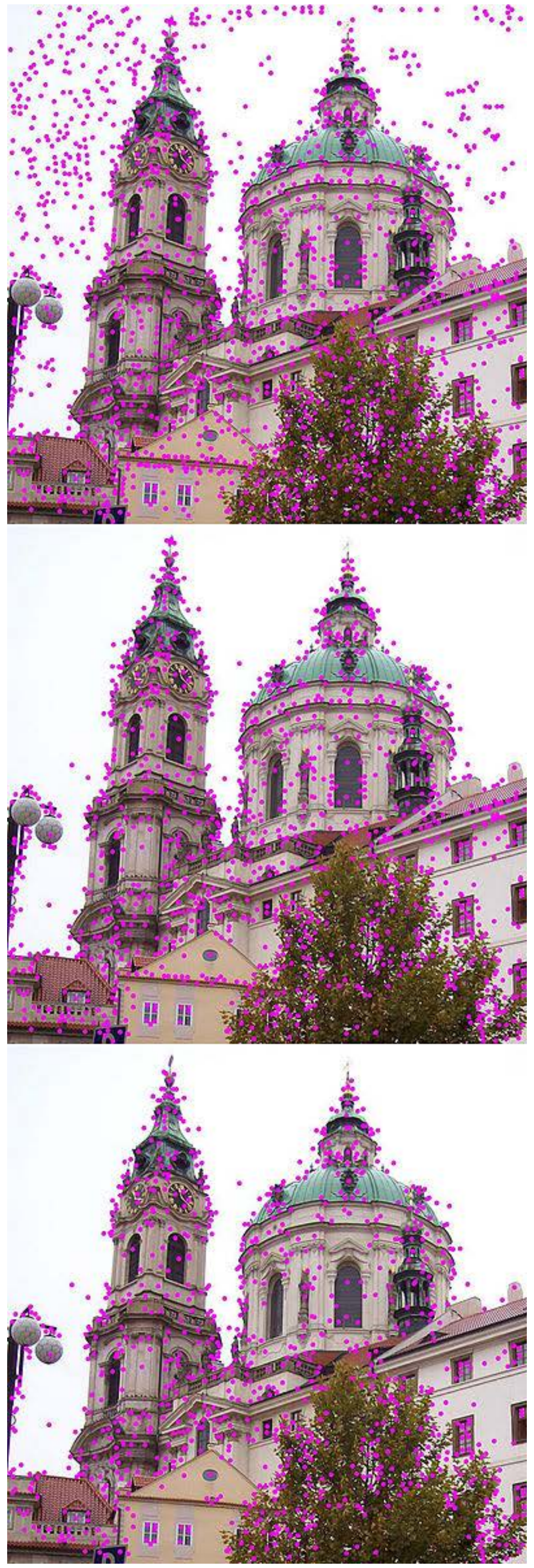
Scale Invariant Feature Transform (**SIFT**)



SIFT describes both a **detector** and **descriptor**

1. Multi-scale extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

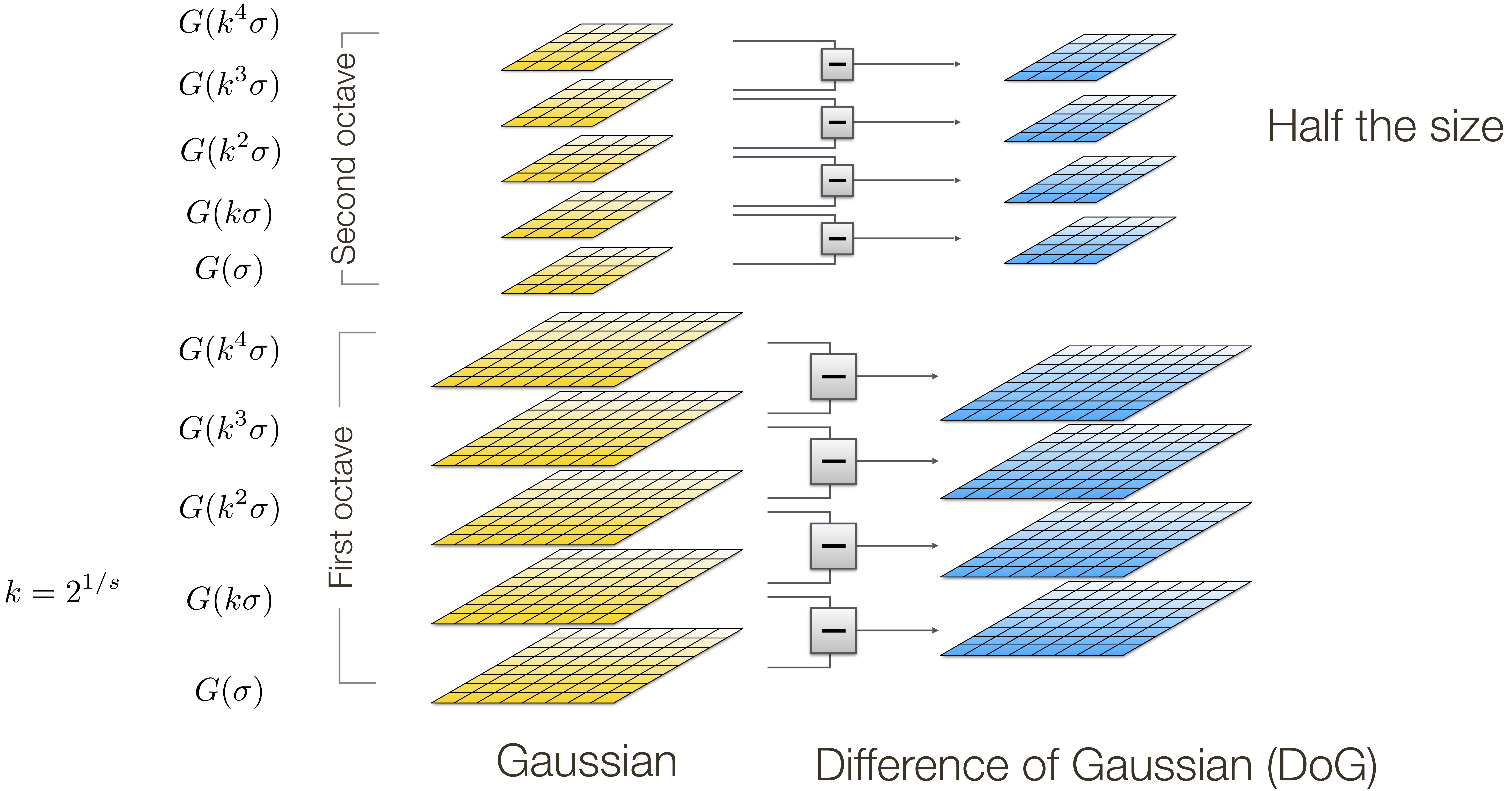
Where do points lead?

Image 1

Image 2

With COTR, we find dense correspondences, which we can reconstruct a dense 3D model from just two calibrated views.

1. Multi-scale Extrema Detection

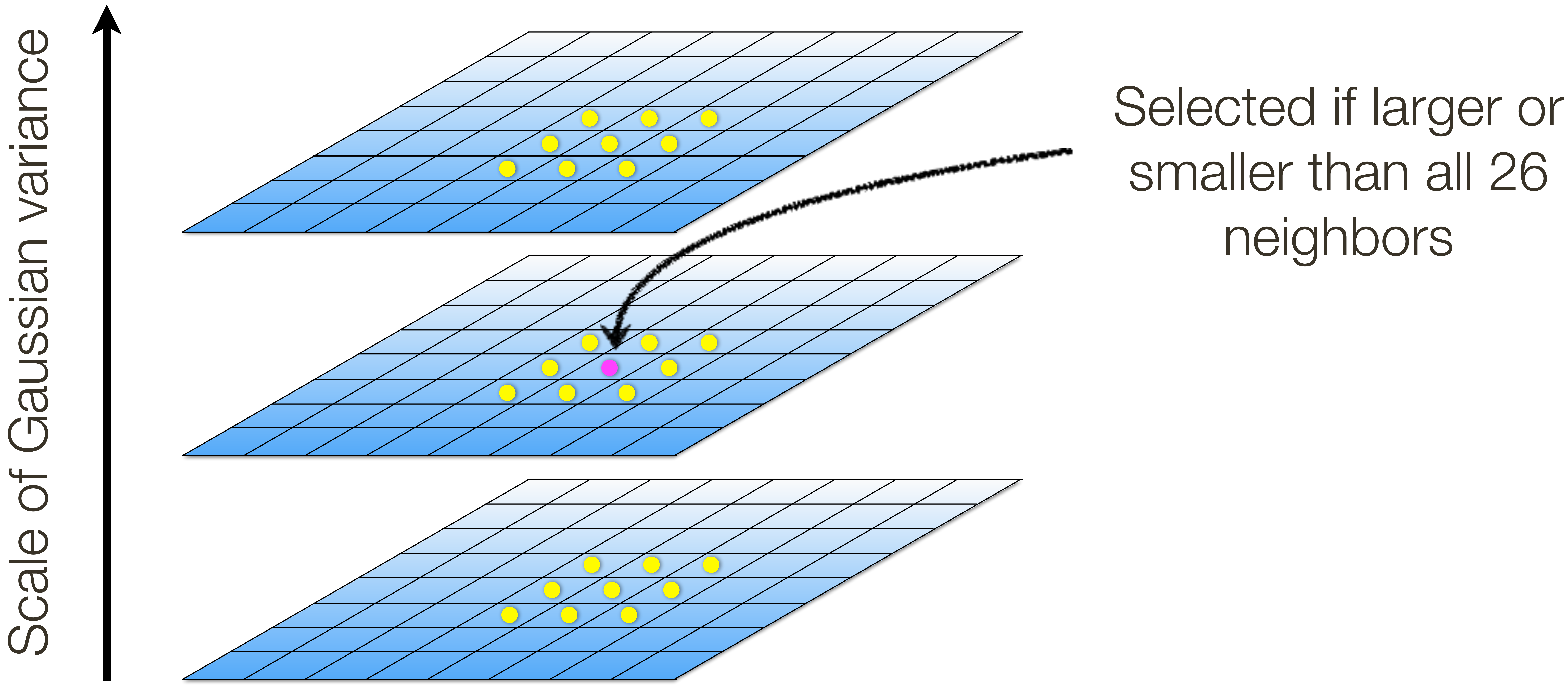


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection

Detect maxima and minima of Difference of Gaussian in scale space

$$\sigma = 2^{1/s}$$



Difference of Gaussian (DoG)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

2. Keypoint Localization

— After keypoints are detected, we remove those that have **low contrast** or are **poorly localized** along an edge

How do we decide whether a keypoint is poorly localized, say along an edge, vs. well-localized?

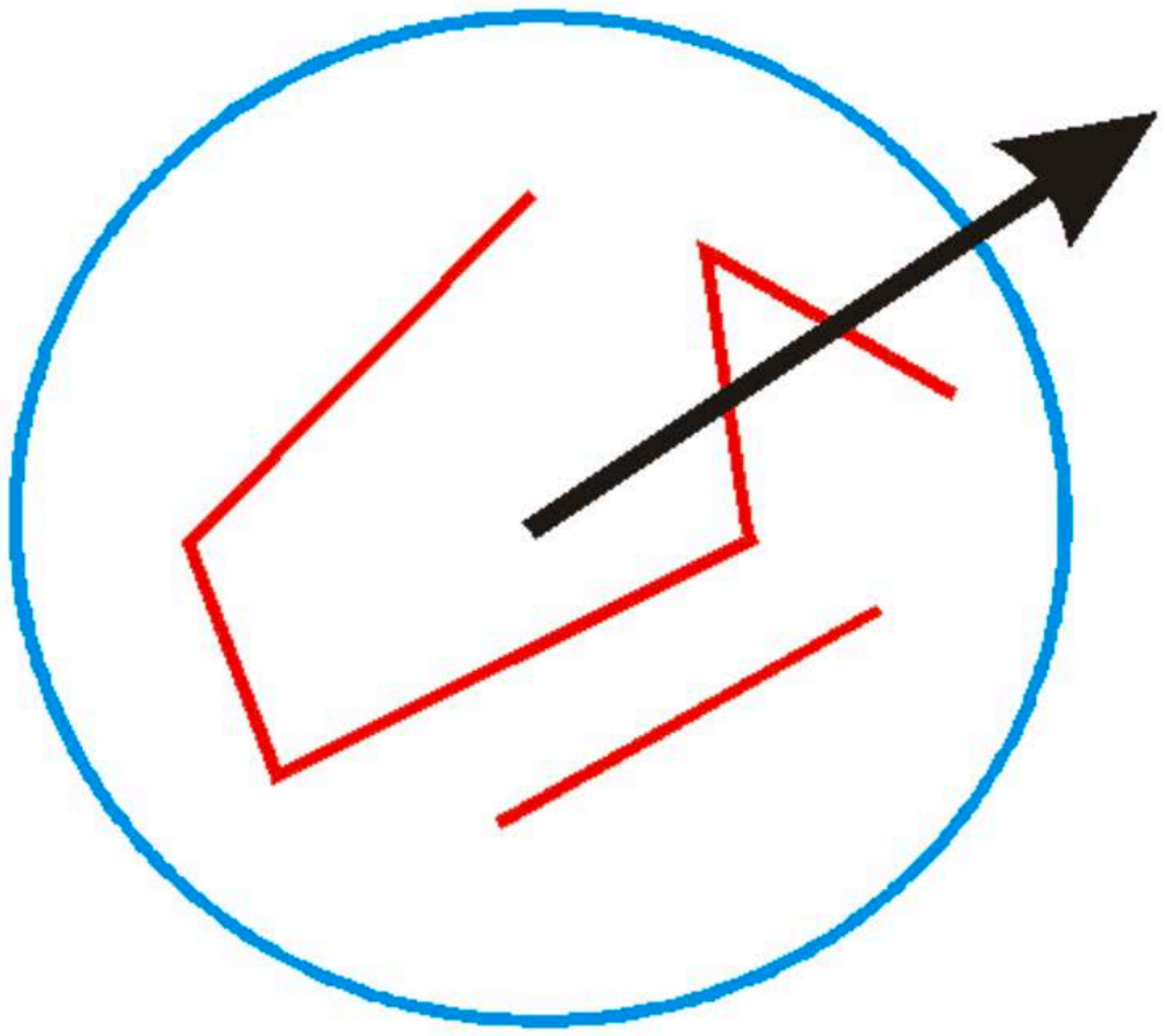
$$C = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

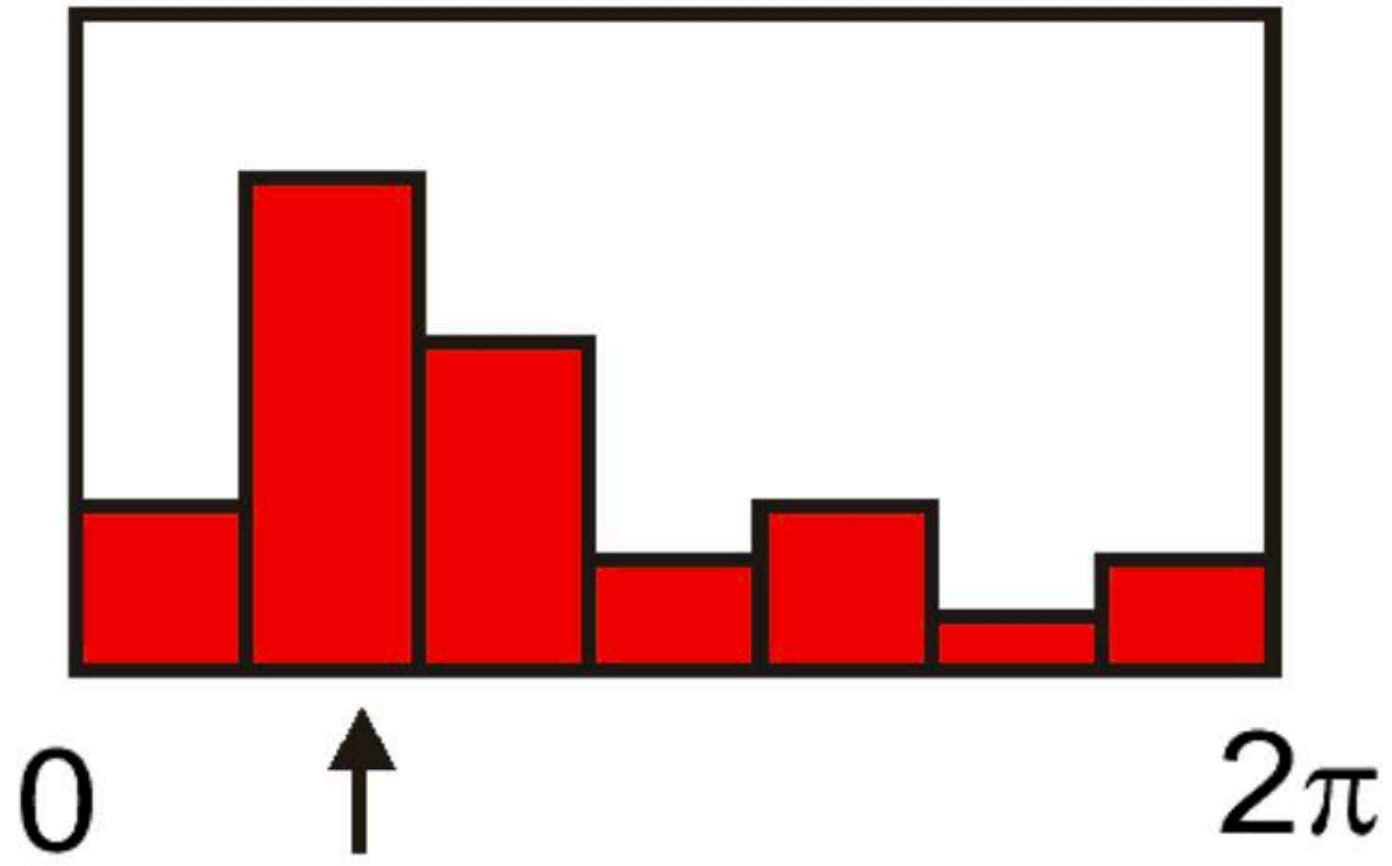
3. Orientation Assignment

— Create **histogram** of local gradient directions computed at selected scale

— Assign **canonical orientation** at peak of smoothed histogram

— Each key specifies stable 2D coordinates (x , y , scale, orientation)



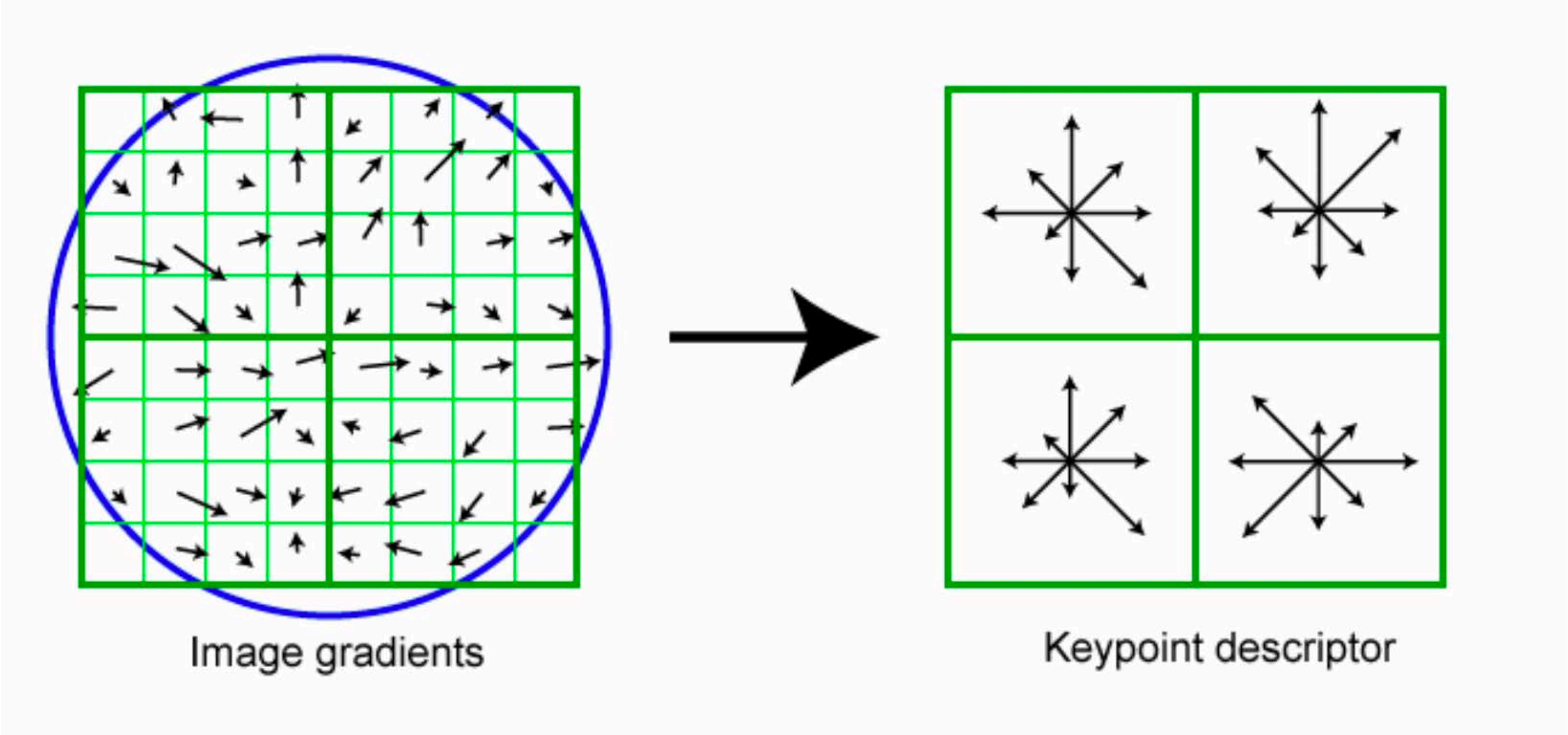


4. SIFT Descriptor

— Image gradients are sampled over 16 × 16 array of locations in scale space (weighted by a Gaussian with sigma half the size of the window)

— Create array of orientation histograms

— 8 orientations × 4 × 4 histogram array



SIFT Matching

— Each SIFT feature is represented by 128-D vector (numbers)

— Feature matching becomes the task of finding the closest 128-D vector

— Nearest-neighbor matching:

$$NN(j) = \arg\min_i |\mathbf{x}_i - \mathbf{x}_j|, \; i \neq j$$

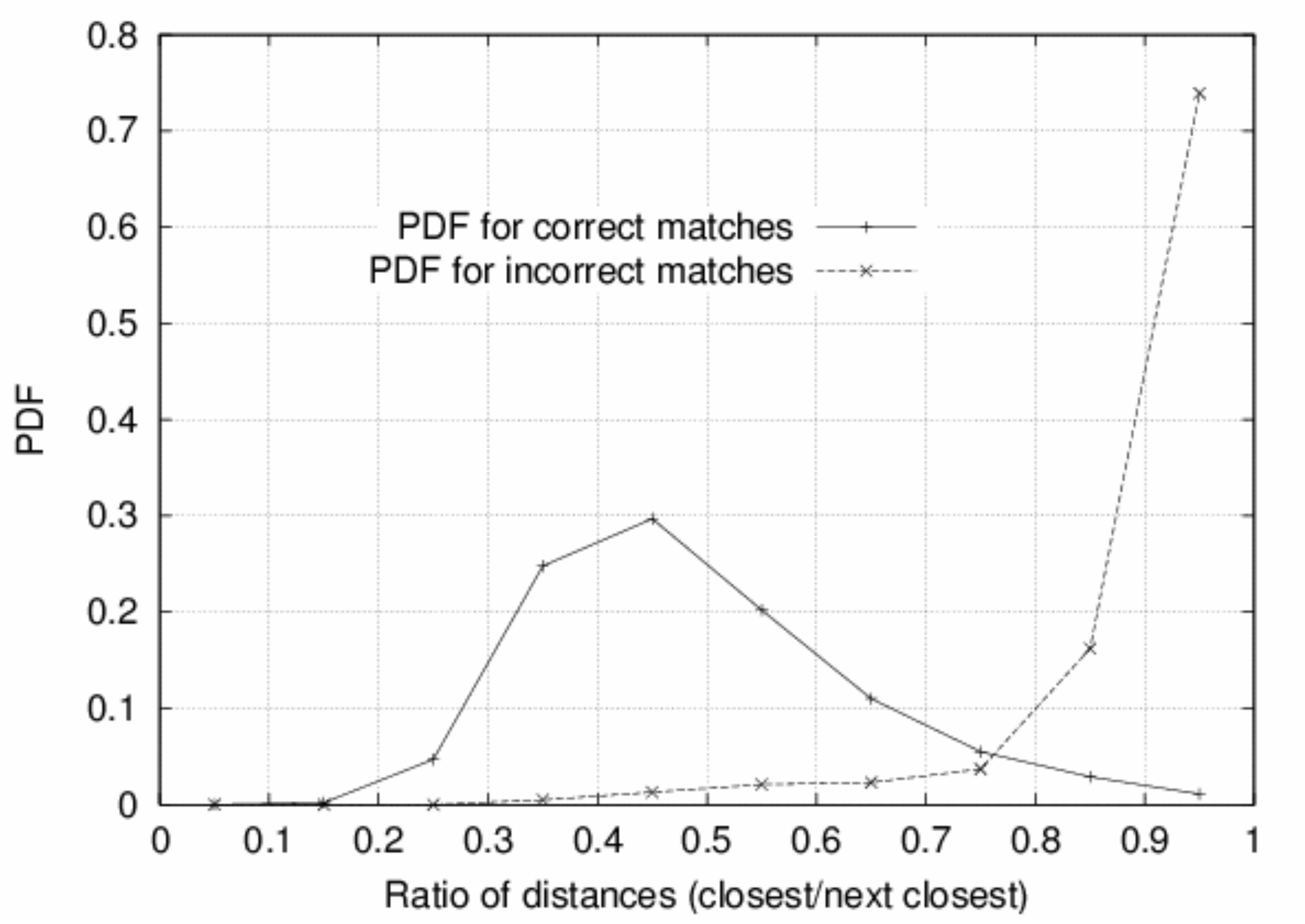
— This is expensive (linear time), but good approximation algorithms exist

e.g., Best Bin First K-d Tree [Beis Lowe 1997], FLANN (Fast Library for Approximate Nearest Neighbours) [Muja Lowe 2009]

Match **Ratio Test**

Compare ratio of distance of **nearest** neighbour (1NN) to **second** nearest (2NN) neighbour — this will be a non-matching point

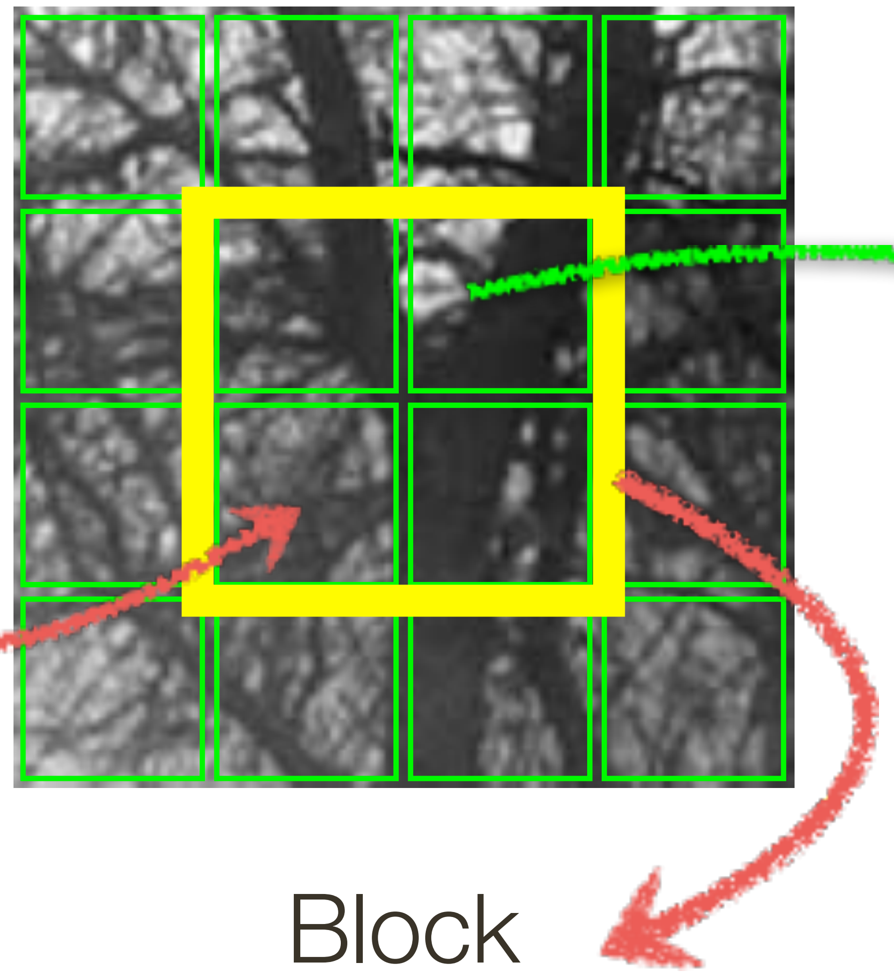
Rule of thumb: d(1NN) < 0.8 * d(2NN) for good match



Histogram of Oriented Gradients (**HOG**) Features

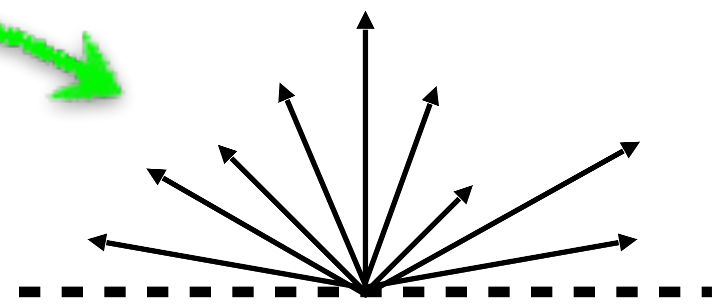
Dalal, Triggs. Histograms of Oriented Gradients for Human Detection. CVPR, 2005

Cell
(8x8 pixels)



Block
(2x2 cells)

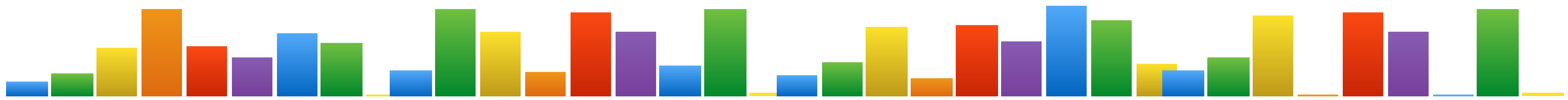
histogram of
'unsigned'
gradients



soft binning

gradient magnitude histogram
(one for each cell)

Concatenate and L-2 normalization



Single scale, no dominant orientation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Histogram of Oriented Gradients (**HOG**) Features

Pedestrian detection

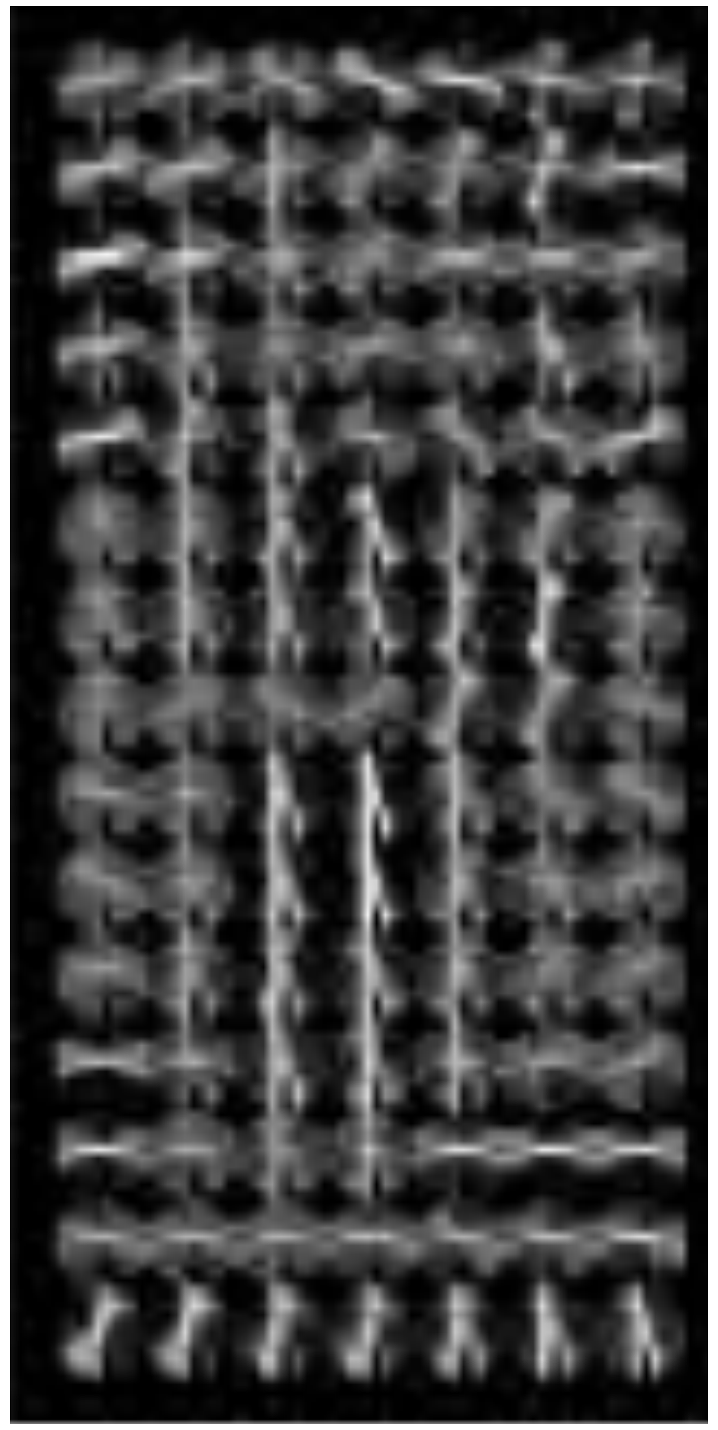
1 cell step size

128 pixels
16 cells
15 blocks

64 pixels
8 cells
7 blocks

15 x 7 x 4 x 9 =
3780

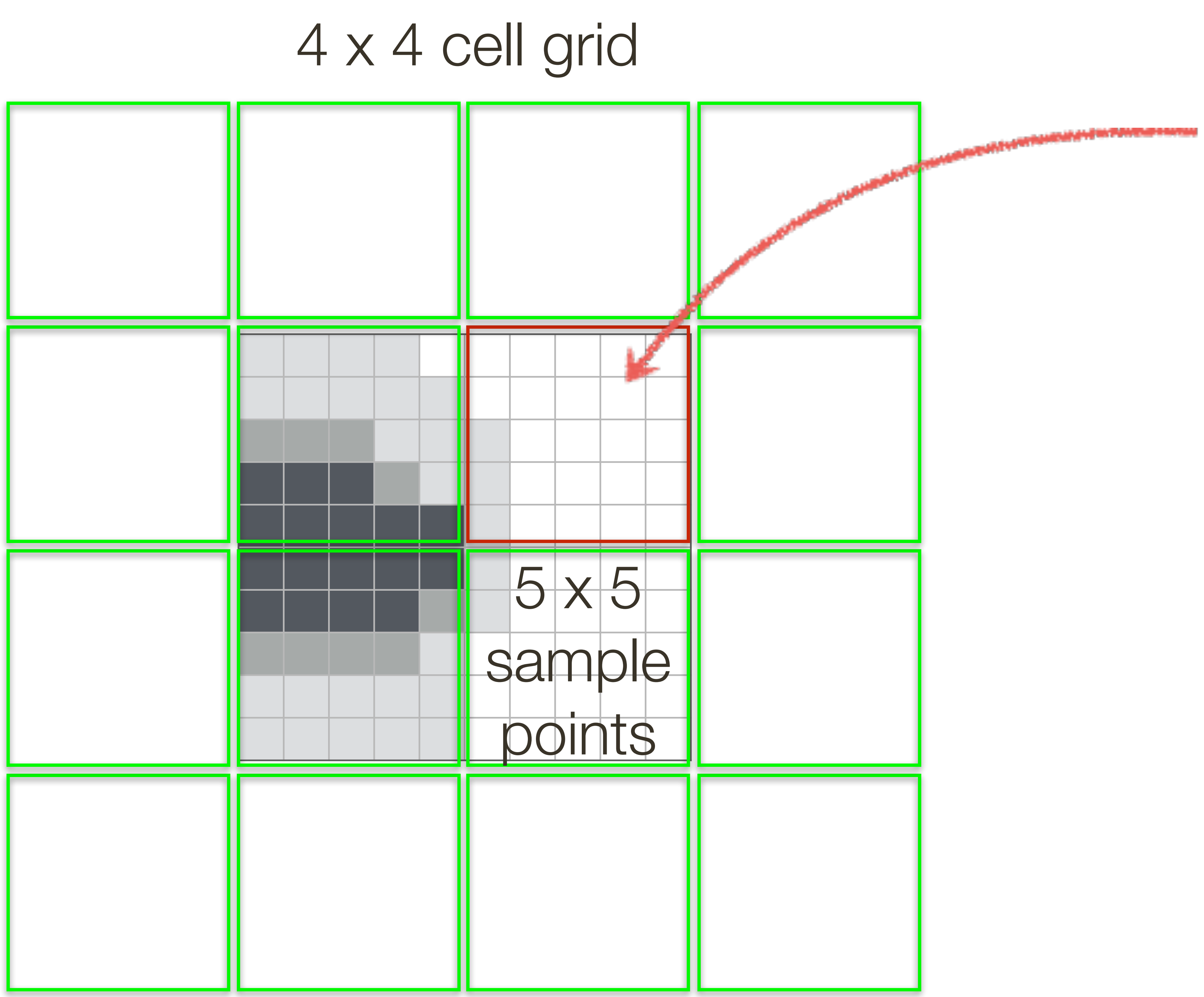
visualization



Redundant representation due to overlapping blocks

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

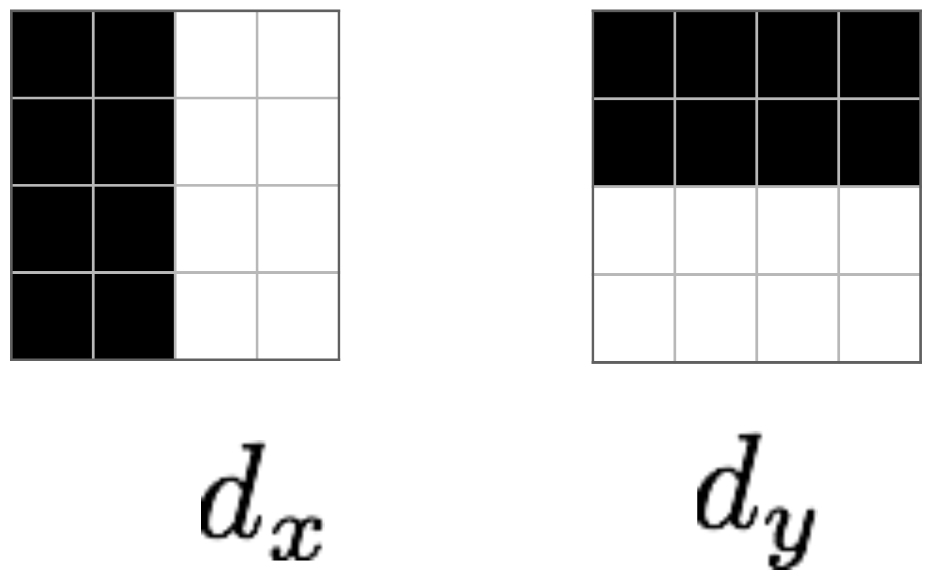
'Speeded' Up Robust Features (**SURF**)



Each cell is represented by 4 values:

$$\left[\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right]$$

Haar wavelets filters

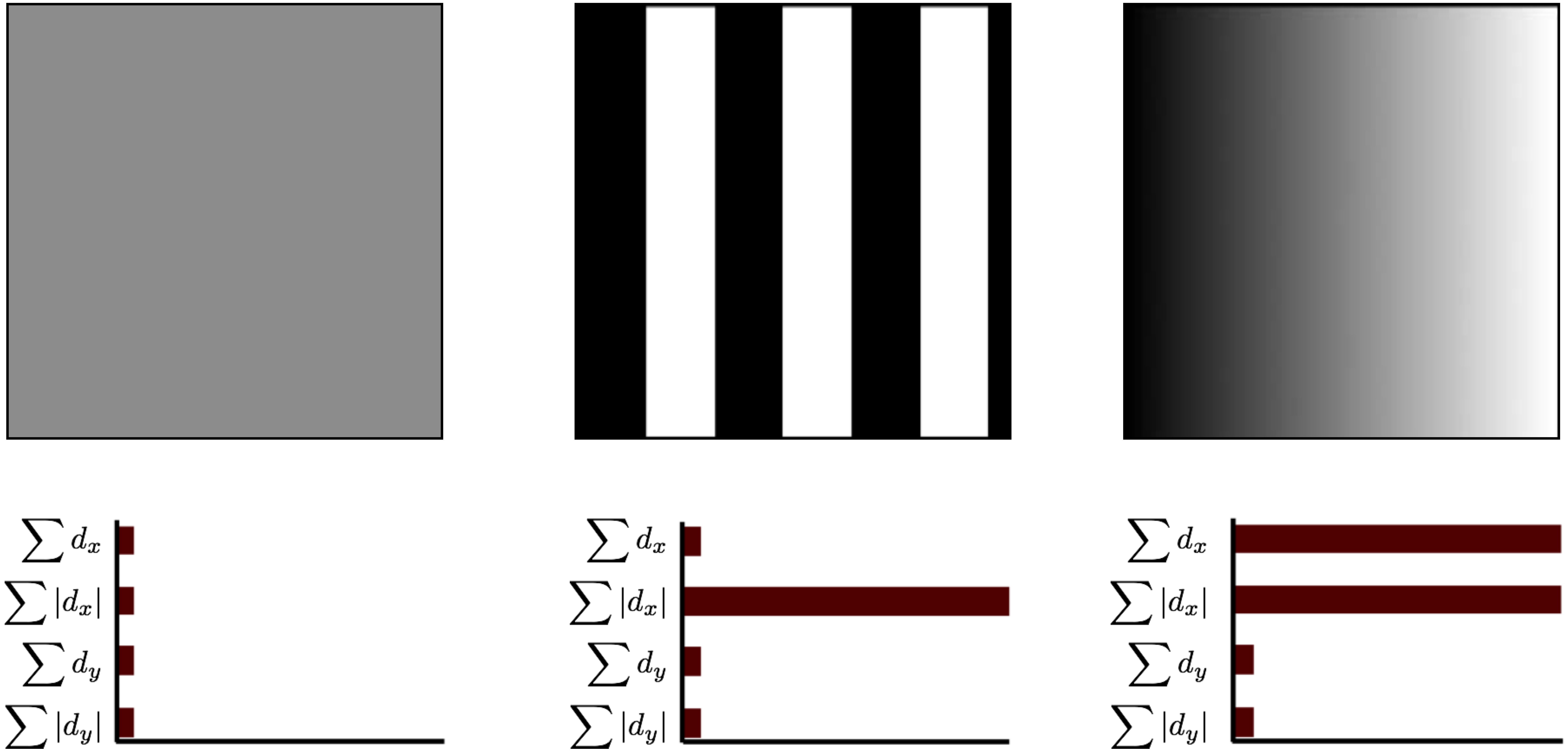


How big is the SURF descriptor?

64 dimensions

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

‘Speeded’ Up Robust Features (**SURF**)



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Keypoint **Detectors** vs. **Descriptors**

- Harris
- Blob (Laplacian)
- SIFT

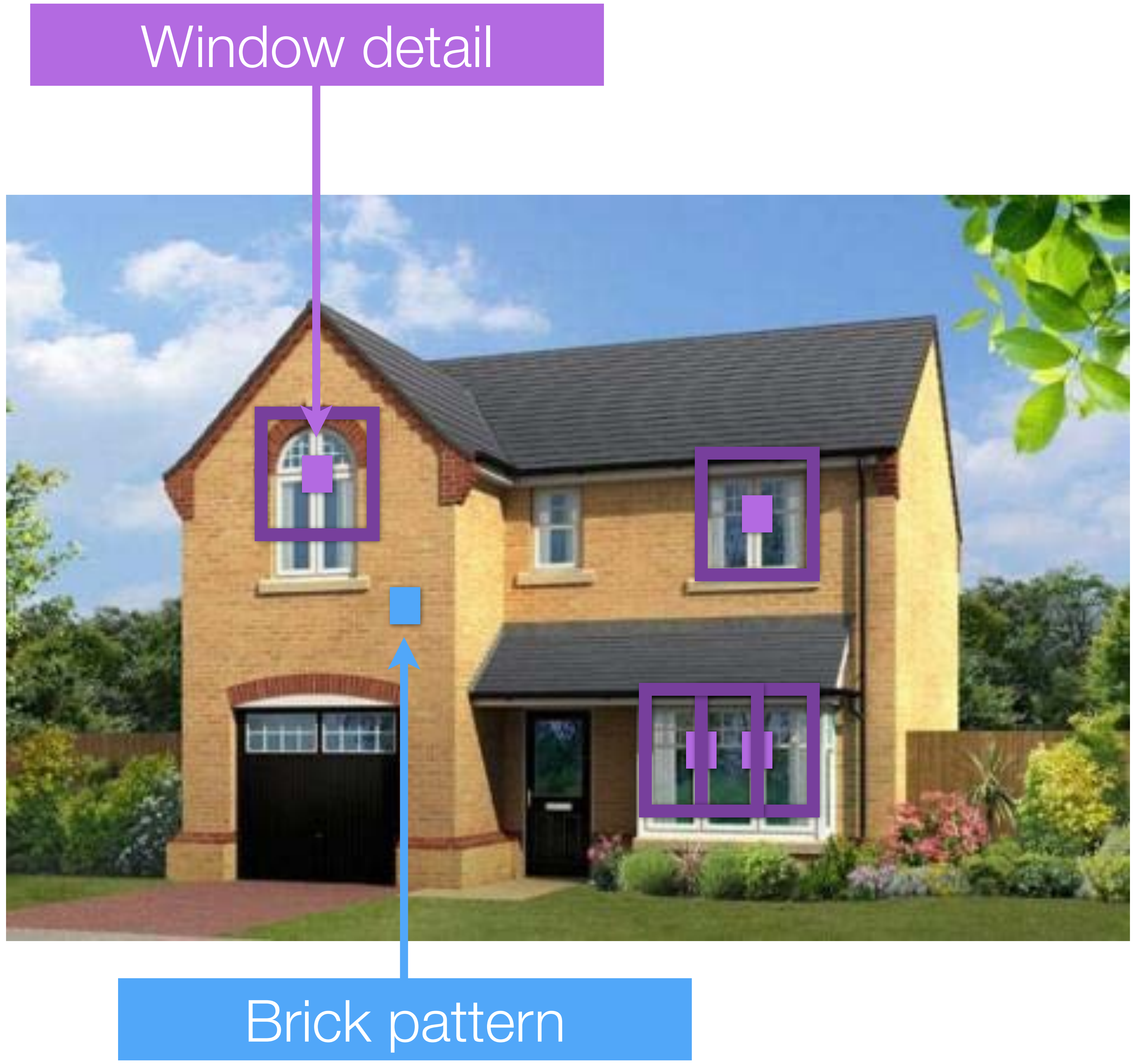
- SIFT
- HoG
- SURF

Failure Case: **Repetitive** Structures

Repetitive structures cause problems for feature matching

Multiple locations in an image provide good matches and have similar matching scores

They are particularly common in man-made environments



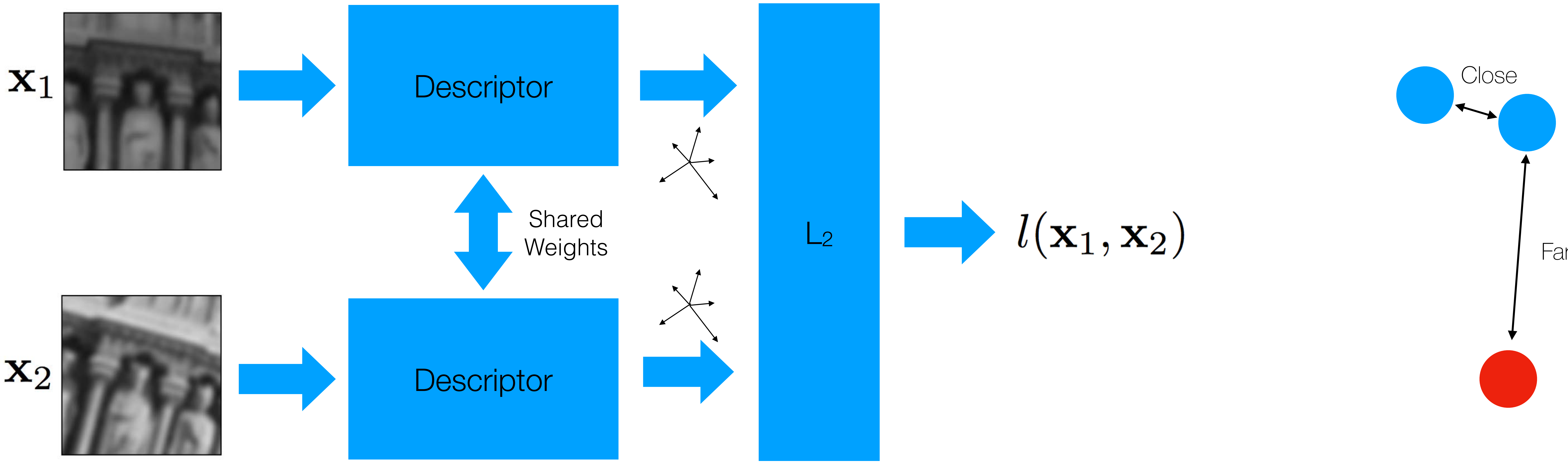
Learning Descriptors

Descriptor design as a learning (embedding) problem

[Winder Brown 2007]

DeepDesc [ICCV 2015]

Learning an "embedding"

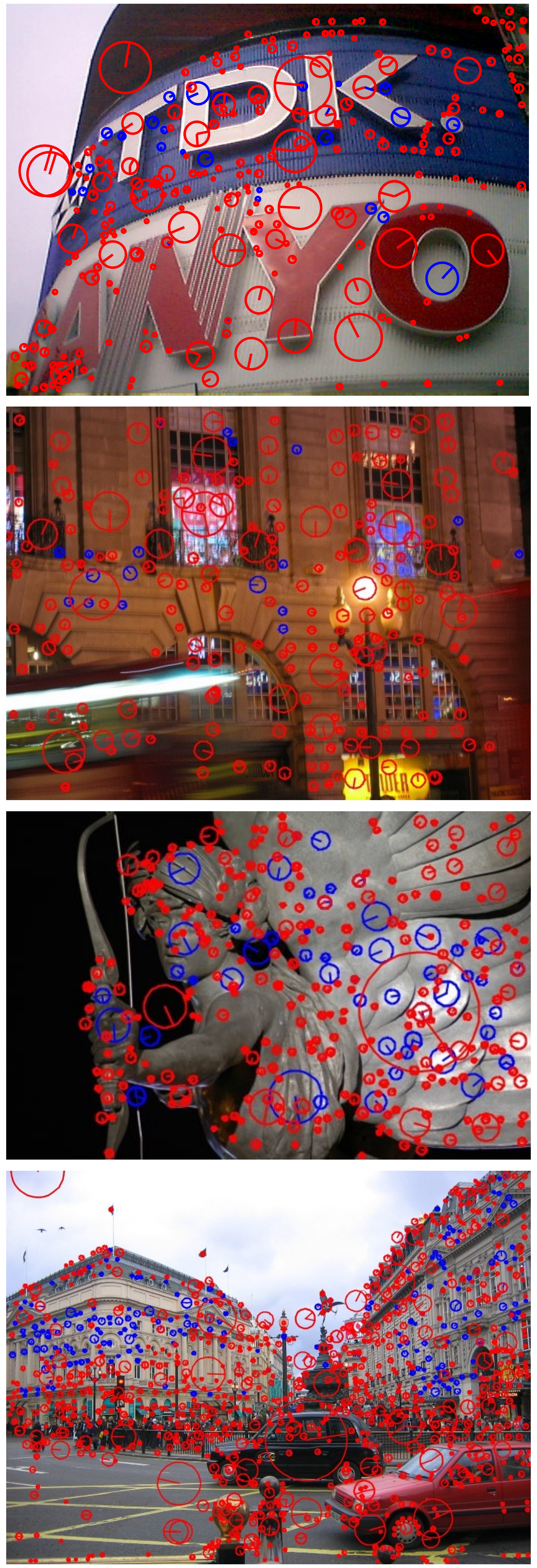


Minimize the distance for corresponding matches.

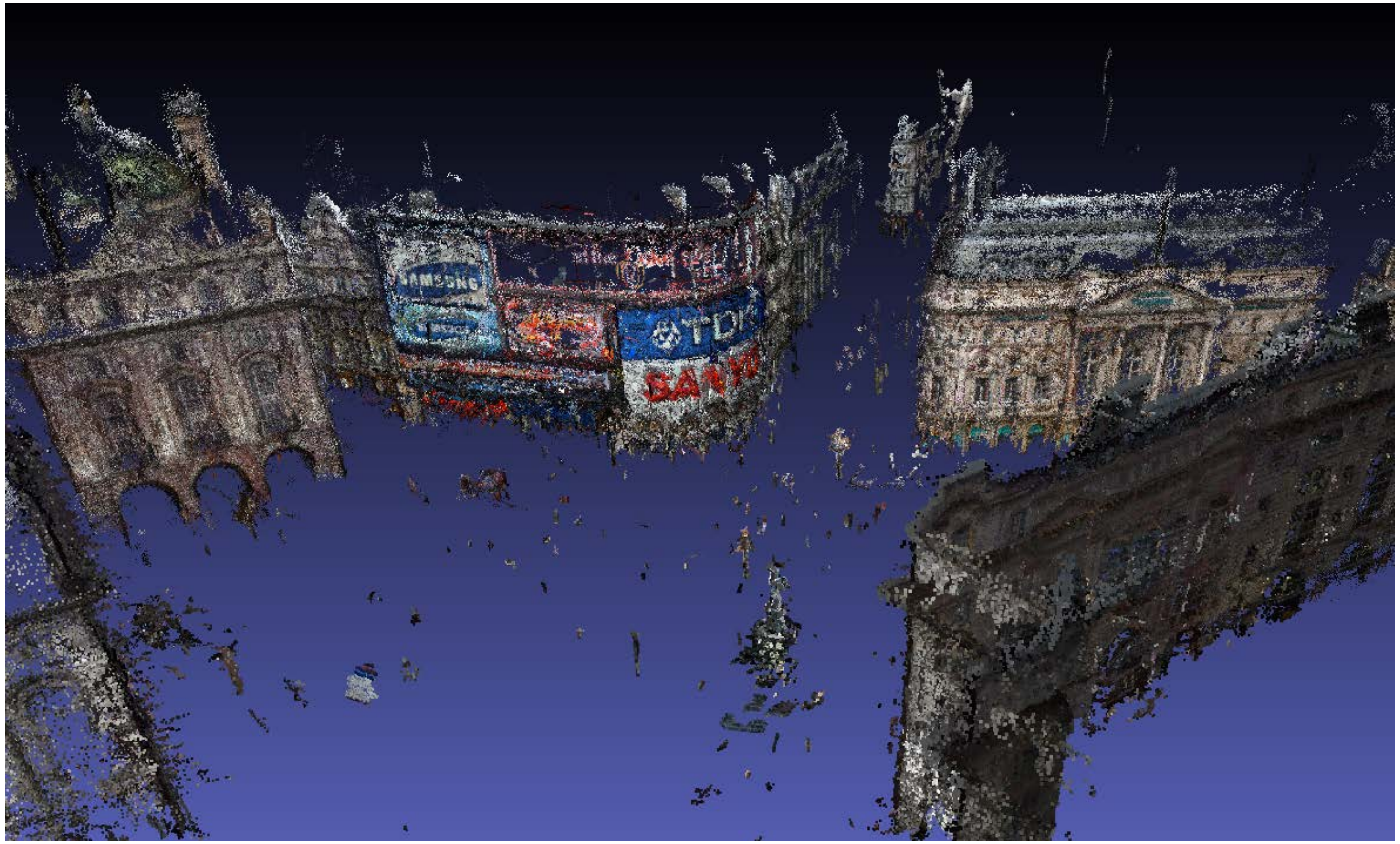
Maximize it for non-corresponding patches.

Slide credits, Eduard Trulls

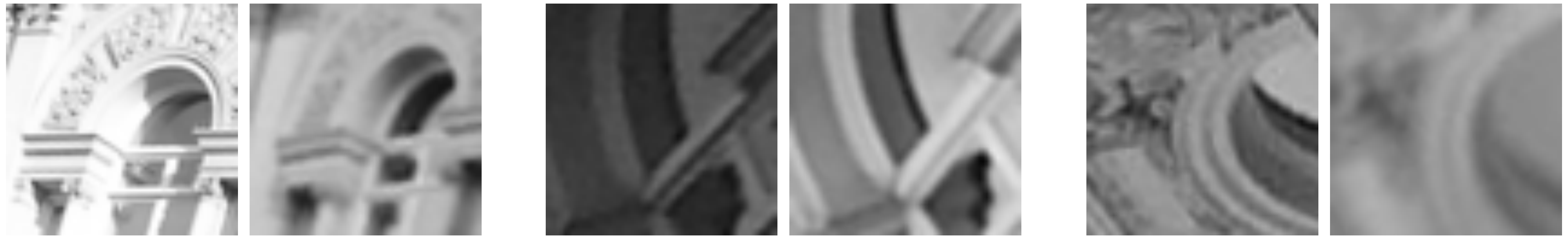
Learning with SfM dataset



Training set #1:

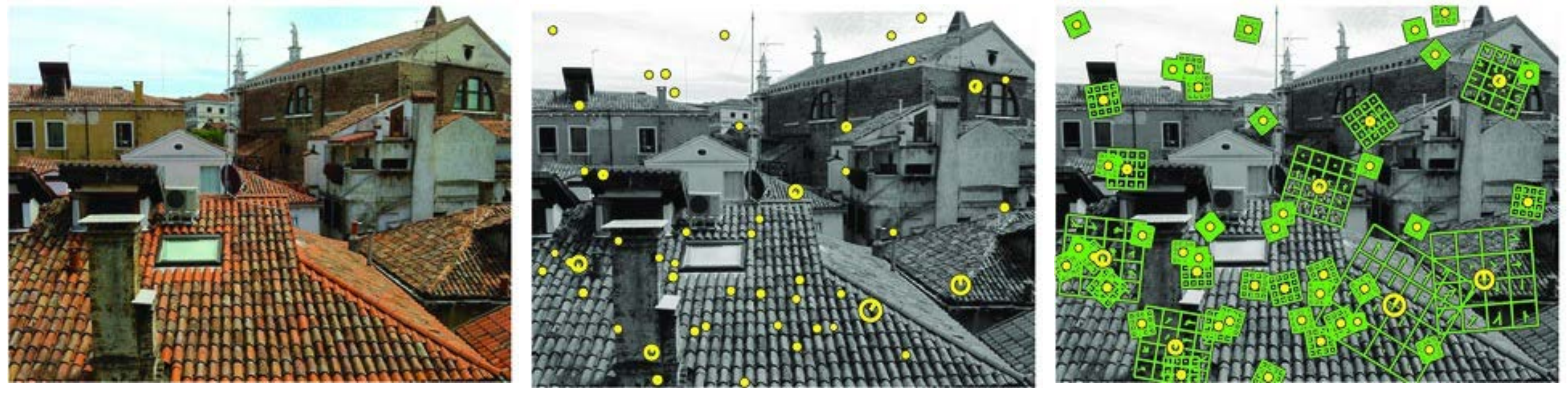


3k images, 59k unique points, 380k



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 3:** Due **Wednesday**!

Learning **Goals**

1. Linear (Projective) Transformations
2. Good results don't happen by chance (or do they?)
3. Good == more support

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

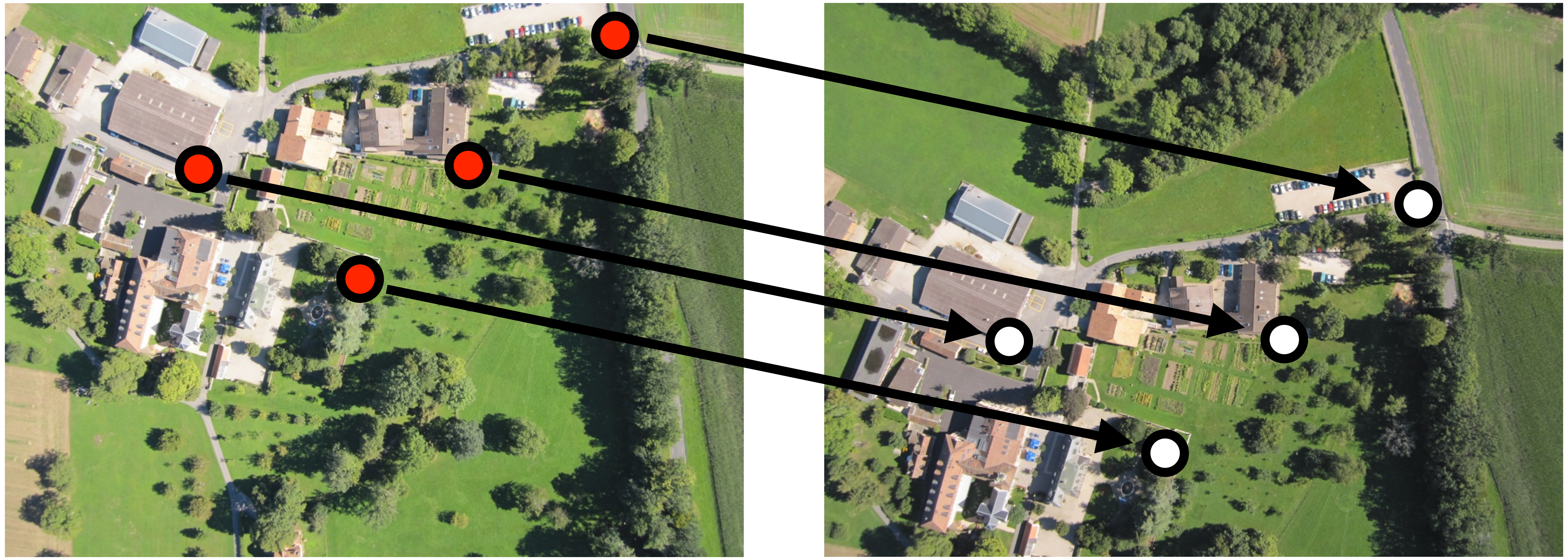


Image **Alignment**

Compute the transformation to align the points

Image **Alignment**

We can also use this transformation to reject outliers

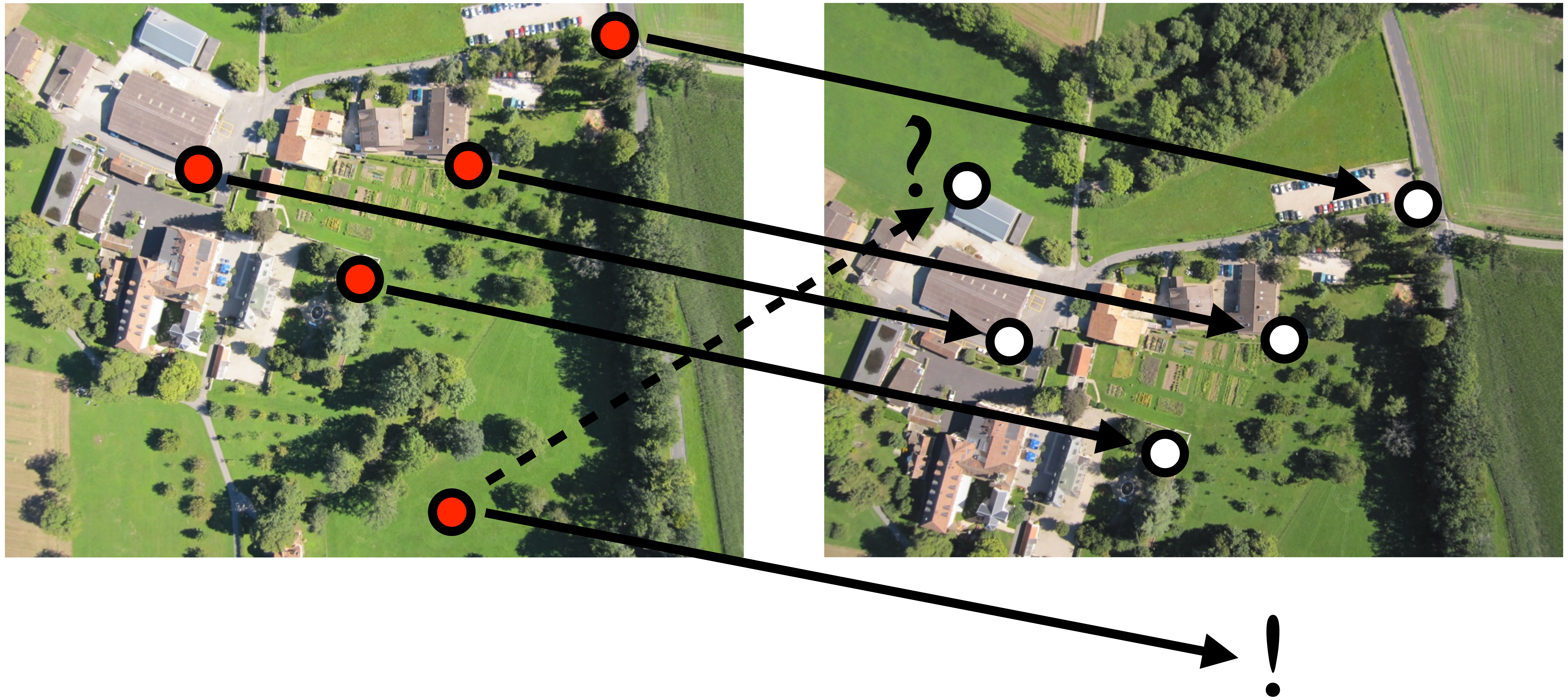


Image **Alignment**

We can also use this transformation to reject outliers

Planar Geometry

— 2D Linear + **Projective** transformations

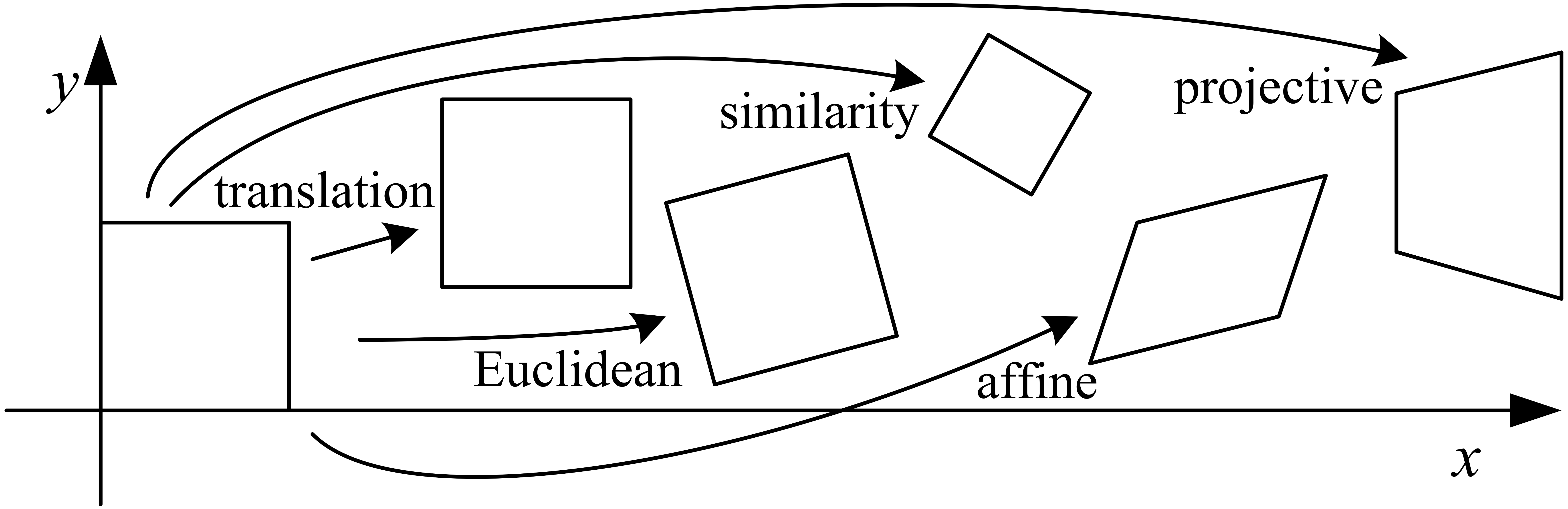
Euclidean, Similarity, Affine, Homography

— Robust Estimation and **RANSAC**

Estimating 2D transforms with noisy correspondences

2D **Transformations**

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



— This group represents perspective projections of **planar surfaces**

Affine Transformation

– Transformed points are a **linear function** of the input points

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

Affine Transformation

— Transformed points are a **linear function** of the input points

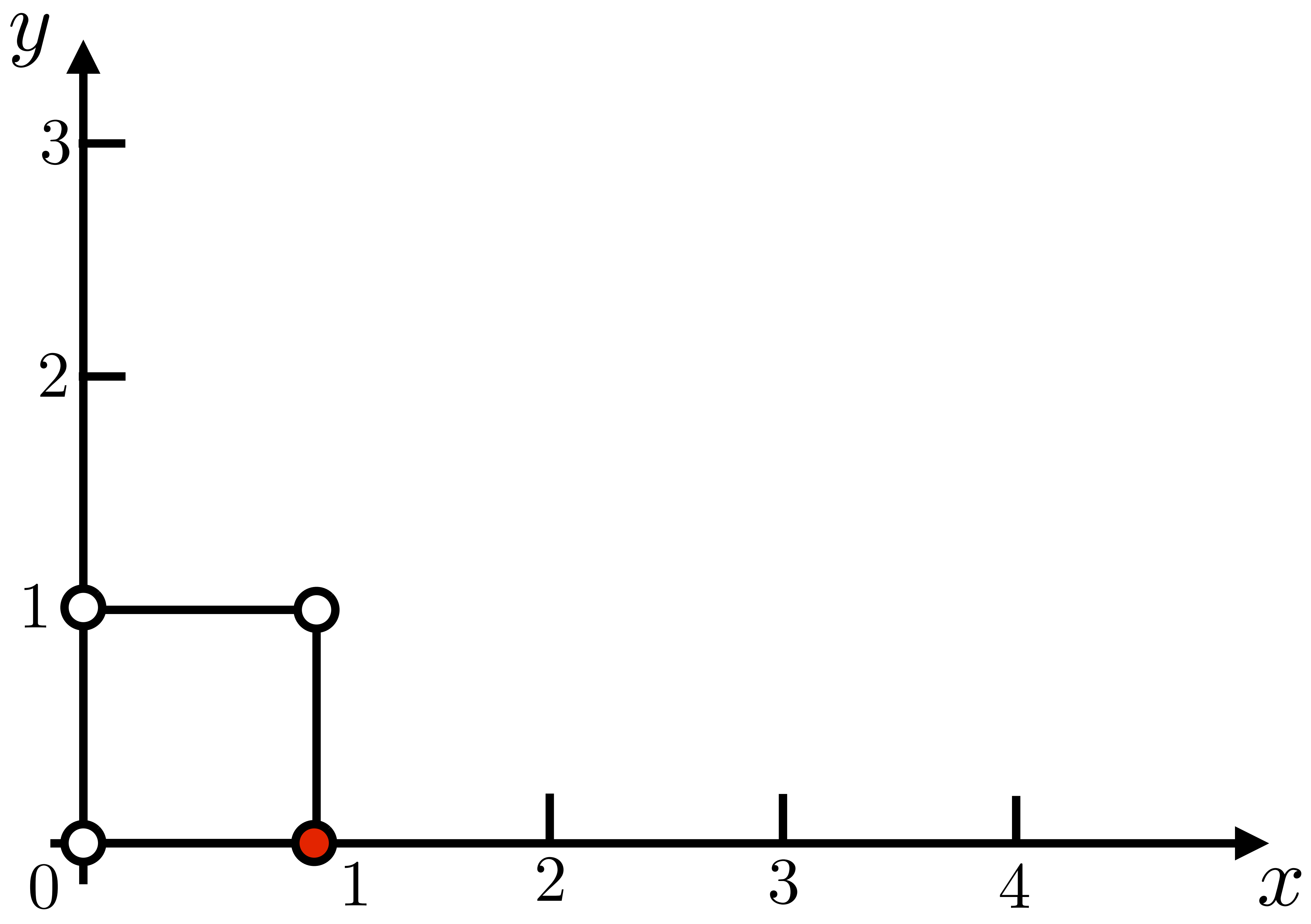
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}$$

— This can be written as a **single matrix multiplication** using **homogeneous** coordinates

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

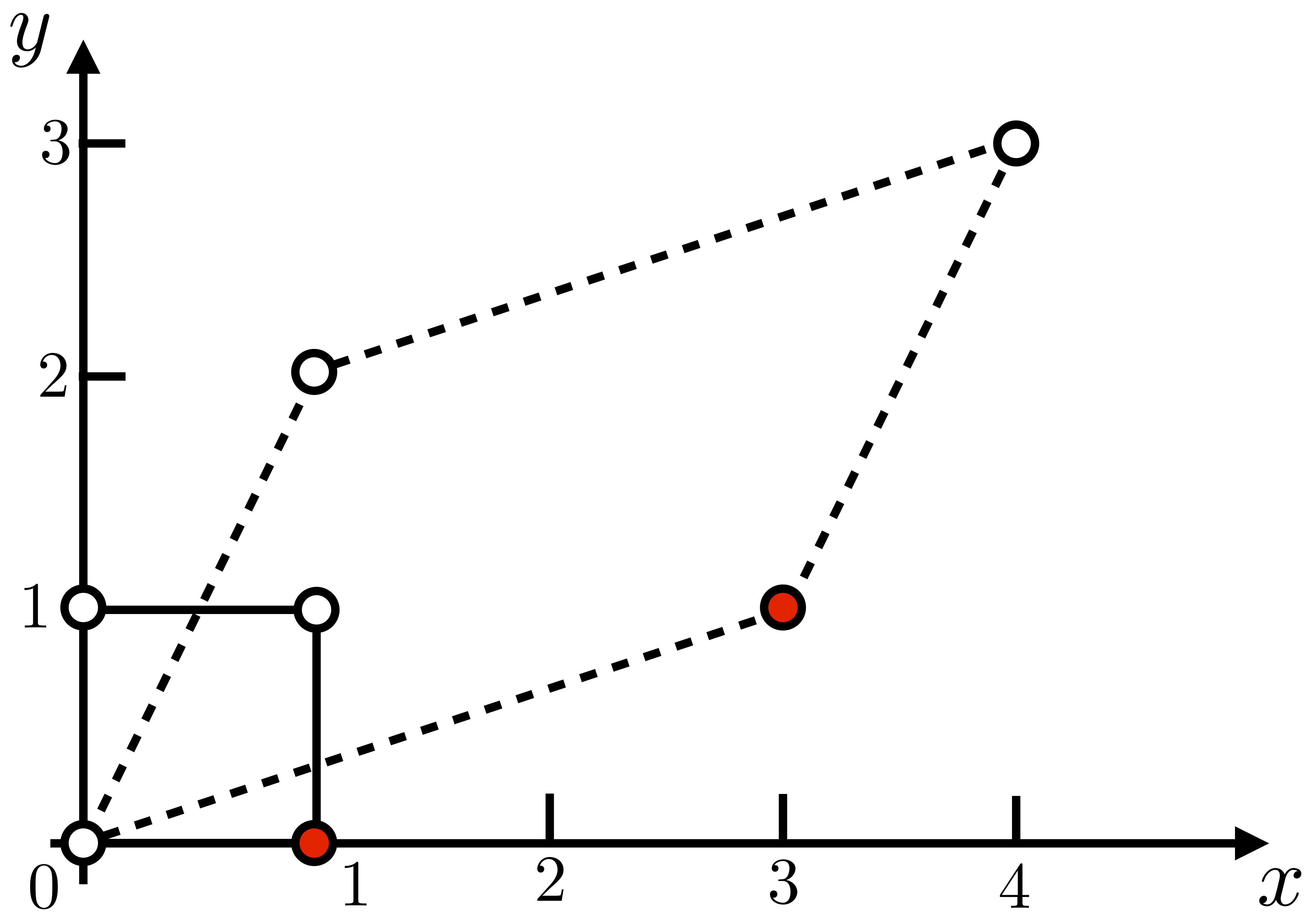
Linear Transformation

– Consider the action of the unit square under, sample transform $\begin{bmatrix} 3 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$



Linear Transformation

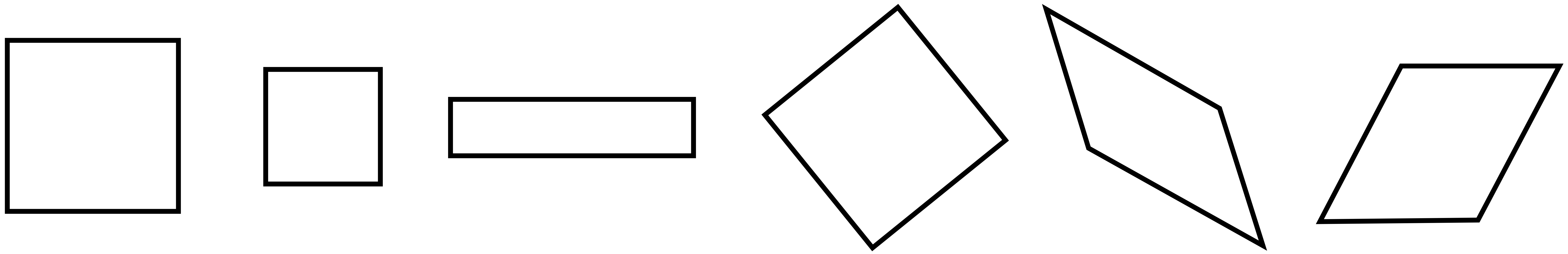
— Consider the action of the unit square under, sample transform $\begin{bmatrix} 3 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$



$$\begin{bmatrix} 3 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 3 & 4 \\ 0 & 2 & 1 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

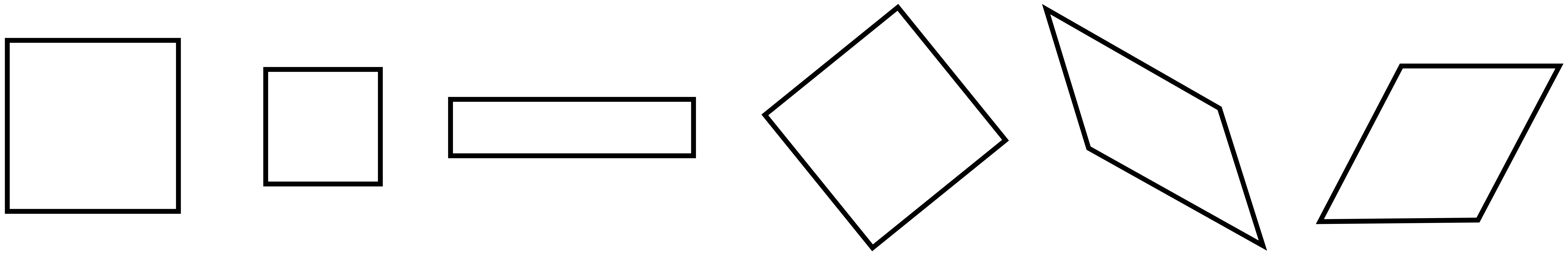
Transformation — Points — Transformed Points

Linear (or Affine) Transformations

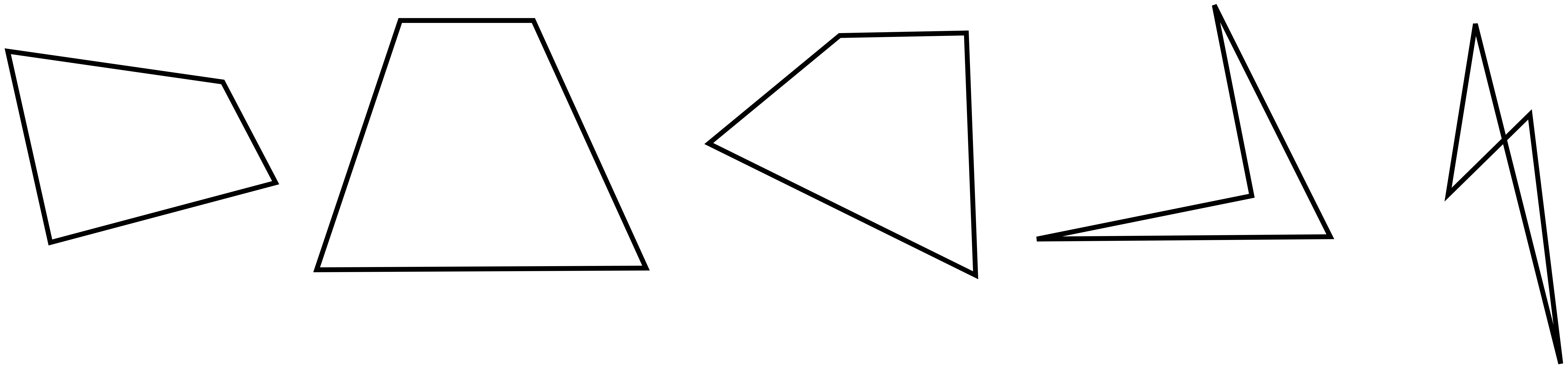


Translation, rotation, scale, shear (parallel lines preserved)

Linear (or Affine) Transformations



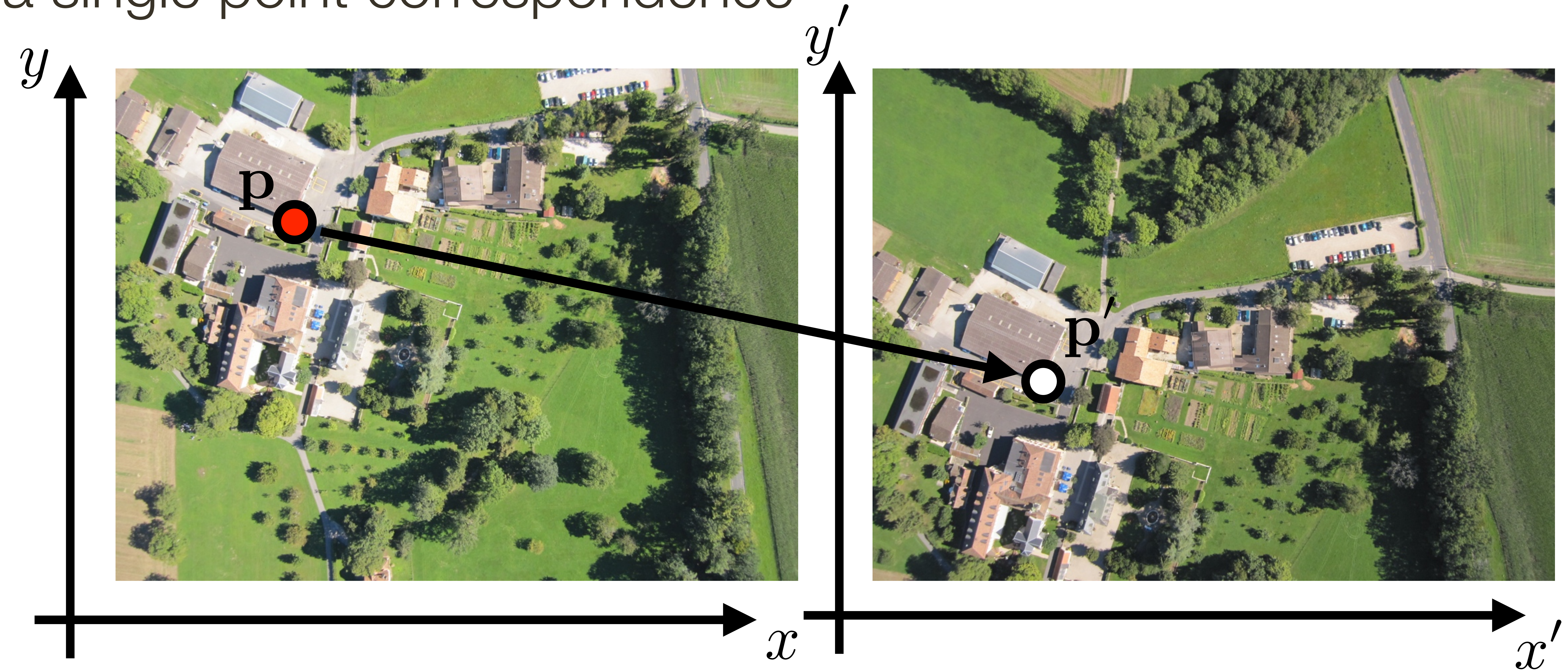
Translation, rotation, scale, shear (parallel lines preserved)



These transforms are not affine (parallel lines not preserved)

Linear (or Affine) Transformations

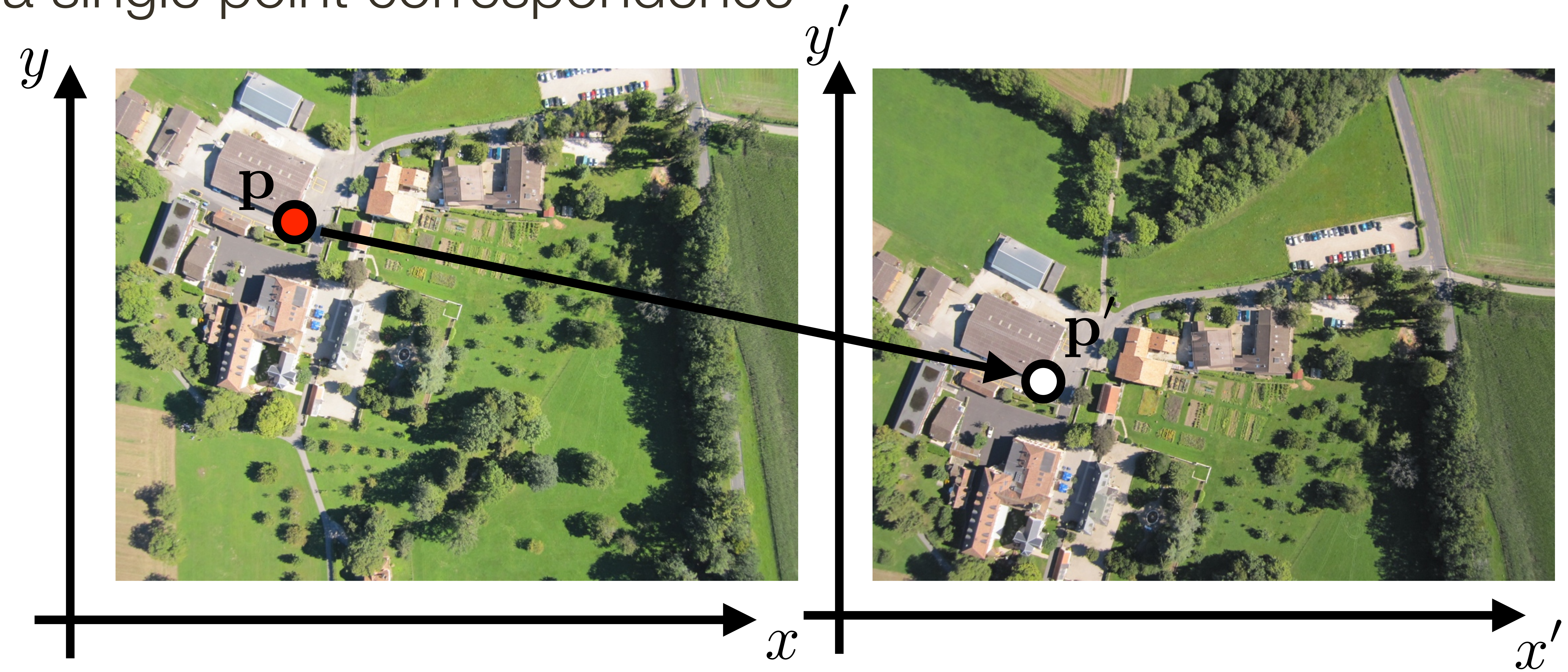
Consider a single point correspondence



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

Linear (or Affine) 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 Transform

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

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

Computing Affine Transform

Linear system in the unknown parameters **a**

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

Of the form

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

Solve for **a** using Gaussian Elimination

Computing Affine Transform

Once we solve for a transform, we can now map any <u>other points</u> between the two images … or resample one image in the coordinate system of the other

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

Computing Affine Transform

Once we solve for a transform, we can now map any <u>other points</u> between the two images … 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** transform:

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

Linear Transformations

Other linear transforms are special cases of **affine** transform:

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

e.g.,

$$\begin{bmatrix} s\cos\theta & s\sin\theta & a_{13} \\ -s\sin\theta & s\cos\theta & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

similarity transform

Linear Transformations

Other linear transforms are special cases of **affine** transform:

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

e.g.,

$$\begin{bmatrix} \cos\theta & \sin\theta & a_{13} \\ -\sin\theta & \cos\theta & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

euclidian transform

Linear Transformations

Other linear transforms are special cases of **affine** transform:

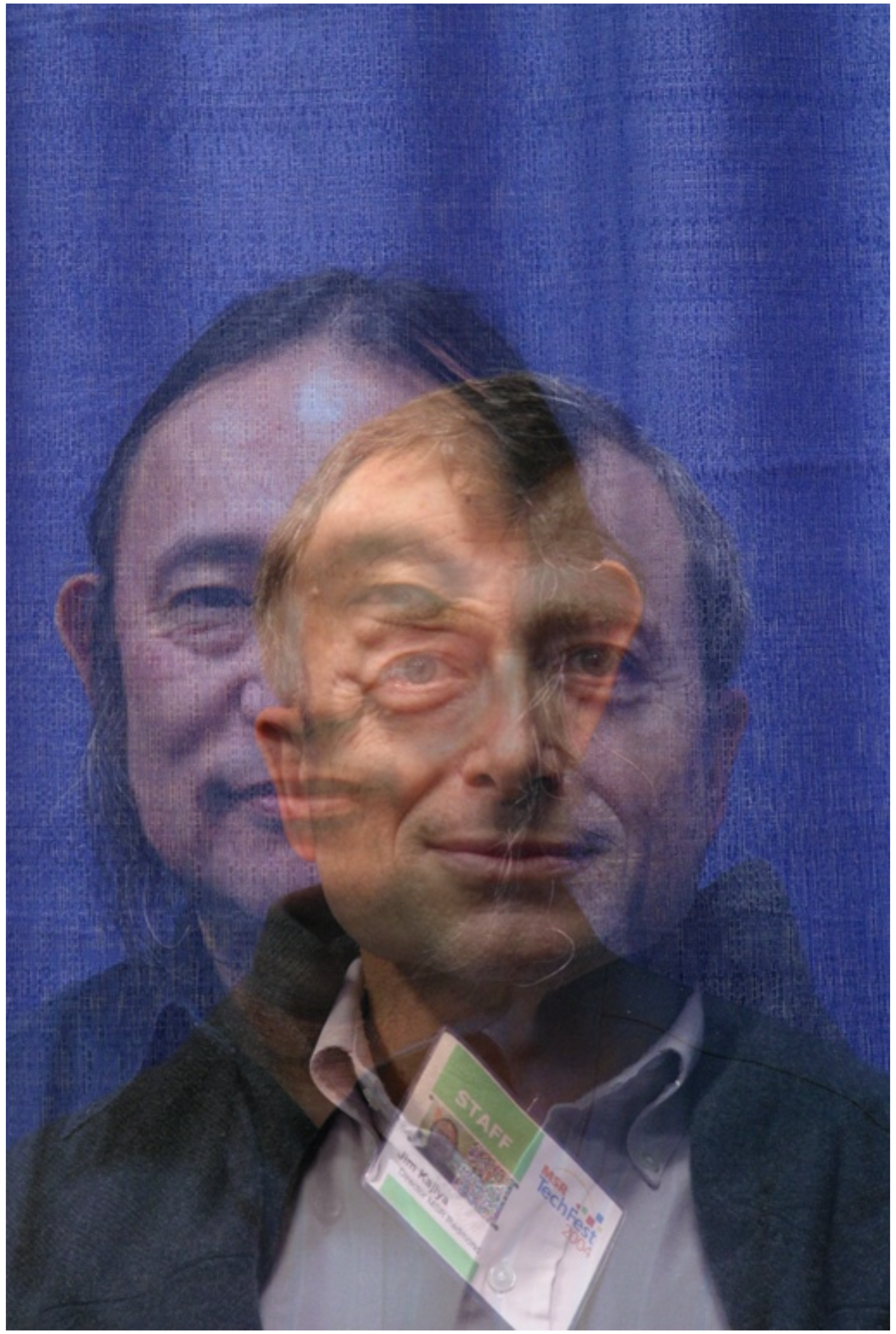
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

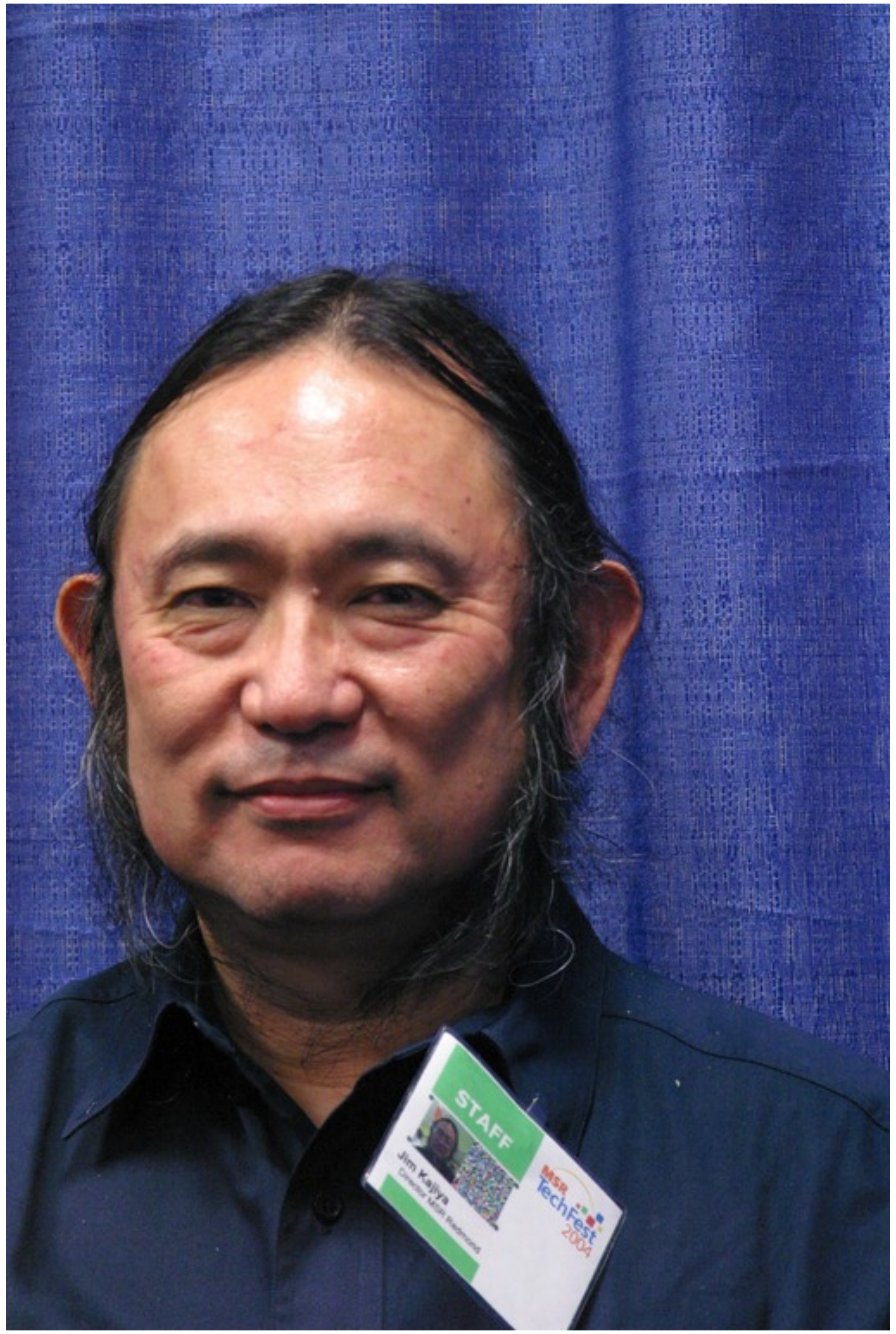
e.g.,

$$\begin{bmatrix} 1 & 0 & a_{13} \\ 0 & 1 & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

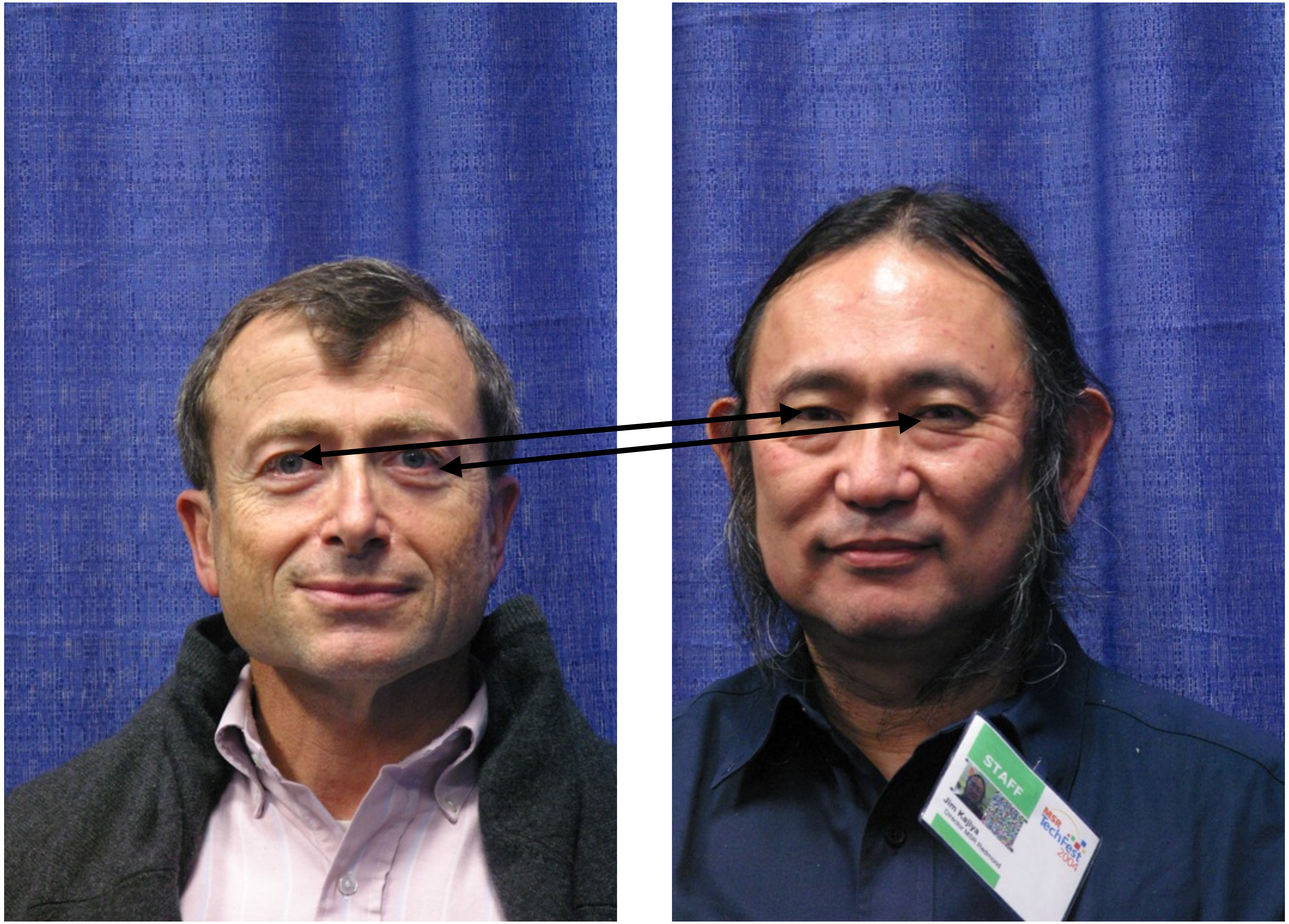
translation transform

Face Alignment

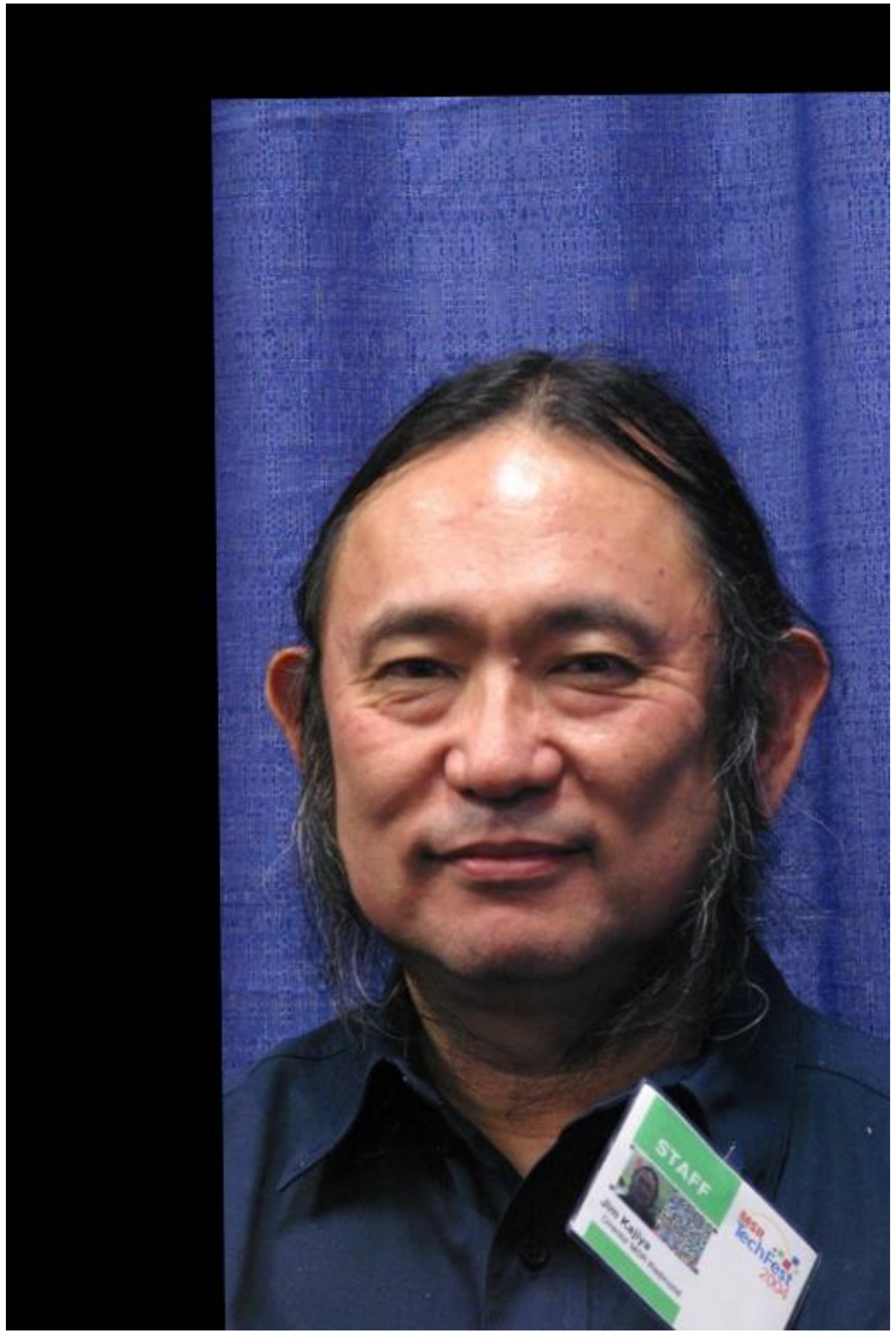


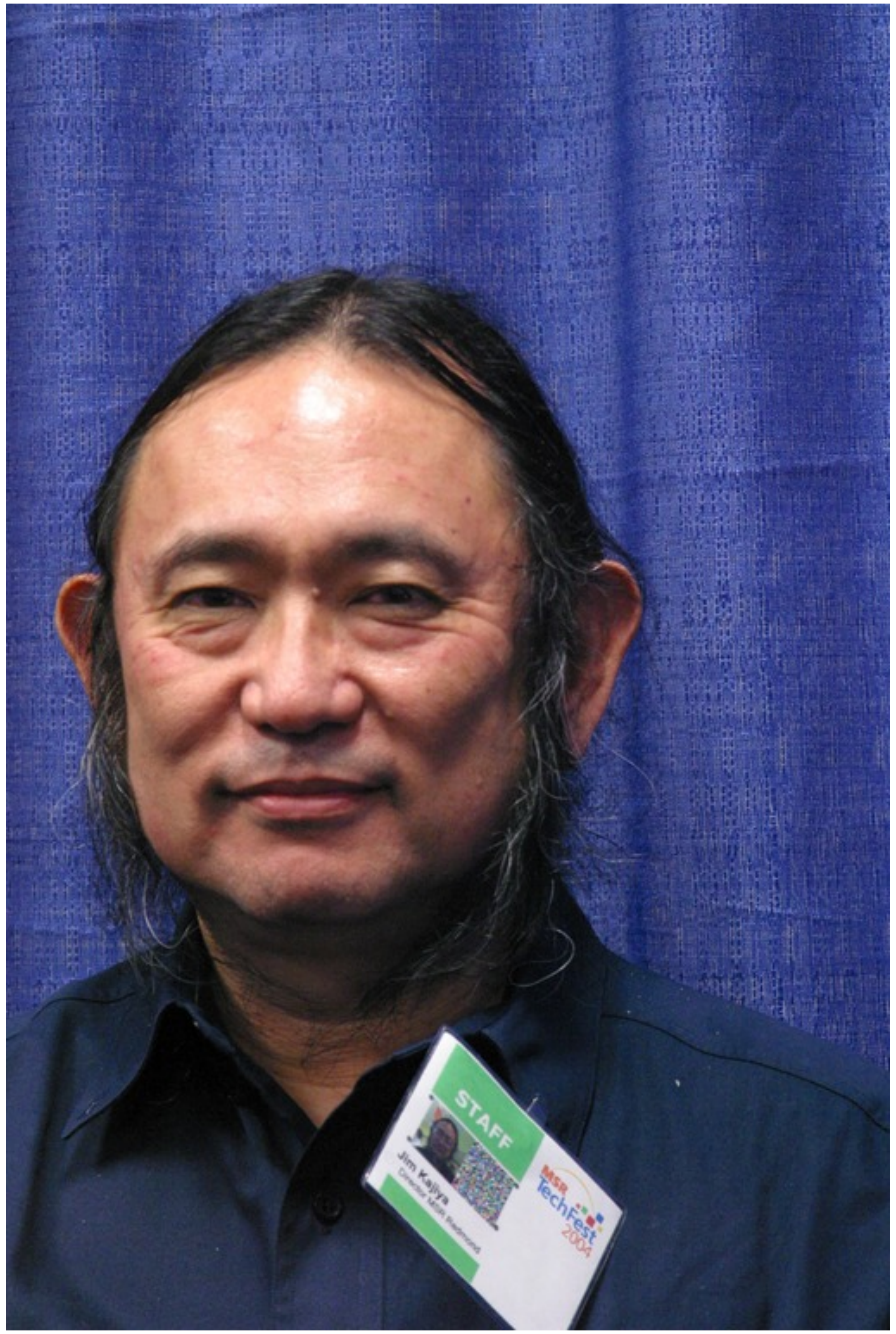


Face Alignment

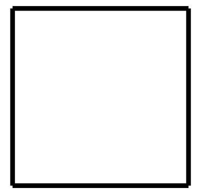
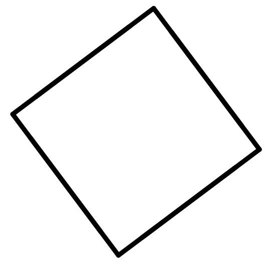
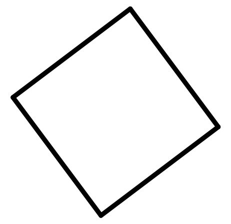
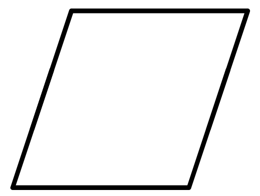
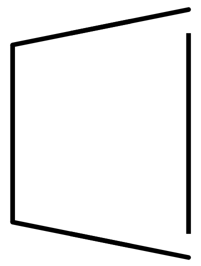


Face Alignment



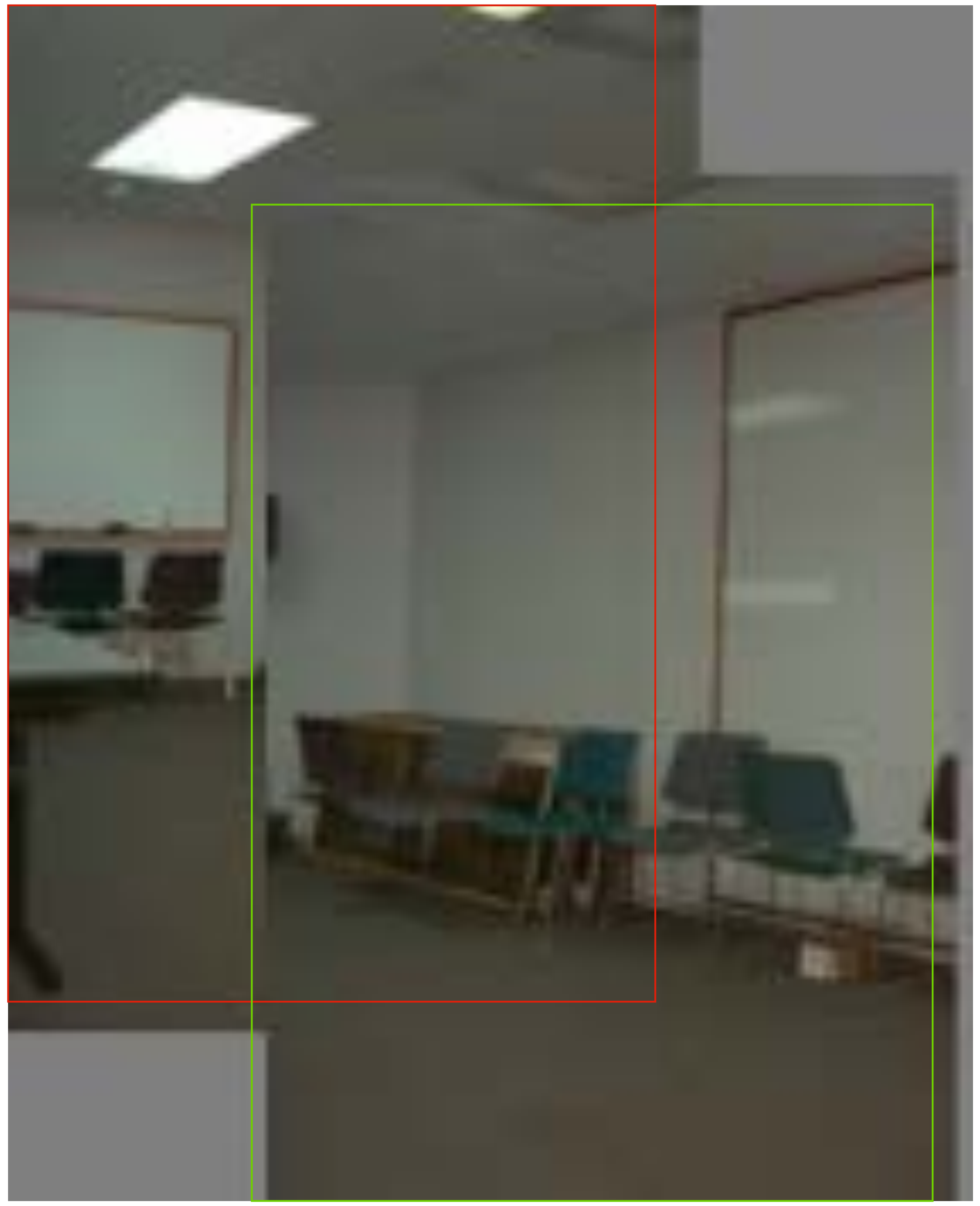


2D **Transformations**

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	

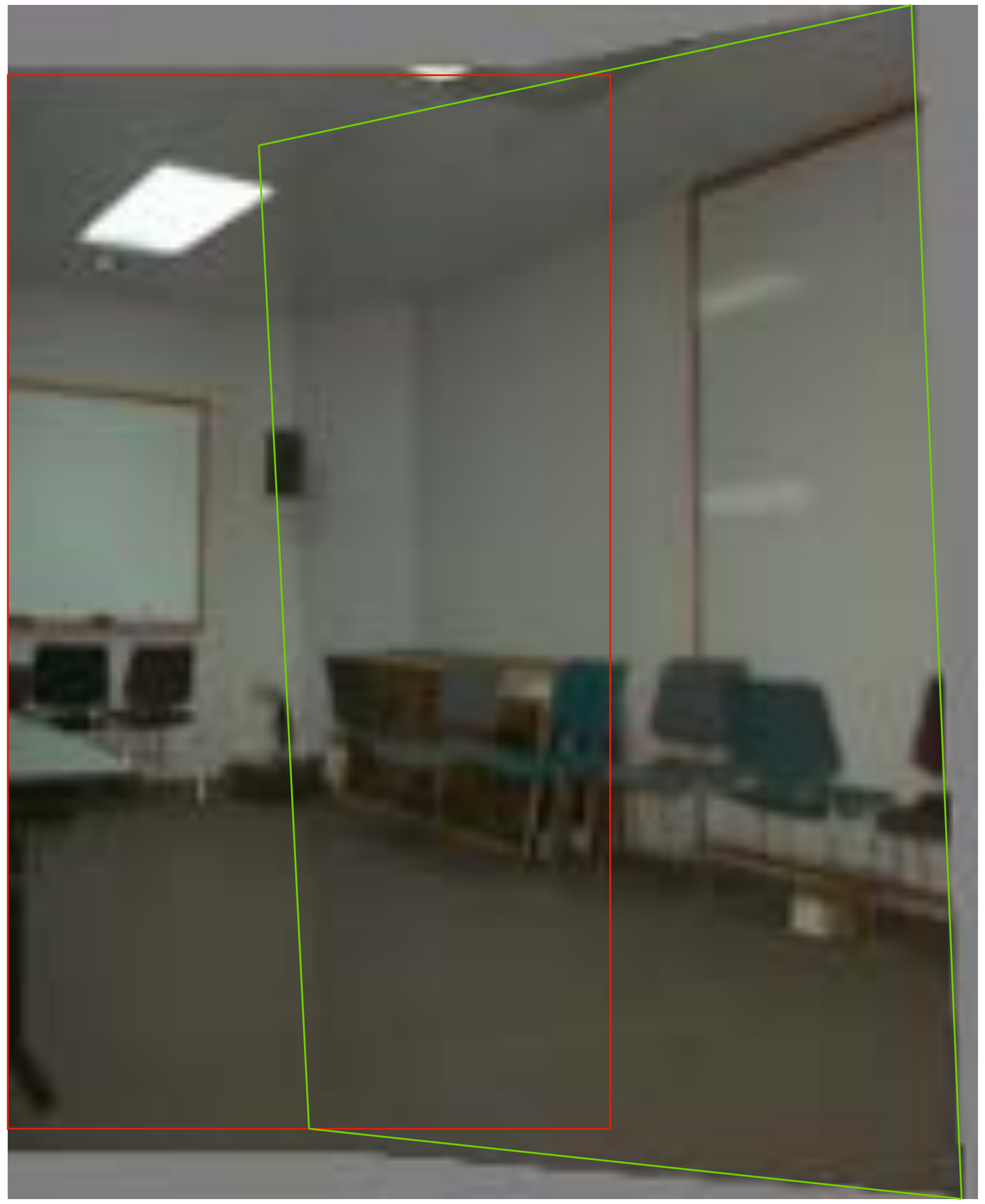
Example: Warping with Different Transformations

Translation



Affine

Projective
(homography)



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aside: We can use homographies when …

1.… the scene is planar; or

2.… the scene is very far or has small (relative) depth variation → scene is approximately planar

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aside: We can use homographies when …

3.… the scene is captured under camera rotation only (no translation or pose change)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Projective Transformation

General 3x3 matrix transformation

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

Projective Transformation

General 3x3 matrix transformation

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

Lets try an example:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 2 & 1 & 2 \end{bmatrix}$$

Transformation Points Transformed Points

Projective Transformation

General 3x3 matrix transformation

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

Lets try an example:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 2 & 1 & 2 \end{bmatrix}$$

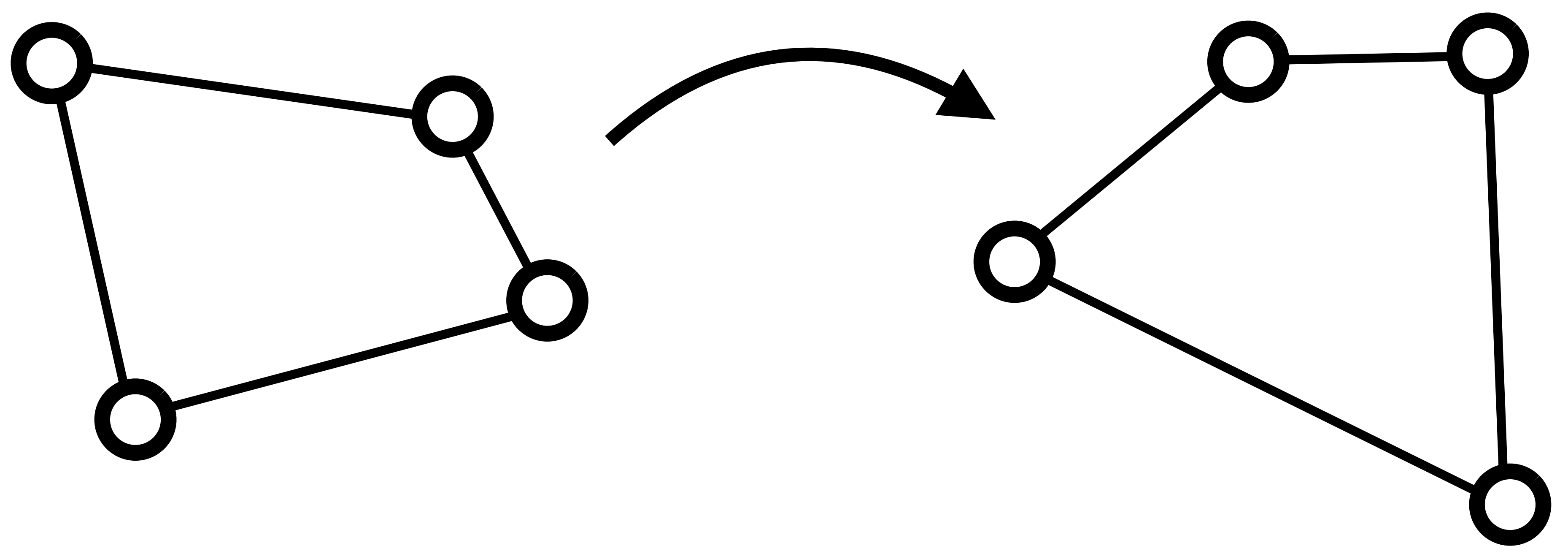
Transformation — Points — Transformed Points

Divide by the last row: $\begin{bmatrix} 0 & 0 & 1 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

Compute **H** from Correspondences

Each match gives 2 equations to solve for **8** parameters

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



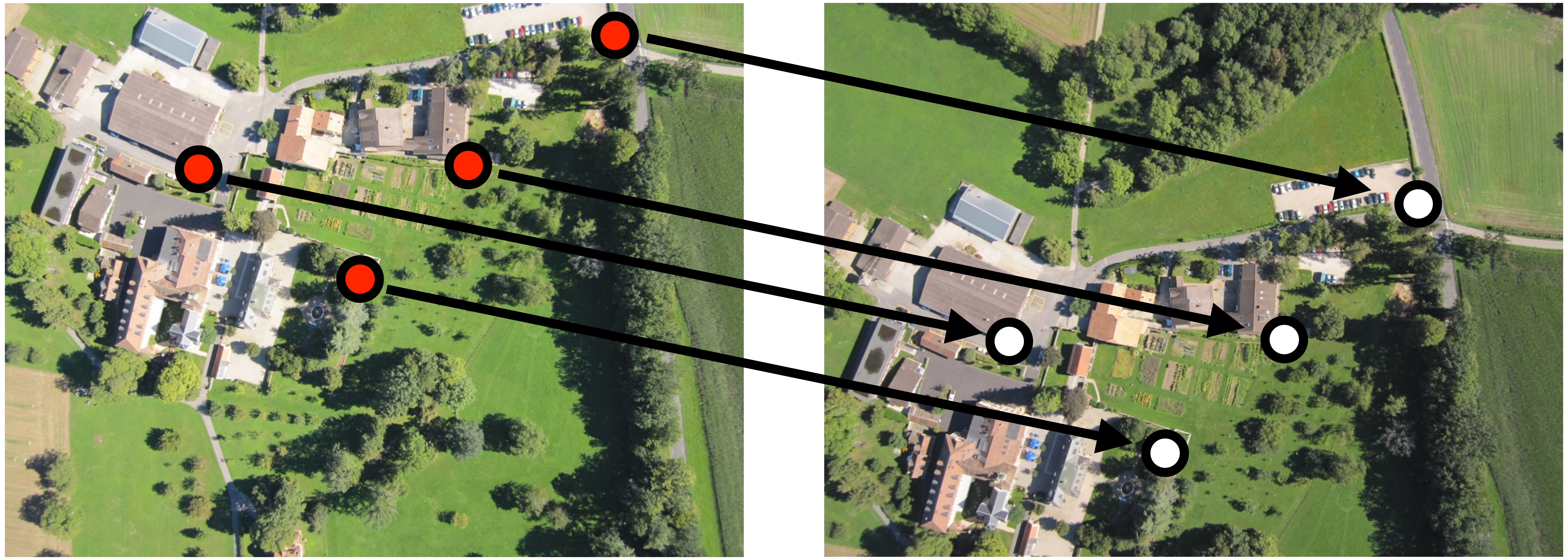
→ 4 correspondences to solve for **H** matrix

Solution uses **Singular Value Decomposition** (SVD)

In **Assignment 4** you can compute this using `cv2.findHomography`

Image **Alignment**

Find **corresponding** (matching) points between the image



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

2 points for Similarity

3 for Affine

4 for Homography

Image **Alignment**

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

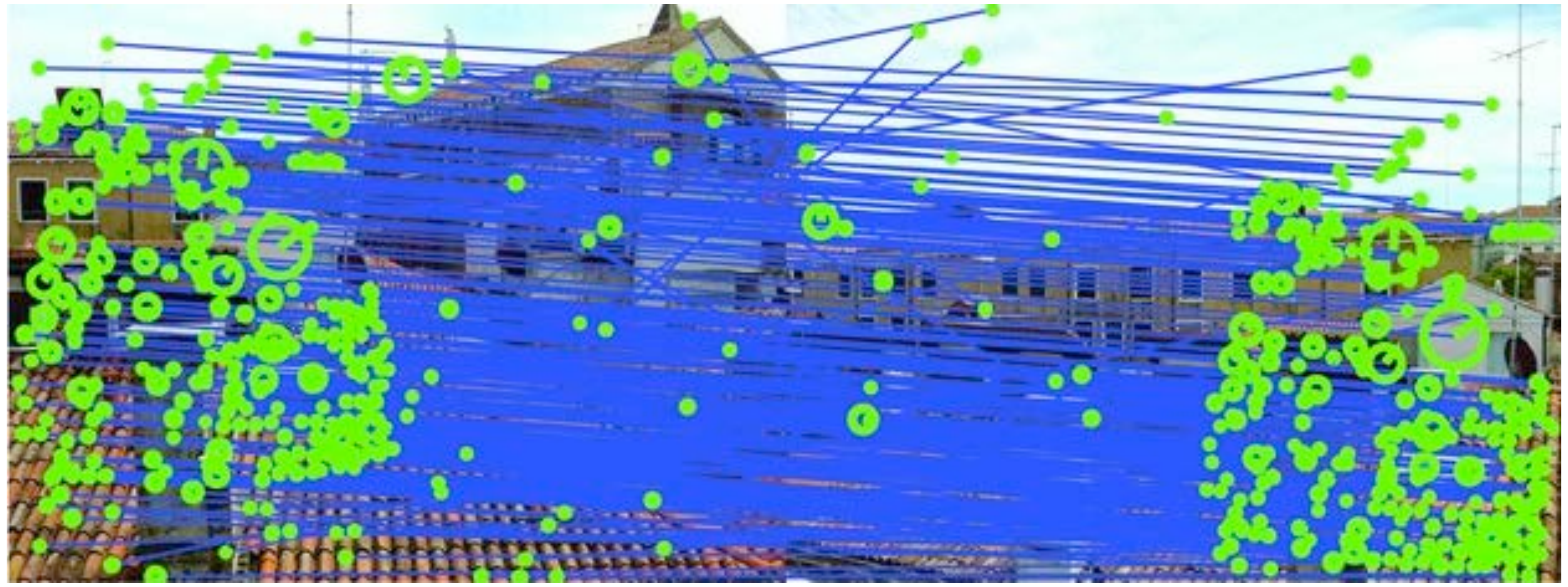


Image **Alignment**

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

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

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

$$\vdots \qquad\qquad \vdots$$

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 …

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)?

— If we draw pairs of points uniformly at random, then about 1/4 of these pairs will consist entirely of 'good' data points (inliers)

— We can identify these good pairs by noticing that a large collection of other points lie close to the line fitted to the pair

— A better estimate of the line can be obtained by refitting the line to the points that lie close to the line

RANSAC (**RAN**dom **SA**mple **C**onsensus)

1. Randomly choose minimal subset of data points necessary to fit model (a **sample**)

2. Points within some distance threshold, t, of model are a **consensus set**. Size of consensus set is model's **support**

3. Repeat for N samples; model with biggest support is most robust fit
   - Points within distance t of best model are inliers
   - Fit final model to all inliers

Slide Credit: Christopher Rasmussen

RANSAC (**RAN**dom **SA**mple **C**onsensus)

1. Randomly choose minimal subset of data points necessary to fit model (a **sample**)

2. Points within some distance threshold, t, of model are a **consensus set**. Size of consensus set is model's **support**

3. Repeat for N samples; model with biggest support is most robust fit
   — Points within distance t of best model are inliers
   — Fit final model to all inliers

RANSAC is very useful for variety of applications

Slide Credit: Christopher Rasmussen

RANSAC (**RAN**dom **SA**mple **C**onsensus)

1. Randomly choose minimal subset of data points necessary to fit model (a **sample**)

   Fitting a Line: 2 points

2. Points within some distance threshold, t, of model are a **consensus set**. Size of consensus set is model's **support**

3. Repeat for N samples; model with biggest support is most robust fit
   - Points within distance t of best model are inliers
   - Fit final model to all inliers

Slide Credit: Christopher Rasmussen

Example 1: Fitting a Line

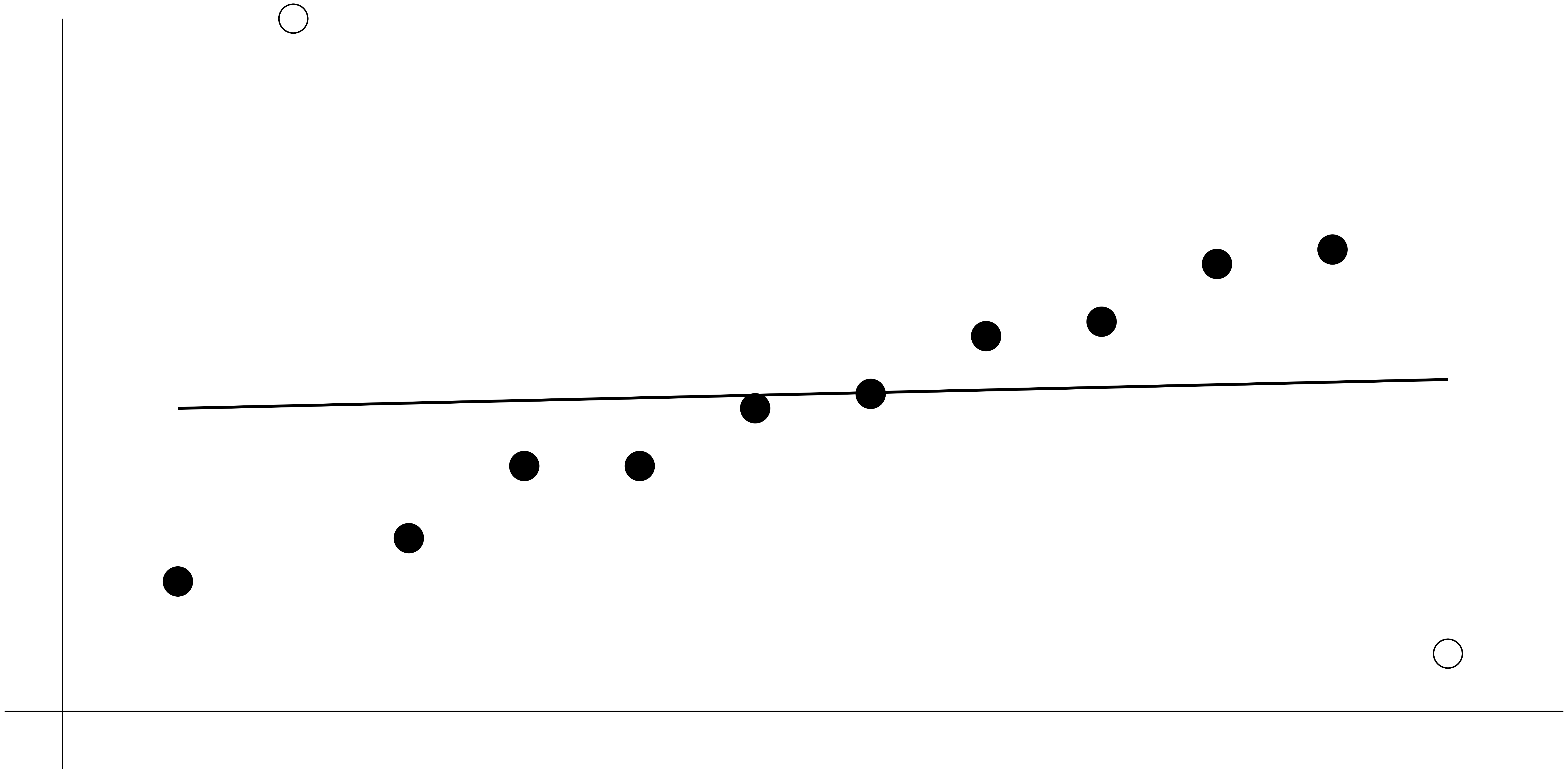


Figure Credit: Hartley & Zisserman

Example 1: Fitting a Line

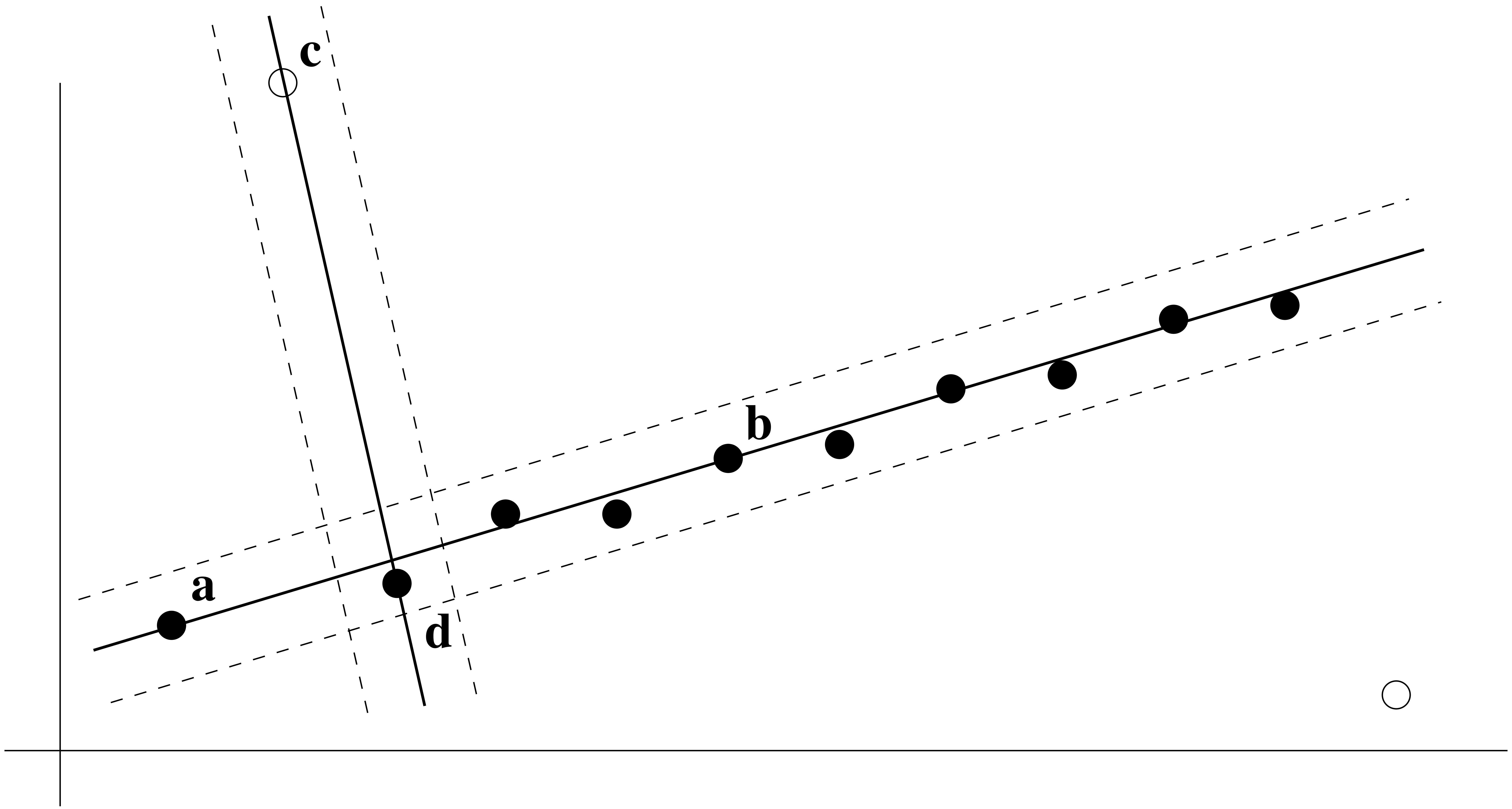


Figure Credit: Hartley & Zisserman

Example 1: Fitting a Line

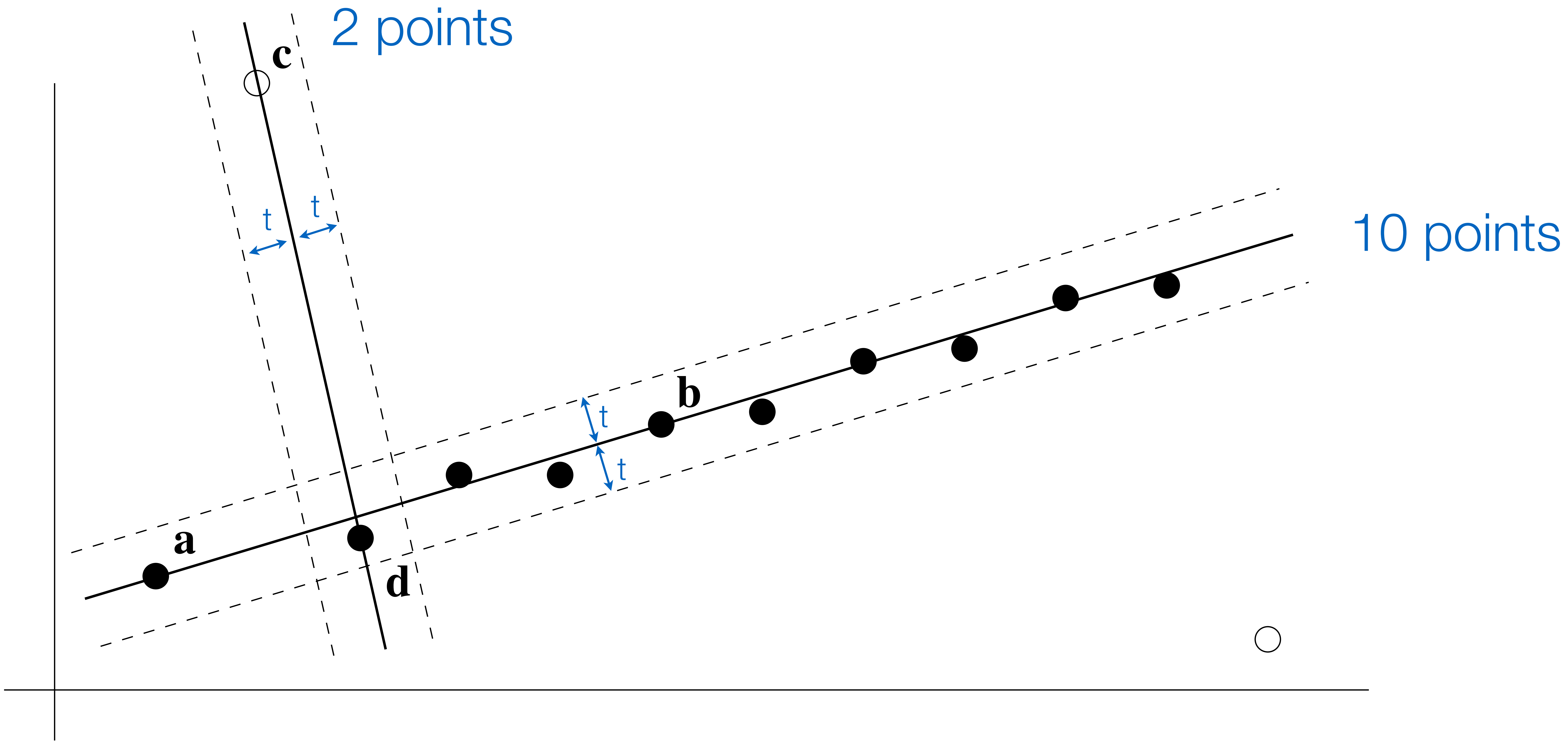


Figure Credit: Hartley & Zisserman

After RANSAC

RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers

Improve this initial estimate with estimation over all inliers (e.g., with standard least-squares minimization)

But this may change inliers, so alternate fitting with re-classification as inlier/outlier

Example 2: Fitting a Line

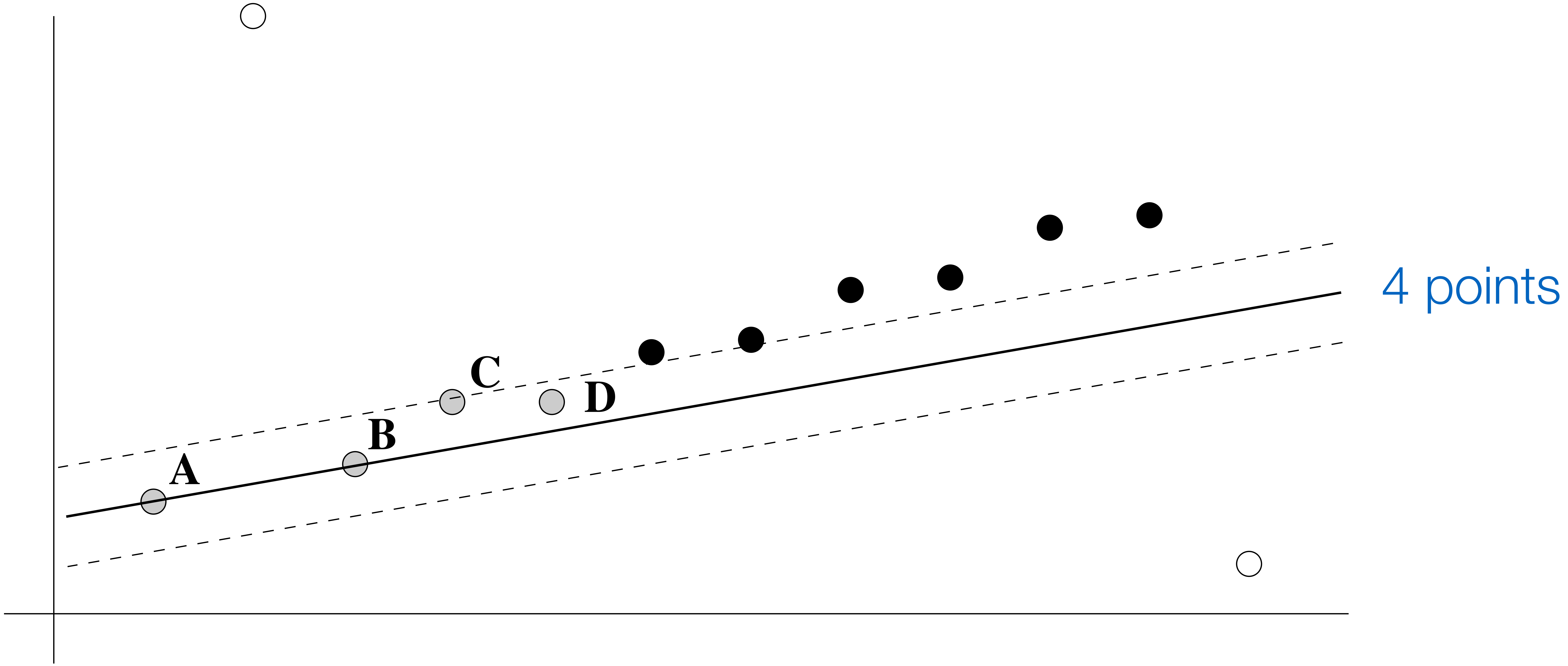


Figure Credit: Hartley & Zisserman

Example 2: Fitting a Line

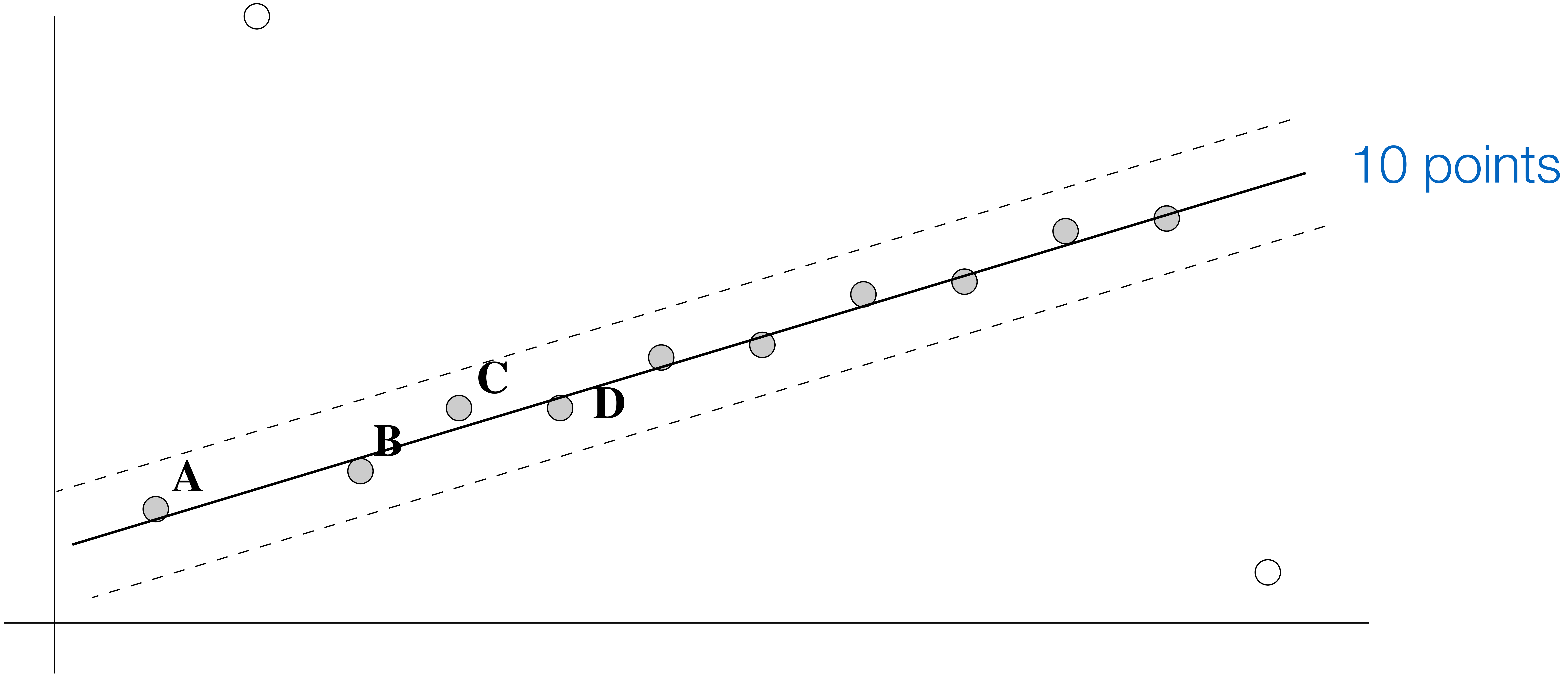


Figure Credit: Hartley & Zisserman

Image **Alignment + RANSAC**

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

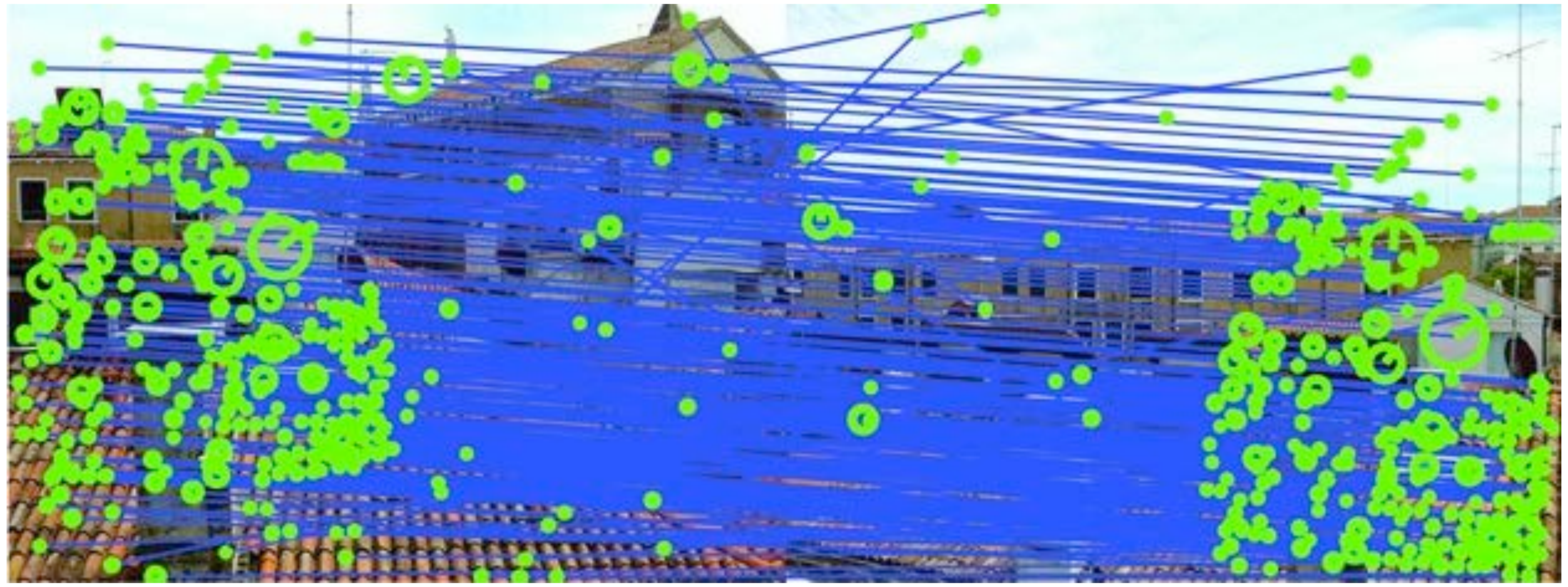


Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

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

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

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

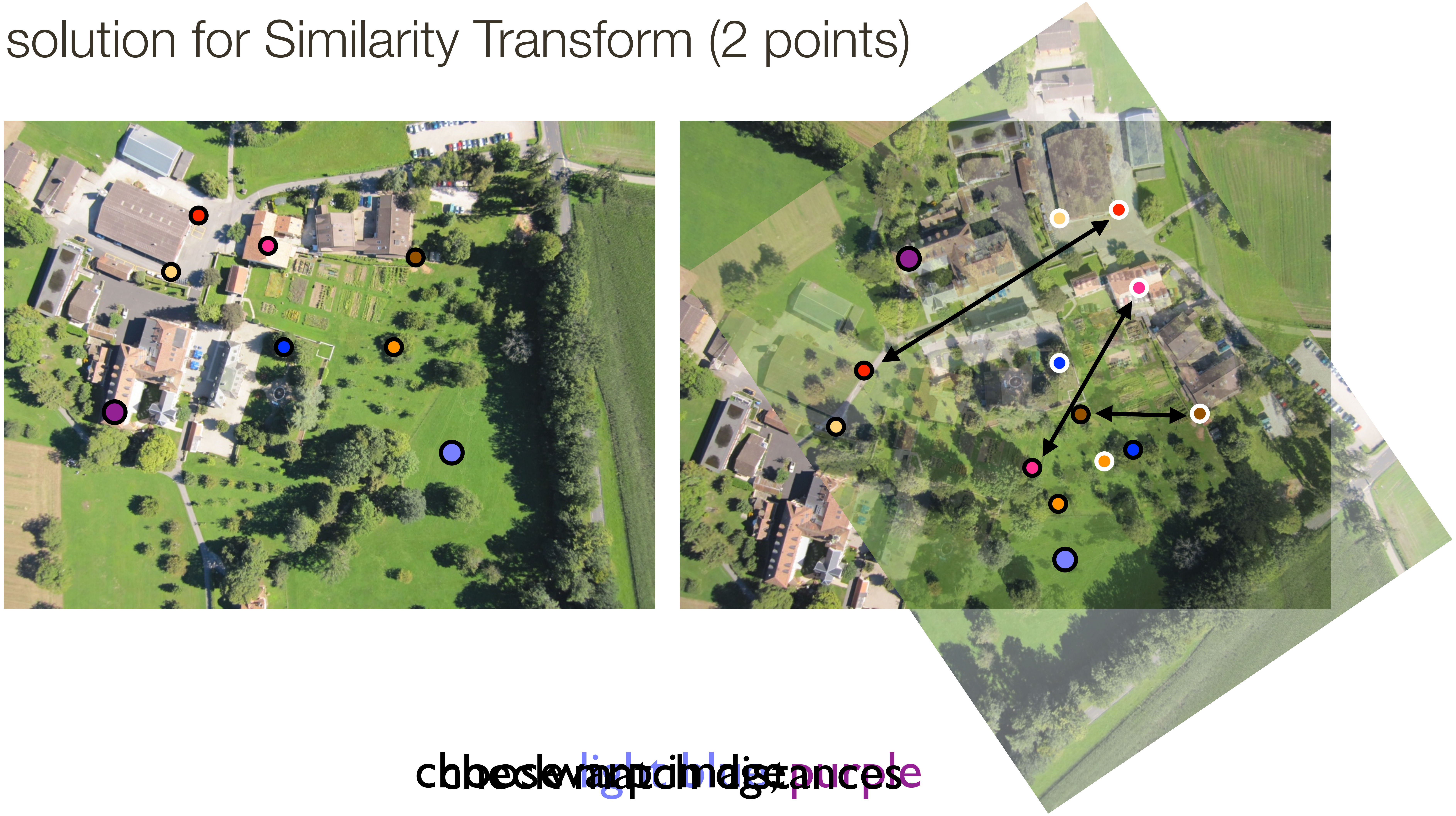
Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

4 inliers (red, yellow, orange, brown),
4 outliers (blue, light blue, purple, pink)

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)



choose light blue, purple

warp image

check match distances

#inliers = 2

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

choose pink, blue

warp image

check distances

#inliers = 2

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

choose red, orange
warp image
check match distances

#inliers = 4

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Image **Alignment + RANSAC**

Assignment 4

1. Match feature points between 2 views

2. Select minimal subset of matches*

3. Compute transformation T using minimal subset

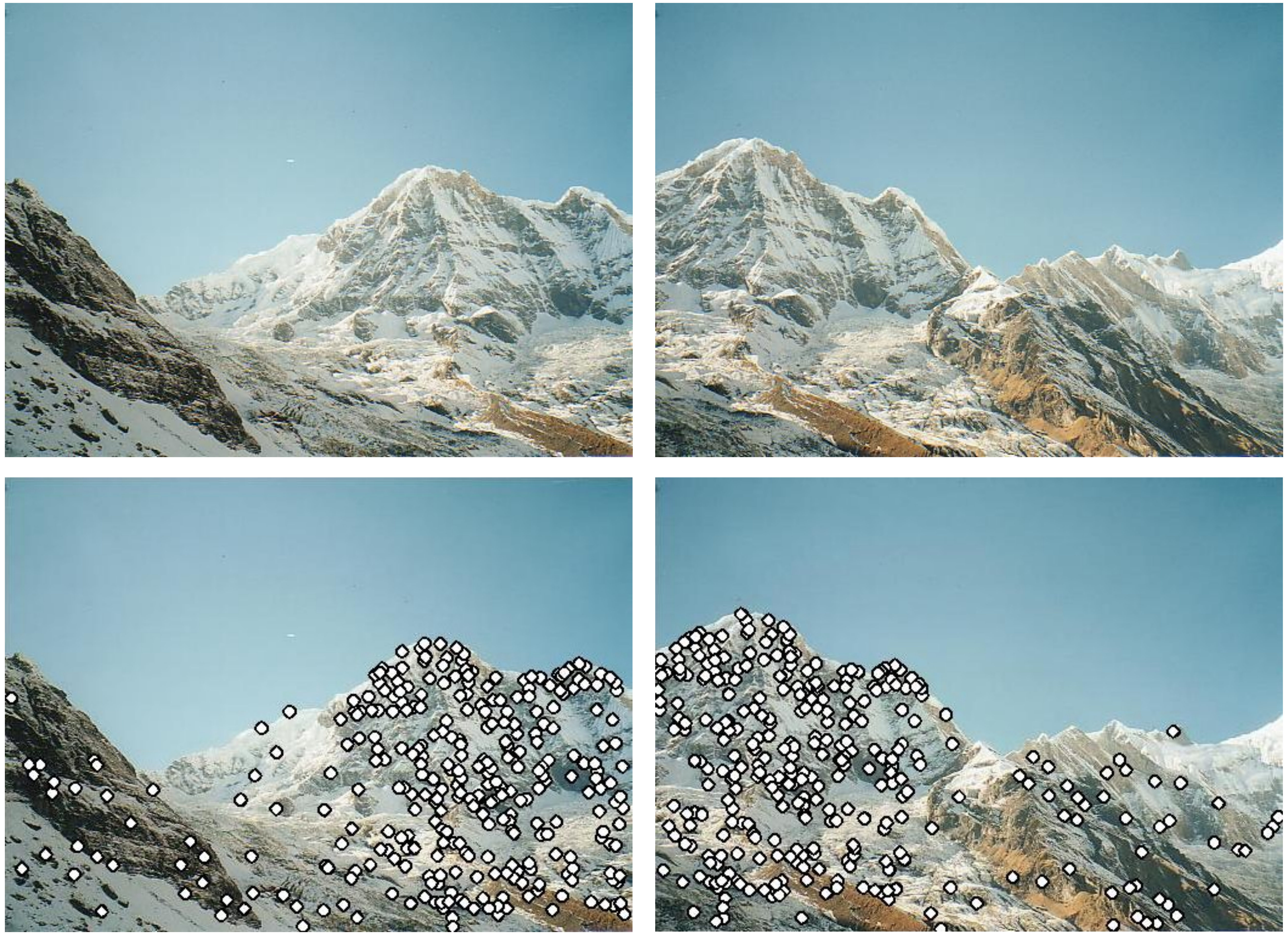
4. Check consistency of all points with T — compute projected position and count #inliers with distance < threshold

5. Repeat steps 2-4 to maximize #inliers

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

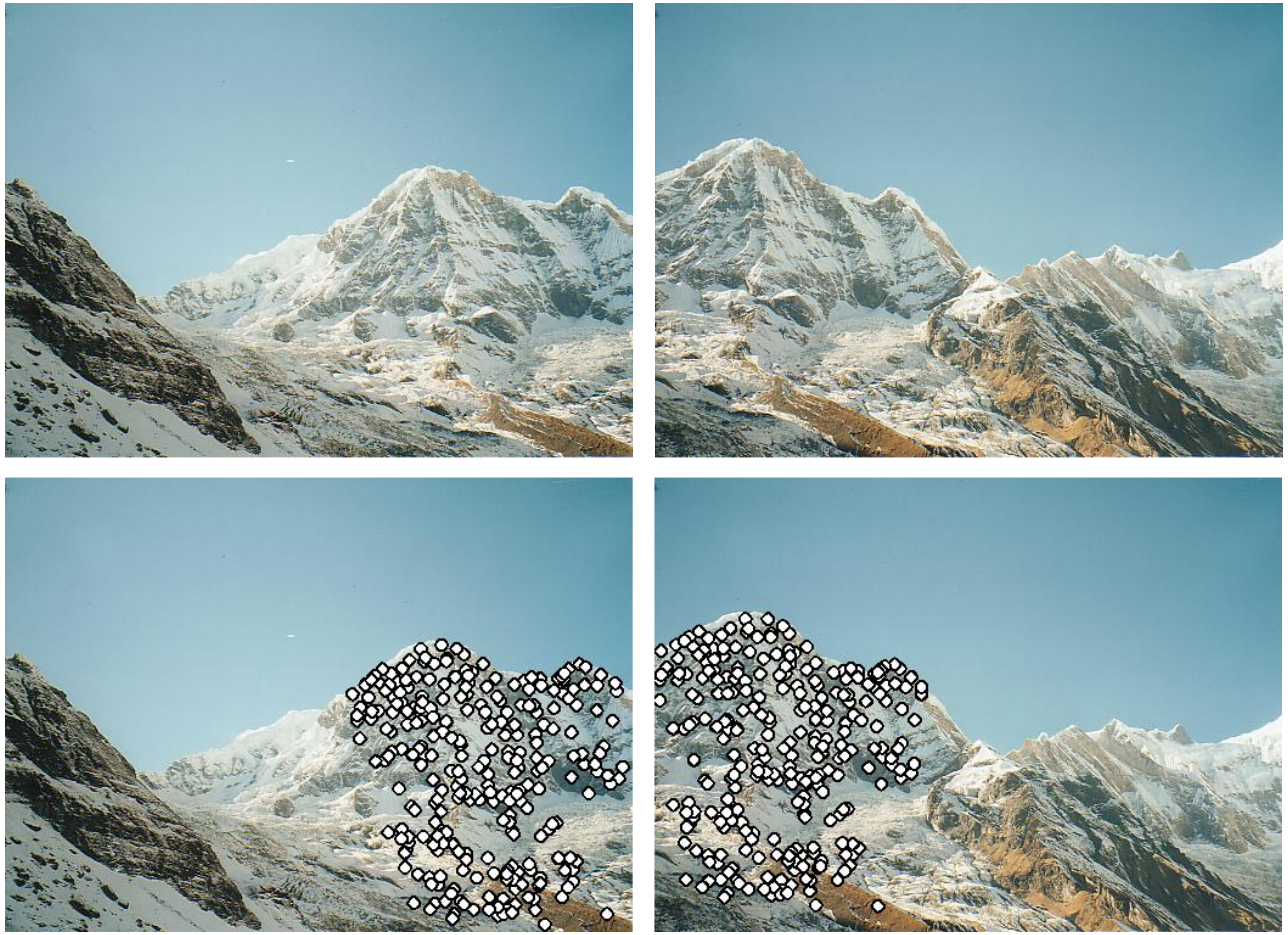
2-view **Rotation** Estimation

Find features + raw matches, use RANSAC to find Similarity



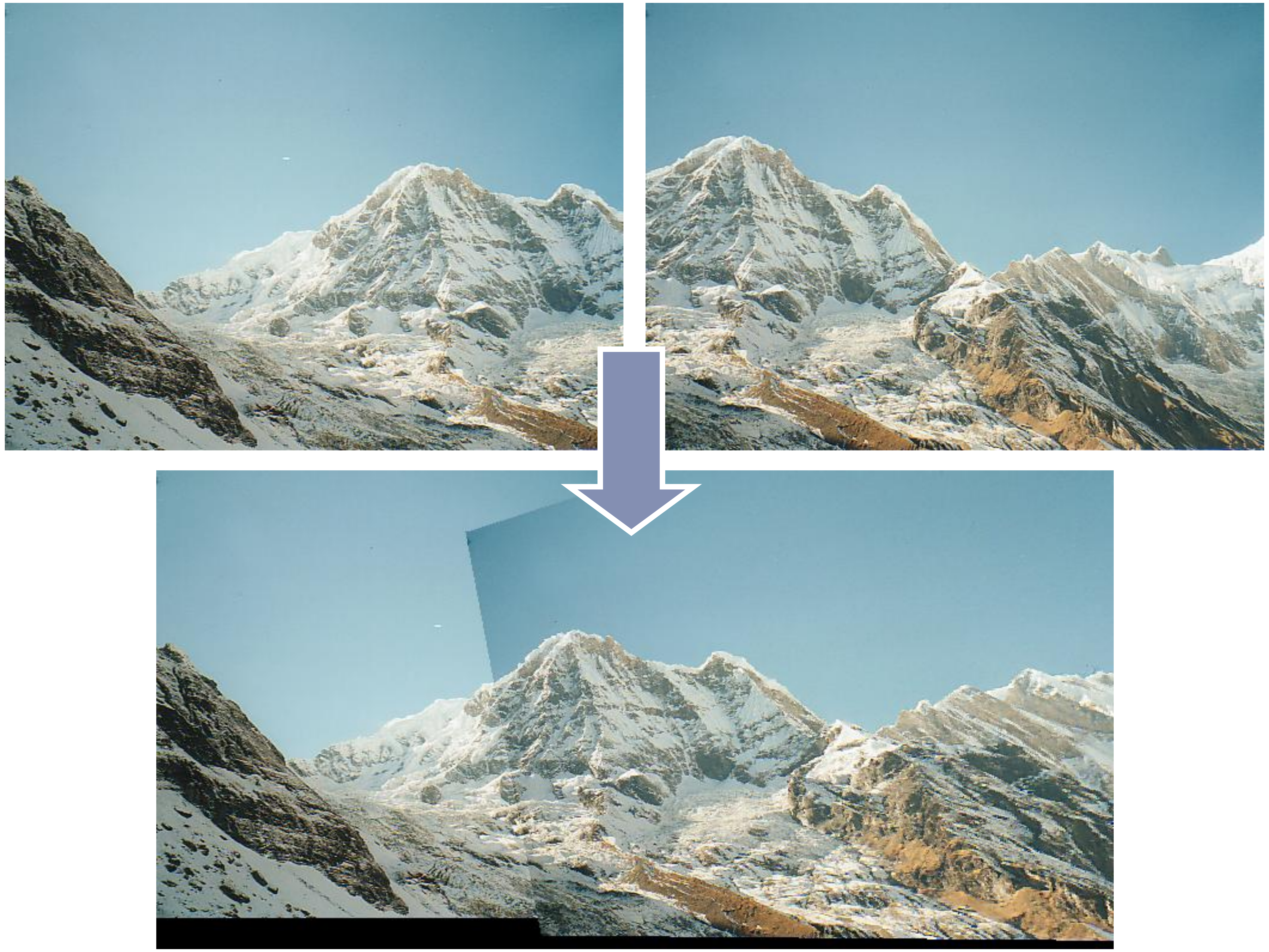
2-view **Rotation** Estimation

Remove outliers, can now solve for R using least squares



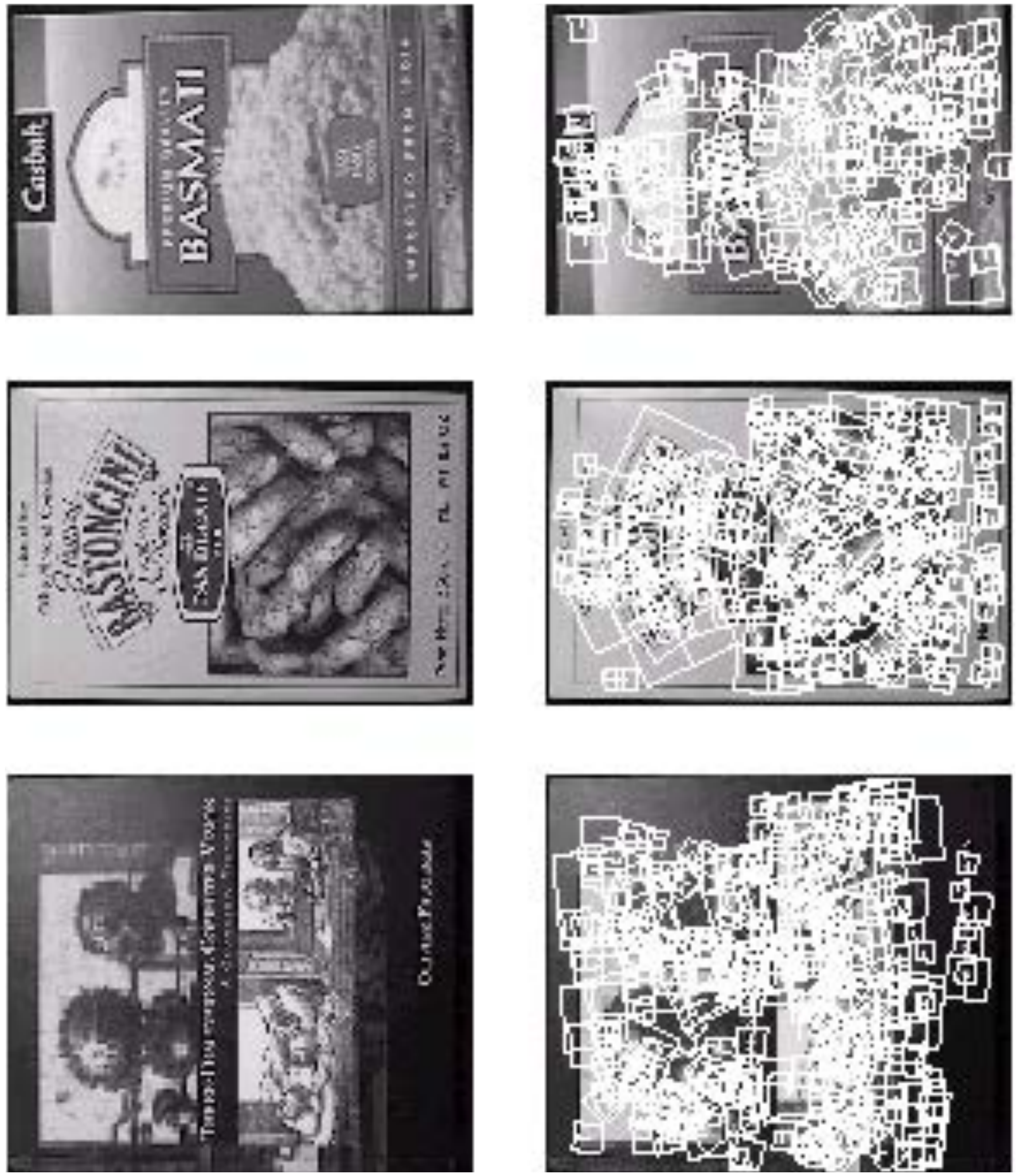
2-view **Rotation** Estimation

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: How many samples? (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

In practice…

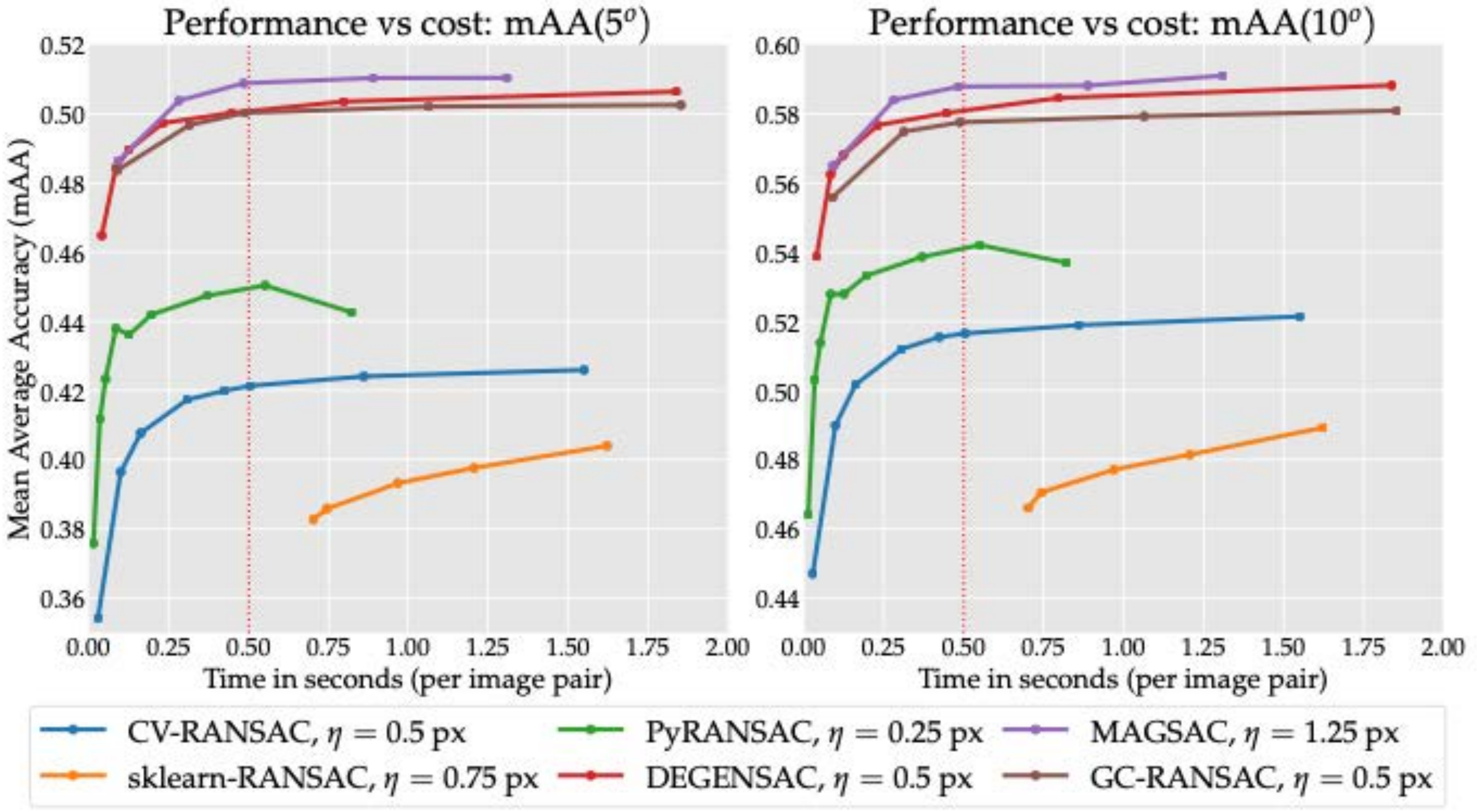


Fig. 9 Validation – Performance vs. cost for RANSAC. We evaluate six RANSAC variants, using 8k SIFT features with "both" matching and a ratio test threshold of r=0.8. The inlier threshold η and iterations limit Γ are variables – we plot only the best η for each method, for clarity, and set a budget of 0.5 seconds per image pair (dotted red line). For each RANSAC variant, we pick the largest Γ under this time "limit" and use it for all validation experiments. Computed on 'n1-standard-2' VMs on Google Compute (2 vCPUs, 7.5 GB).

[Jin et al., 2021]

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 3:** Due Wednesday!