

Homographies and Image Panoramas

Matching, RANSAC, Homographies



Antonio Carta

2026-10-01

University of Pisa

antonio.cart@unipi.it

Roadmap



Image Panoramas

PA FORTH PROJECT WP 1



Recap: Detectors and Descriptors

- **Keypoint Detectors**: finding repeatable keypoints and computing descriptors
- **SIFT**: scale-invariant keypoints + 128-dim gradient histogram descriptor
- Each keypoint carries a **location**, **scale**, **orientation**, and a **descriptor vector**

SIFT keypoints - image 1



Today: From Descriptors to Correspondences

- We have two sets of keypoints and descriptors: $\{(p_i, d_i)\}$ from image A , $\{(q_j, e_j)\}$ from image B
- **Matching**: find pairs (i, j) such that d_i and e_j describe the same scene point
- Good matches = **inliers**; wrong matches = **outliers** (mismatches)
- Good matches will then be used to estimate a rigid transformation between the two images

Large feature matches



Full Pipeline of Image Panoramas:

1. **Keypoint Detection and Description:** SIFT
2. **Descriptor matching:** nearest-neighbor search and Lowe's ratio test
3. **Robust estimation:** RANSAC framework
4. **Homographies:** definition and DLT estimation
5. **Warp**

Missing Match: Occlusions



Figure 7:

Bad Match: Repeated Structures

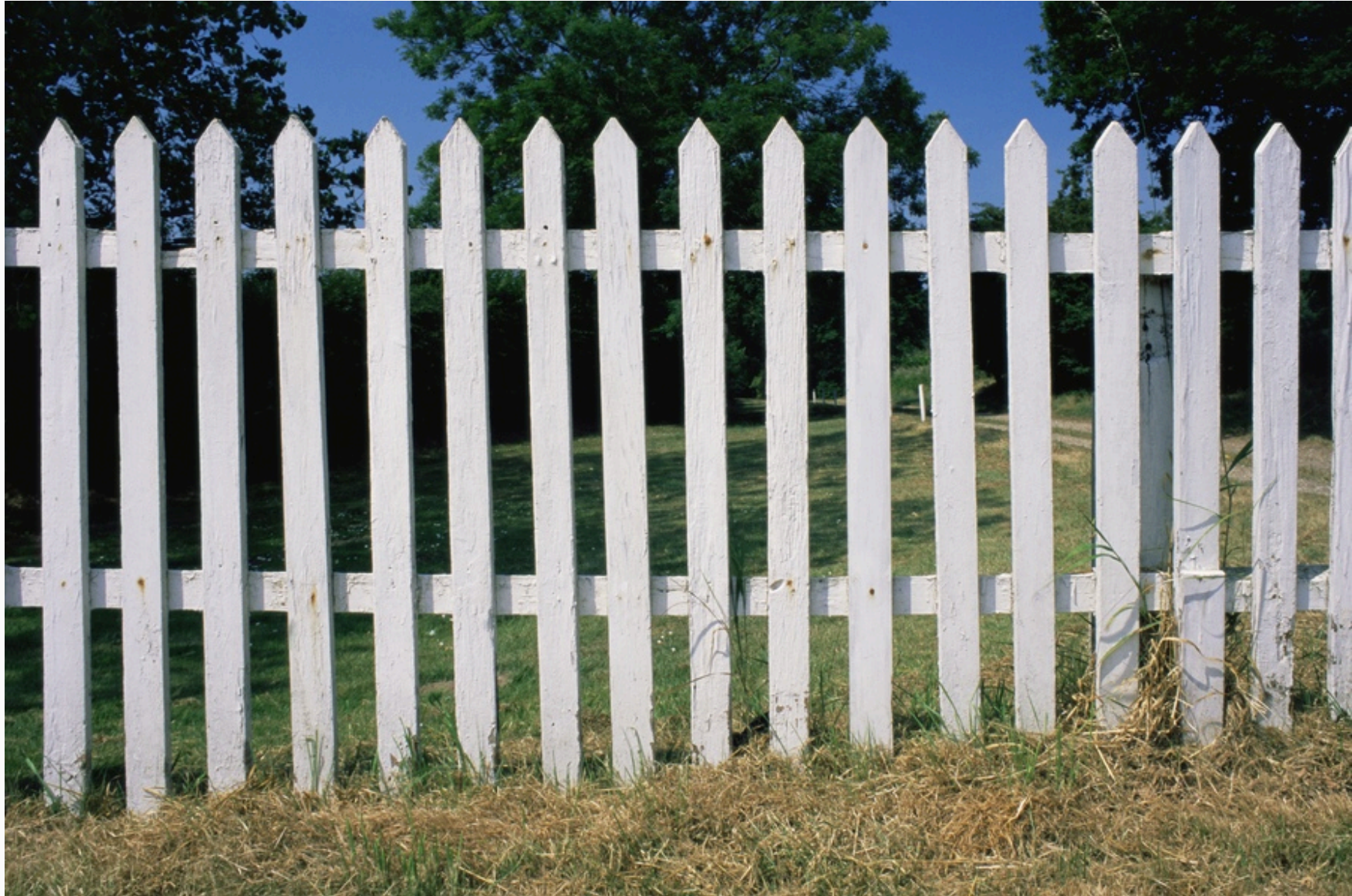


Figure 8:

Descriptor Matching

- We have two images A and B and have already extracted keypoints and descriptors
 - A has a set $\{(p_i, d_i)\}$, where p_i is the point and d_i its descriptor
 - B has a set $\{(q_j, e_j)\}$
- **Matching**: find pairs (i, j) such that d_i and e_j describe the same scene point
- find good matches = **inliers** and wrong matches = **outliers** (mismatches)

Solution: the closest descriptor is the correct match

for each descriptor d_i in image A , find its **nearest neighbor** in image B

$$j^* = \arg \min_j \| d_i - e_j \|_2$$

- Produces a set of candidate pairs (i, j^*) , one per keypoint in A

Problem: every descriptor gets matched, even if the true match is absent (e.g., occlusion) or the matching is poor

- **Euclidean distance** $\|d_i - e_j\|_2$: standard for floating-point descriptors (SIFT, learned)
- **Hamming distance** $\|d_i \oplus e_j\|_0$ (count of differing bits): for **binary descriptors** (ORB, BRIEF, BRISK)
 - Computed in a single XOR + popcount: extremely fast on modern CPUs
- Choice of metric must match the descriptor type

- Many regions look similar (windows, grass, brick walls): the nearest neighbor in descriptor space may still be a wrong match
- **IDEA**: a **good** match is close to d_i , and **not too close** to any other descriptor
- A **bad** match has a nearest neighbor that is nearly as close as the second-nearest

Intuition: if the nearest neighbor is distinctly closer than all other keypoints, it is likely the true match

- For each d_i , retrieve the **two nearest neighbors** e_{j_1} (best) and e_{j_2} (second-best)
- Accept the match only if:

$$\frac{\|d_i - e_{j_1}\|}{\|d_i - e_{j_2}\|} < \tau$$

- We use $\tau = 0.75$ in the demo notebook, which is a common choice

Choosing the Ratio Threshold

- precision: how many matches are correct among those found by the method
- recall: how many correct matches existing in the image pair are found by the method
- Smaller τ (e.g., 0.6): **higher precision**, fewer but more reliable matches
 - use when outlier tolerance is low
- Larger τ (e.g., 0.9): **higher recall**, more matches including more outliers
 - useful when the scene is sparse and we don't want to remove too many correspondences

In practice:

- tune τ to achieve the desired precision/recall trade-off
- a manual tune starting around $\tau = 0.75$ is feasible in several applications (e.g. image panoramas)
- dependent on the scene (how many keypoints we have, how many reliable), and the application
 - in videos, you can sample frequently and have very similar images
 - in image classification, A and B can be very different

- **Mutual NN (MNN)**: accept pair (i, j) only if i is the NN of e_j in A **and** j is the NN of d_i in B
- Filters out one-to-many matches (one point in B matched by many in A)
- Can be combined with the ratio test for additional precision
- Slightly reduces recall but substantially cuts false positives

Visual Inspection

- Visual inspection of all NN matches on a typical image pair reveals many obvious errors
 - Lines crossing dramatically across the image pair
 - Clusters of parallel lines (correct) mixed with scattered crossing lines (wrong)
- After ratio test: mostly parallel lines remaining, much cleaner correspondence set
- Residual outliers are handled by RANSAC in the next stage

Large feature matches



- we have a **robust set of matches: for each keypoint in A we know the corresponding keypoint in B**
- it's not a perfect correspondence: next steps need to be robust to noise and errors
- the amount of noise depends on the application

Homographies



Image Panoramas: stitch together two images taken by a *fixed photographer*

Problem: the images are from the same scene, but they have different coordinates

- to stitch the two images together we need to **convert coordinates from A to B**
- when taking image B, the photographer (and camera) stays in the same place, while the camera is rotated compared to image A

Viewpoint as a Change in coordinates

when taking image B, the photographer (and camera) stays in the same place, while the camera is rotated compared to image A

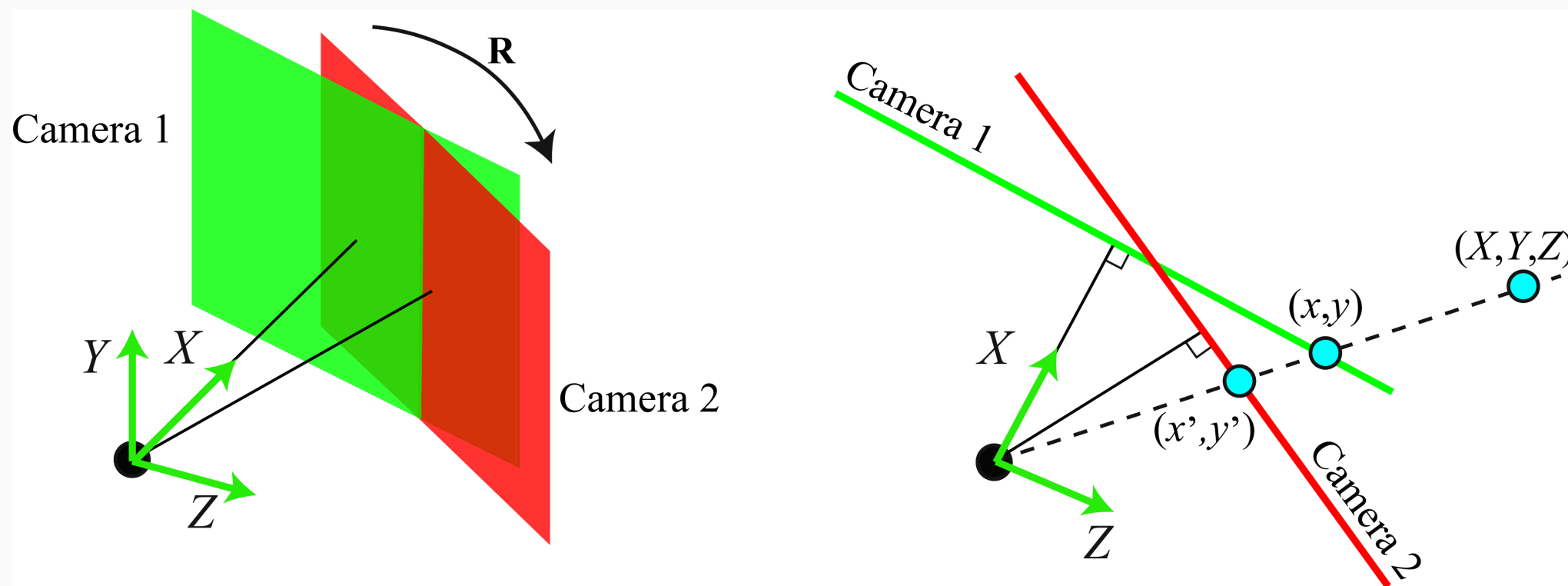


Figure 10: source: visionbook

Homography

- **notation**: we use homogeneous coordinates: ordinary points $\begin{pmatrix} x \\ y \end{pmatrix}$ represented as homogeneous triplets $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Definition: A homography (or projective transformation) is a geometric transformation that preserves straight lines

- in **image panoramas** we stitch together images related by a homography
- given an homography $p' = h(p)$
- if p_1, p_2 are colinear, then $h(p_1), h(p_2)$ are also colinear
- for our purposes, a homography will always be a transformation from a 2D plane to a 2D plane (i.e. we don't need to care about 3D coordinates yet)

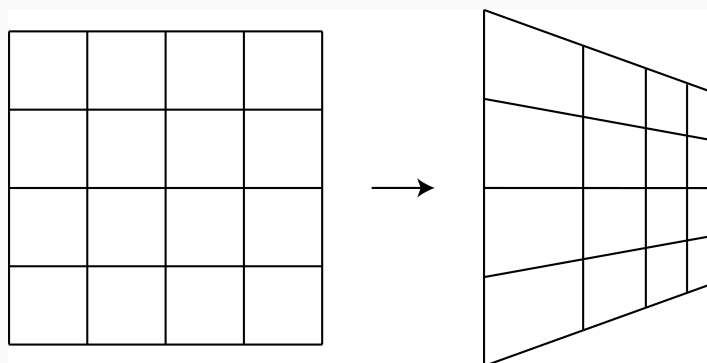


Figure 11: source: visionbook

Example

When the scene is planar you also get a homography.



Figure 12: Homographies preserve straight line. Source: visionbook

- Given H and a point $p = (x, y)$:
 1. Convert to homogeneous: $\tilde{p} = (x, y, 1)^T$
 2. Apply the homography: $\tilde{q} = H\tilde{p}$
 3. Dehomogenize: $q = \left(\frac{\tilde{q}_1}{\tilde{q}_3}, \frac{\tilde{q}_2}{\tilde{q}_3} \right)$

The Homography Matrix H

- A homography is represented by a 3×3 matrix H (in homogeneous coordinates):

$$\tilde{q} = H\tilde{p}, \quad H \in \mathbb{R}^{3 \times 3}$$

- due to homogeneous coordinates, H is defined **up to scale** (nonzero scalar multiple):

$$H \equiv \lambda H$$

- In Cartesian coordinates:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{1}{h_{31}x + h_{32}y + h_{33}} \begin{pmatrix} h_{11}x + h_{12}y + h_{13} \\ h_{21}x + h_{22}y + h_{23} \end{pmatrix}$$

- Because H is defined up to scale, it has **8 degrees of freedom** (9 entries minus 1 for scale)
- **note**: we will explain the scale-invariance and the heterogeneous-to-homogeneous conversion in the next lecture

- H has 8 DoF $\rightarrow$ need at least **4 point correspondences** (each gives 2 equations) to solve for H
- With exactly 4 (general position), the solution is unique
- With $n > 4$, the system is overdetermined and we solve in a least-squares sense (DLT)

Homographies:

- **Planar scene:** any two views of a flat surface are related by a homography (regardless of camera motion)
- **Pure camera rotation:** for a rotating camera (no translation), any scene is related by a homography
- A homography is a 3×3 matrix. Next step is to estimate it from the data

Not everything is a homography: non-planar scenes (most 3D scenes) + camera translation breaks the homography model

Estimating a Homography: DLT

- we have set of keypoint correspondences between A and B
- we have a model that tells us how to convert from coordinates in A to B (homography)

Next: fit the parameters of the homography using the correspondences with the Direct Linear Transform (DLT)

DLT: given a set of correspondences (at least 4) we obtain an estimate of the homography matrix H

we will see DLT in detail in the next lectures. Today, we assume that DLT provides us with an estimate of H that is consistent with the data.

Outliers and Robust Estimation

- **Repetitive patterns**: many identical-looking regions (windows, tiles). NN is ambiguous
- **Occlusions**: a point visible in A has no match in B , forcing an incorrect match
- **Illumination/viewpoint change**: descriptor may change enough that the true match is not the NN
- **Background clutter**: distractors with accidentally similar descriptors

Given a rigid geometric transformation from coordinates in A to coordinates in B (e.g. a homography):

- **Inlier**: a match (p_i, q_j) that is consistent with the geometric transformation
- **Outlier**: a match that violates the geometric transformation

We don't know the outliers because we don't have the ground truth.

In typical real-world matching pipelines, even after the ratio test, you will get outliers

Problem: we optimize a least squares objective.

- a least-squares objective penalizes $\sum_i r_i^2$ where r_i is the residual error for match i (i.e. how far the point predicted by the estimated homography is from the actual point)
- a single large residual (outlier) can dominate the sum of residuals
- even with a single outlier, we can shift the prediction as much as we want by making the outlier bigger

Conclusion: least squares is not robust to outliers

Example with 1D regression

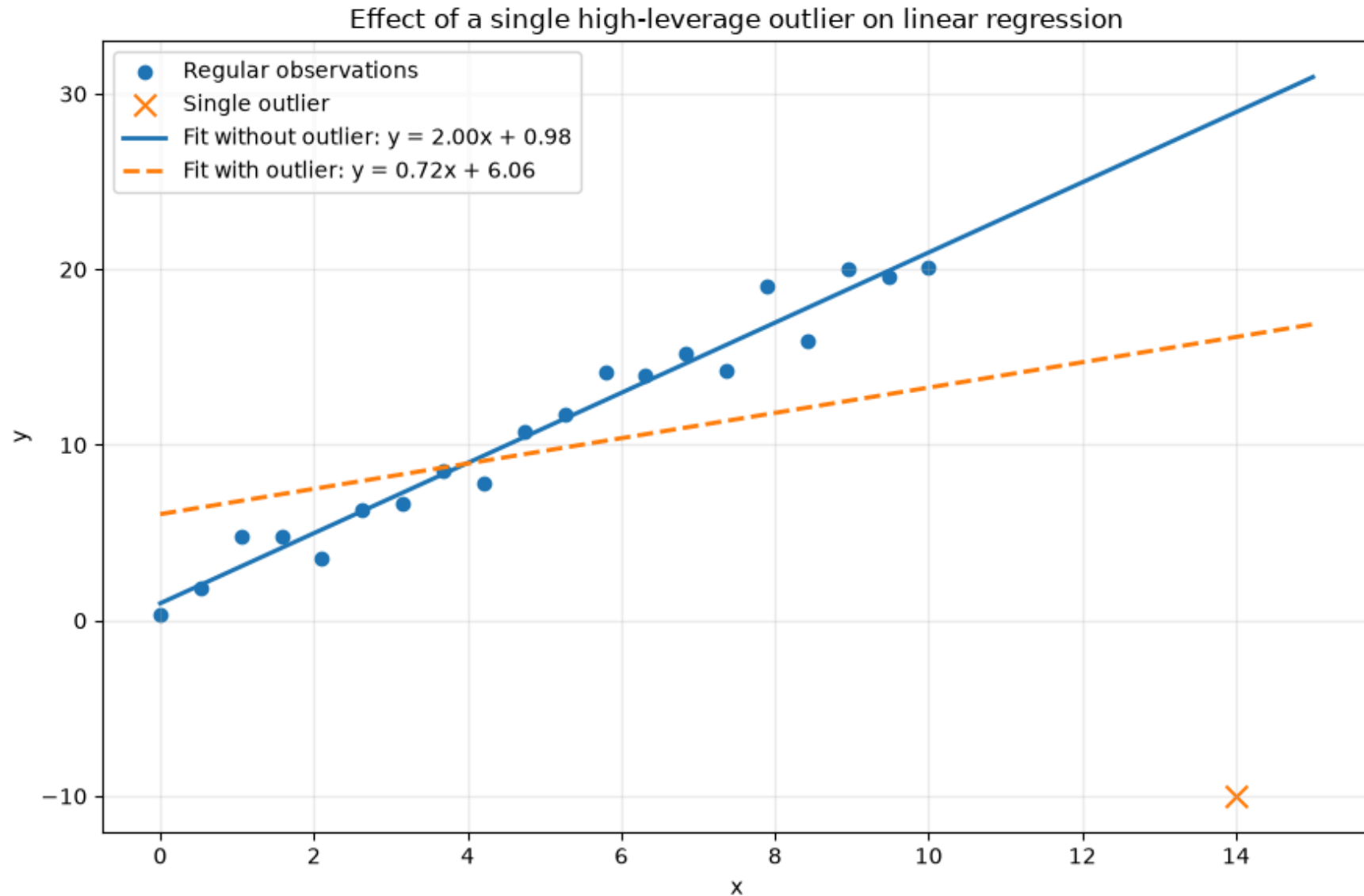


Figure 13: Even a single outlier can completely change the result.

- outliers are few but have outsized impact
- **Robust estimators** gain robustness to outliers by limiting the influence of individual points
- **RANSAC** (Random Sample Consensus): find the largest set consistent with a model
 - a very popular component in computer vision pipelines

RANSAC



- We want to estimate H from these noisy correspondences
- DLT (least squares) is corrupted by even a few outliers
- **RANSAC**: estimate multiple homographies and find the one supported by the **largest consensus set** (inliers), ignoring outliers entirely

repeat N times:

1. Sample: draw the minimum number of correspondences (4 for H) uniformly at random
 2. Fit: compute H from the samples using DLT
 3. Score: count inliers matches (p_i, q_i) with error $< t$ (threshold)
 4. Update: keep H if current-inlier-count $>$ best-so-far
- return H^* with most inliers; re-estimate H^* using all inliers (DLT)

- **Symmetric transfer error:** $d(q, Hp)^2 + d(p, H^{-1}q)^2$
- Or **one-way transfer error:** $d(q, Hp)^2$ (project p using H , compare to q)
- A match is an **inlier** if the transfer error is below threshold t
- Threshold t in pixels (e.g., $t = 3-5$ px): chosen based on expected noise level
 - 4.0 in the notebook

Example

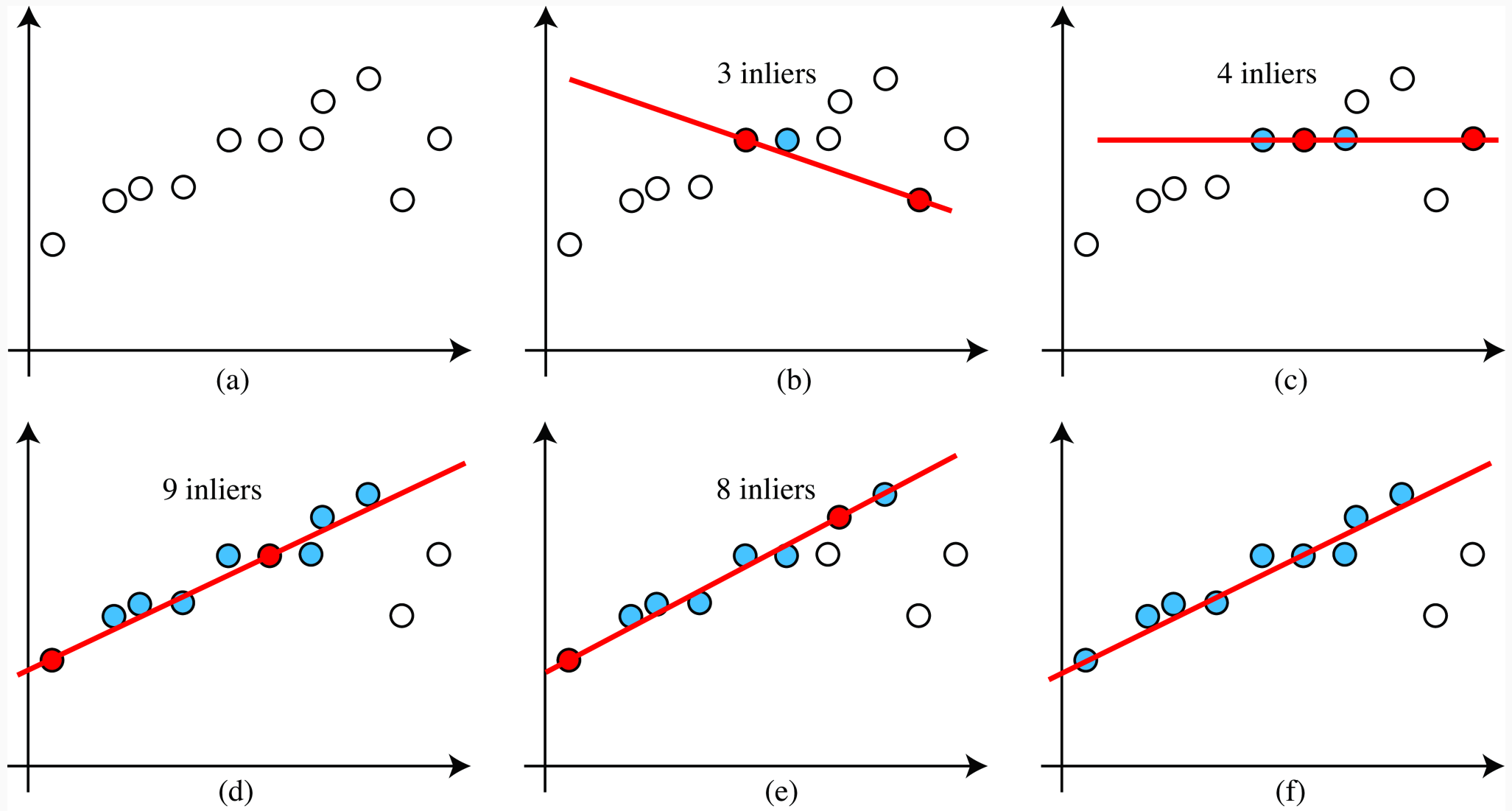


Figure 14: source: visionbook

- given w ratio of inliers in the data and n samples for the training (4 in our case)
 - w^n is the probability that all points are inliers
 - $1 - w^n$ is the probability to have at least one outlier
 - with k trials, $(1 - w^n)^k$ is the probability that all the trials have at least one outlier
- given p the probability that all points are inliers, we have $1 - p = (1 - w^n)^k$

Taking the logarithm we solve for k :

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

Conclusion: given an estimate of the ratio of outliers in the data, you can find how many trials you need to get a desired probability p to obtain the correct model (all inliers used for the estimate).

- **Threshold t :**
 - too small: true inliers rejected
 - too large: outliers accepted
- **Iteration count N :** more iterations = higher confidence, but slower
- **Outlier rate $1 - w$:** RANSAC works well for $1 - w < 50\%$; very high outlier rates require many iterations or more sophisticated variants
- **Final refinement:** always re-estimate H from all inliers after RANSAC for best accuracy

Learned Matching

- Ratio test + RANSAC works well but fails in:
 - **Extreme viewpoint changes** ($>60^\circ$): descriptors become unrecognizable
 - **Low-texture regions**: too few keypoints detected
 - **Repetitive patterns**: ratio test cannot disambiguate many similar descriptors
 - **Dark/noisy images**: low SNR degrades both detection and description

SuperGlue: Attention-Based Matching

- **SuperGlue** (Sarlin et al., 2020): treats matching as a **graph assignment problem**
- Keypoints and descriptors from both images are nodes in a cross-image graph
- **Self-attention** refines each descriptor using context from its own image
- **Cross-attention** propagates information across the two images
- Dramatically outperforms classical matching on challenging scenes (indoors, low light)

Advantages:

- learned methods are trained on large set of image pairs
- they can use more contextual information more effectively (descriptor of the other keypoints through attention)

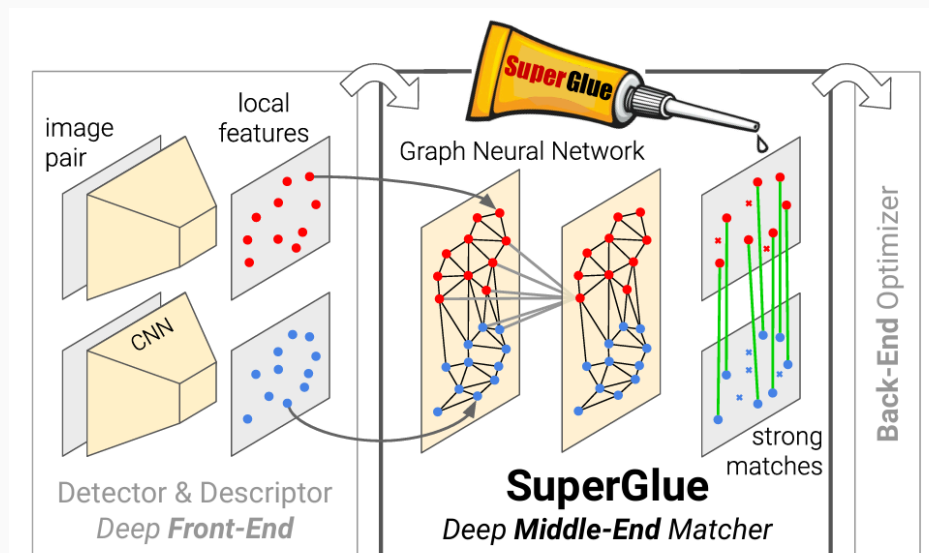


Figure 1: **Feature matching with SuperGlue.** Our approach establishes pointwise correspondences from off-the-shelf local features: it acts as a middle-end between hand-crafted or learned front-end and back-end. SuperGlue uses a graph neural network and attention to solve an assignment optimization problem, and handles partial point visibility and occlusion elegantly, producing a partial assignment.

Figure 15: source: 'SuperGlue: Learning Feature Matching with Graph Neural Networks'.

LightGlue: Efficient Attention Matching

- **LightGlue** (Lindenberger et al., 2023): redesigns SuperGlue for **efficiency**
- Adaptive depth: exits early for easy pairs (fewer transformer layers)
- Confidence-based pruning: removes low-confidence keypoints during inference
- Runs 2–10× faster than SuperGlue with comparable or better accuracy
- Becoming the de-facto learned matcher for real-time applications
- <https://github.com/cvg/lightglue>



Figure 16: source: LightGlue github.

Full Pipeline



detect $\rightarrow$ describe $\rightarrow$ match $\rightarrow$ RANSAC $\rightarrow$ DLT $\rightarrow$ warp

- Input: two images I_A, I_B of the same scene (planar or rotation-only)
- Output: a warped image $I_{B'}$ aligned to I_A 's coordinate frame

- **Inverse warping**: for each pixel (x', y') in the output, compute $(x, y) = H^{-1}(x', y')$ and sample I_B
 - Avoids holes (forward warping can leave unmapped pixels)
 - Use bilinear interpolation for sub-pixel accuracy
 - we will discuss more in detail inverse and forward warping in the next lectures
- Compose I_A and the warped $I_{B'}$ by blending (e.g. using the multi-scale image blending seen in the multi-scale lecture)

Conclusions

- DLT fails if points are collinear: H is not uniquely determined
- Homography is valid only for planar scenes or pure camera rotation
- It is possible to apply the same pipeline with a different transform model
 - example: OpenCV Stitcher supports homographies (for image panoramas) and affine (e.g. for scans) https://docs.opencv.org/4.13.0/d8/d19/tutorial_stitcher.html

- **Ratio test**: removes ambiguous matches by comparing best vs. second-best NN
- **Homography** H : encodes planar projective transform; estimated by DLT
- **RANSAC**: robust estimation by random sampling
- **Full pipeline**: detect $\rightarrow$ describe $\rightarrow$ match $\rightarrow$ RANSAC/DLT $\rightarrow$ warp
 - a similar pipeline is used in many other applications

- Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, IJCV 2004 for ratio test and SIFT matching
- visionbook Ch. 41