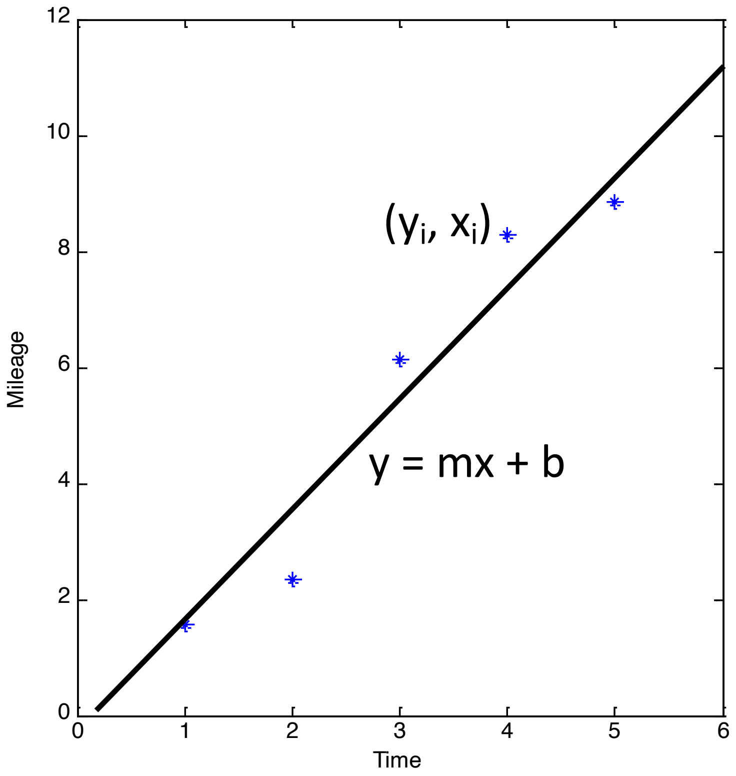
Robust Estimation w/ RANSAC

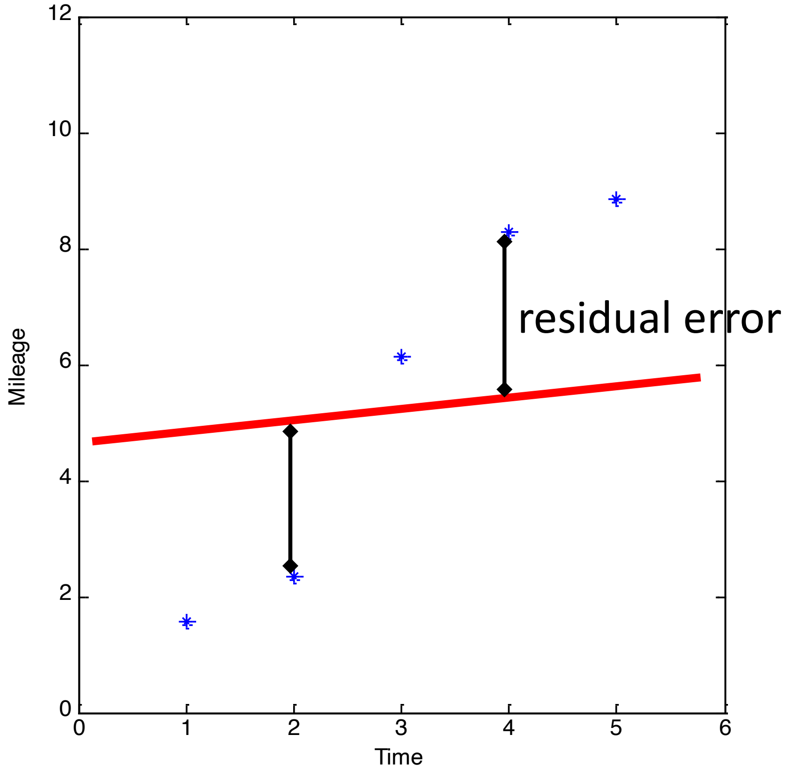
Dealing with outliers

- Estimating E relies on correspondences
- What if correspondences are incorrect?
- Fitting: find the parameters of a model that best fit the data
- Other examples:
  - least squares linear regression

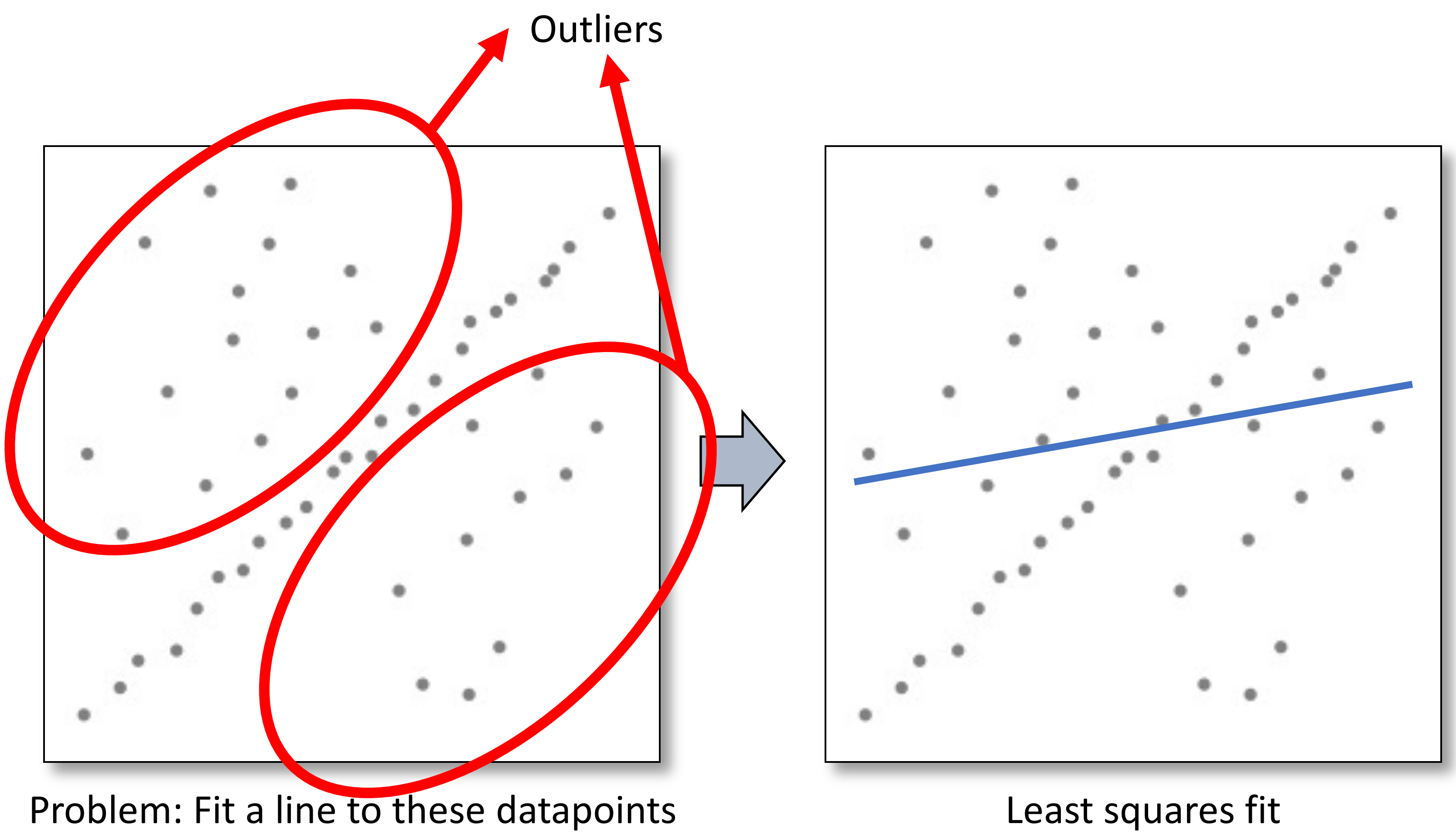
Example: Fitting lines



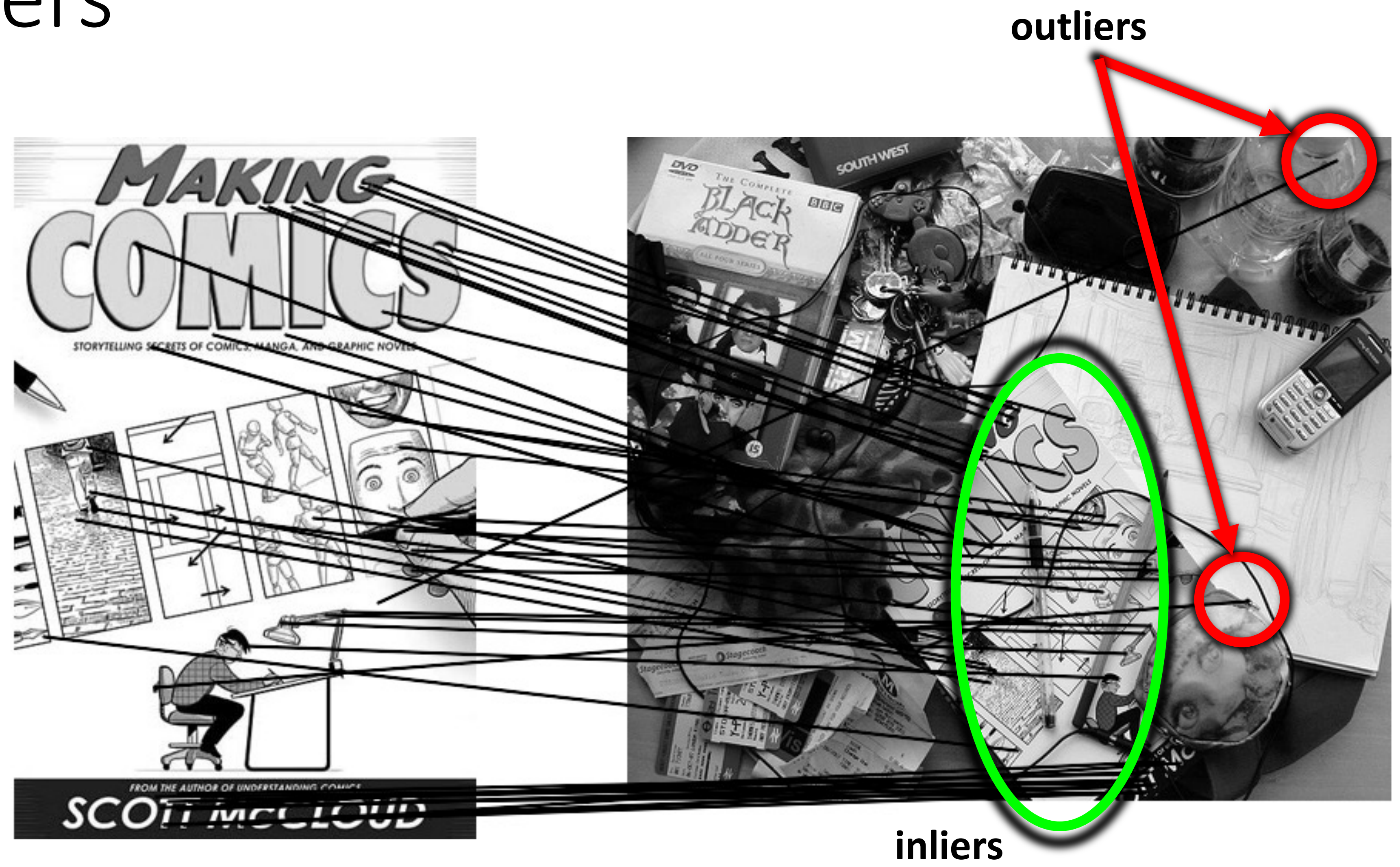
Linear regression



Outliers in linear regression



Outliers



Outliers

- Grossly incorrect
- Dominate objective
- Lead to incorrect solutions
- Must be eliminated
- But how do we know which data points are outliers?

More general problem setup

- Given
  - A ***noisy*** dataset $D = \{ p_1, p_2, \ldots, p_N\}$ with some ***completely incorrect*** outliers
    - Example 1: Line fitting: $\{ (x_1, y_1), \ldots, (x_n, y_n)\}$
    - Example 2: Fundamental matrix: $\{ (\vec{p}_1, \overrightarrow{q_1}), (\overrightarrow{p_2}, \overrightarrow{q_2}), \ldots, (\overrightarrow{p_N}, \overrightarrow{q_N})\}$
  - A set of parameters θ that need to be fitted
    - Line fitting: $\theta = (m, b)$
    - F estimation $\theta = F$, $||f|| = 1$
  - A cost function $C(p, \theta)$
    - Line fitting: $C\big((x, y), (m, b)\big) = ||y - (mx + b)||^2$
    - F estimation: $C\big((\vec{p}, \vec{q}), F\big) = \overrightarrow{p^T} F \vec{q}$(Reprojection error)
- Find θ

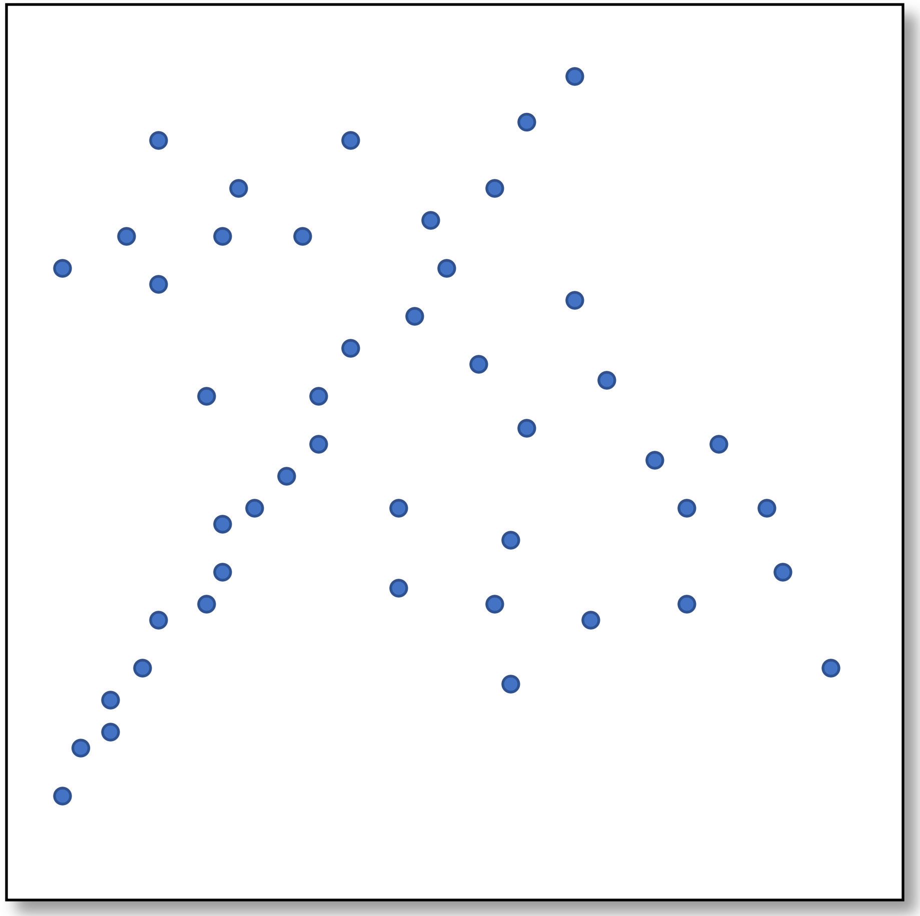
Anna Karenina principle

- “Happy families are all alike; every unhappy family is unhappy in its own way.” – Leo Tolstoy, *Anna Karenina*
- Inliers *bound to agree with each other*
- Outliers are all outliers in different ways
    - *So assume outliers will not all point towards same hypothesis*
- More precise assumption:
    - Outliers either <50%
    - Or noisy points don’t all agree

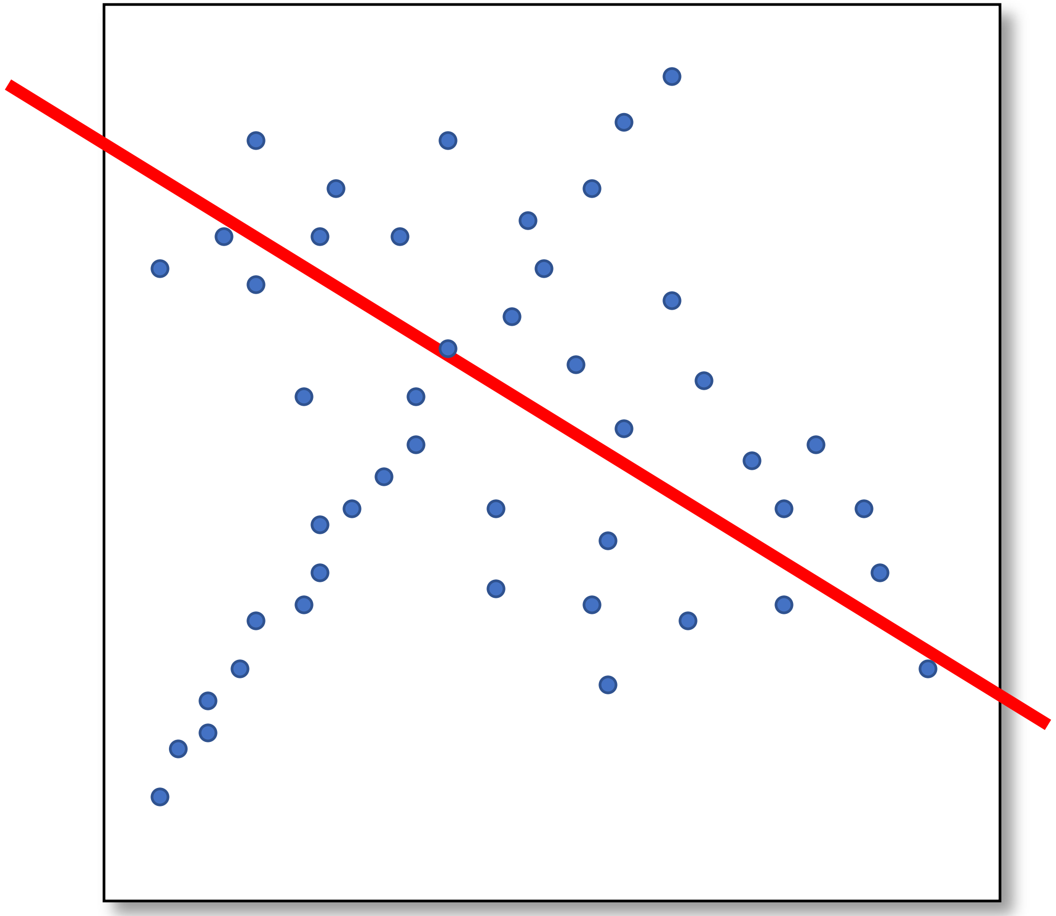
Approach

- Search through all possible hypotheses
  - E.g., all possible lines
- For every point count number of potential *inliers*
  - Points that agree with the line
- Find line with maximum # of inliers
  - Since outliers don't agree with each other, they won't all lie on the same line
  - So the points on this line must be true inliers

Counting inliers

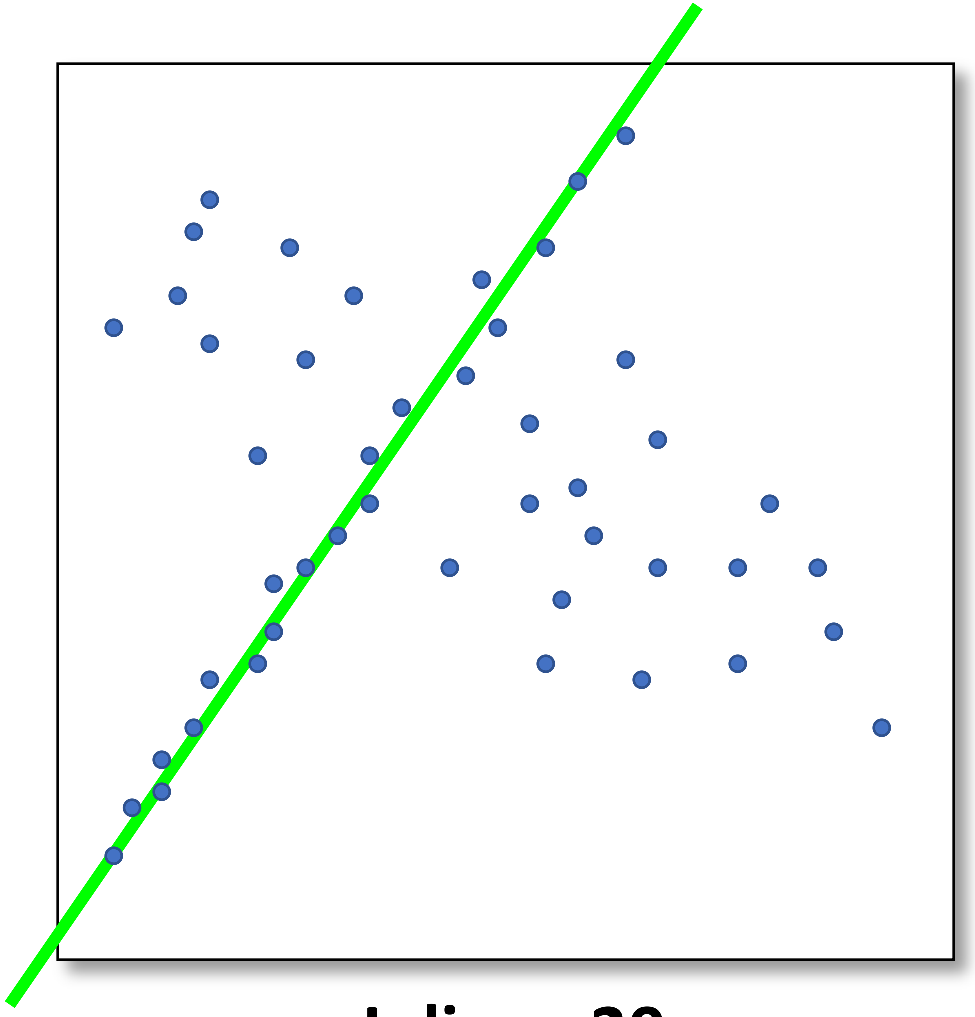


Counting inliers



Inliers: 3

Counting inliers



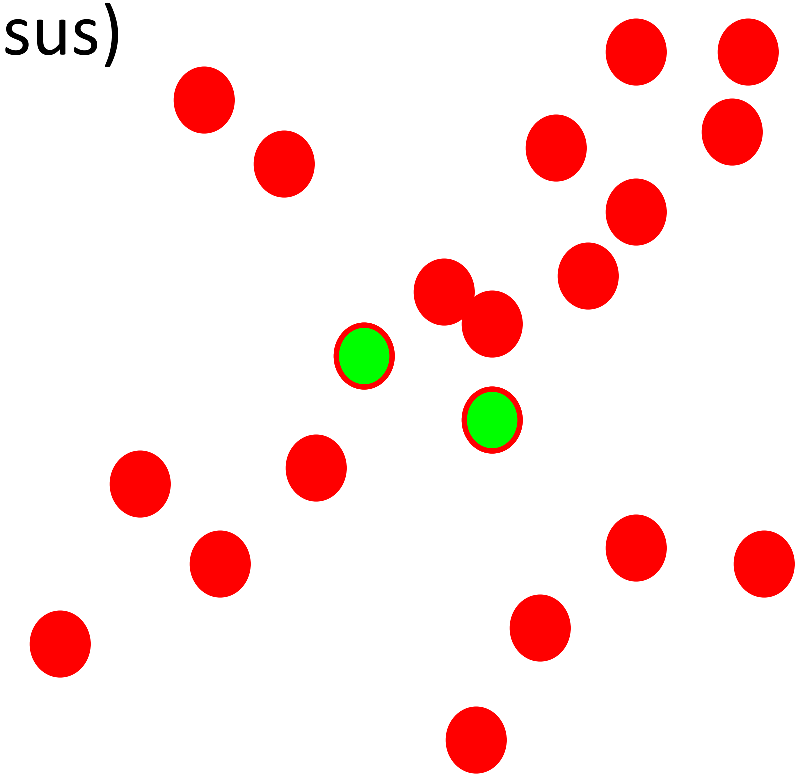
Inliers: 20

Which hypotheses?

- Sample hypotheses randomly?
  - Might sample useless hypotheses that doesn’t fit any data
- Only want hypotheses that fit *at least some data*
- Idea: sample minimum points to fit hypothesis
- This yields candidate hypothesis

RANSAC (Random Sample Consensus)

Line fitting example



Algorithm:

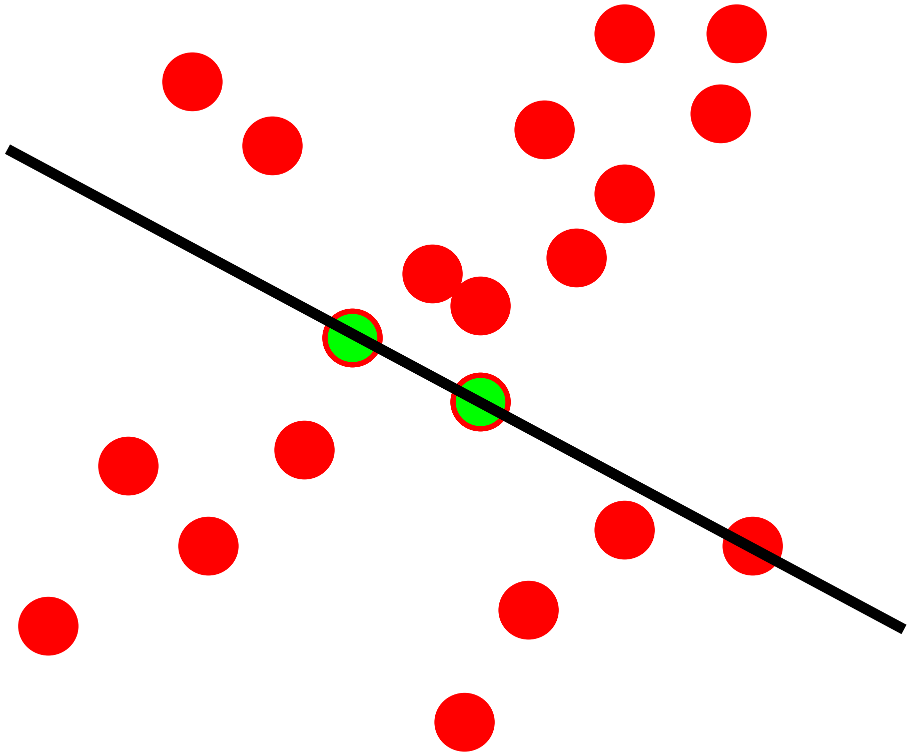
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example



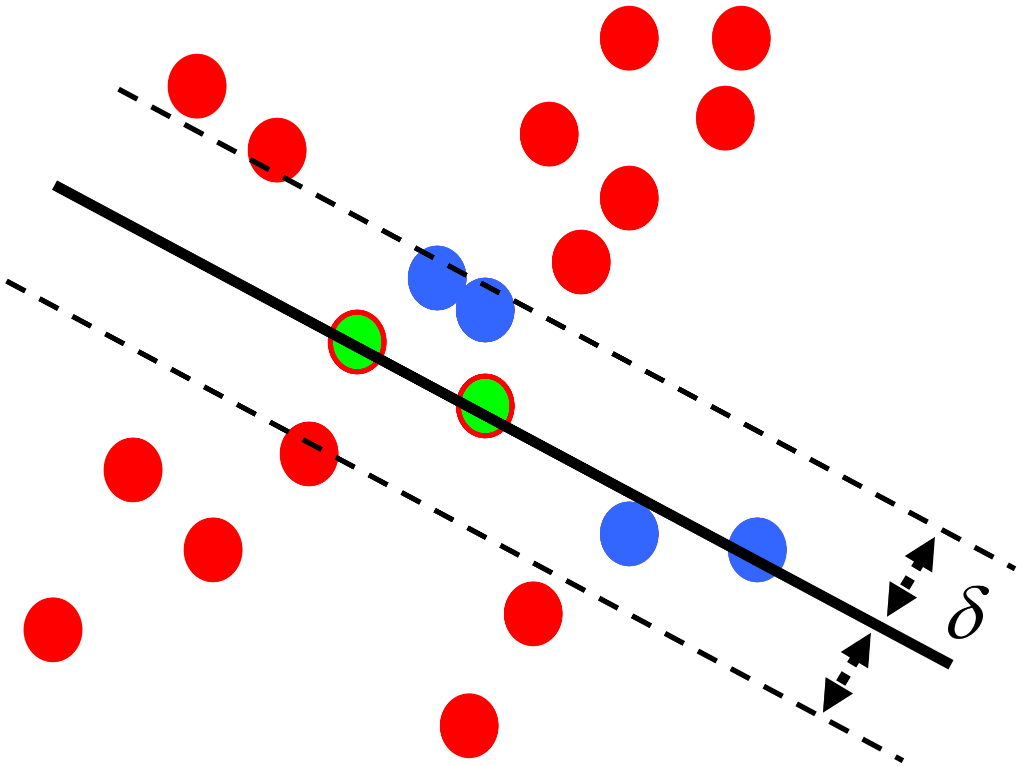
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. $\boxed{\textbf{Solve} \text{ for model parameters using samples}}$
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



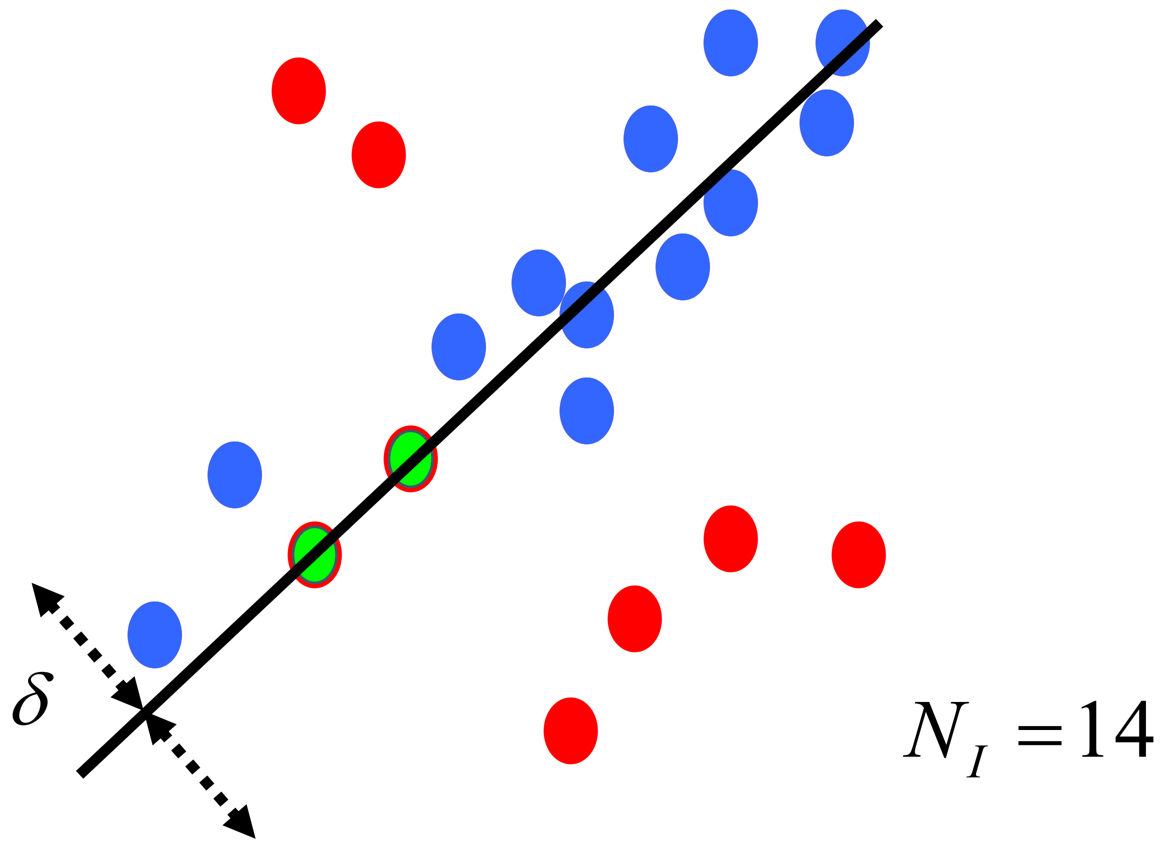
$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Problem setup (again)

- Given
  - A dataset $D = \{ p_1, p_2, \dots, p_N\}$
    - Example 1: Line fitting: $\{ (x_1, y_1), \dots, (x_n, y_n)\}$
    - Example 2: Fundamental matrix: $\{ (\vec{p}_1, \overrightarrow{q_1}), (\overrightarrow{p_2}, \overrightarrow{q_2}), \dots, (\overrightarrow{p_N}, \overrightarrow{q_N})\}$
  - A set of parameters θ that need to be fitted
    - Line fitting: $\theta = (m, b)$
    - F estimation $\theta = F, ||f|| = 1$
  - A cost function $C(p, \theta)$
    - Line fitting: $C\big((x, y), (m, b)\big) = ||y - (mx + b)||^2$
    - F estimation: $C\big((\vec{p}, \vec{q}), F\big) = \overrightarrow{p^T} F \vec{q}$(Reprojection error)
  - A minimum number needed k
    - Line fitting: 2
    - F estimation: 8

RANSAC (RAndom SAmple Consensus)

- Repeat:
  - Sample minimum number of points k to fit hypothesis
  - Fit hypothesis
  - Count number of inliers in entire dataset
- Choose hypothesis with most number of inliers
- Re-update hypothesis with estimated inliers

RANSAC - hyperparameters

- **Inlier threshold** related to the amount of noise we expect in inliers
  - Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee

RANSAC

- An example of a "voting"-based fitting scheme
- Each hypothesis gets voted on by each data point, best hypothesis wins

- There are many other types of voting schemes
  - E.g., Hough transforms…

The correspondence problem

Till now

- Geometry of image formation
- Stereo reconstruction
  - Given 3D → 2D correspondence, find K, R, t
  - Given 2 images, correspondence, K, R, t, find 3D points
  - Given 2 images, correspondence, find F, E, R, t, 3D points

Till now

- Geometry of image formation
- Stereo reconstruction
  - Given 3D → 2D correspondence, find K, R, t
  - Given 2 images, correspondence, K, R, t, find 3D points
  - Given 2 images, correspondence, find F, E, R, t, 3D points

Correspondence can be challenging

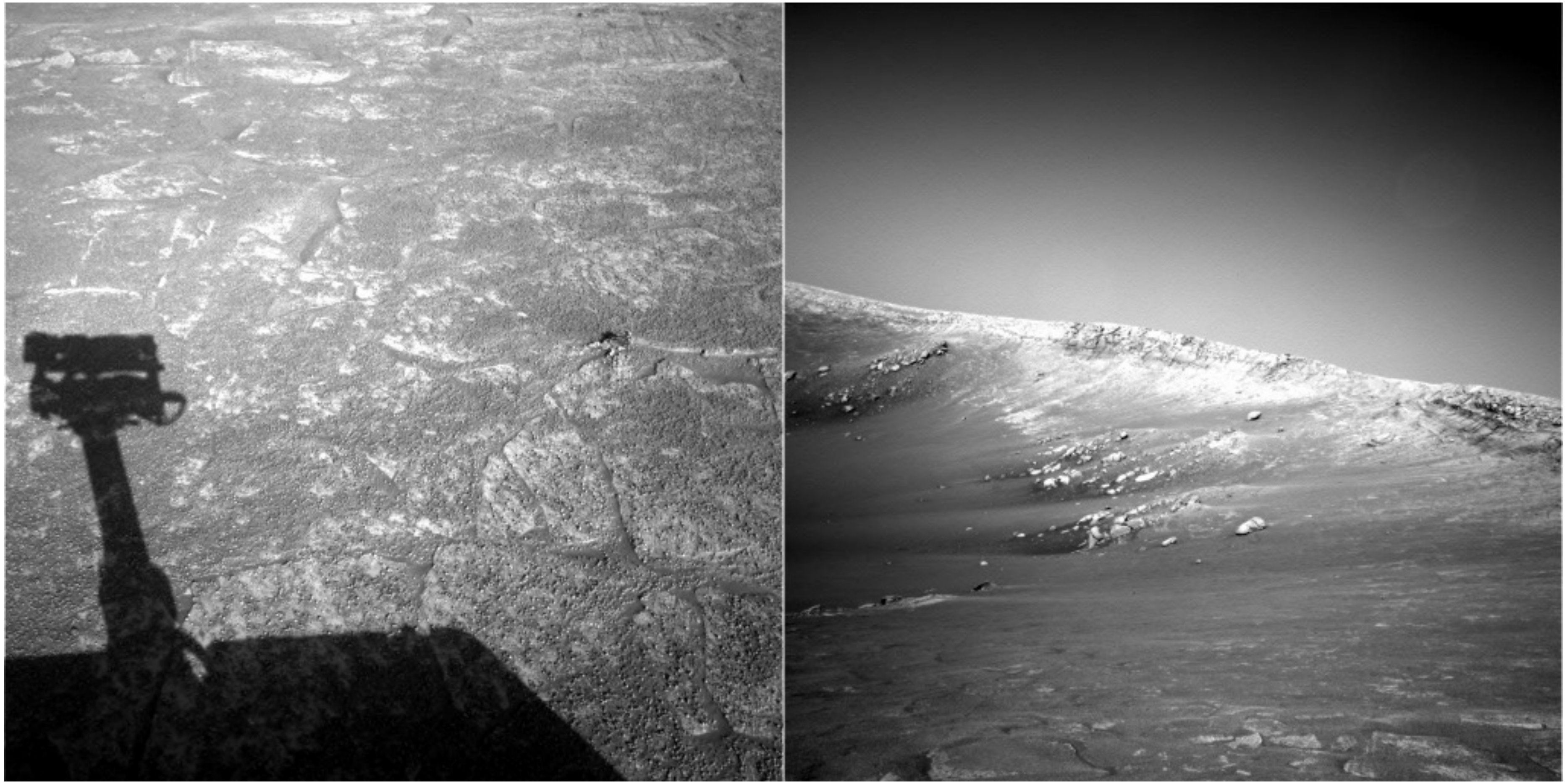
Fei-Fei Li

Harder case

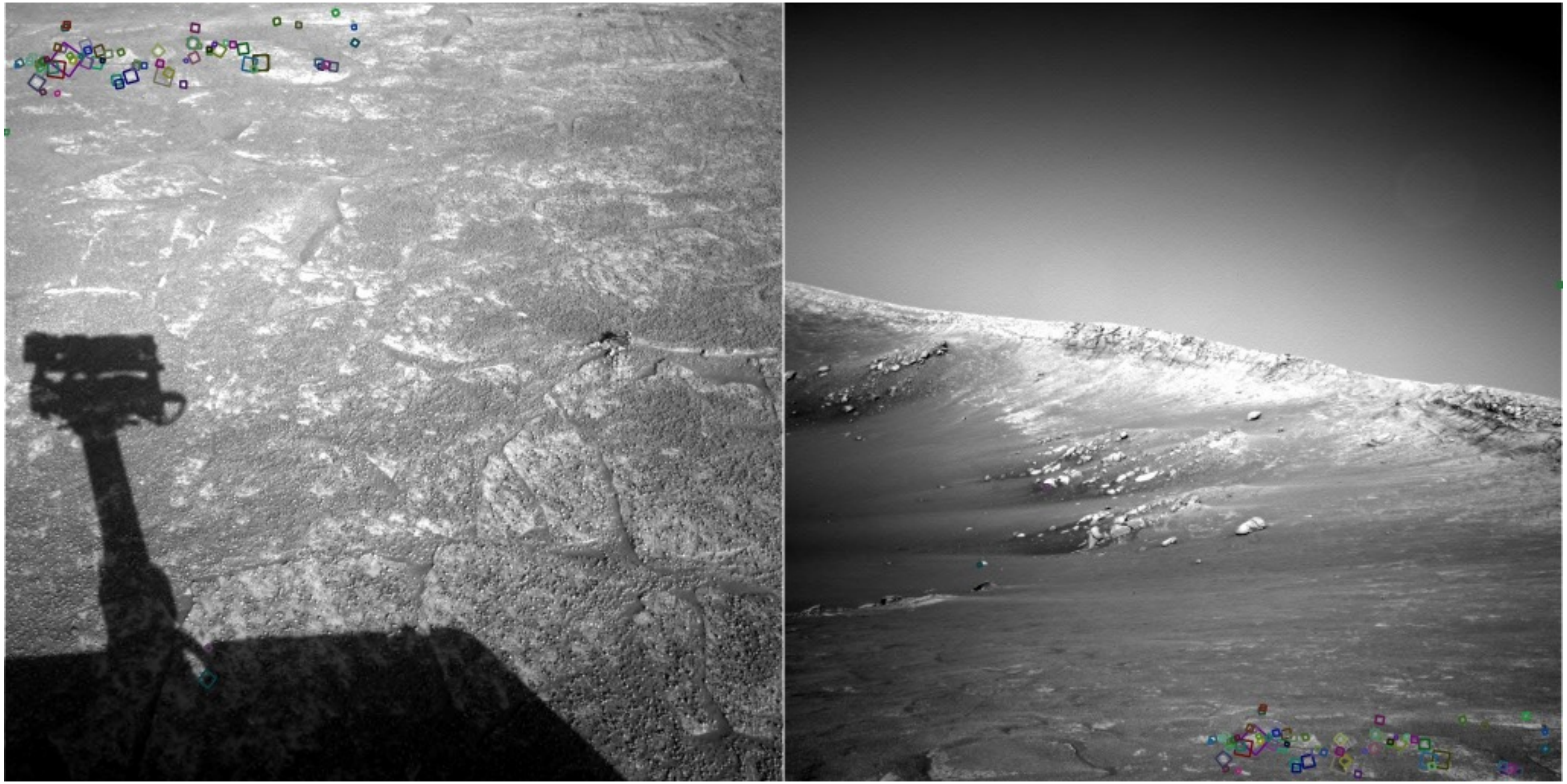
by Diva Sian

by scgbt

Harder still?



Answer below (look for tiny colored squares…)



NASA Mars Rover images
with SIFT feature matches

The correspondence problem

The aperture problem

- When viewed from a small “aperture”, correspondence is ambiguous

The aperture problem

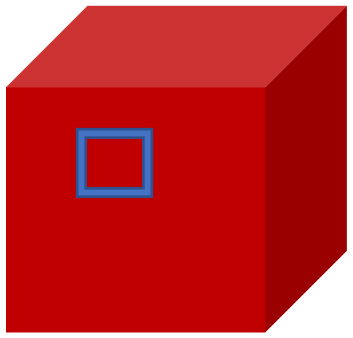
- Individual pixels are ambiguous
- Idea: Look at whole patches!

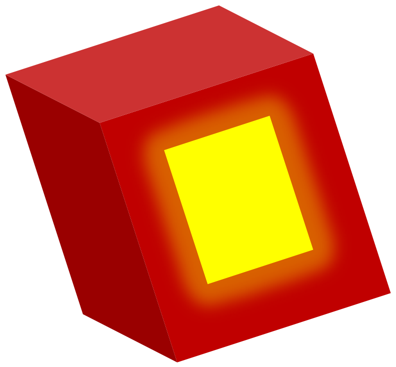
The aperture problem

- Individual pixels are ambiguous
- Idea: Look at whole patches!

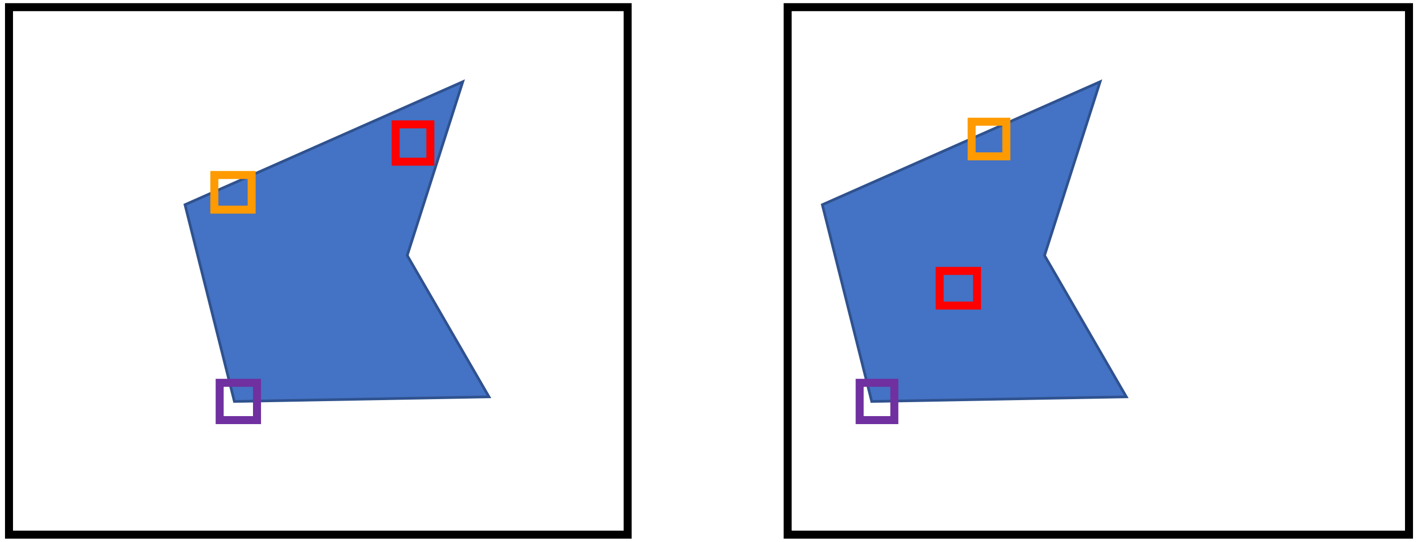
The aperture problem

- *Some local neighborhoods* are ambiguous



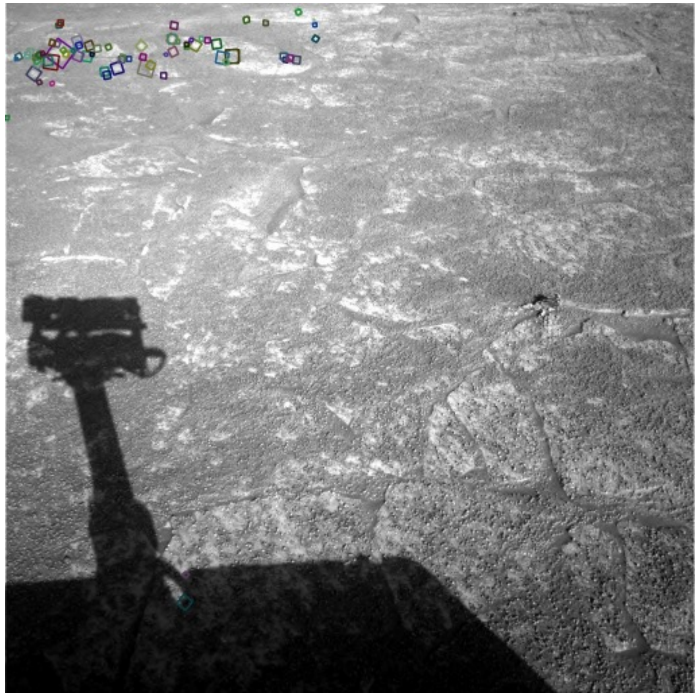


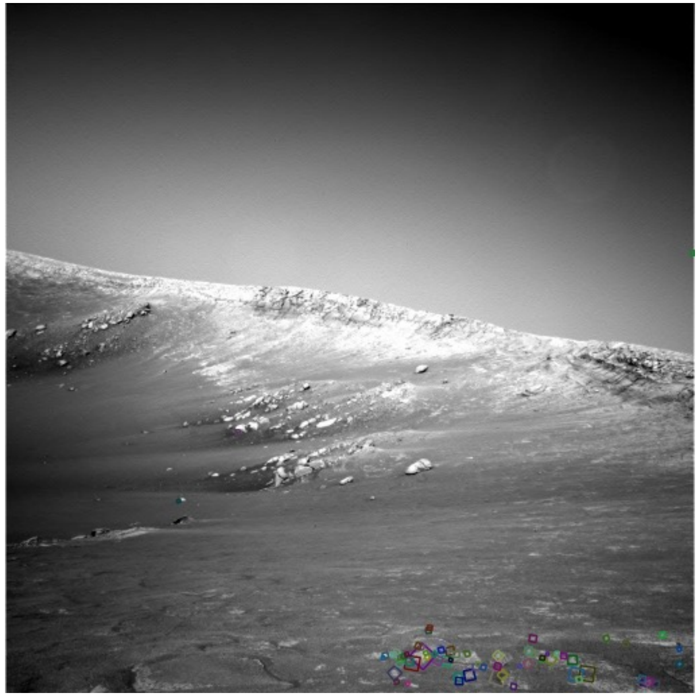
The aperture problem



Sparse vs dense correspondence

- Sparse correspondence: produce a few, high confidence matches
  - Good enough for estimating pose or relationship between cameras
- Dense correspondence: try to match every pixel
  - Needed if we want 3D location of every pixel (e.g., stereo)



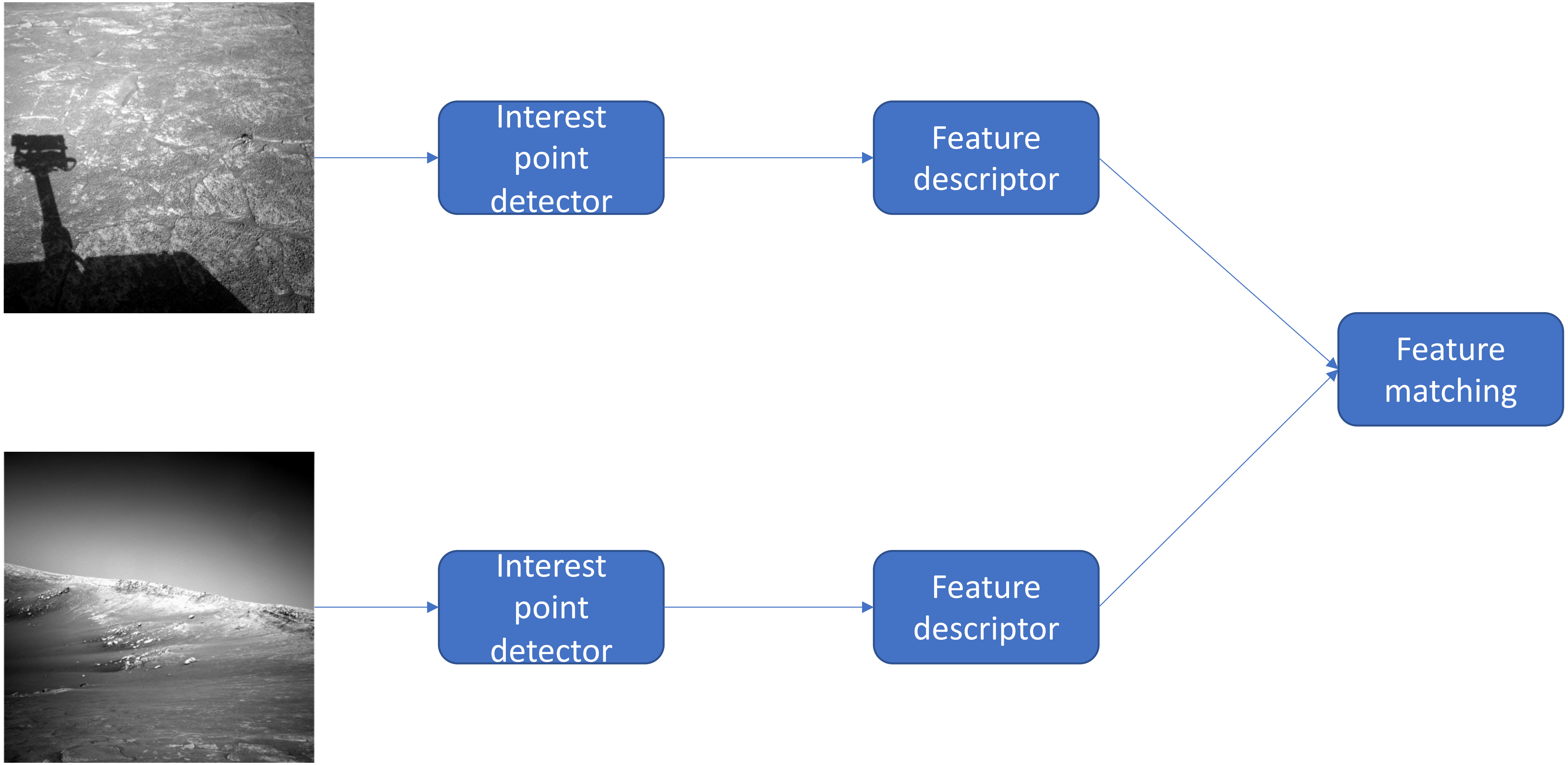


Sparse correspondences

- For many applications, a few good correspondences suffice
  - Camera calibration
  - Estimating essential matrix
  - Reconstructing a sparse cloud of 3D points

- Detect points that will produce good correspondences
- Match detected points from both images

Sparse correspondence pipeline



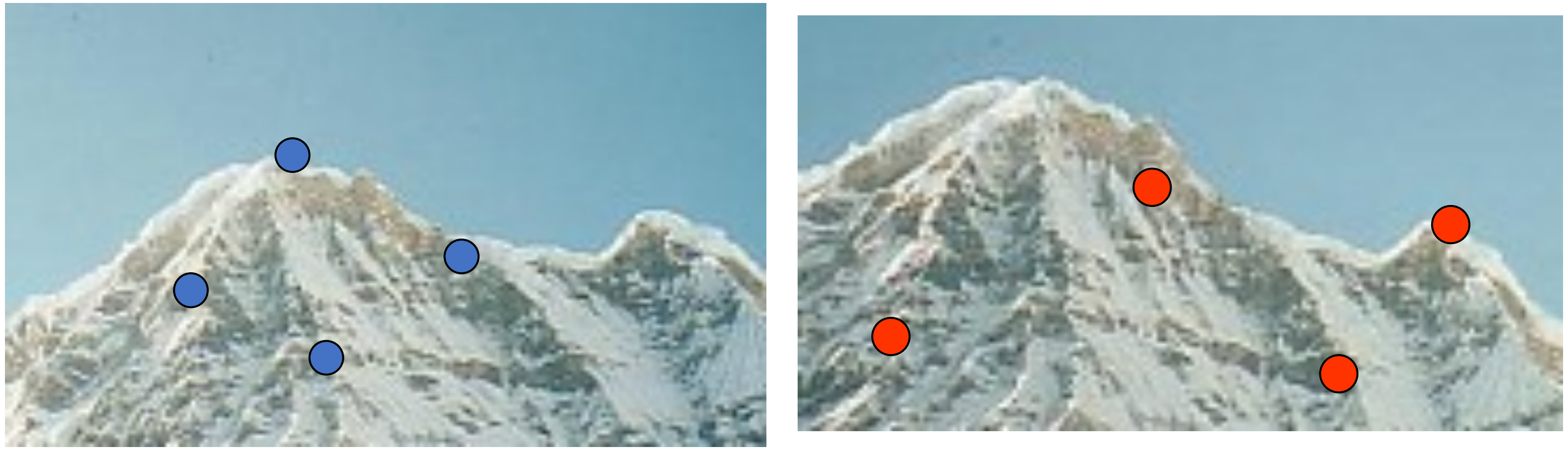
Characteristics of good feature points

- Repeatability / invariance
  - The same feature point can be found in several images despite geometric and photometric transformations
- Saliency / distinctiveness
  - Each feature point is distinctive
  - Fewer ”false” matches / less ambiguity

Slide credit: Kristen Graumanc

Goal: repeatability

- We want to detect (at least some of) the same points in both images.



No chance to find true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

Slide credit: Kristen Graumanc

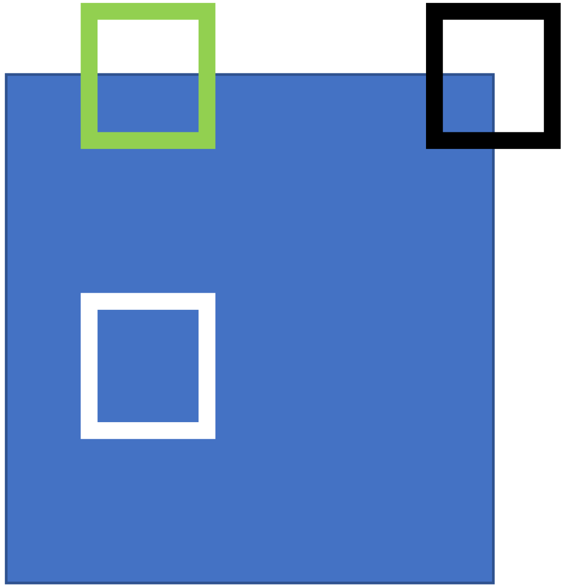
Goal: distinctiveness

- The feature point should be distinctive enough that it is easy to match
  - Should *at least* be distinctive from other patches nearby

Slide credit: Kristen Graumanc

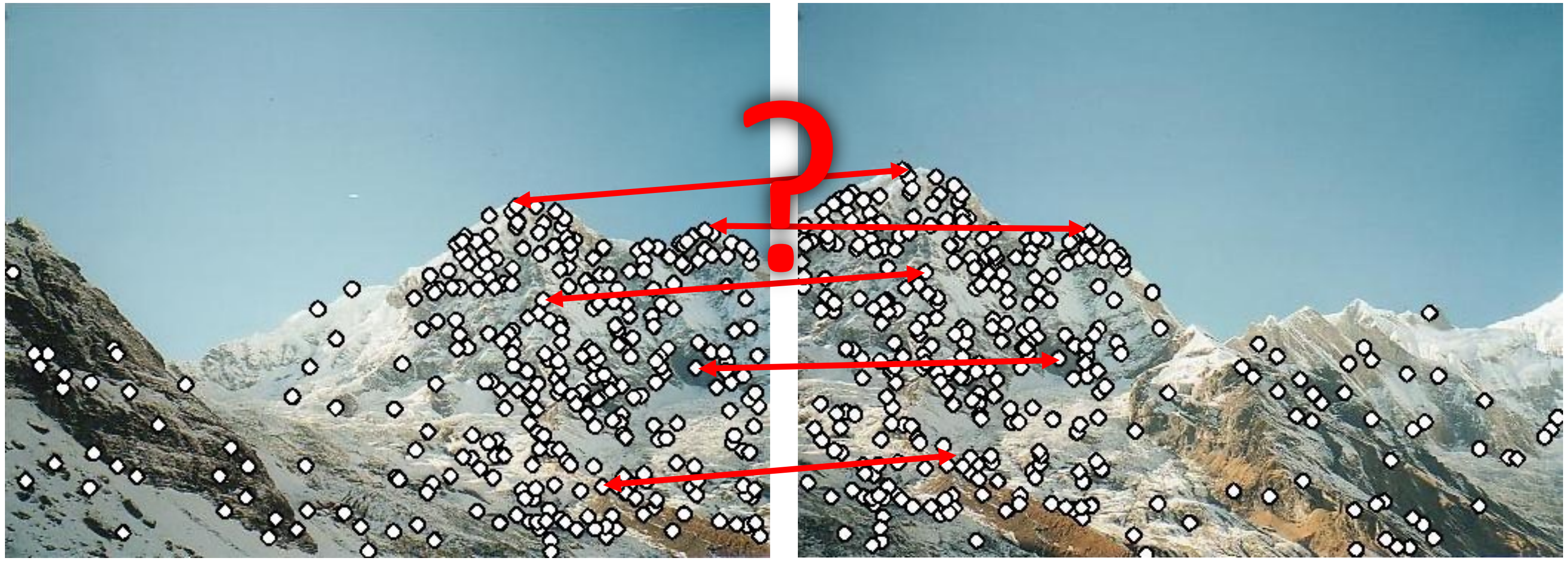
Harris corner detector

- Let us tackle second goal
- Main idea: Translating patch should cause large differences
- An example of an *interest point detector*



Matching feature points

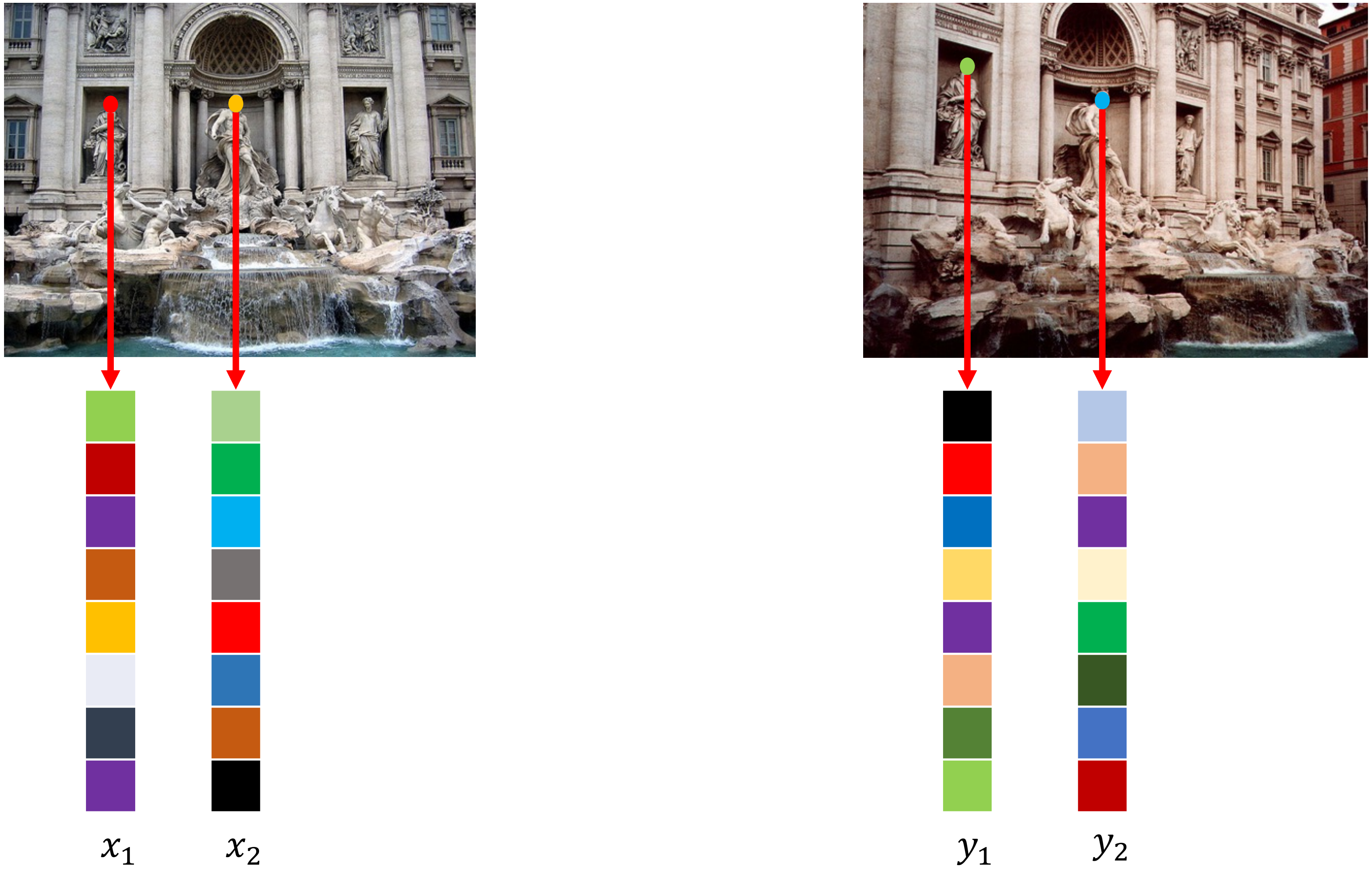
We know how to detect good points
Next question: **How to match them?**



Two interrelated questions:

1. How do we *describe* each feature point?
2. How do we *match* descriptions?

Feature descriptor



Feature matching

- Measure the distance between (or similarity between) every pair of descriptors

	y_1	y_2
x_1	$d(x_1, y_1)$	$d(x_1, y_2)$
x_2	$d(x_2, y_1)$	$d(x_2, y_2)$

Invariance vs. discriminability

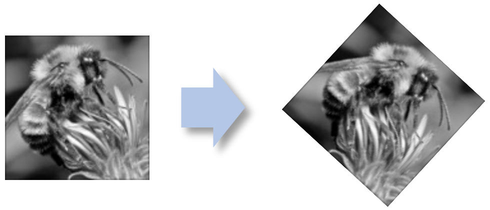
- Invariance:
  - Distance between descriptors should be small even if image is transformed

- Discriminability:
  - Descriptor should be highly unique for each point (far away from other points in the image)

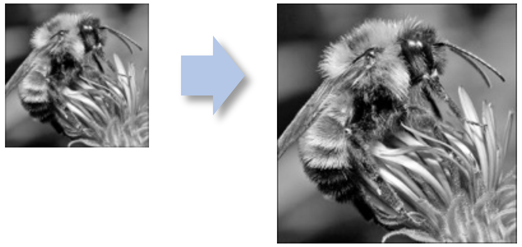
Image transformations

- Geometric

Rotation

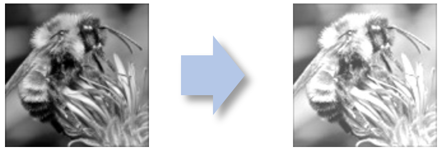


Scale



- Photometric

Intensity change

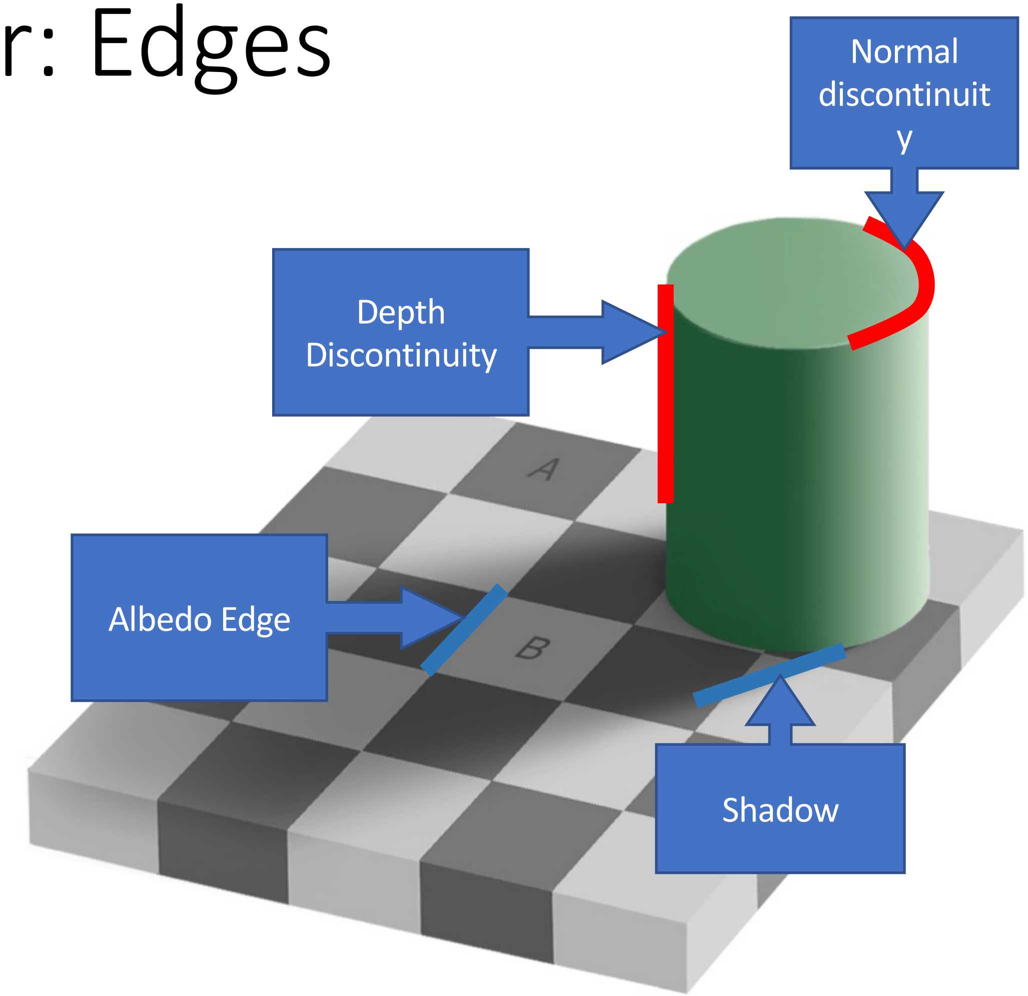


Invariance

- Most feature descriptors are designed to be invariant to
  - Translation, 2D rotation, scale

- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 60 degrees)
  - Limited affine transformations (some are fully affine invariant)
  - Limited illumination/contrast changes

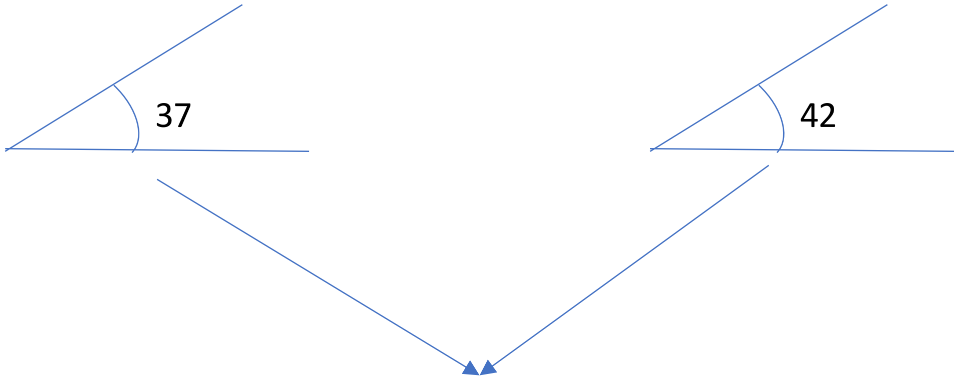
Better representation than color: Edges



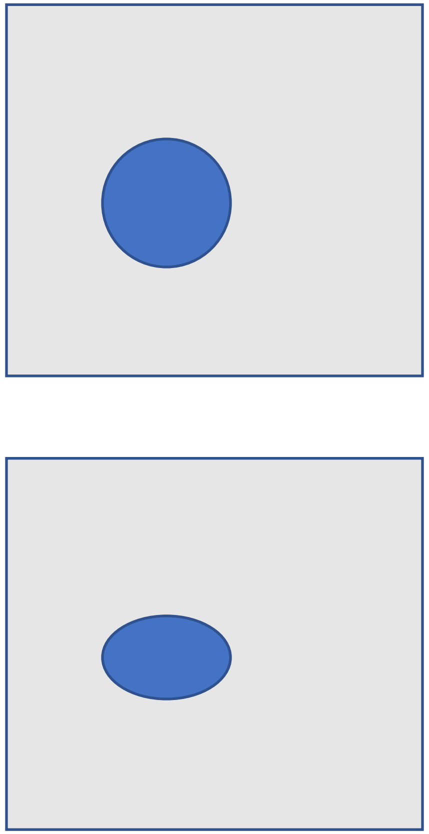
Towards a better feature descriptor

- Match *pattern of edges*
  - Edge orientation – clue to shape
- Be resilient to *small deformations*
  - Deformations might move pixels around, but slightly
  - Deformations might change edge orientations, but slightly

Invariance to deformation by quantization



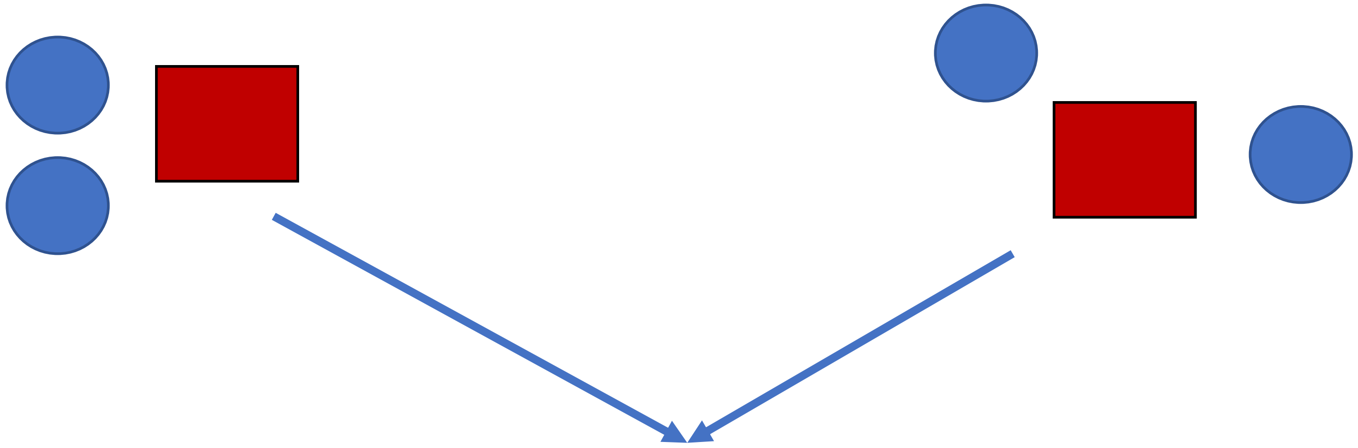
Between 30 and 45



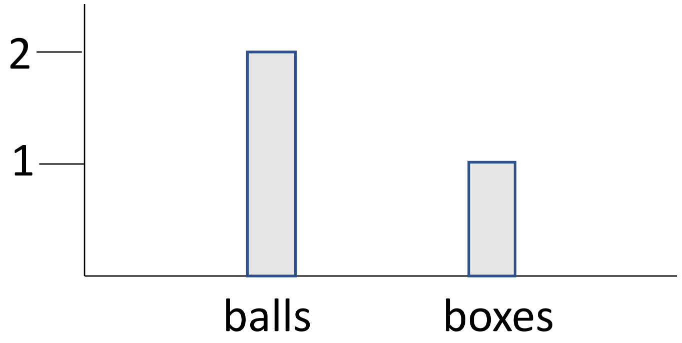
Invariance to deformation by quantization

$$g(\theta) = \begin{cases} 0 & \text{if } 0 < \theta < 2\pi/N \\ 1 & \text{if } 2\pi/N < \theta < 4\pi/N \\ 2 & \text{if } 4\pi/N < \theta < 6\pi/N \\ & \dots \\ N-1 & \text{if } 2(N-1)\pi/N \end{cases}$$

Spatial invariance by histograms



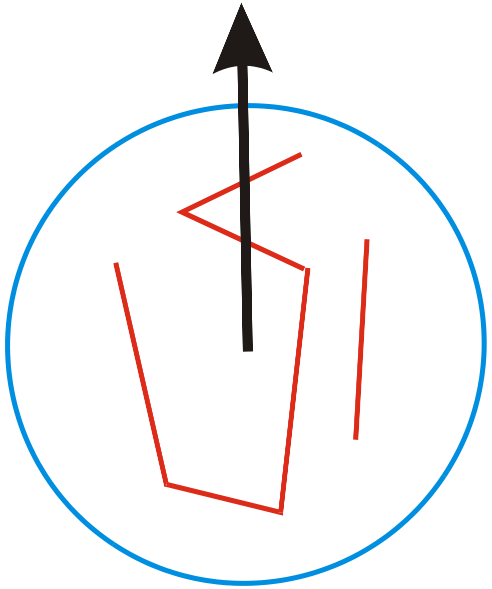
2 blue balls, one red box

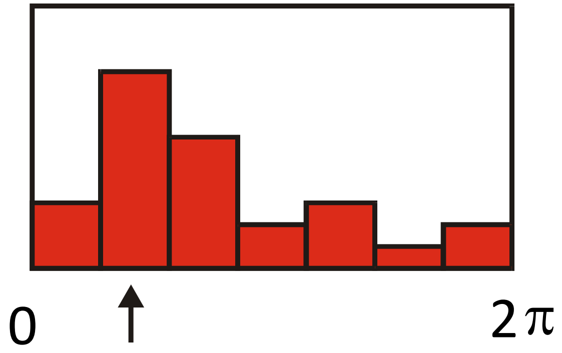


Rotation Invariance by Orientation Normalization

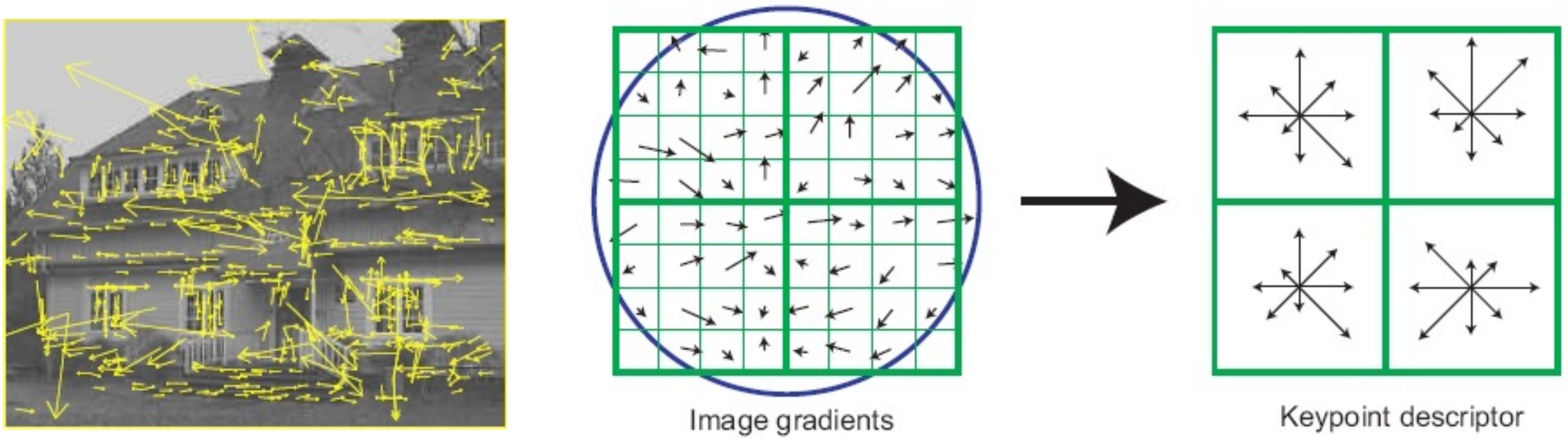
[Lowe, SIFT, 1999]

- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation





The SIFT descriptor



SIFT – Lowe IJCV 2004

Scale Invariant Feature Transform

Basic idea:

- DoG for scale-space feature detection
- Take 16x16 square window around detected feature
    - Compute gradient orientation for each pixel
    - Throw out weak edges (threshold gradient magnitude)
    - Create histogram of surviving edge orientations

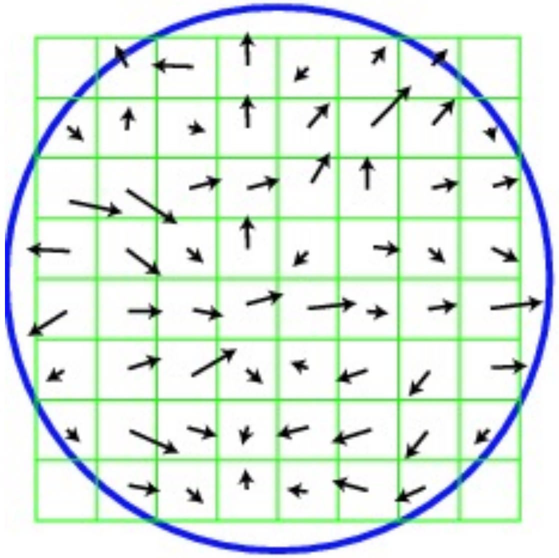
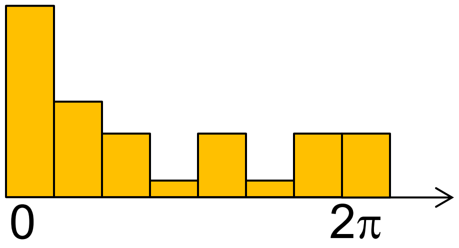
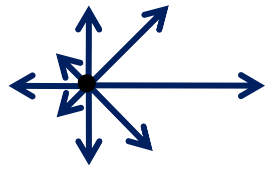


Image gradients



angle histogram



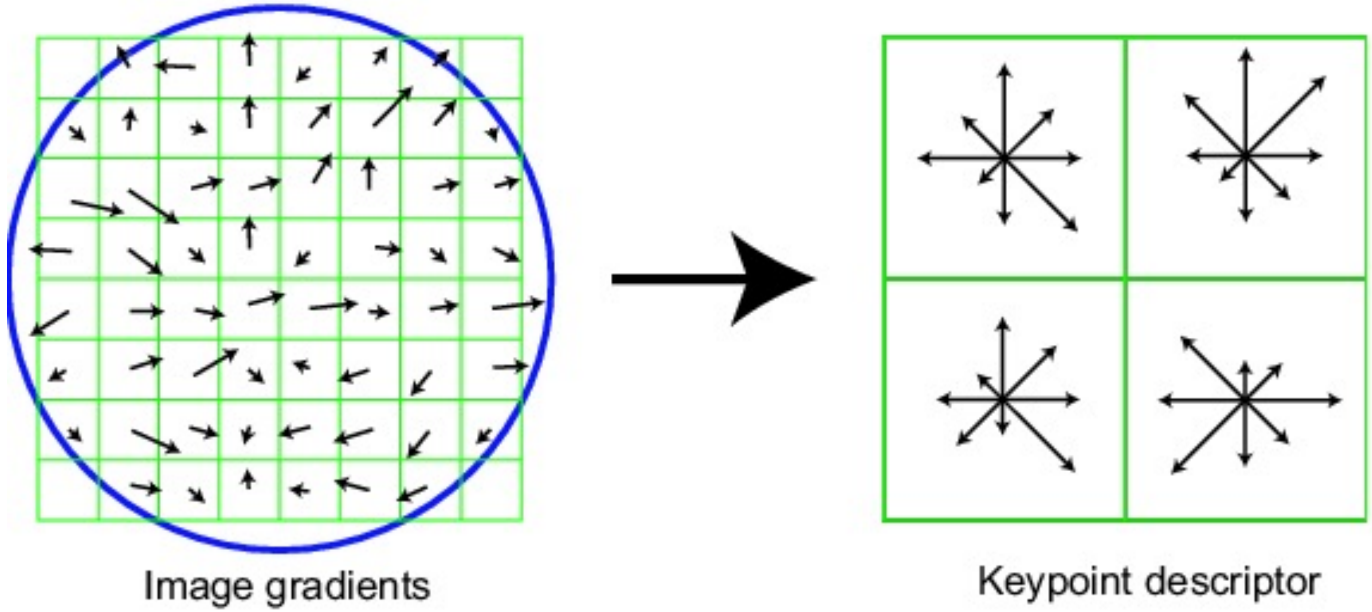
Keypoint descriptor

Adapted from slide by David Lowe

SIFT descriptor

Create histogram

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



Adapted from slide by David Lowe

SIFT vector formation

- Computed on rotated and scaled version of window according to computed orientation & scale
  - resample the window
- Based on gradients weighted by a Gaussian

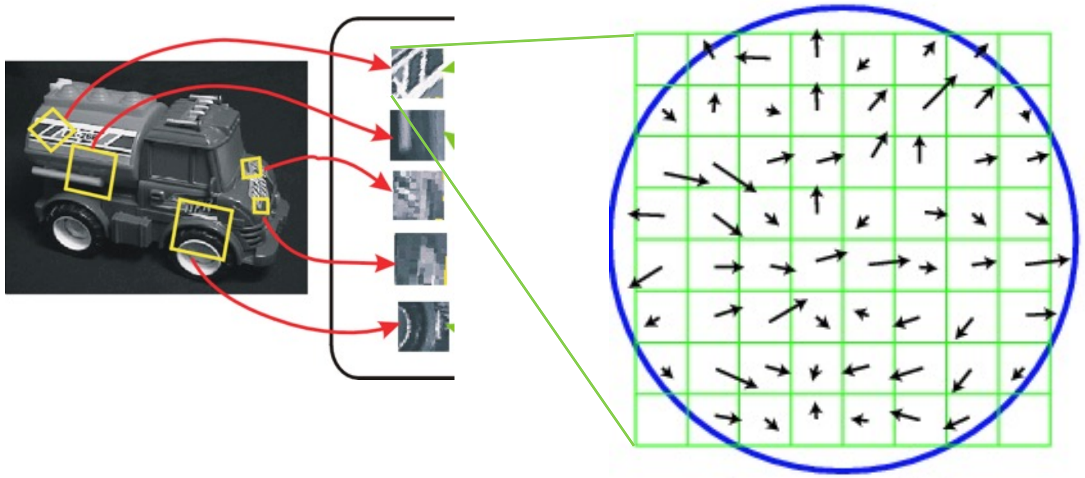


Image gradients

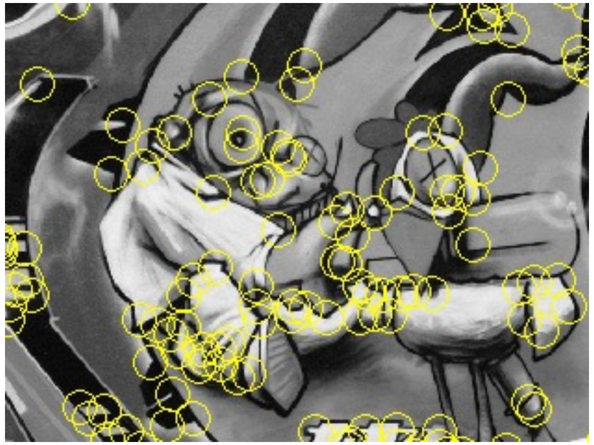
Properties of SIFT

Extraordinarily robust matching technique

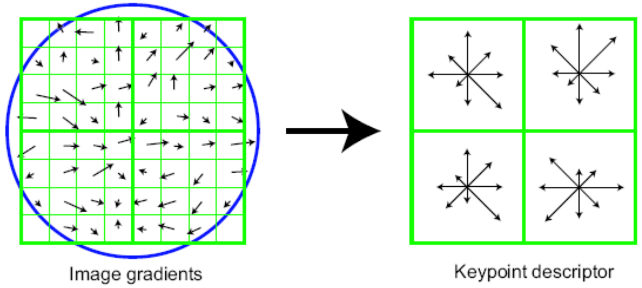
- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available:
  http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT

Summary

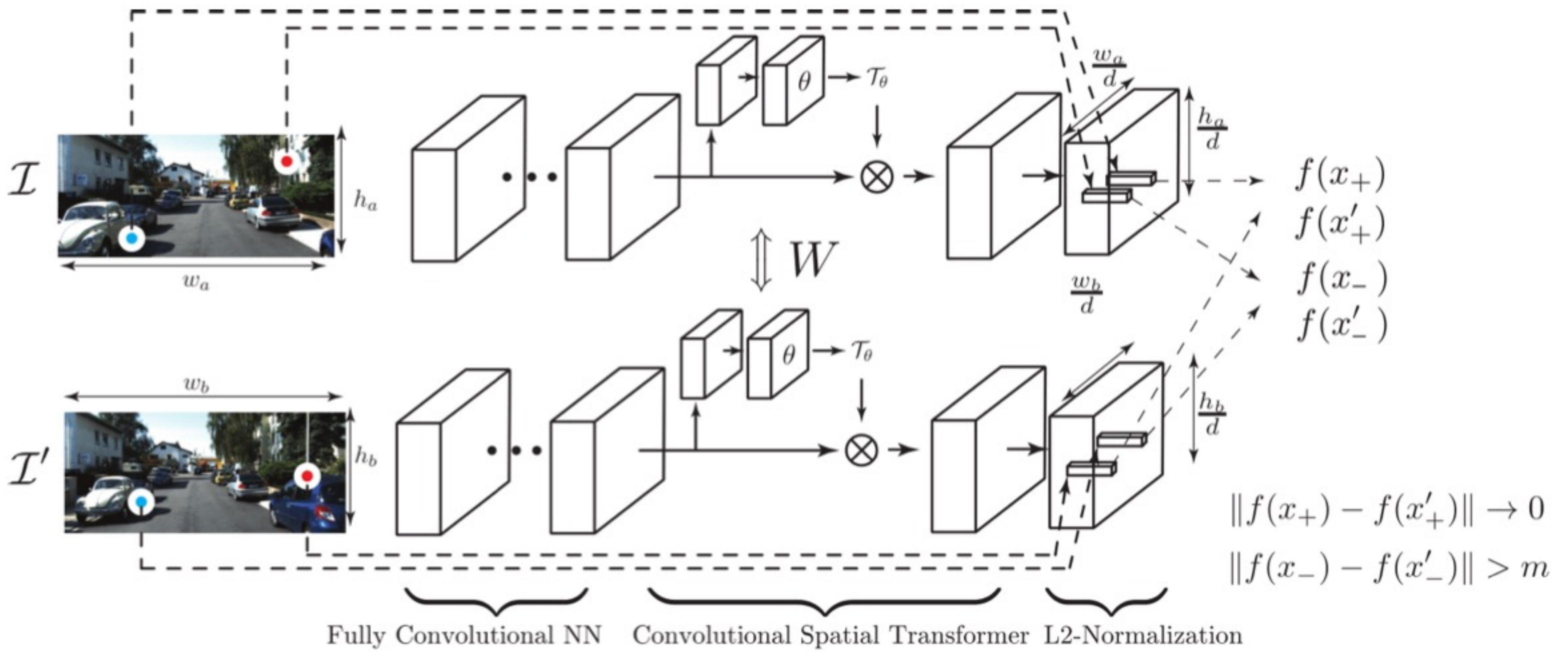
- Keypoint detection: repeatable and distinctive
  - Corners, blobs, stable regions
  - Harris, DoG



- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT and variants are typically good for stitching and recognition
  - But, need not stick to one

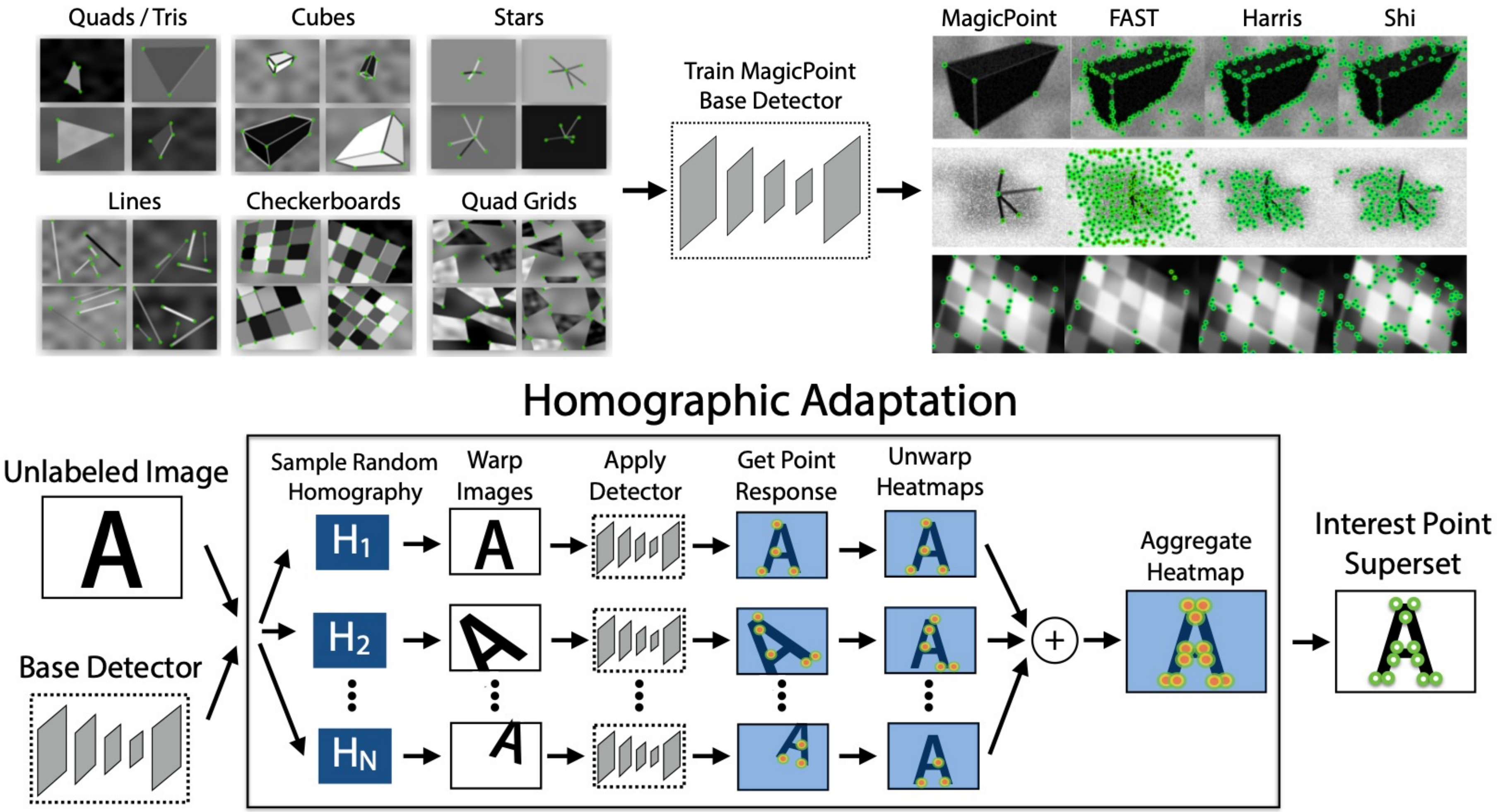


Learning-based correspondence



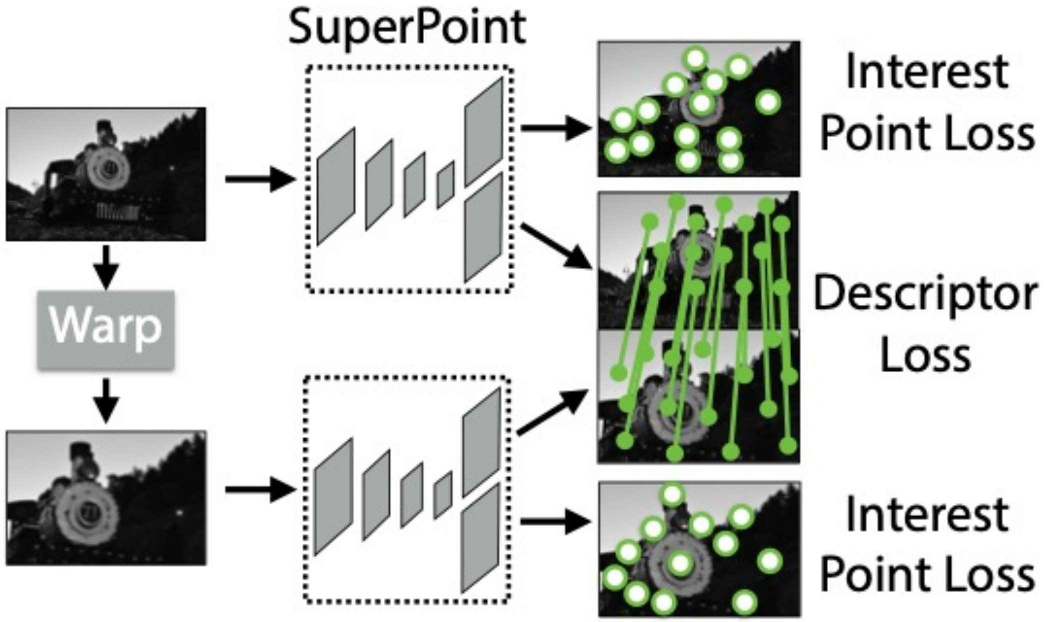
Choy, Christopher B., et al. "Universal correspondence network." *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016.

Learning interest points



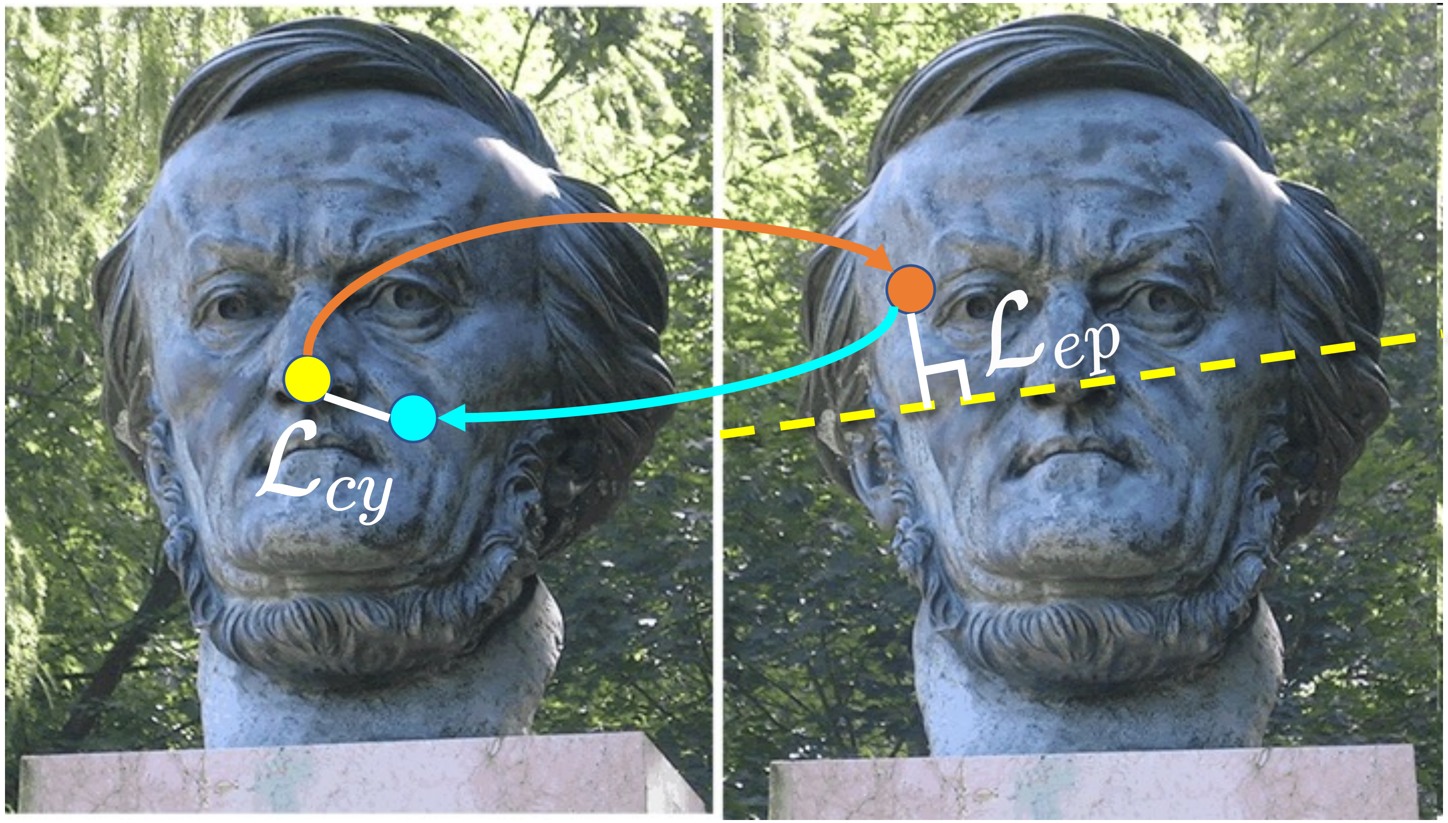
DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description." *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018.

Learning descriptors without supervision



DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description." *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018.

Epipolar constraint → Epipolar loss



Query point

Ground truth epipolar line

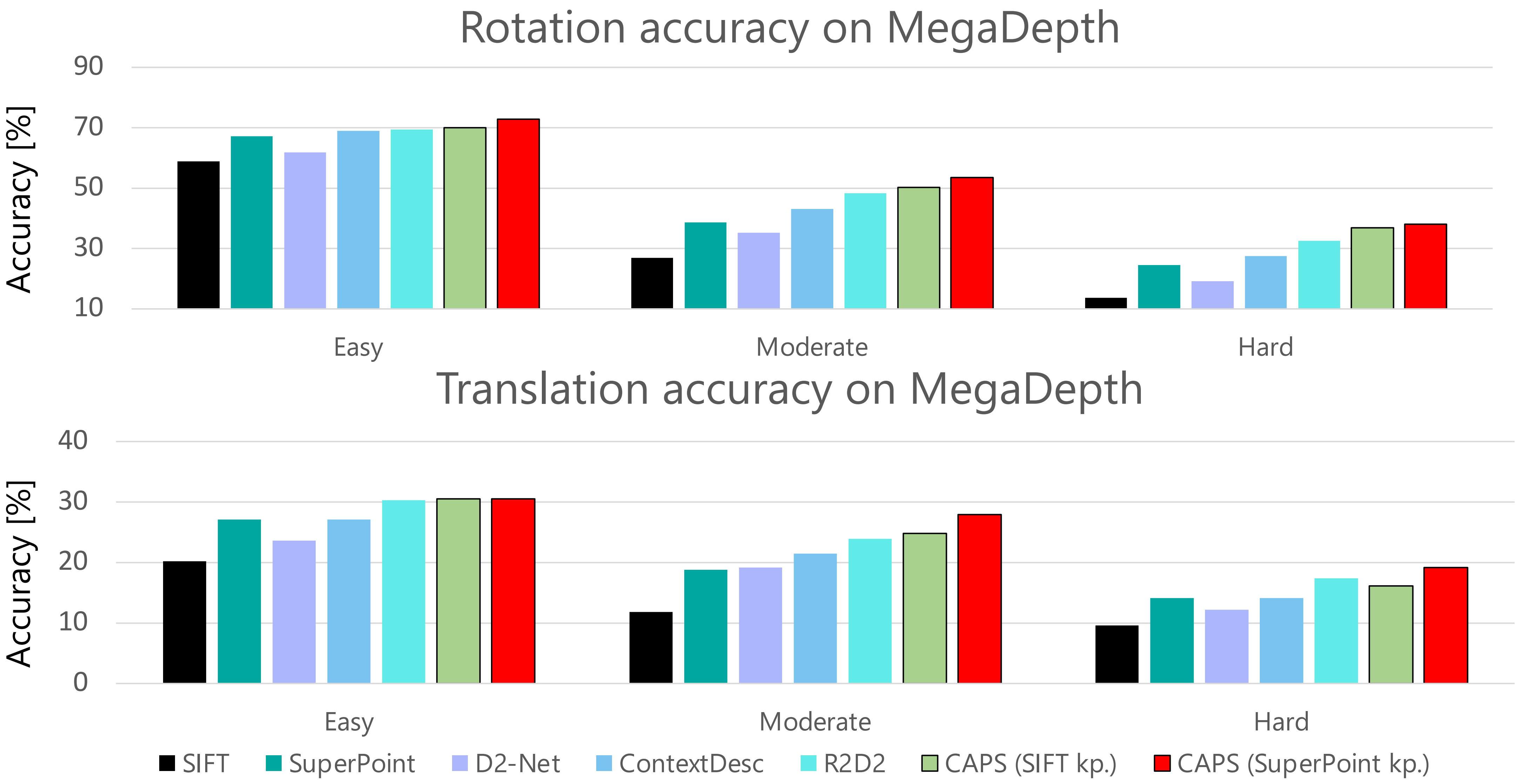
Predicted correspondence

$\mathcal{L}_{ep}$ Epipolar loss

$\mathcal{L}_{cy}$ Cycle consistency loss

Wang, Qianqian, et al. "Learning feature descriptors using camera pose supervision." *European Conference on Computer Vision*. Springer, Cham, 2020.

Evaluation on relative pose estimation



The structure from motion pipeline

- Image matching
  - Estimate correspondences, use epipolar geometry + RANSAC to clean correspondences
- Incremental 3D reconstruction
  - Reconstruct keypoints from a pair of images
  - Add images in, do triangulation to reconstruct more 3D points
- Bundle adjustment
  - Take all 3D points and all cameras and minimize reprojection error
- Lots of details; decades of work in getting this right!

Image matching

- Given a collection of images
- Extract interest points and descriptors (e.g., SIFT)
- Look at image pairs and use correspondences to:
  - Decide if image pair has some overlap
  - Estimate E (or F) (or a homography H if no translation)
  - Use RANSAC for outlier sensitivity
- Obtain:
  - Verified image pairs
  - Verified inlier correspondences
  - Transformation between cameras (relative pose, i.e., R and t)

Scene graph

Incremental 3D Reconstruction

- Given scene graph
- Pick initial pair
  - Use inlier correspondences + known relative pose for triangulation
  - Obtain initial set of 3D points, say S
- Repeat:
  - Pick an unregistered image
  - Use known 3D points S and their corresponding 2D location to calibrate (Use RANSAC)
  - Use other correspondences between registered images to grow S
  - Bundle adjustment: minimize reprojection error for all points and cameras
- Output: 3D point cloud S and camera pose for every registered image

Bundle adjustment

$$\min_{\{\mathbf{P}_c\},\{\mathbf{X}_k\}} \sum_{c,k} \rho_{ck} \left(\|\pi(\mathbf{P}_c, \mathbf{X}_k) - \mathbf{x}_{ck}\|^2 \right)$$

Cameras

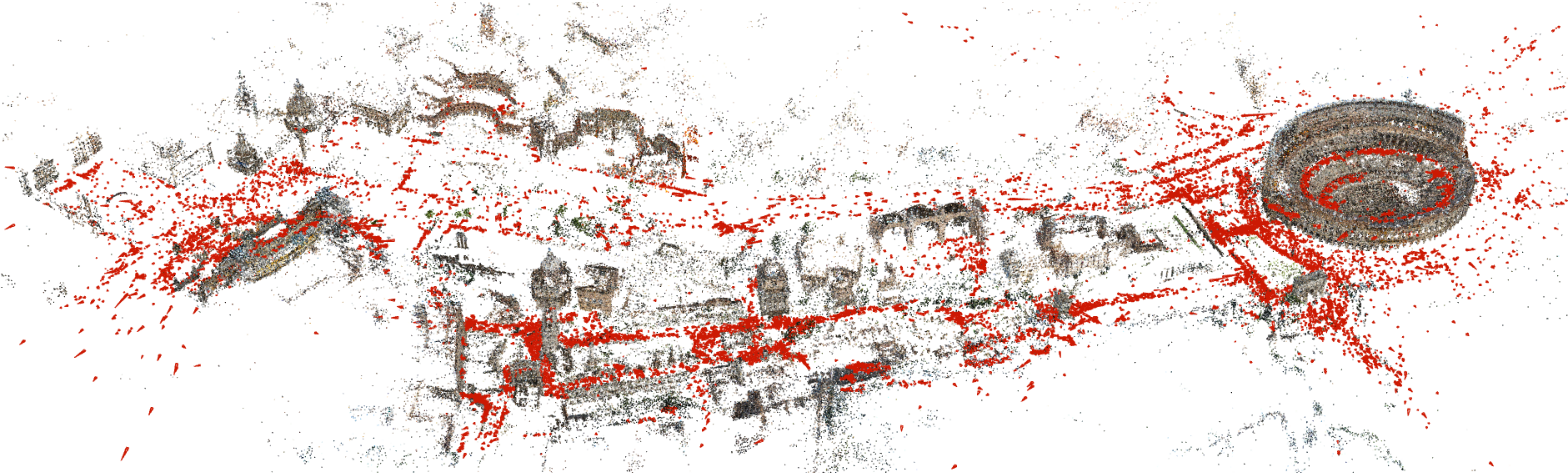
3D points

Perspective projection

2D Image location

Outlier downweighting

The structure-from-motion pipeline



https://colmap.github.io