HuStar AI Course: Computer Vision

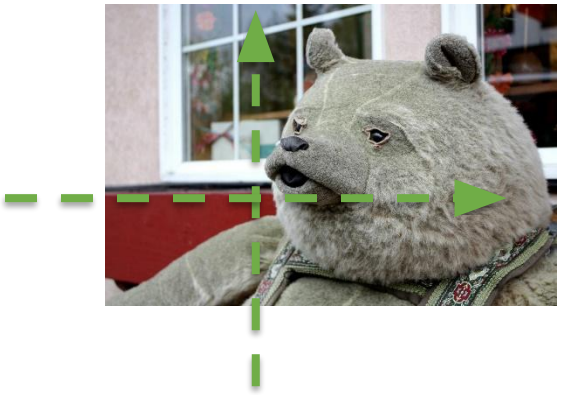
Image Warping and Alignment

Janghun Jo Geonung Kim

Computer Graphics Lab.

POSTECH

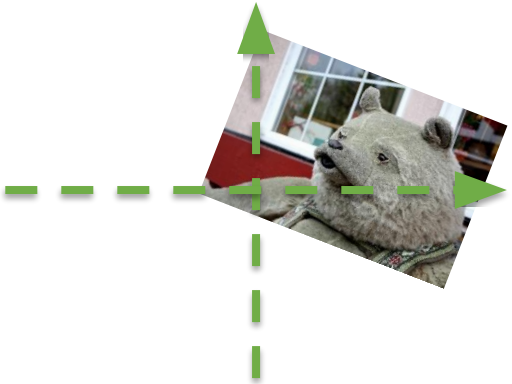
Parametric (global) warping



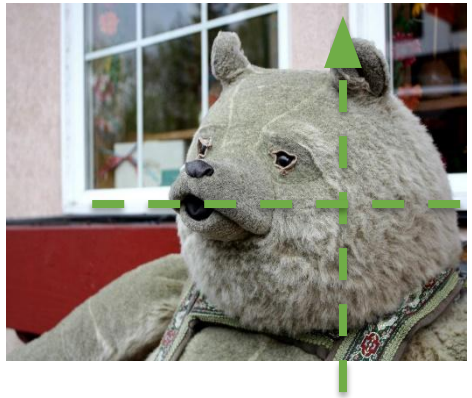
Translation

Rotation

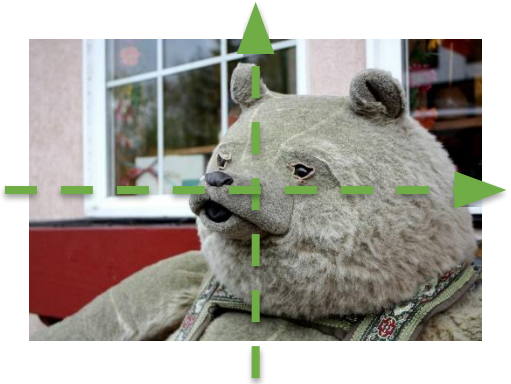
Rigid-body (a.k.a Euclidean)
(translation + rotation)



Similarity
(translation + rotation
+ scaling)



Affine
(preserve parallel
lines)



Perspective

Projective Transformation

- Translation

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

- Shear

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

- Scale

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

- Rotation

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

Affine Matrix Derivation

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

$$\begin{aligned} x' &= ax + by + e \\ y' &= cx + dy + f \end{aligned}$$

$$\begin{aligned} x'_1 &= ax_1 + by_1 + e \\ y'_1 &= cx_1 + dy_1 + f \\ x'_2 &= ax_2 + by_2 + e \\ y'_2 &= cx_2 + dy_2 + f \\ x'_3 &= ax_3 + by_3 + e \\ y'_3 &= cx_3 + dy_3 + f \end{aligned}$$

$$\begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ x_3 & y_3 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_3 & y_3 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}$$

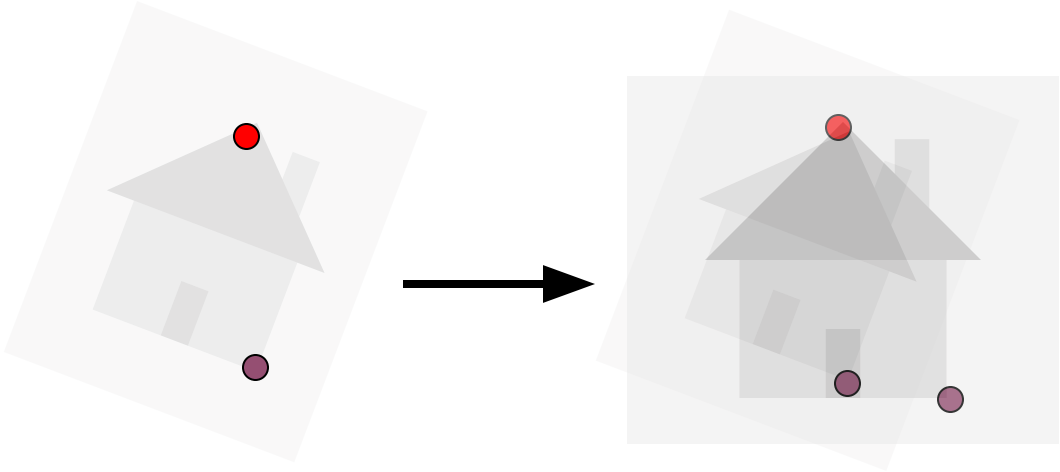
$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ x_3 & y_3 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_3 & y_3 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}$$

Projective Matrix Derivation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{12} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix}$$

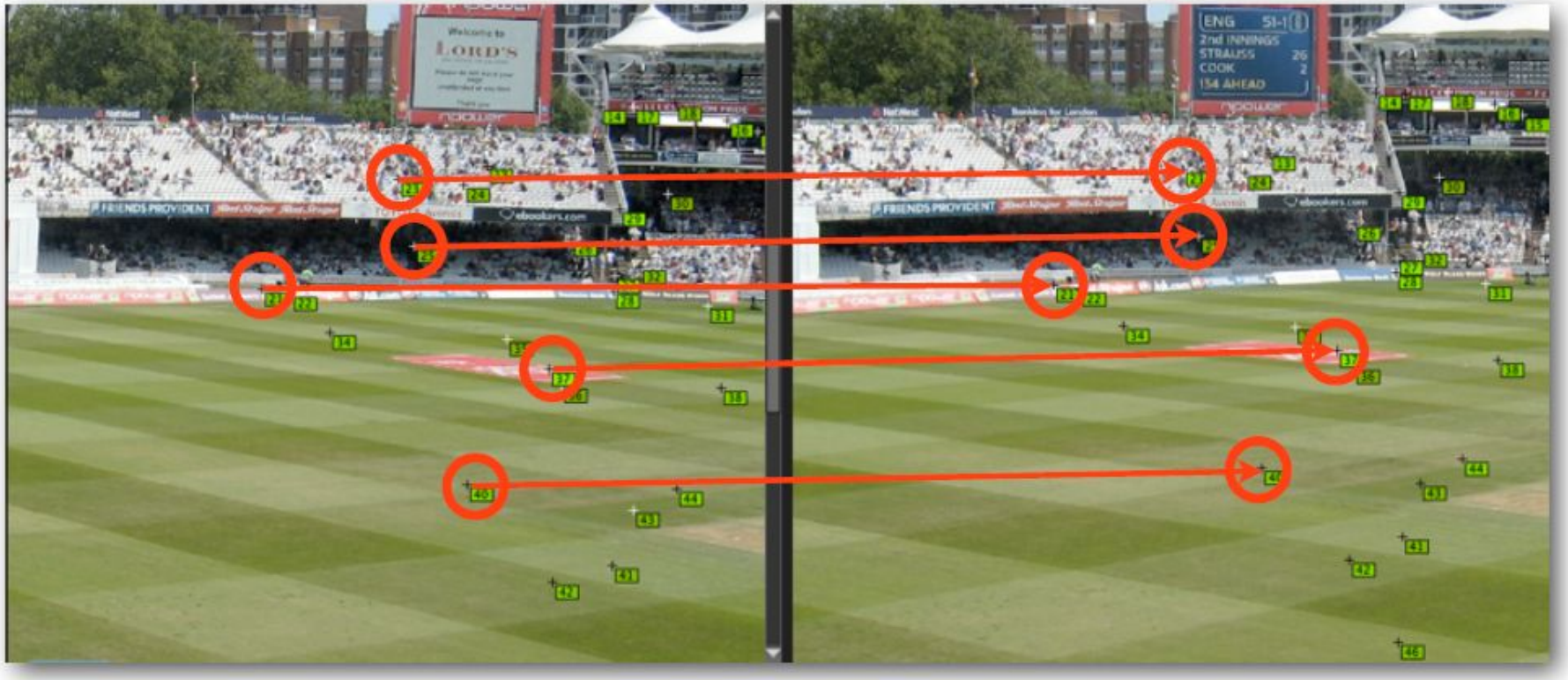
Image alignment



- How do we align two images automatically?
  - i.e., how to find **transformation parameters**?
- Two approaches:
  - Feature-based alignment
    - Find a few matching features in both images
    - compute alignment
  - Direct (pixel-based) alignment
    - Search for alignment where most pixels agree

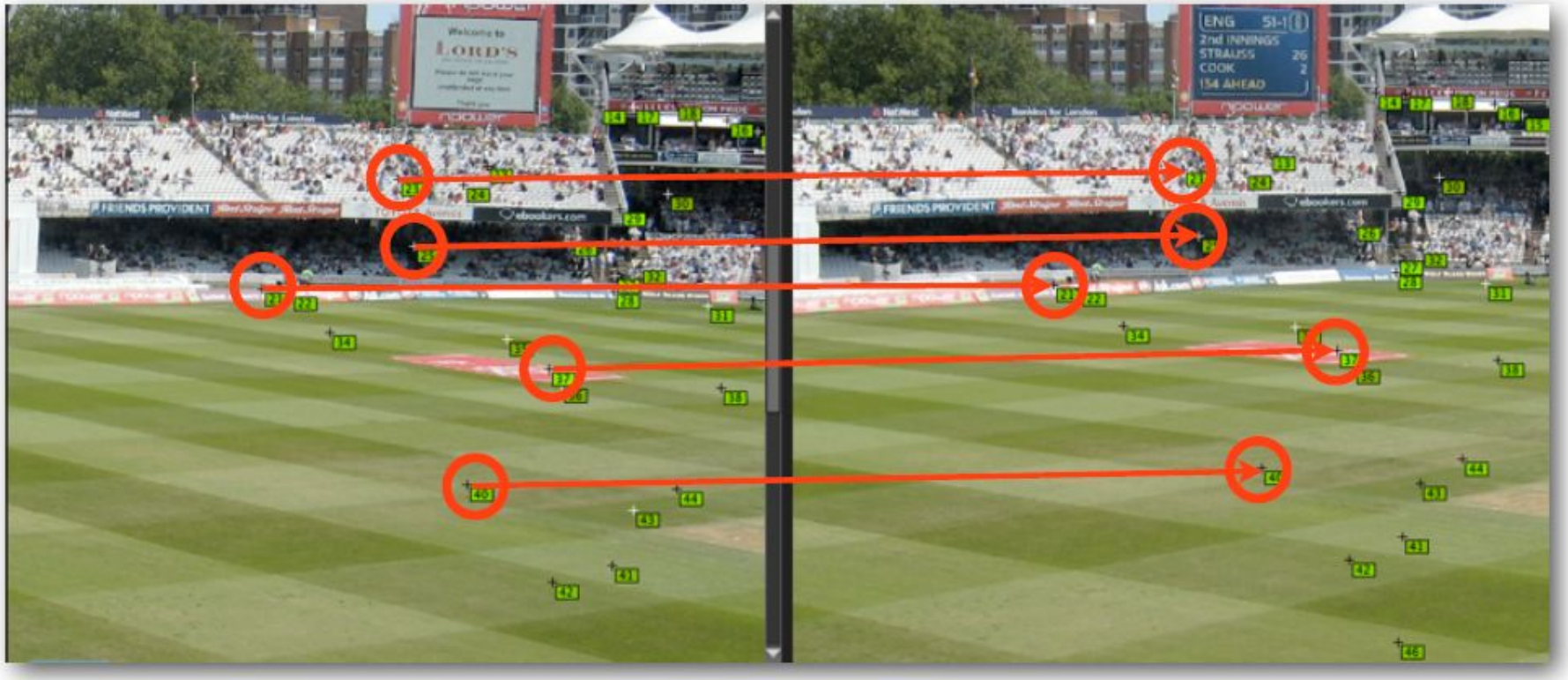
Feature-based alignment

- Procedure
  - Find a few important feature points (a.k.a interest points)
  - Match them across two images
  - Compute image transformation from the matches (e.g., homography)



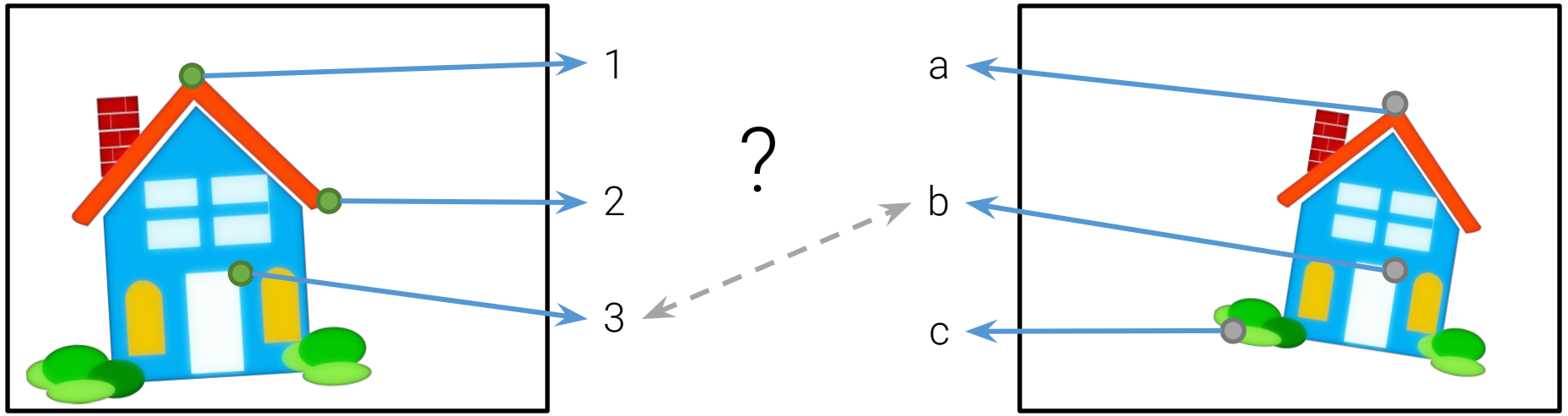
Feature detection

- Goal – Find points in an image that can be:
  - Found in other images
  - Found precisely – well localized
  - Found reliably – well matched



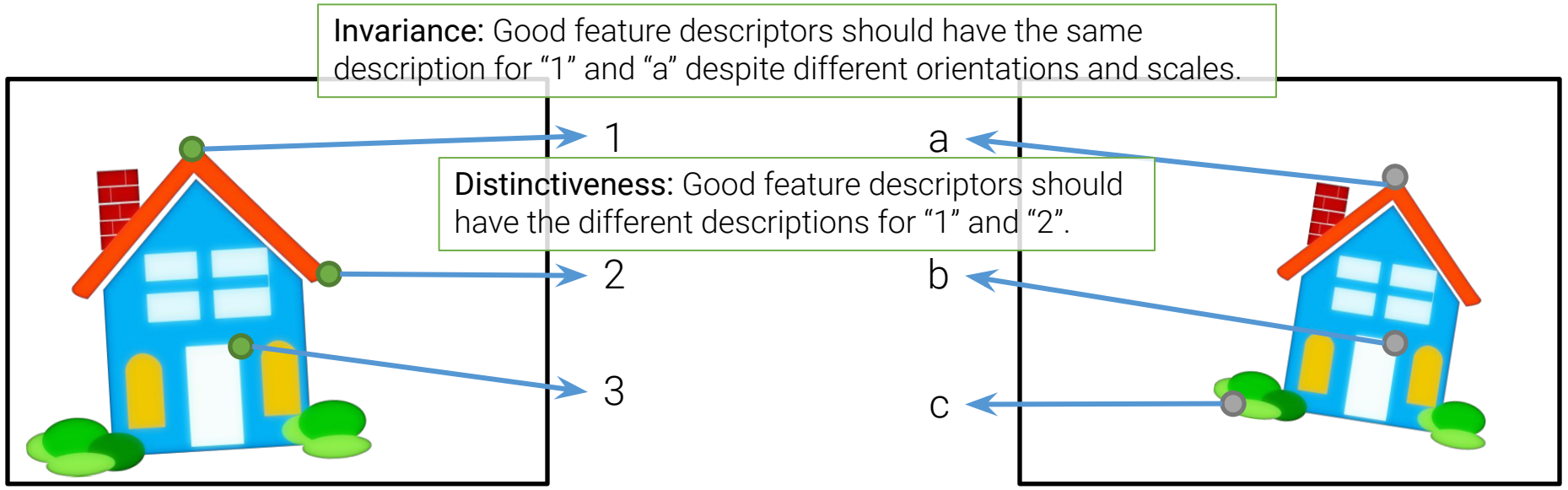
Feature matching

- We need a way to describe feature points to match them
  - A better way than using a single pixel color or a simple patch



Feature descriptors

- Feature descriptor describes what kind of feature the point is
  - E.g., a pixel color, a local patch
- "Good" feature descriptor should be:
  - **Invariant** to any possible transformations e.g., rotation, scaling, or brightness change
  - **Distinctive** - different features should be distinguishable

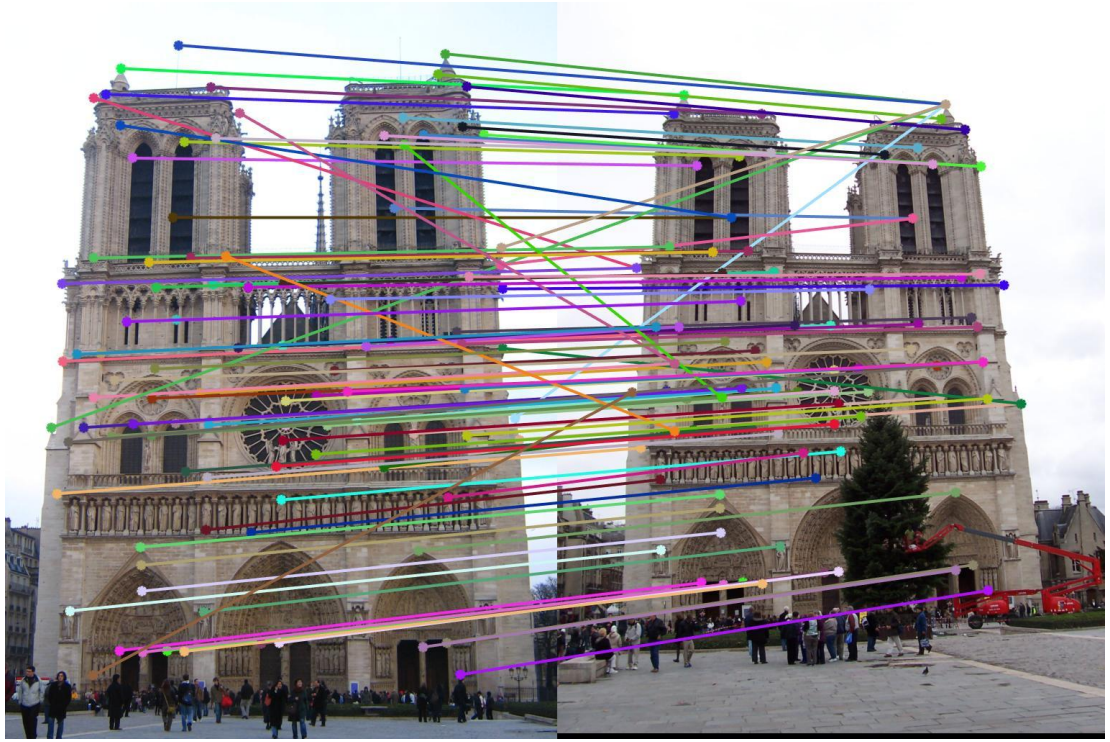


Feature matching

- We have matches now

- Can we estimate a transformation now?
  - No. Because of outliers (wrongly matched pairs)

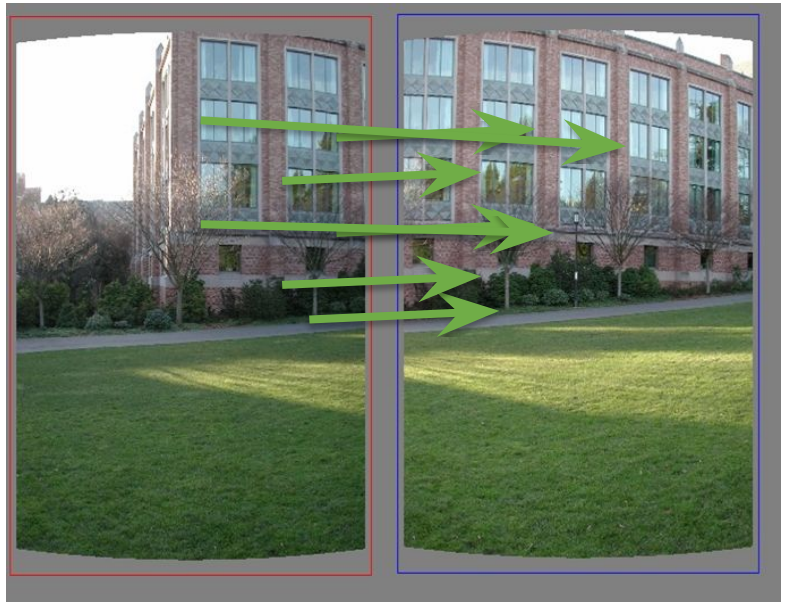
- We have to exclude outliers from estimating a transformation
  - **RAN**dom **SA**mple **C**onsensus
    - Most widely used outlier rejection method

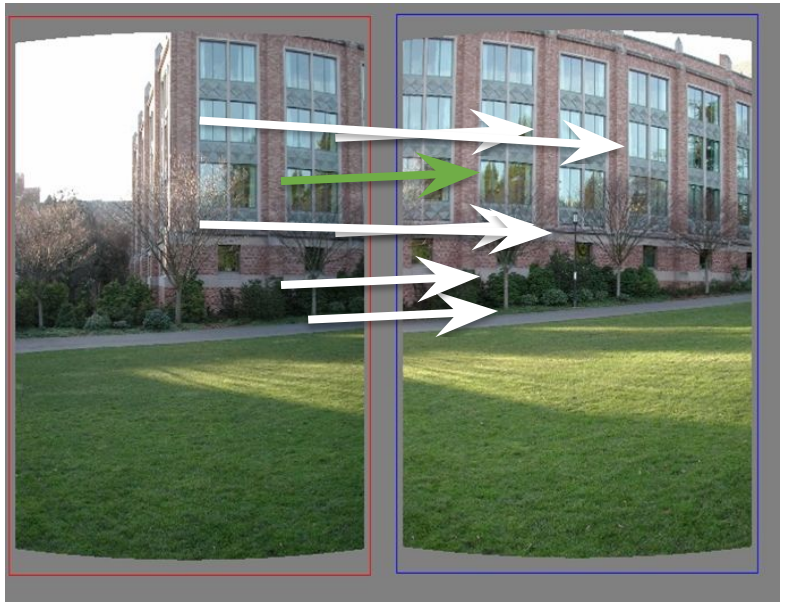


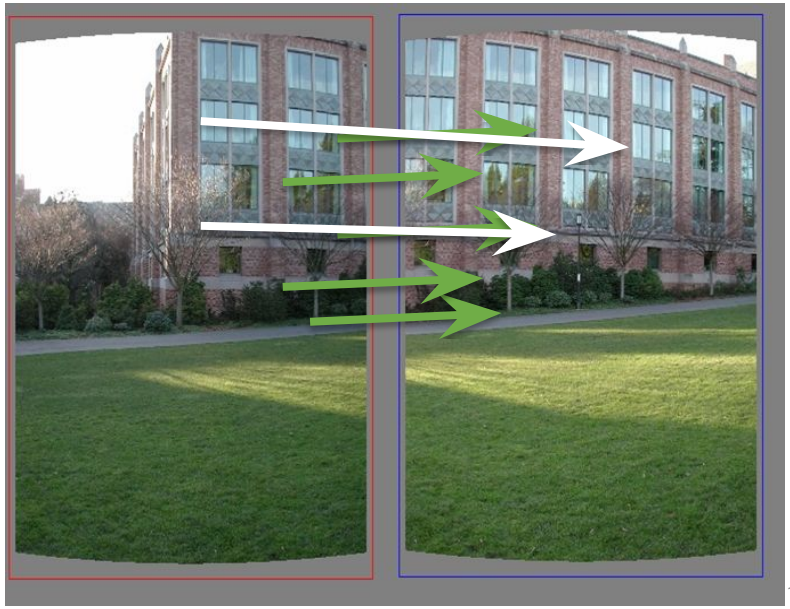
Outlier rejection

- RANSAC
  - Let's assume that we want to find translation that best matches two images

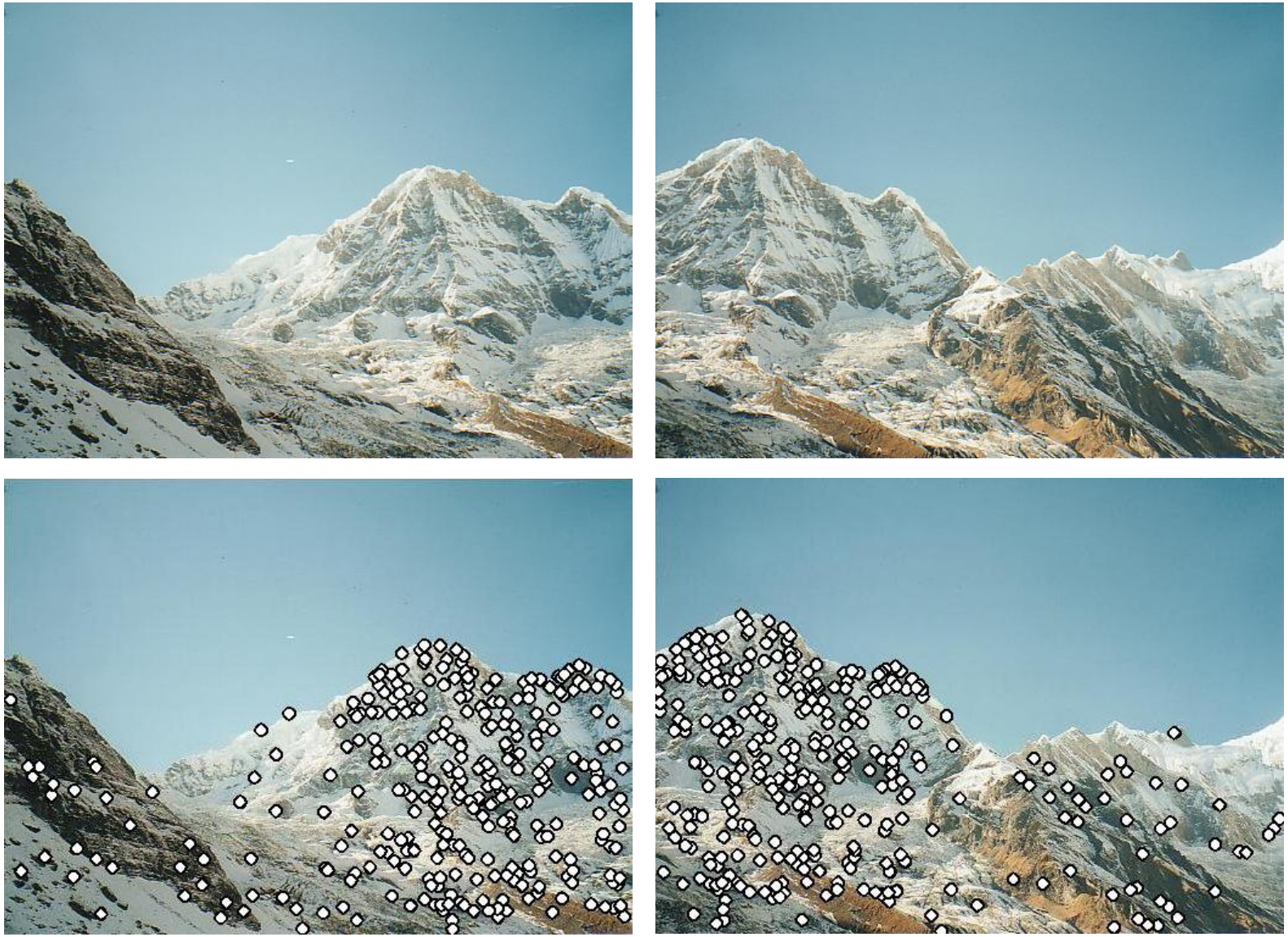
1. Randomly select one match
2. Compute the translation vector from the selected match
3. Count inliers that agree with the computed translation vector
   - In this case, there are 5 inliers
4. Repeat the process and keep the largest set of inliers
5. After N repetitions, compute the "average" translation vector from the largest set of inliers



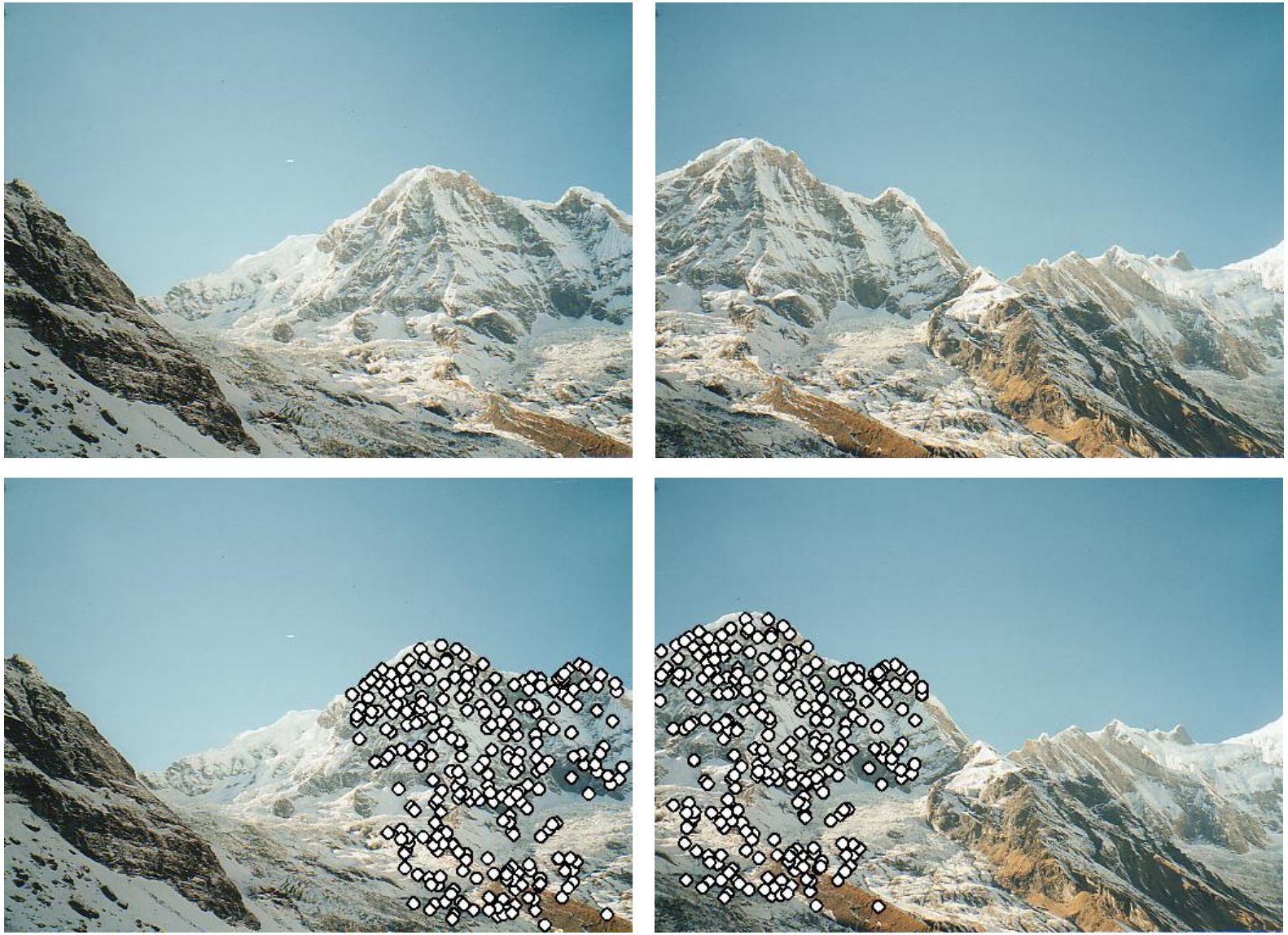




Example – feature detection



Example – feature matching & homography estimation using RANSAC



Example – align two images using estimated homography