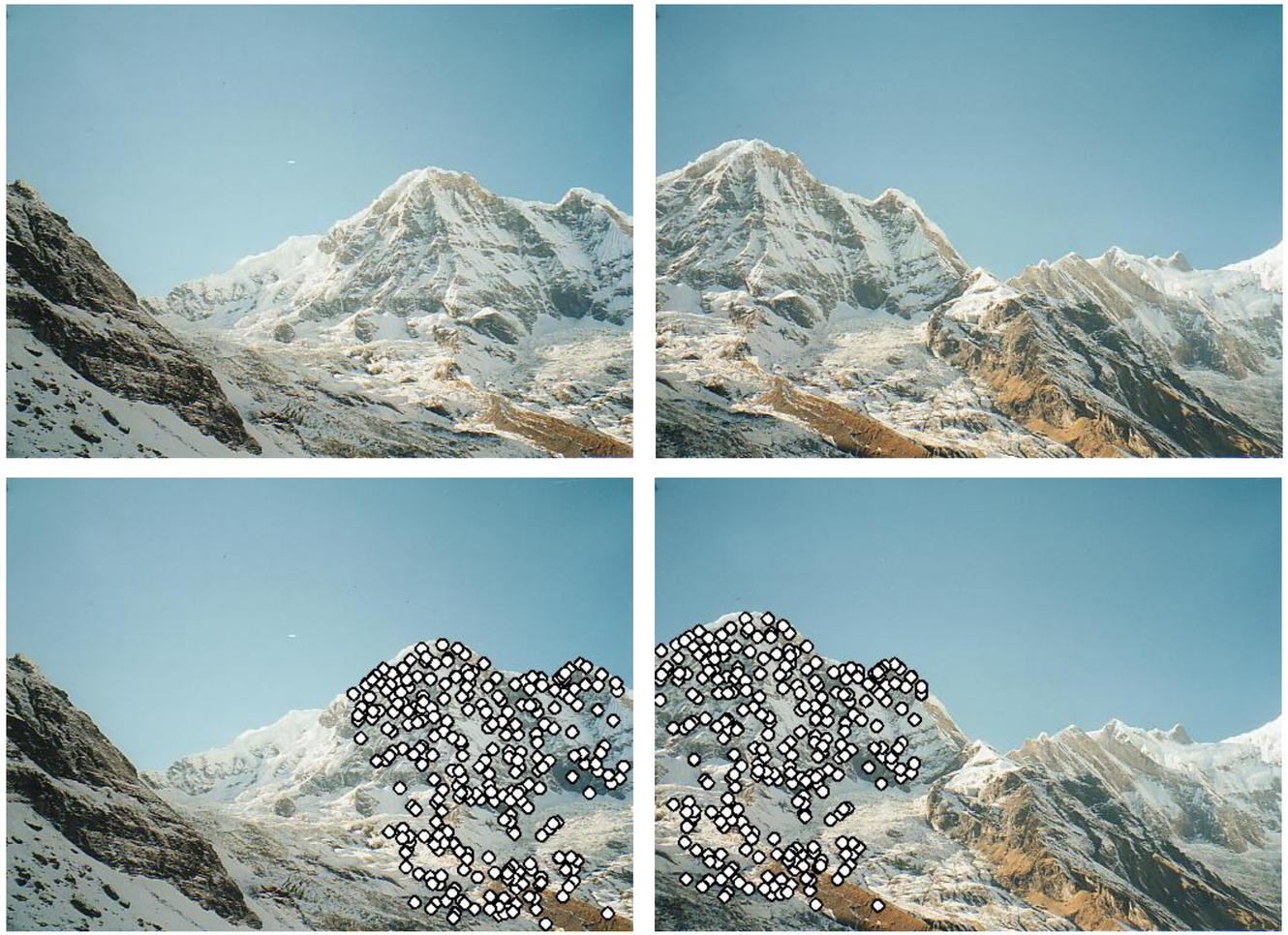
2-view Rotation Estimation

- Find features + raw matches, use RANSAC to find Similarity



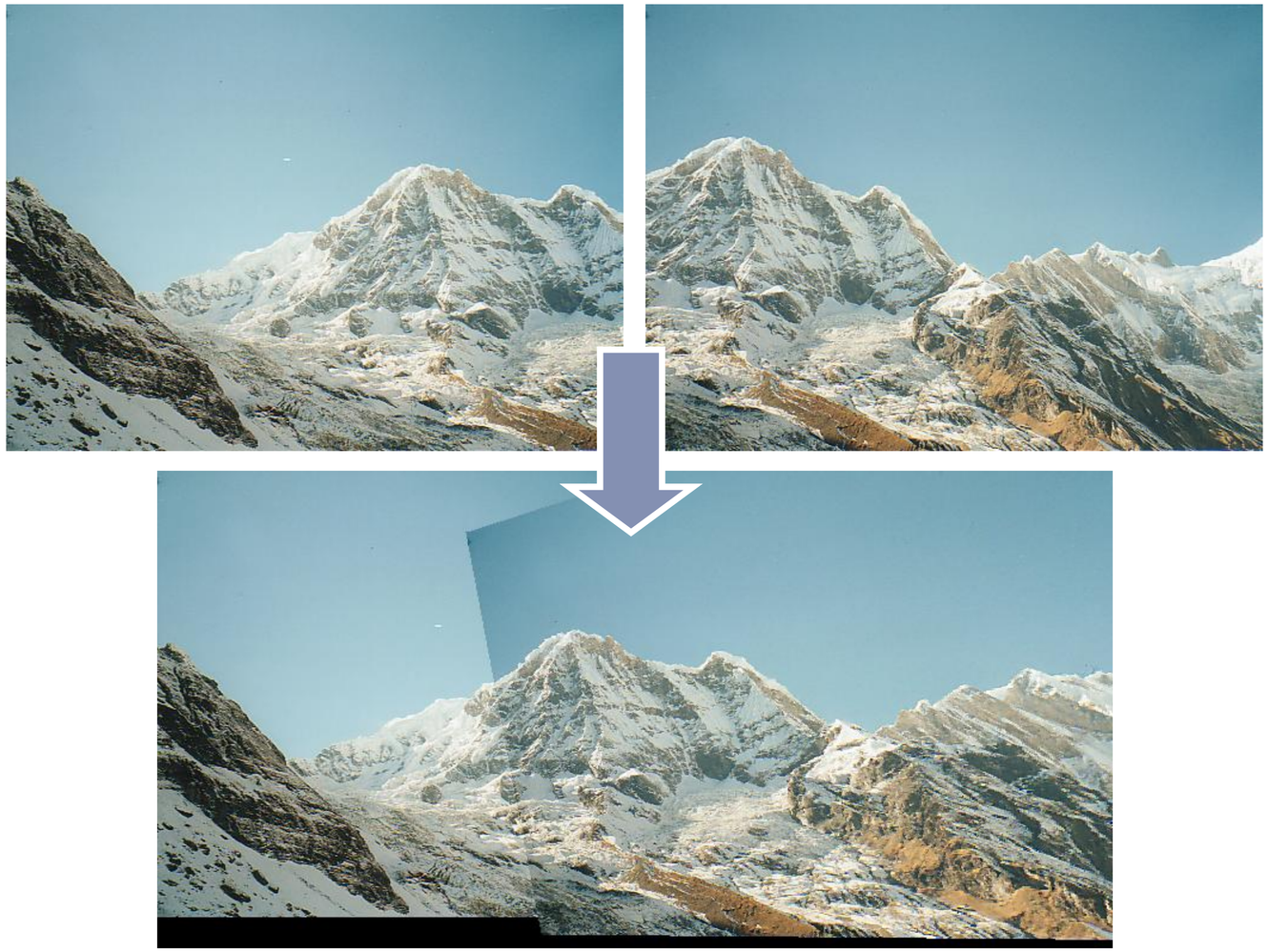
2-view Rotation Estimation

- Remove outliers, can now solve for R using least squares



2-view Rotation Estimation

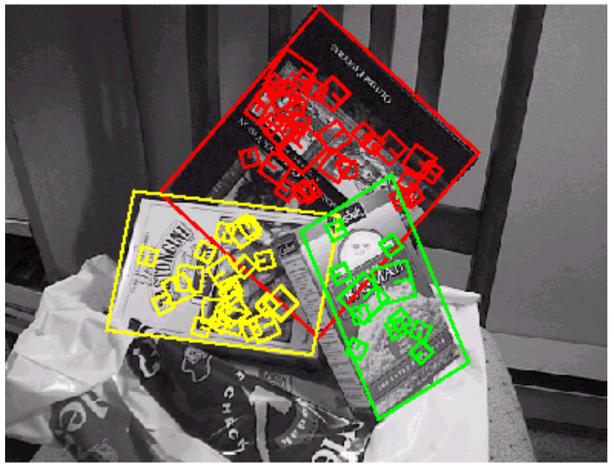
- Final rotation estimation



Object **Instance Recognition**

Database of planar objects

Instance recognition



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Object **Instance Recognition** with SIFT

Match SIFT descriptors between **query image** and a database of known keypoints extracted from **training examples**

— use fast (approximate) nearest neighbour matching

— threshold based on ratio of distances between 1NN and 2NN

Use **RANSAC** to find a **subset of matches** that all agree on an object and geometric transform (e.g., **affine transform**)

Optionally **refine pose estimate** by recomputing the transformation using all the RANSAC inliers

Fitting a Model to Noisy Data

Suppose we are **fitting a line** to a dataset that consists of 50% outliers

We can fit a line using two points

If we draw pairs of points uniformly at random, what fraction of pairs will consist entirely of ‘good’ data points (inliers)?

RANSAC: How many samples?

Let p_0 be the fraction of outliers (i.e., points on line)

Let n be the number of points needed to define hypothesis

($n = 2$ for a line in the plane)

Suppose k samples are chosen

How many samples do we need to find a good solution?


13.7

RANSAC: *k* Samples Chosen (p = 0.99)

Sample size	Proportion of outliers						
n	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Figure Credit: Hartley & Zisserman

Lecture 14: Re-cap RANSAC

RANSAC is a technique to fit data to a model

— divide data into inliers and outliers

— estimate model from minimal set of inliers

— improve model estimate using all inliers

— alternate fitting with re-classification as inlier/outlier

RANSAC is a general method suited for a wide range of model fitting problems

— easy to implement

— easy to estimate/control failure rate

RANSAC only handles a moderate percentage of outliers without cost blowing up

Menu for Today

Topics:

- **Planar** Geometry
- **Image Alignment**, Object Recognition
- **RANSAC**

Readings:

- **Today's** Lecture: Szeliski 2.1, 8.1, Forsyth & Ponce 10.4.2

Reminders:

- **Assignment 4**: RANSAC and Panorama Stitching — **now available**