Practical Course: Vision-based Navigation WS 2018/2019

Lecture 3. Keypoints

Vladyslav Usenko, Nikolaus Demmel,
Prof. Dr. Daniel Cremers

What We Will Cover Today

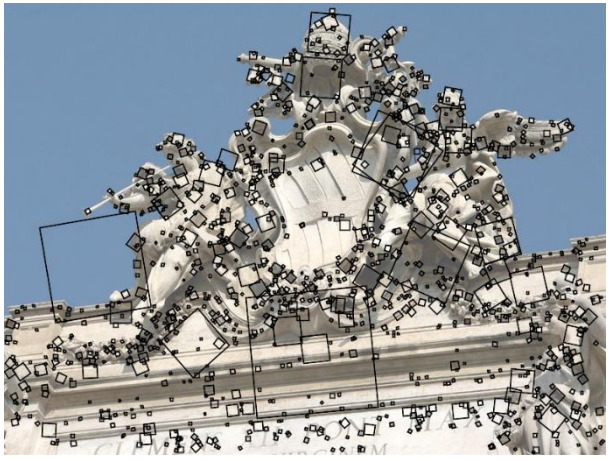
- Keypoint detection
  - Corner detection
  - Blob detection
  - Scale selection
- Keypoint description
  - Scale-Invariant Feature Transform (SIFT)
- State-of-the-art detectors and descriptors
- Keypoint matching
- RANSAC

Recap: Keypoint Detection

- Desirable properties of keypoint detectors for visual odometry:
  - high repeatability,
  - localization accuracy,
  - robustness,
  - invariance,
  - computational efficiency

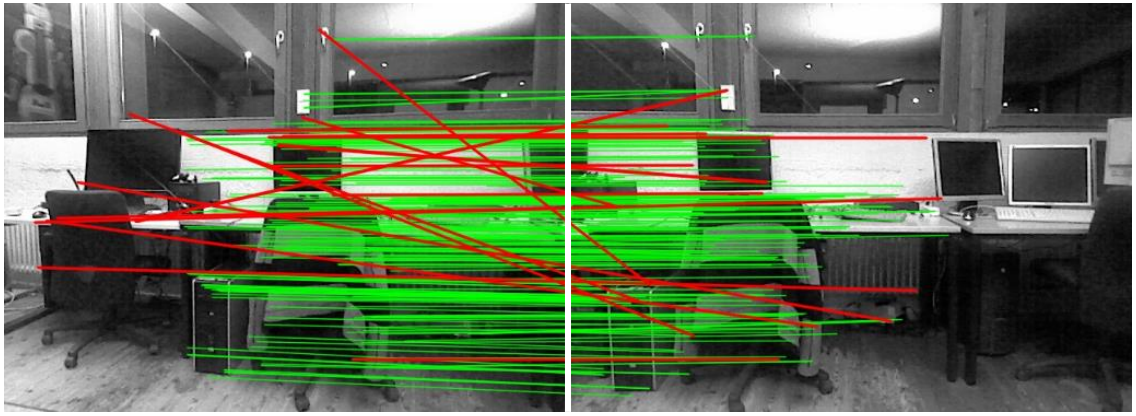
Harris Corners

Image source: Svetlana Lazebnik



DoG (SIFT) Blobs

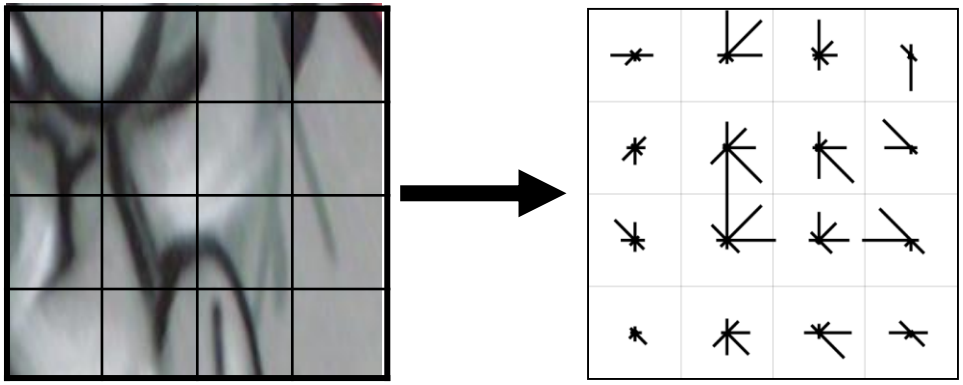
Recap: Keypoint Matching



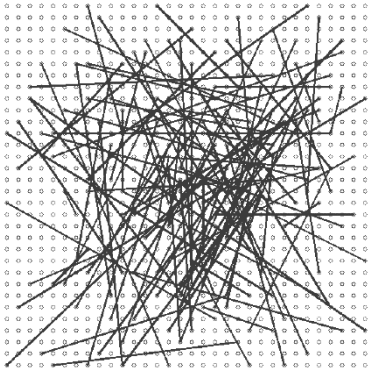
- Desirable properties for VO:
  - High recall
  - Precision
  - Robustness
  - Computational efficiency
- One possible approach to keypoint matching: by descriptor

Recap: Local Feature Descriptors

- Desirable properties for VO: distinctiveness, robustness, invariance
- Extract signatures that describe local image regions, examples:
  - Histograms over image gradients (SIFT)
  - Histograms over Haar-wavelet responses (SURF)
  - Binary patterns (BRIEF, BRISK, FREAK, etc.)
  - Learning-based descriptors (f.e. Calonder et al., ECCV 2008)
- Rotation-invariance: Align with dominant orientation in local region
- Scale-invariance: Adapt described region extent to keypoint scale



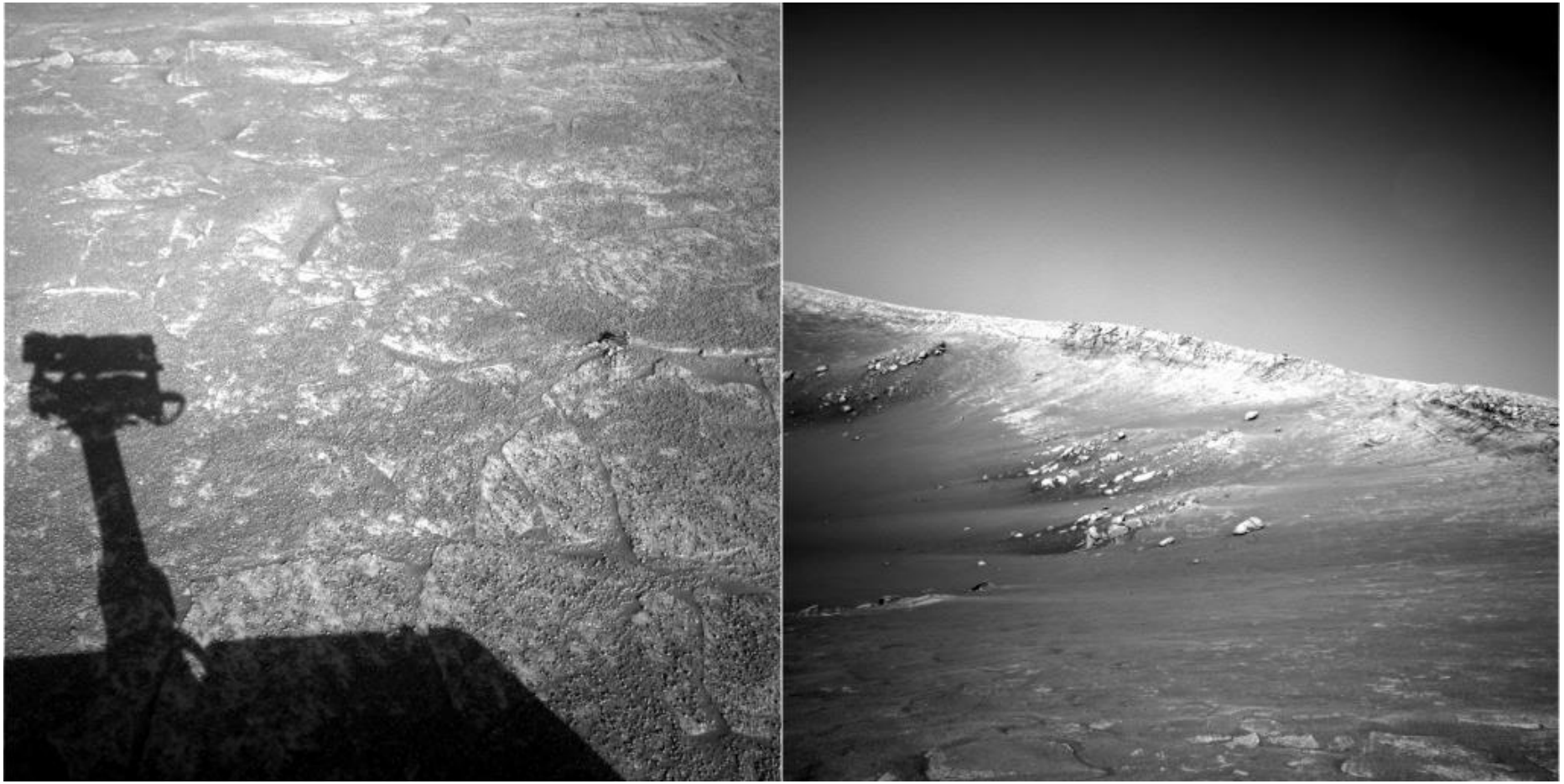
SIFT gradient pooling



BRIEF test locations

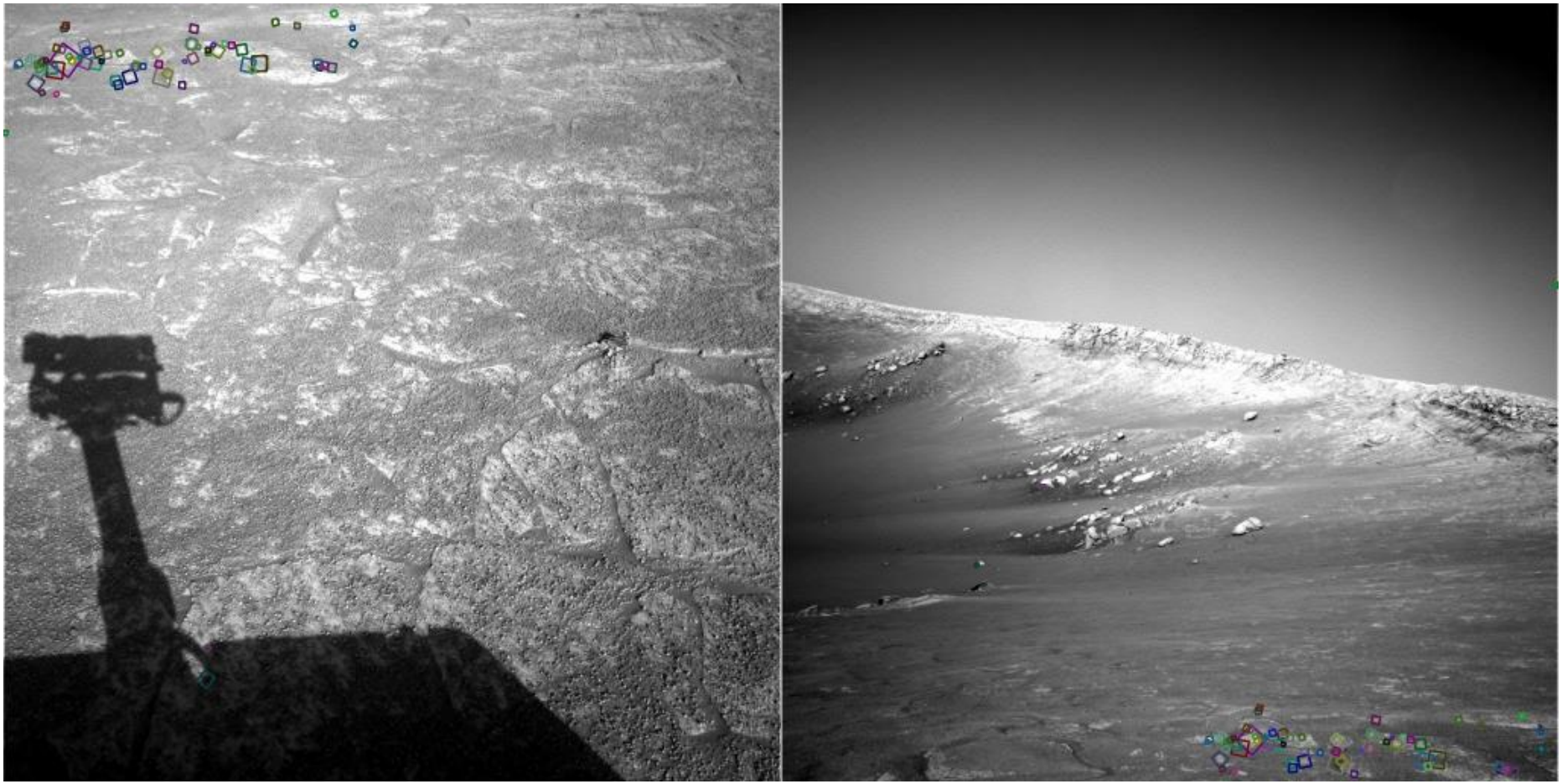
Image source: Svetlana Lazebnik / Calonder et al., ECCV 2010

Image Matching



NASA Mars Rover images

Image Matching

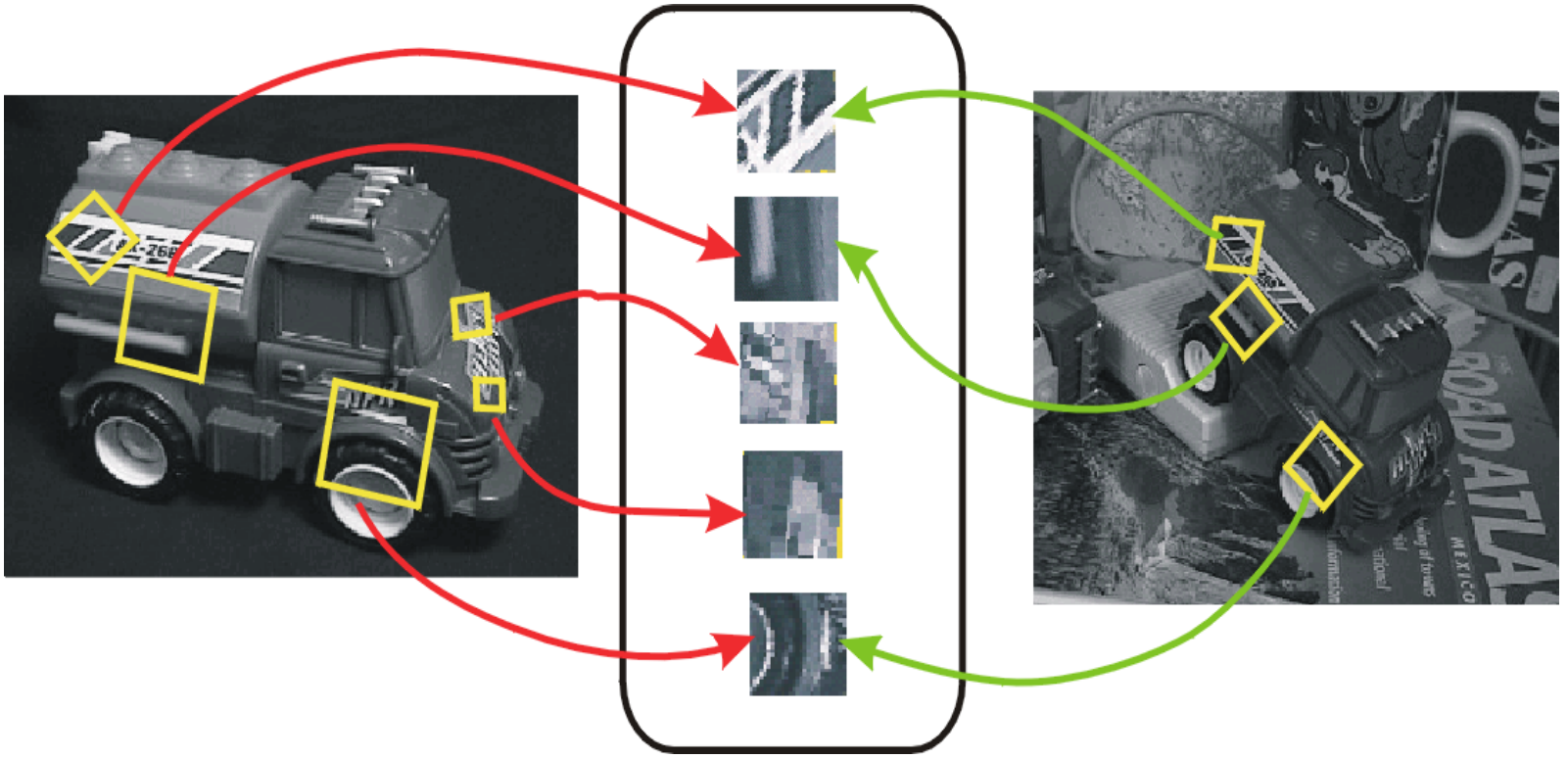


NASA Mars Rover images
with SIFT feature matches

Invariant Local Features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …



Feature Descriptors

Advantages of Local Features

Locality

- features are local, so robust to occlusion and clutter

Distinctiveness:

- can differentiate a large database of objects

Quantity

- hundreds or thousands in a single image

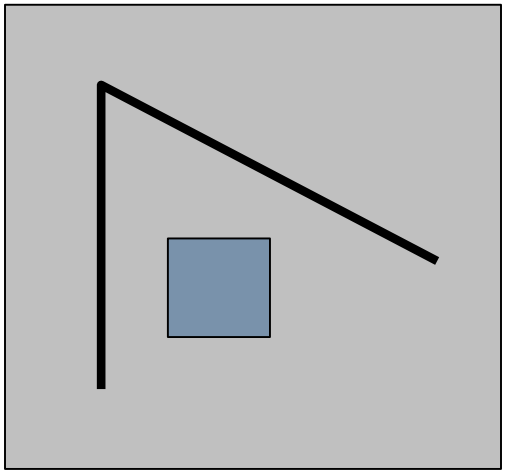
Efficiency

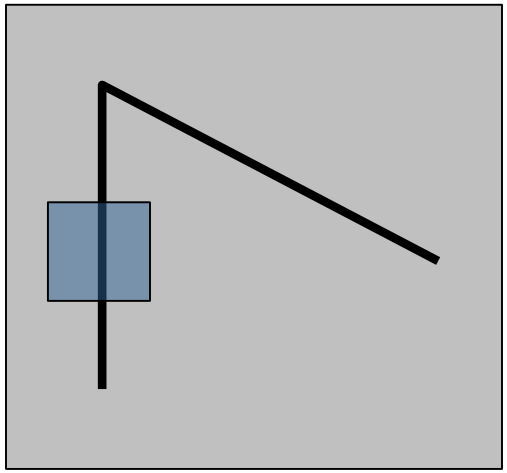
- real-time performance achievable

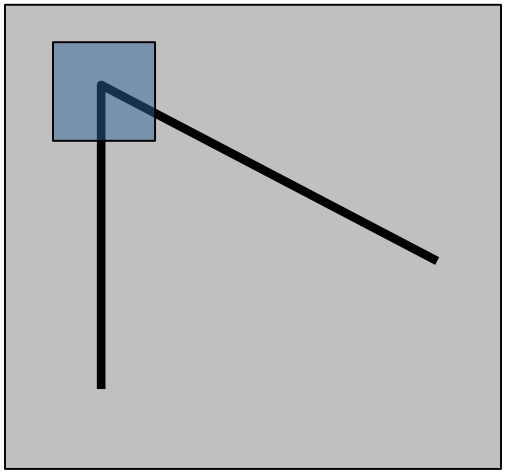
Local Measures of Uniqueness

Suppose we only consider a small window of pixels

- What defines whether a feature is well localized and unique?

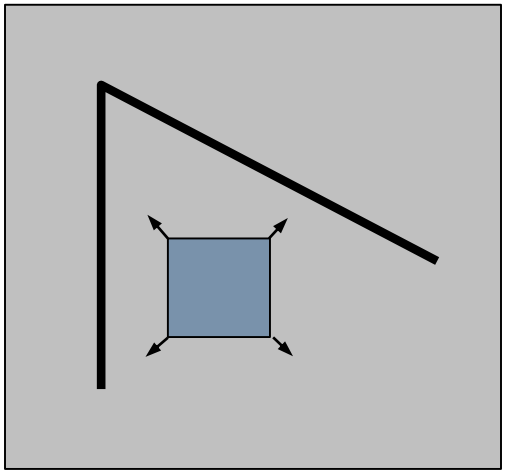




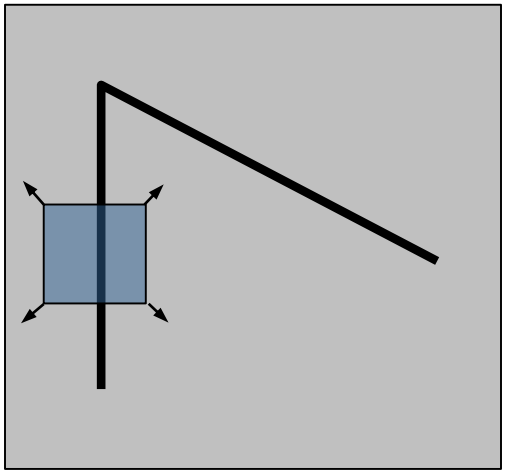


Local Measure of Uniqueness

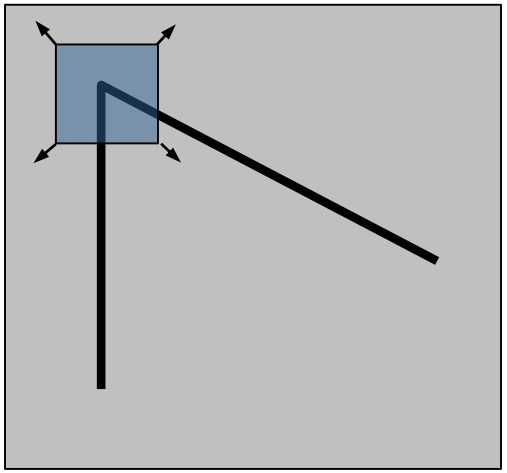
- How does the window change when you shift by a small amount?



"flat" region:
no change in all directions



"edge":
no change along the edge direction

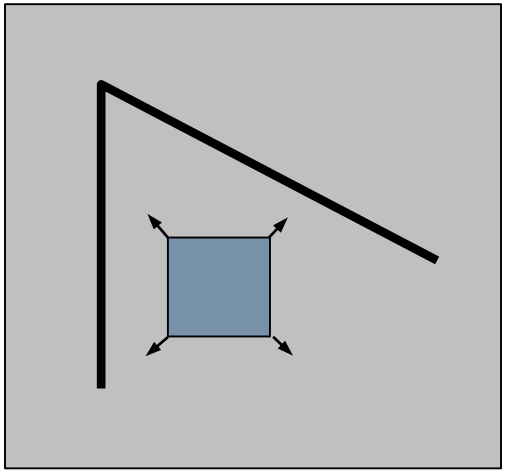


"corner":
significant change in all directions

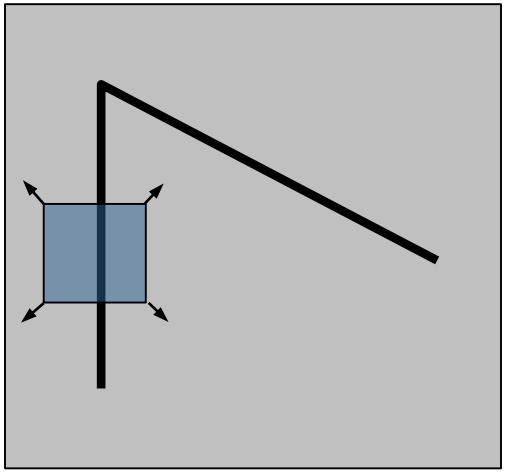
Locally Unique Features (Corners)

Define

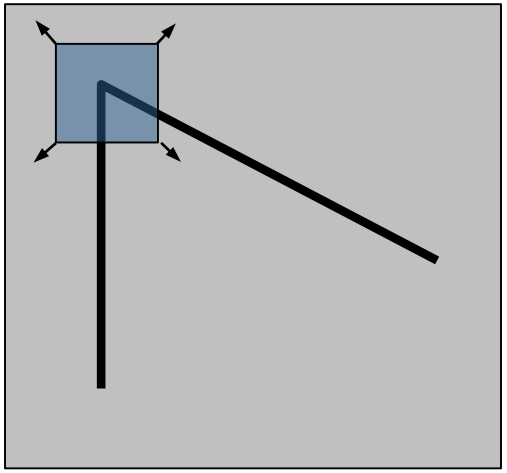
E(u,v) = amount of change when you shift the window by (u,v)



E(u,v) is small
for all shifts



E(u,v) is small
for some shifts



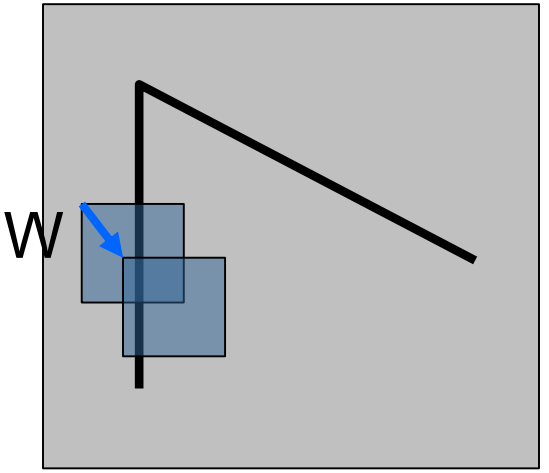
E(u,v) is small
for no shifts

We want $\min_{(u,v)} E(u,v)$ to be?

Corner Detection

Consider shifting the window W by (u,v)

- how do the pixels in W change?
- compare each pixel before and after by Sum of the Squared Differences (SSD)
- this defines an SSD “error” E(u,v):



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

Sum of Squared Differences (SSD)

Small Motion Assumption

Taylor Series expansion of I:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (u,v) is small, then first order approx is good

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

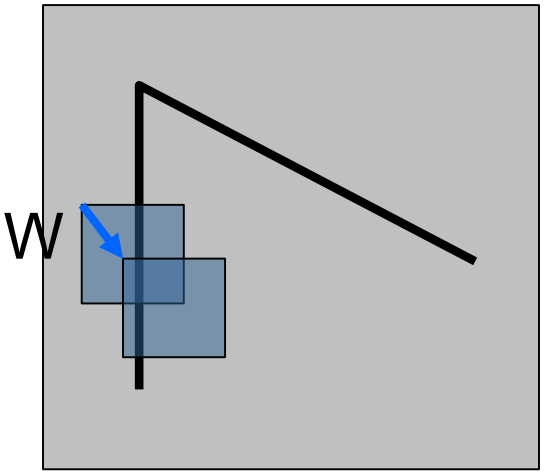
$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide…

Corner Detection

Consider shifting the window W by (u,v)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences
- this defines an “error” of E(u,v):

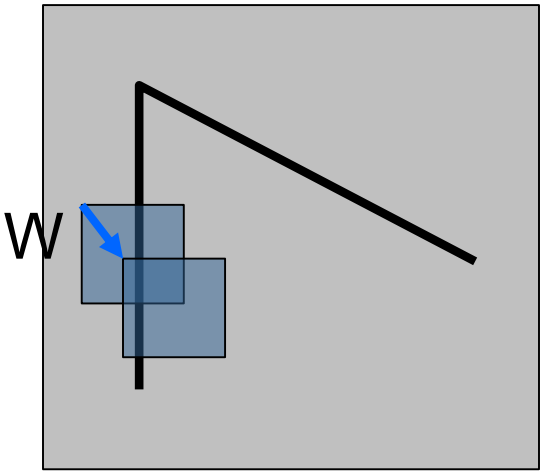


$$E(u,v) = \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2$$

Corner Detection

Consider shifting the window W by (u,v)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences
- this defines an "error" of E(u,v):



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2$$

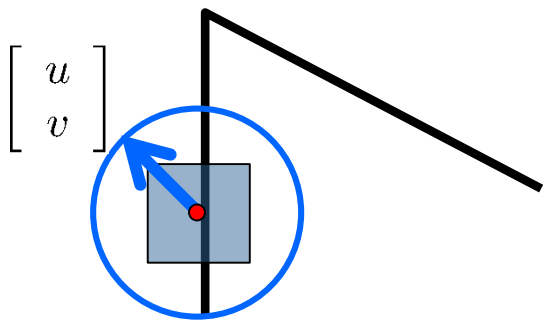
$$\approx \sum_{(x,y)\in W} [I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)]^2$$

$$\approx \sum \left[[I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$

Structure Tensor

This can be rewritten:

$$E(u,v) = \sum_{(x,y)\in W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$



For the example above

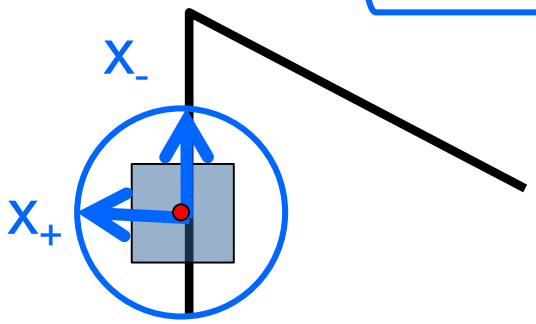
- You can move the center of the green window to anywhere on the blue unit circle
- Which directions will result in the largest and smallest E values?
- We can find these directions by looking at the eigenvectors of H

Structure Tensor

This can be rewritten:

$$E(u,v) = [u\ v] \underbrace{\left(\sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \right)}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

H „structure tensor“



Eigenvalues and eigenvectors of H

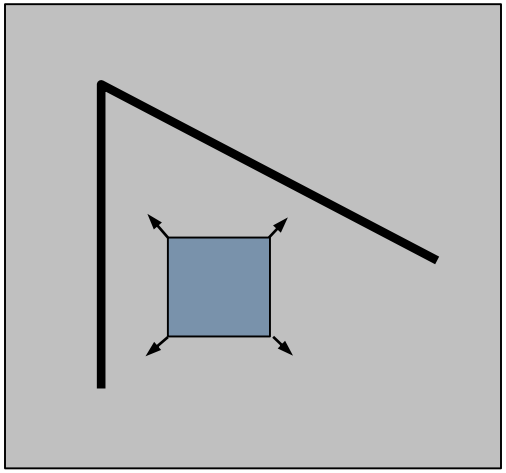
- Define shifts with the smallest and largest change (E value)
- x_+ = direction of largest increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of smallest increase in E.
- λ- = amount of increase in direction x_-

$$Hx_+ = \lambda_+ x_+$$
$$Hx_- = \lambda_- x_-$$

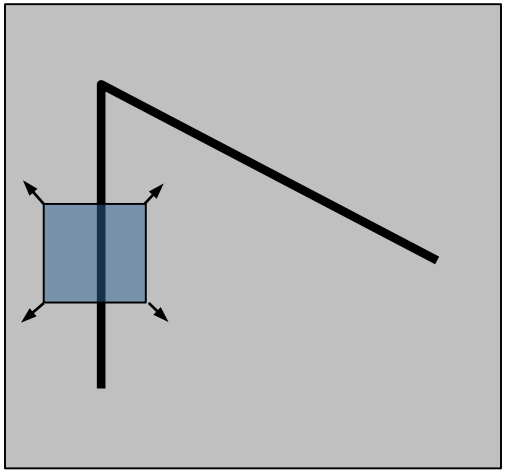
Corner Detection

Define

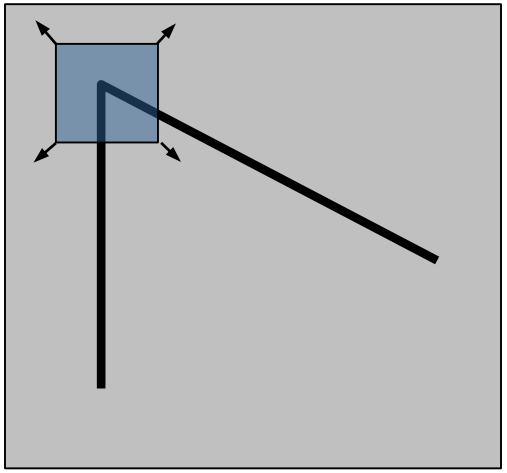
E(u,v) = amount of change when you shift the window by (u,v)



E(u,v) is small
for all shifts



E(u,v) is small
for some shifts

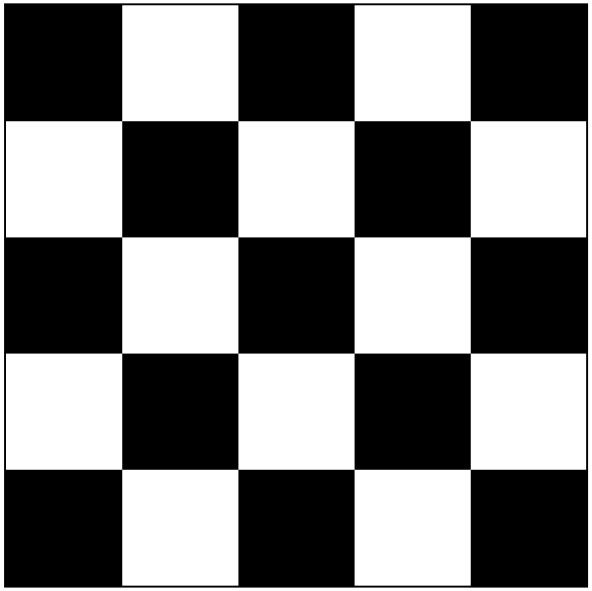


E(u,v) is small
for no shifts

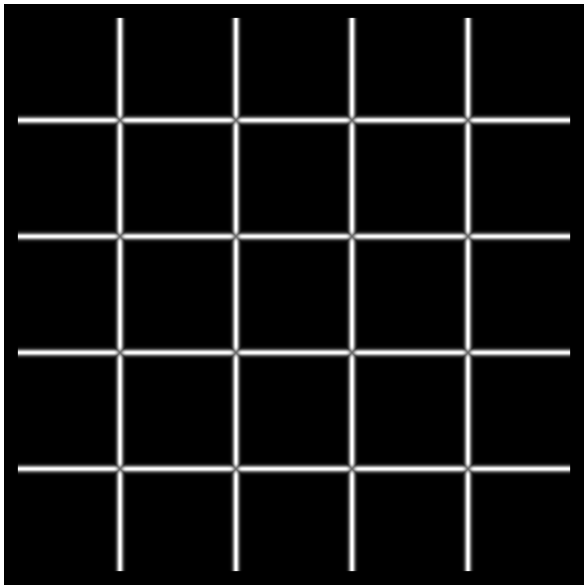
We want $\min_{(u,v)} E(u,v)$ to be large: maximize λ_

Corner Detection Recipe

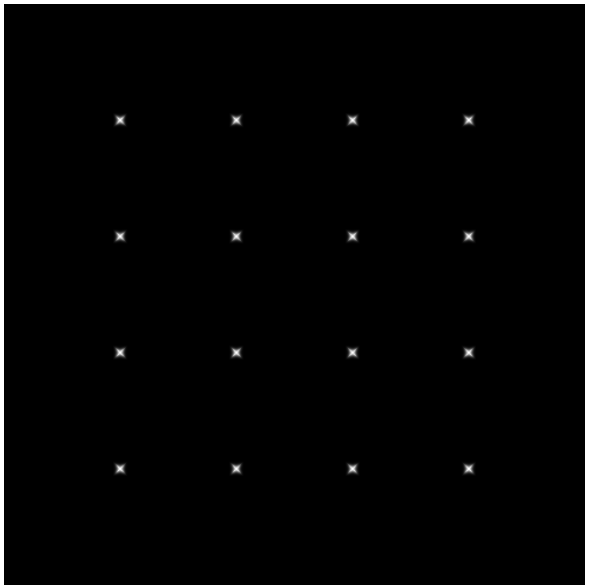
- Compute the gradient at each point in the image
- Create the H matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features



I



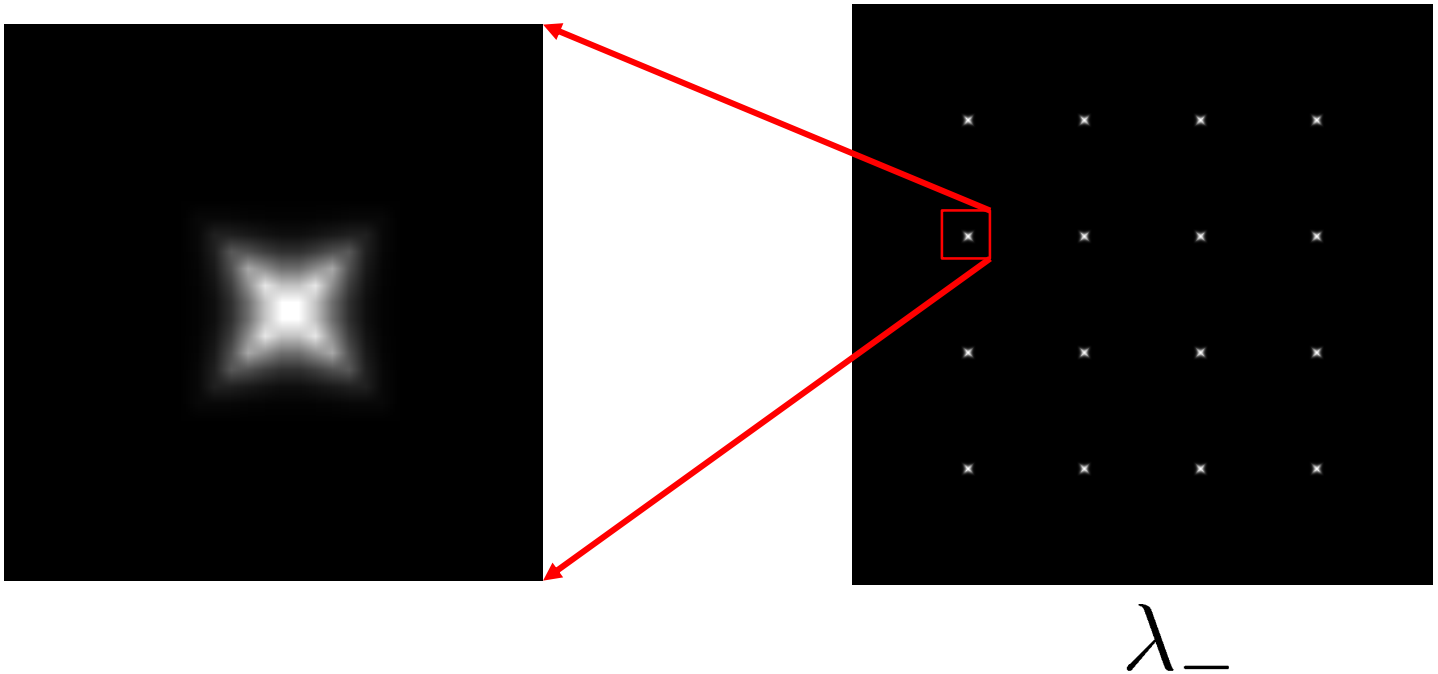
λ_+



λ_-

Corner Detection Recipe

- Compute the gradient at each point in the image
- Create the H matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features



Harris Operator

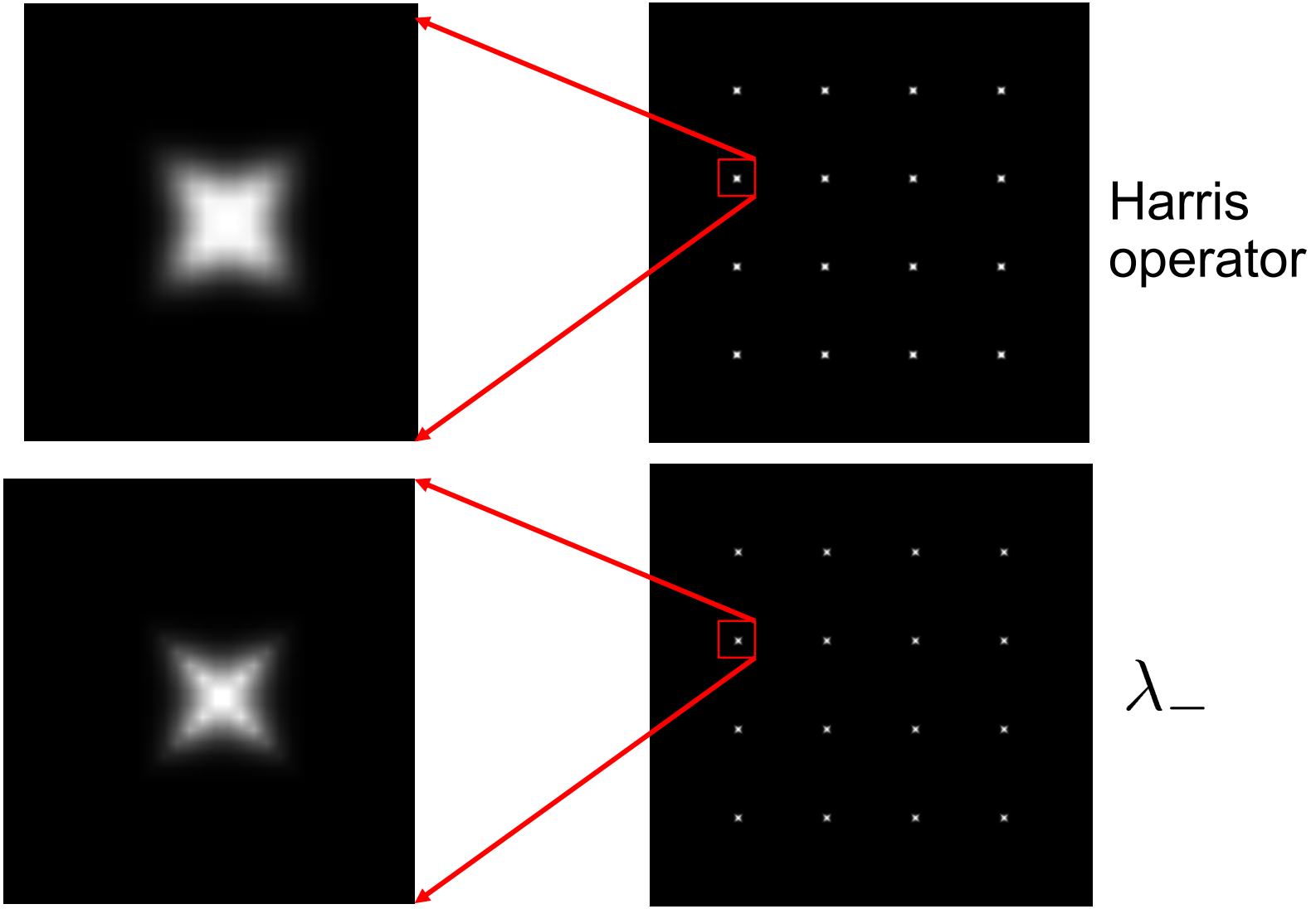
- λ_- is a variant of the "Harris operator" for corner detection

$$f = \frac{\lambda_- \lambda_+}{\lambda_- + \lambda_+}$$

$$= \frac{determinant(H)}{trace(H)}$$

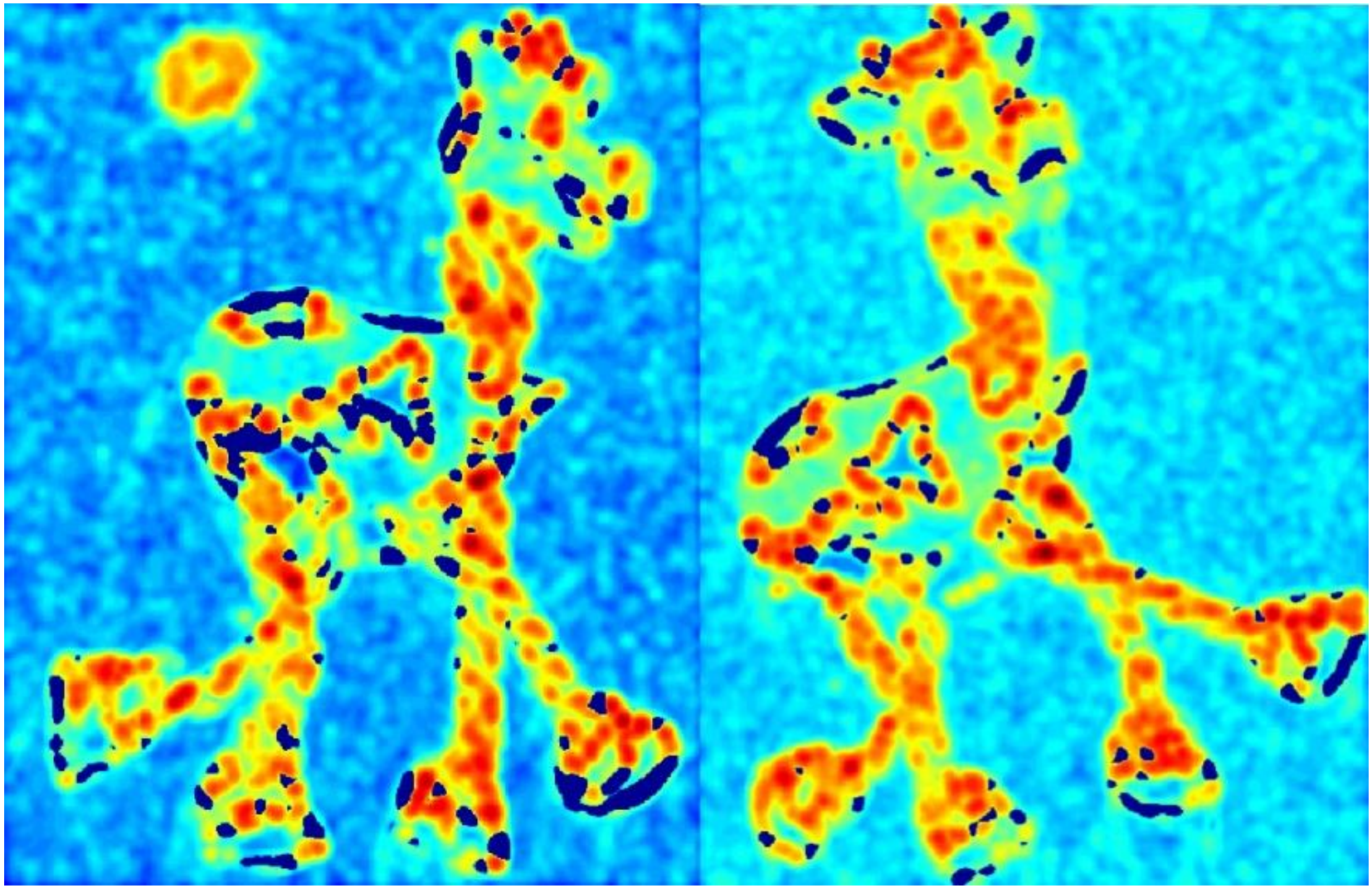
  - The trace is the sum of the diagonals, i.e., trace(H) = $h_{11} + h_{22}$
  - Very similar to λ_- but less expensive (no square root)
  - Called the "Harris Corner Detector" or "Harris Operator"
  - Lots of other detectors, this is one of the most popular

Harris Operator

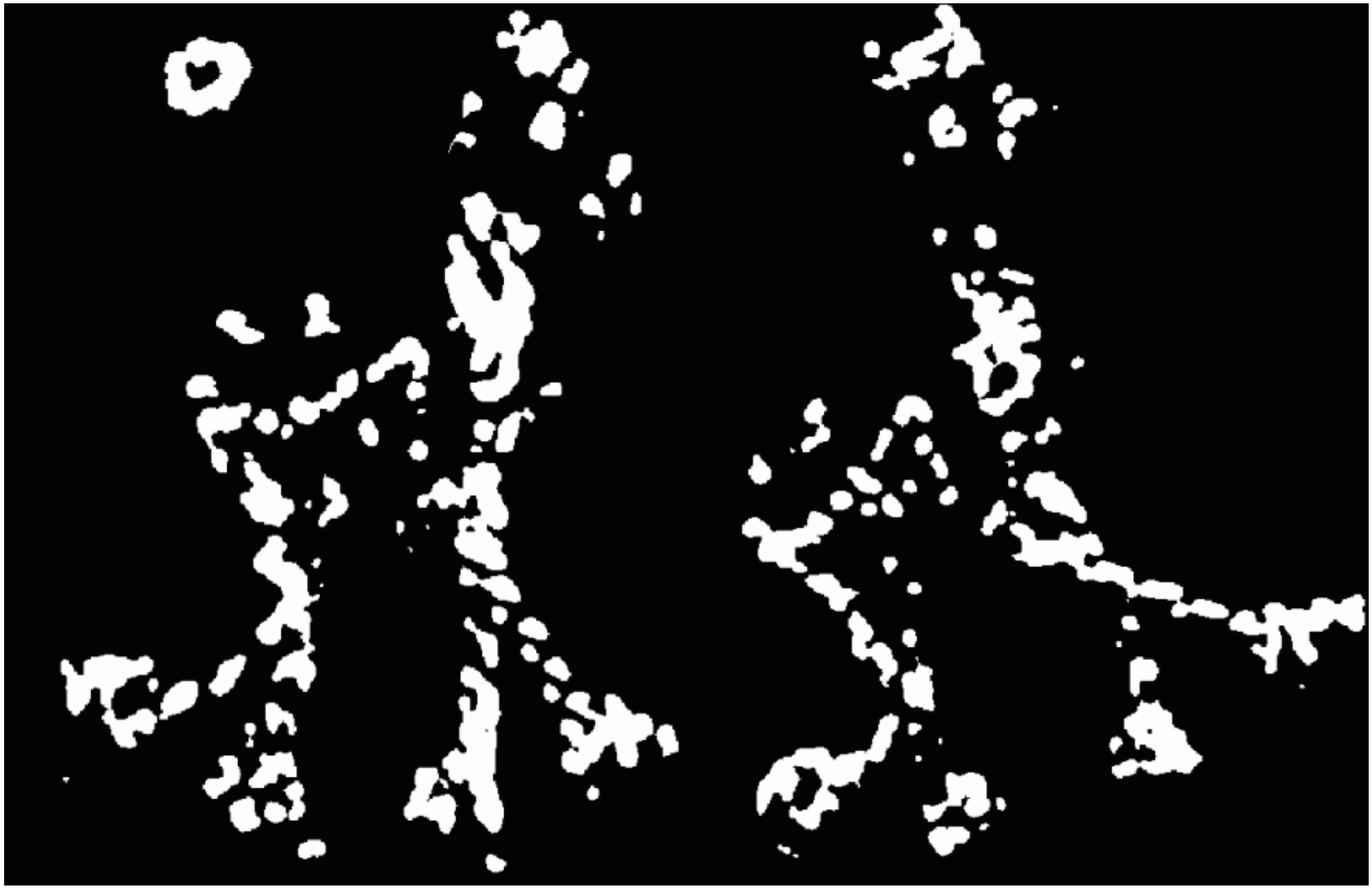


Harris Detector Example

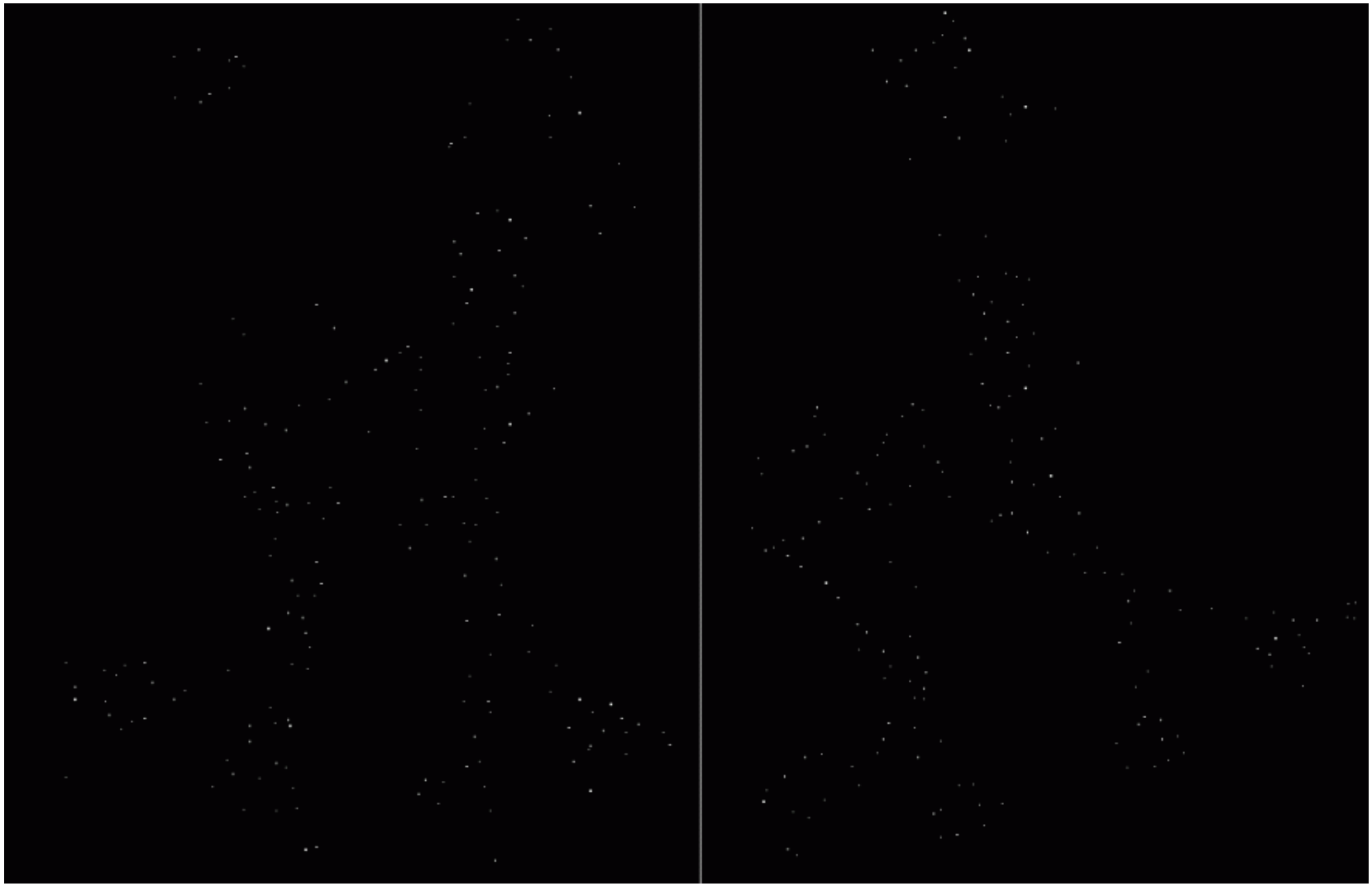
Harris Corner Response



Thresholded Harris Corner Response



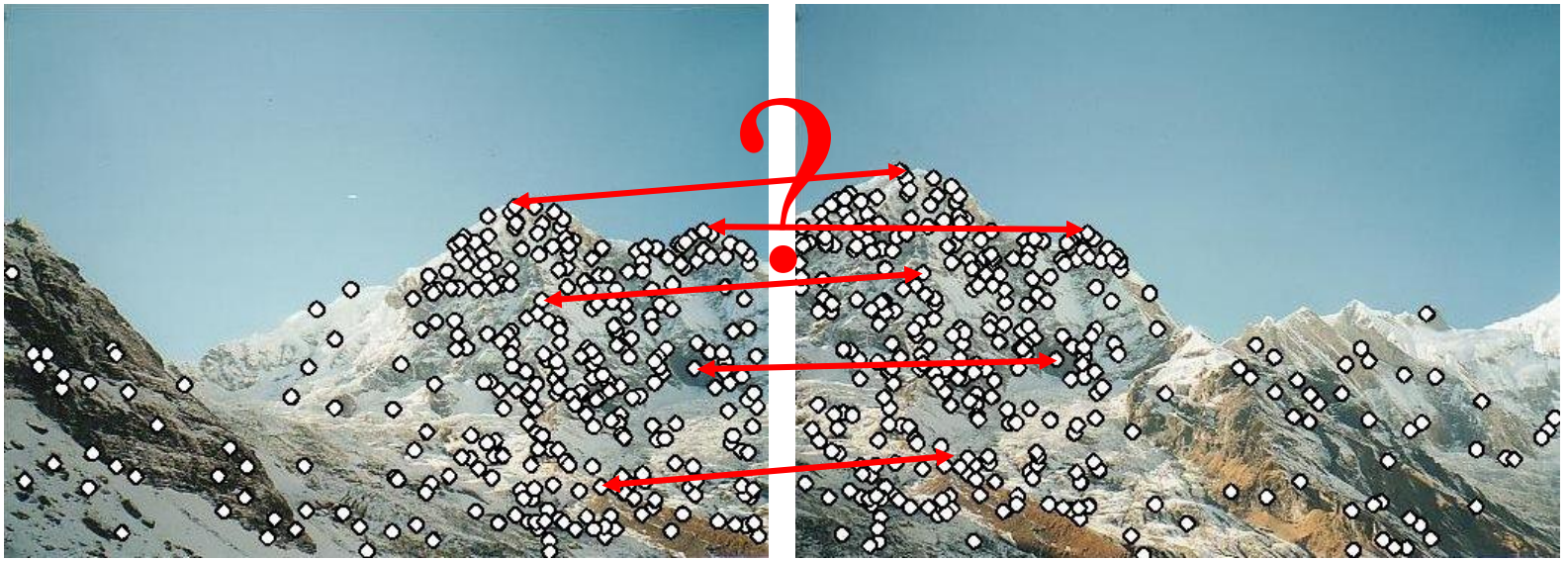
Local Maxima of Harris Corner Response



Harris Corners

Keypoint Descriptors

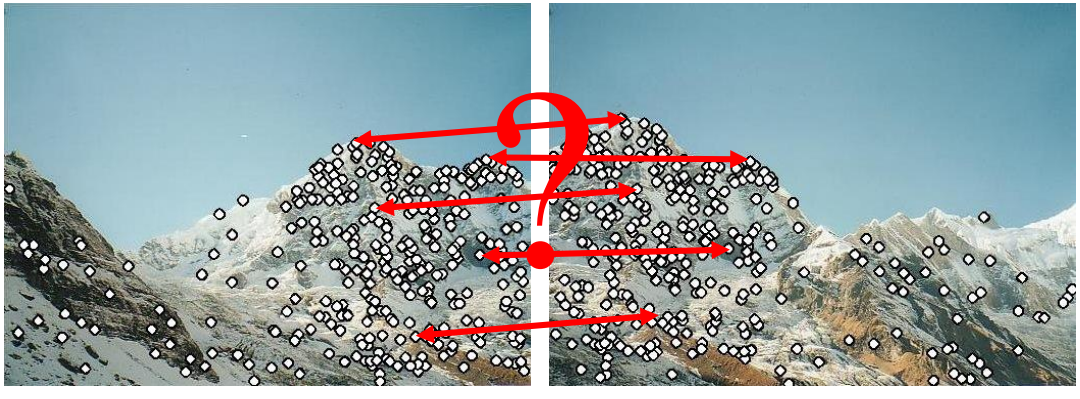
- We know how to detect good points
- Next question: How to match them?



- Idea: extract distinctive descriptor vector from a local patch around the keypoint

Invariance

- Goal: match keypoints regardless of image transformation
  - This is called transformational invariance
- Most keypoint detection and description methods 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



Invariant Detection and Description

- Make sure your detector is invariant
  - Harris is invariant to translation and rotation
  - Scale is trickier
    - Scale selection for blobs (f.e. SIFT)
    - Keypoints at multiple scales for same location

- Design an invariant feature descriptor
  - A descriptor captures the information in a region around the detected feature point
  - The simplest descriptor: a square window of pixels
    - What's this invariant to?
  - Let's look at some better approaches…

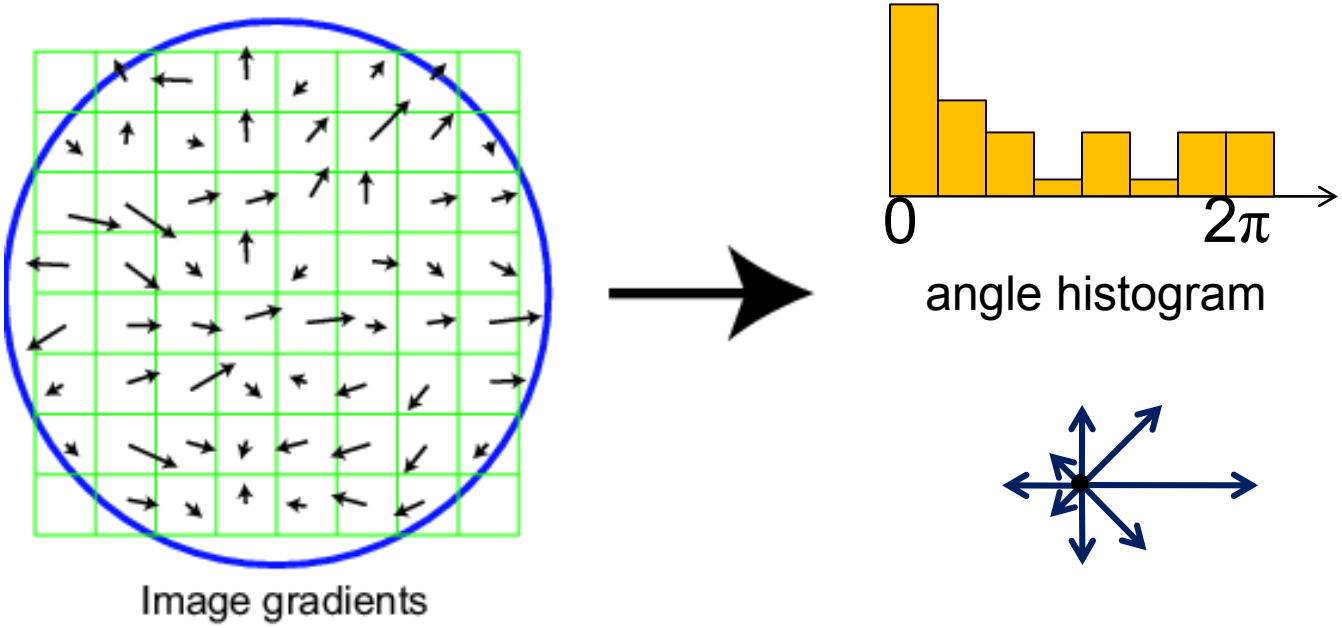
2D Rotation Invariance

- Idea: align the descriptor with a dominant 2D orientation
- Example approach: Use the eigenvector of H corresponding to larger eigenvalue

Figure by Matthew Brown

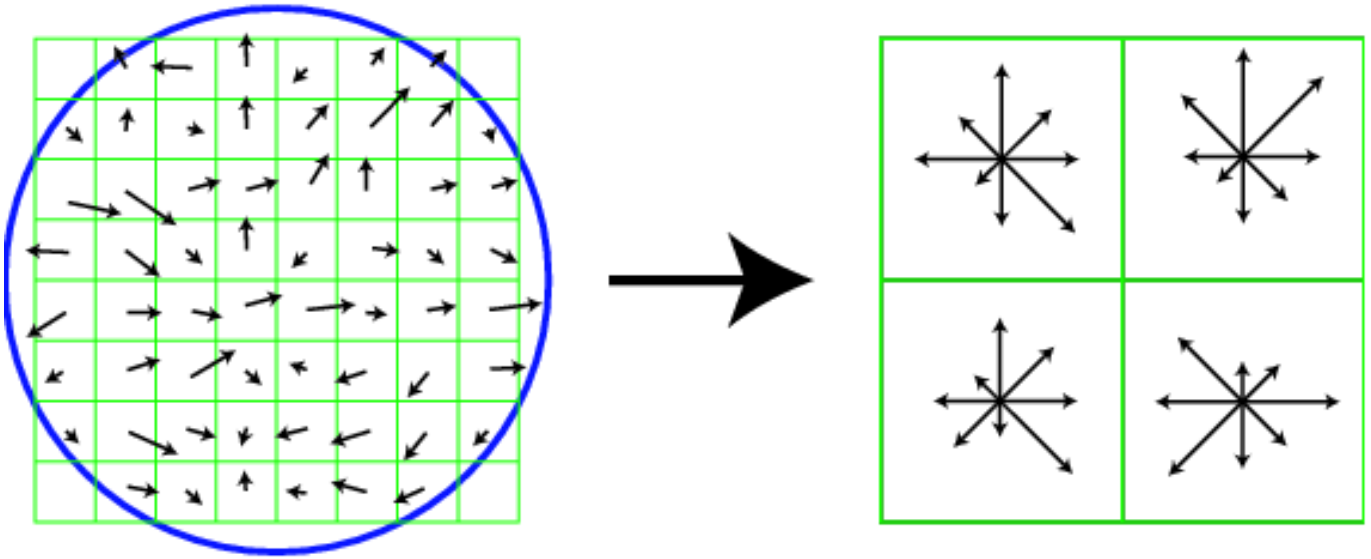
Scale Invariant Feature Transform (SIFT)

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations
- Select two strongest orientations and create two descriptors



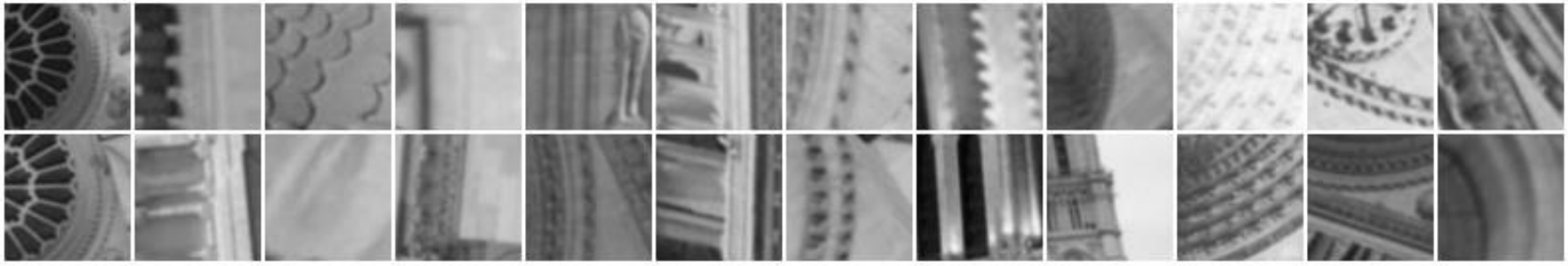
SIFT Descriptor

- 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



Properties of SIFT

- 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
- But: false positive matches



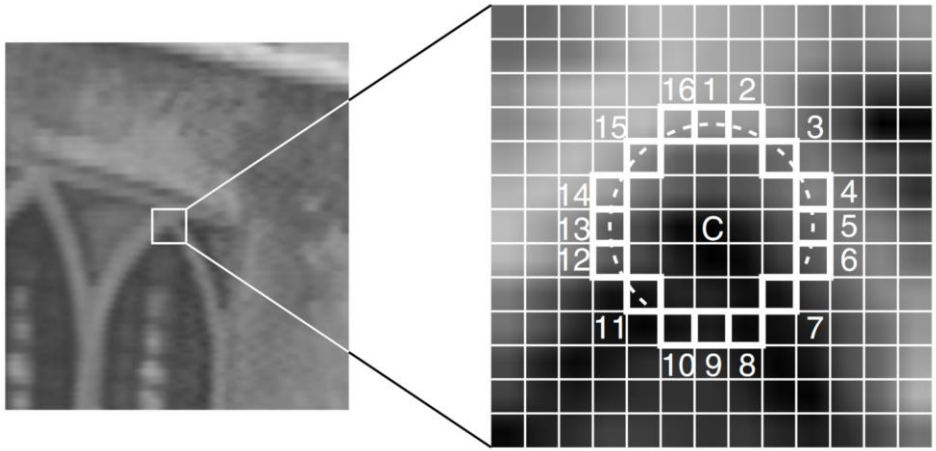
SURF

- Speeded Up Robust Features
- Approximates LoG and descriptor calculation in SIFT using Haar wavelets
  - Faster computation
  - Similar performance like SIFT

Bay, Tuytelaars, Van Gool, Speeded Up Robust Features, ECCV 2006

FAST Detector

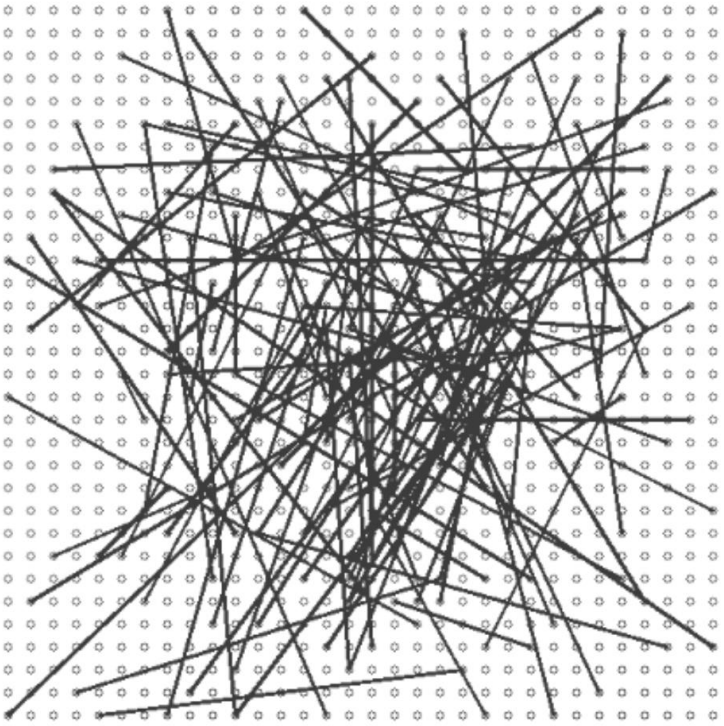
- Features from Accelerated Segment Test
- Check relation of brightness values to center pixel along circle
- Specific number of contiguous pixels brighter or darker than center
- Very fast corner detection



Rosten, Drummond, Fusing Points and Lines for High Performance Tracking, ICCV 2005

BRIEF Descriptor

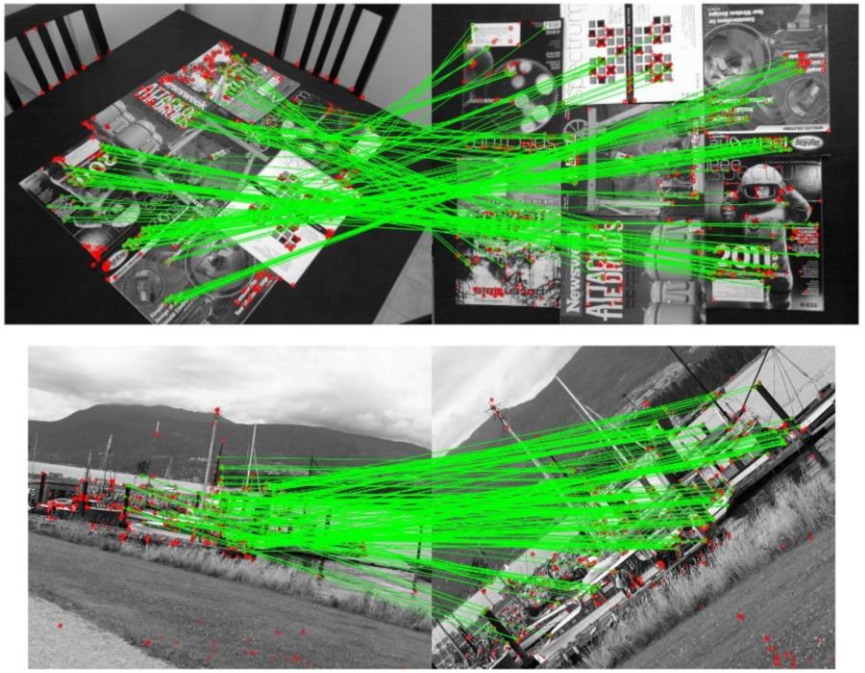
- Binary Robust Independent Elementary Features
- Binary descriptor from intensity comparisons at sample positions
- Very efficient to compute
- Fast matching distance through Hamming distance



Calonder, Lepetit, Strecha, Fua, BRIEF: Binary Robust Independent Elementary Features, ECCV'10

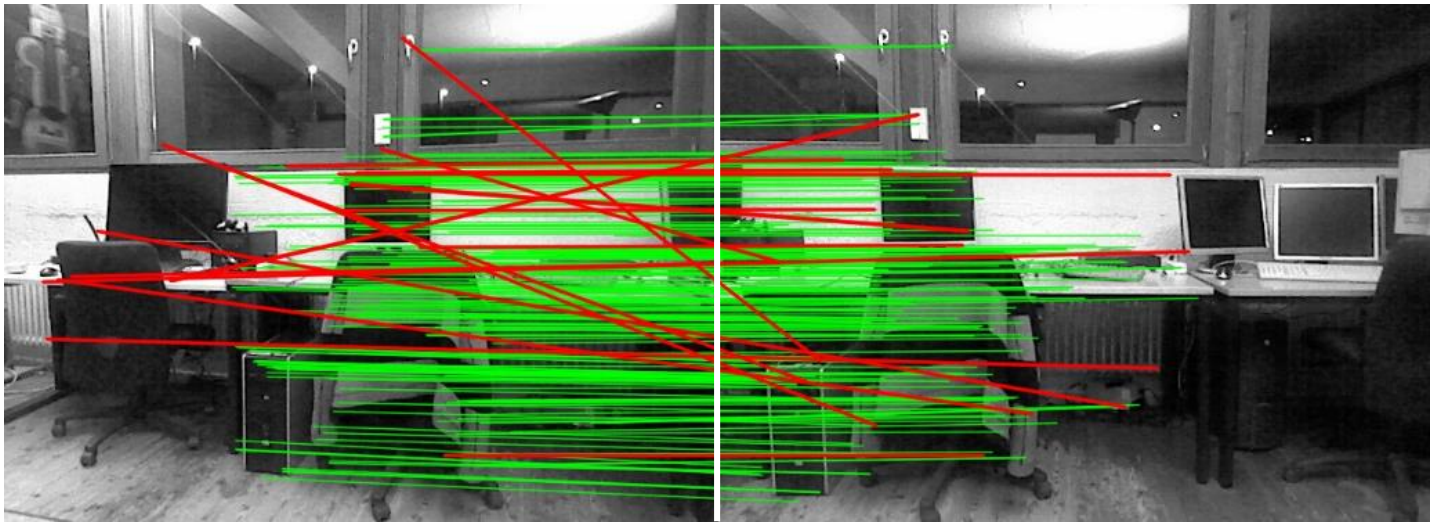
ORB Descriptor

- Oriented Fast and Rotated BRIEF
  - Combination of FAST detector and BRIEF descriptor
  - Rotation-invariant BRIEF: Estimate dominant orientation from patch moments
- Very popular for VO



Rublee, Rabaud, Konolige, Bradski, ORB: an efficient alternative to SIFT or SURF, ICCV 2011

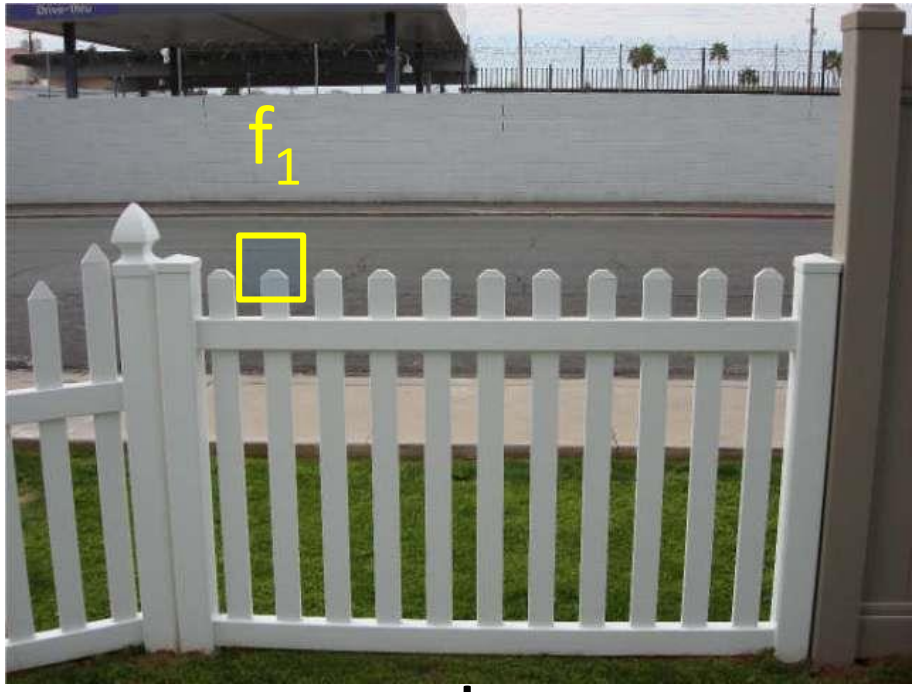
Keypoint Matching



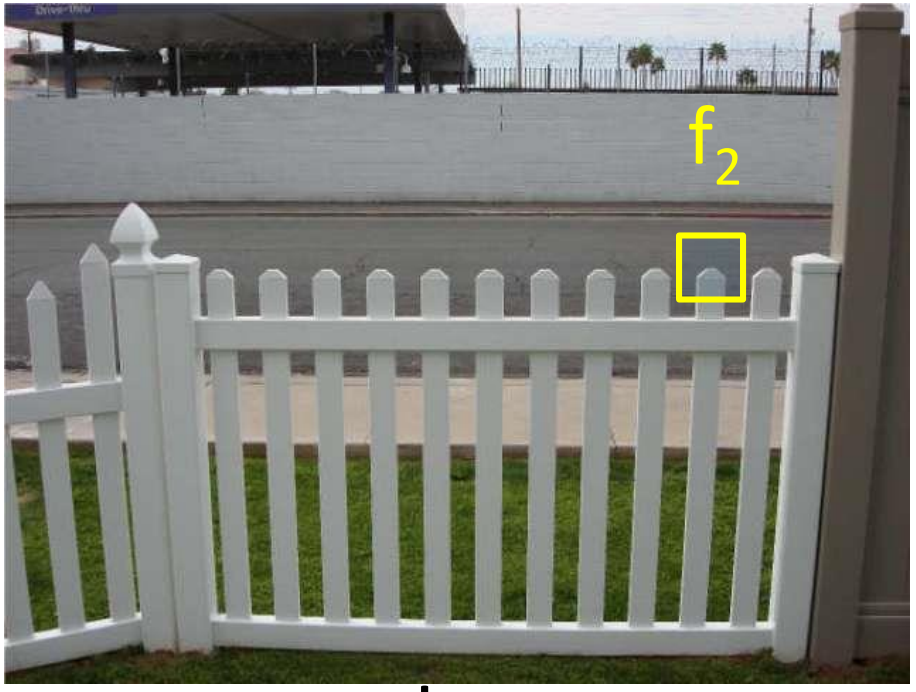
- Match keypoints with similar descriptors

Matching Distance

- How to define the difference between two descriptors f1, f2?
- Simple approach is to assign keypoints with minimal sum of square differences SSD(f1, f2) between entries of the two descriptors



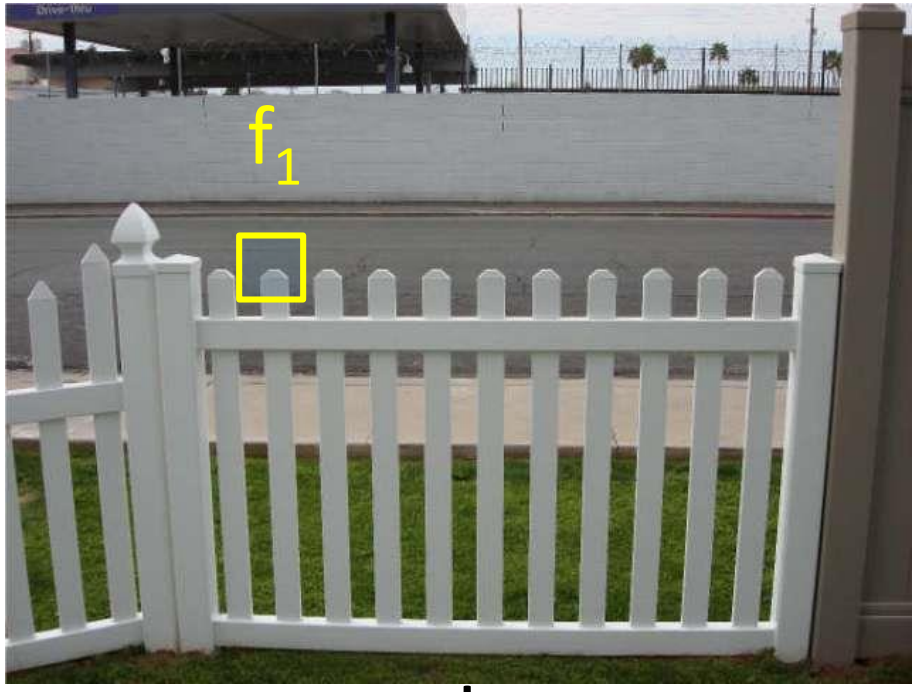
I_1



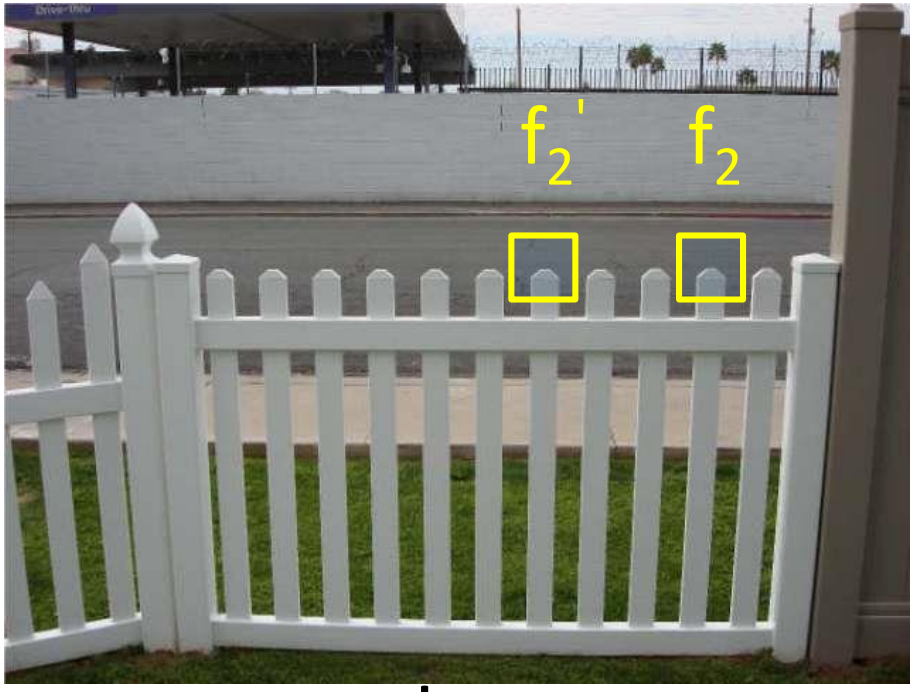
I_2

Matching Distance

- Better approach:
  best to second best ratio distance = SSD(f1, f2) / SSD(f1, f2')
  - f2 is best SSD match to f1 in I2
  - f2' is 2nd best SSD match to f1 in I2

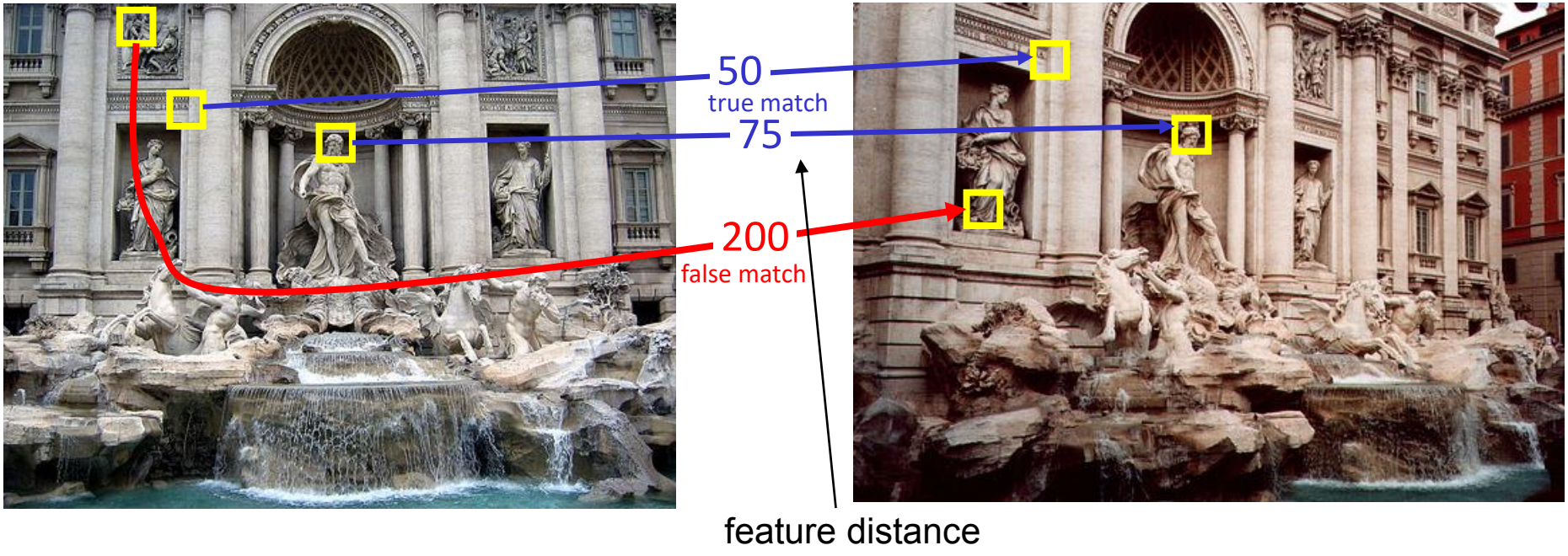


I_1



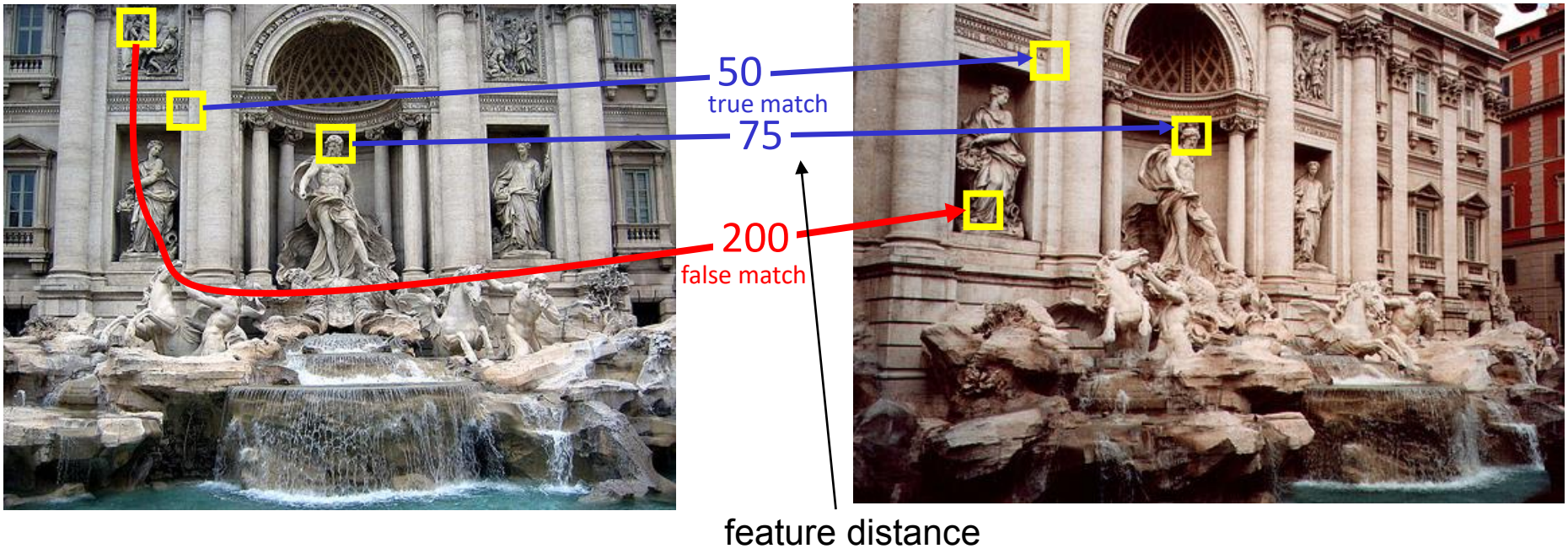
I_2

Eliminating Bad Matches



- Only accept matches with distance smaller a threshold
- How to choose the threshold?

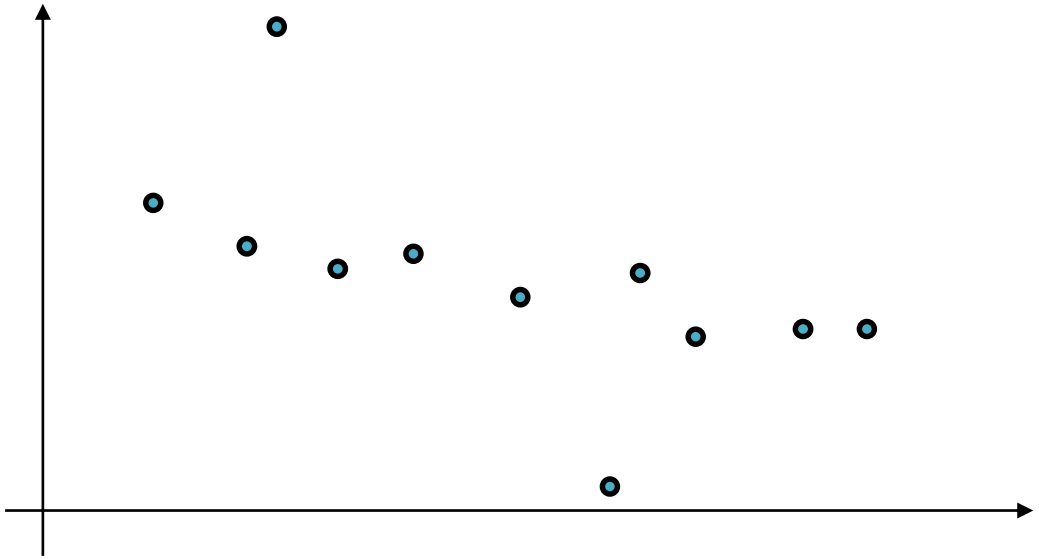
True/False Positives



- Choice of threshold affects performance
  - Too restrictive: less false positives (#false matches) but also less true positives (#true matches)
  - Too lax: more true positives but also more false positives
- Can we do more?

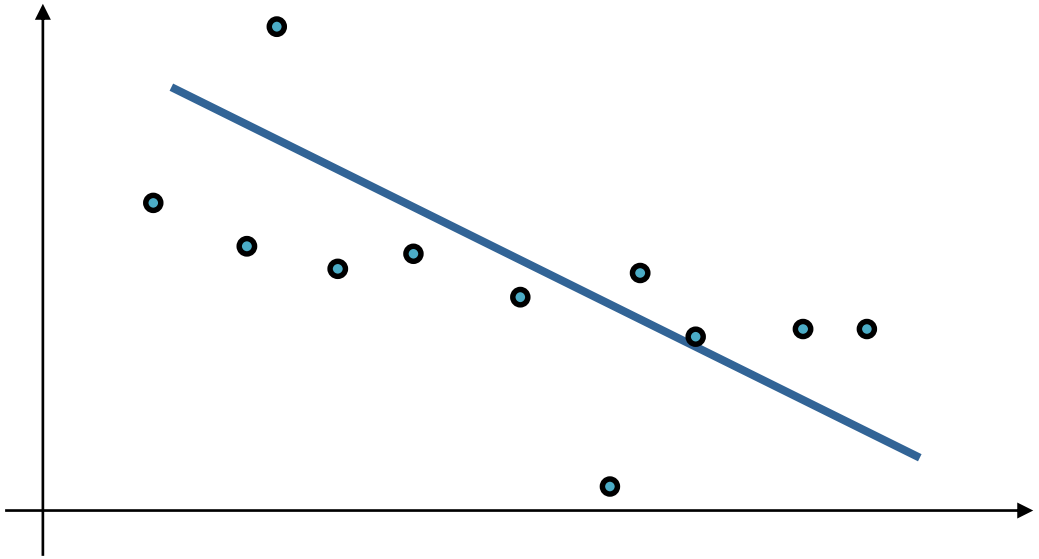
Random Sample Consensus (RANSAC)

- Model fitting in presence of noise and outliers
- Example: fitting a line through 2D points



RANSAC

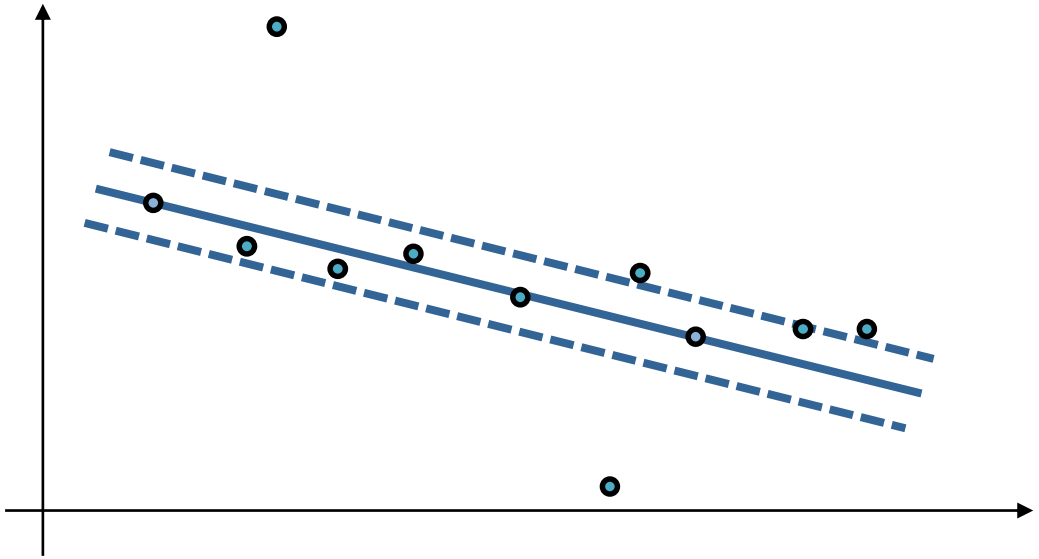
- Least-squares solution, assuming constant noise for all points



Bad!

RANSAC

- We only need 2 points to fit a line. Let's try 2 random points

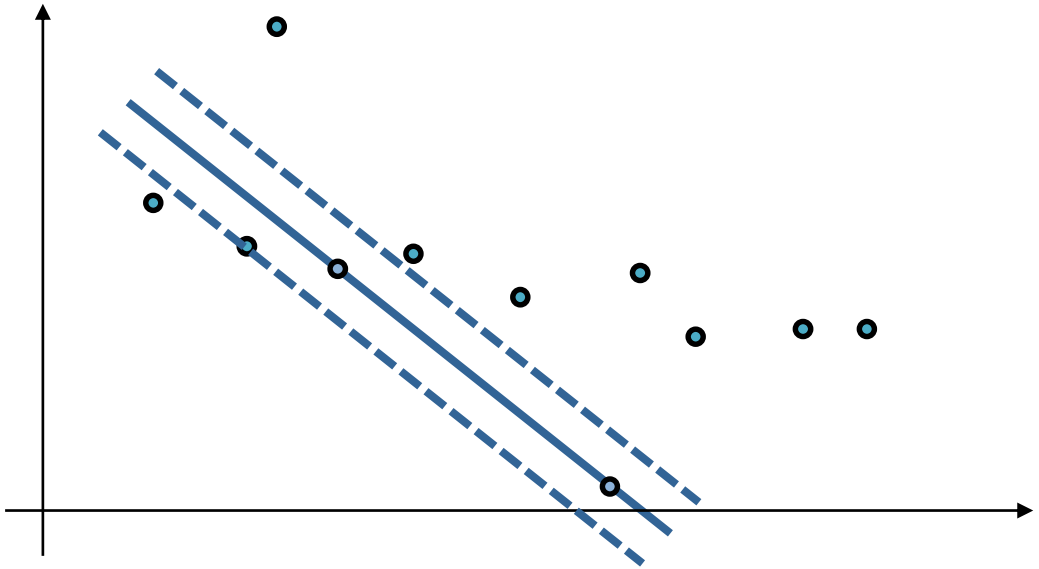


Quite ok..

7 inliers
4 outliers

RANSAC

- Let’s try 2 other random points

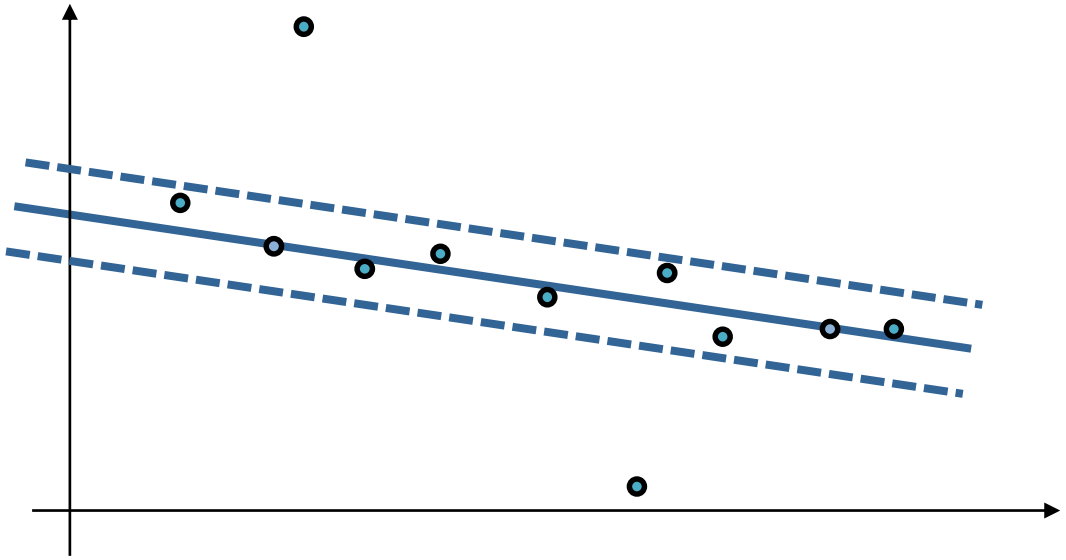


Quite bad..

3 inliers
8 outliers

RANSAC

- Let’s try yet another 2 random points

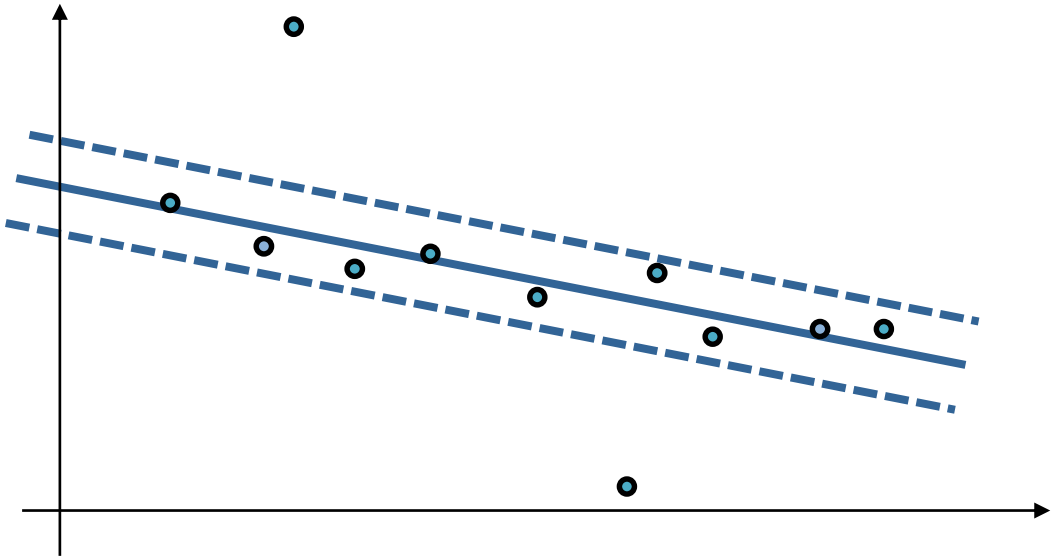


Quite good!

9 inliers
2 outliers

RANSAC

- Let’s use the inliers of the best trial so far to perform least squares fitting



Even better!

RANSAC Algorithm

- RANdom SAmple Consensus algorithm formalizes this idea
- Algorithm:

Input: data D, s required #data points for fitting, success probability p, outlier ratio ϵ

Output: inlier set

1. Compute required number of iterations $N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}$
2. For N iterations do:
   1. Randomly select a subset of s data points
   2. Fit model on the subset
   3. Count inliers and keep model/subset with largest number of inliers
3. Refit model using found inlier set

RANSAC

N for $p = 0.99$

	Required points	Outlier ratio ϵ						
	s	10%	20%	30%	40%	50%	60%	70%
Line	2	3	5	7	11	17	27	49
Plane	3	4	7	11	19	35	70	169
Essential matrix	8	9	26	78	272	1177	7025	70188

Lessons Learned Today

- Keypoint detection, description and matching is a well researched topic
- Highly performant corner and blob detectors exist
- Corners are optimized for localization accuracy
- Blobs have a natural notion of scale through the scale-normalized LoG
- ORB is currently most popular detector/descriptor combination for visual motion estimation
- Keypoint matching by descriptor distance
- Robust matching based on model fitting using RANSAC

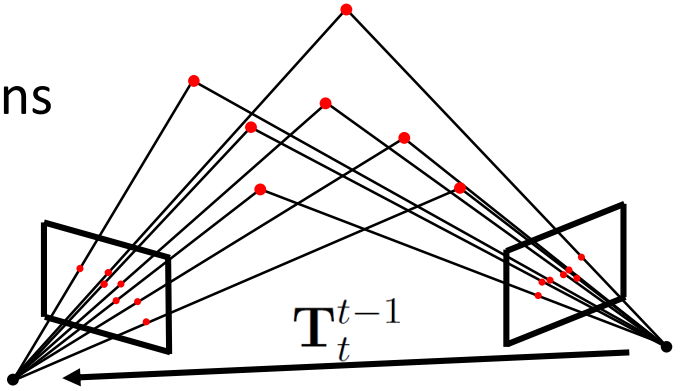
Recap: 2D-to-2D Motion Estimation

- Given corresponding image point observations

$$\mathcal{Y}_t = \{\mathbf{y}_{t,1}, \ldots, \mathbf{y}_{t,N}\}$$

$$\mathcal{Y}_{t-1} = \{\mathbf{y}_{t-1,1}, \ldots, \mathbf{y}_{t-1,N}\}$$

of unknown 3D points $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
(expressed in camera frame at time t)
determine relative motion $\mathbf{T}_t^{t-1}$ between frames



- Naive try: minimize reprojection error using least squares

$$E(\mathbf{T}_t^{t-1}, \mathcal{X}) = \sum_{i=1}^{N} \left\| \bar{\mathbf{y}}_{t,i} - \pi\left(\bar{\mathbf{x}}_i\right) \right\|_2^2 + \left\| \bar{\mathbf{y}}_{t-1,i} - \pi\left(\mathbf{T}_t^{t-1} \bar{\mathbf{x}}_i\right) \right\|_2^2$$

- Convexity? Uniqueness (scale-ambiguity)?
- Alternative algebraic approach

Recap: Eight-Point Algorithm

- First proposed by Longuet and Higgins, Nature 1981
- Algorithm:
  1. Rewrite epipolar constraints as a linear system of equations

$$\widetilde{\mathbf{y}}_i^\top \mathbf{E} \widetilde{\mathbf{y}}_i' = \mathbf{a}_i \mathbf{E}_s = 0 \quad \longrightarrow \quad \mathbf{A}\mathbf{E}_s = \mathbf{0} \qquad \mathbf{A} = \left(\mathbf{a}_1^\top, \ldots, \mathbf{a}_N^\top\right)^\top$$

  using Kronecker product $\mathbf{a}_i = \widetilde{\mathbf{y}}_i \otimes \widetilde{\mathbf{y}}_i'$ and $\mathbf{E}_s = (e_{11}, e_{12}, e_{13}, \ldots, e_{33})^\top$

  2. Apply singular value decomposition (SVD) on $\mathbf{A} = \mathbf{U}_\mathbf{A}\mathbf{S}_\mathbf{A}\mathbf{V}_\mathbf{A}^\top$ and unstack the 9th column of $\mathbf{V}_\mathbf{A}$ into $\widetilde{\mathbf{E}}$

  3. Project the approximate $\widetilde{\mathbf{E}}$ into the (normalized) essential space: Determine the SVD of $\widetilde{\mathbf{E}} = \mathbf{U}\,\mathrm{diag}\,(\sigma_1, \sigma_2, \sigma_3)\,\mathbf{V}^\top$ with $\mathbf{U}, \mathbf{V} \in \mathbf{SO}(3)$ and replace the singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3$ with $1, 1, 0$ to find

$$\mathbf{E} = \mathbf{U}\,\mathrm{diag}(1, 1, 0)\mathbf{V}^\top$$

Recap: Error Metric of the Eight-Point Algorithm

- What is the physical meaning of the error minimized by the eight-point algorithm?
- The eight-point algorithm finds E that minimizes

$$\text{argmin}_{\mathbf{E}_s} \|\mathbf{A}\mathbf{E}_s\|_2^2$$

  subject to $\|\mathbf{E}_s\|_2^2 = 1$ through the SVD on A
- We find a least squares fit to the epipolar constraints
- A violated epipolar constraint

$$\widetilde{\mathbf{y}}^\top (\mathbf{t} \times \mathbf{R}\widetilde{\mathbf{y}}') = 0$$

  quantifies the volume spanned by y, t, and Ry‘
- No clear interpretation in terms of distance or angular error

Algorithm: 2D-to-2D Visual Odometry

Input: image sequence $I_{0:t}$, camera calibration

Output: aggregated camera poses $\mathbf{T}_{0:t}$

Algorithm:

For each current image I_k :

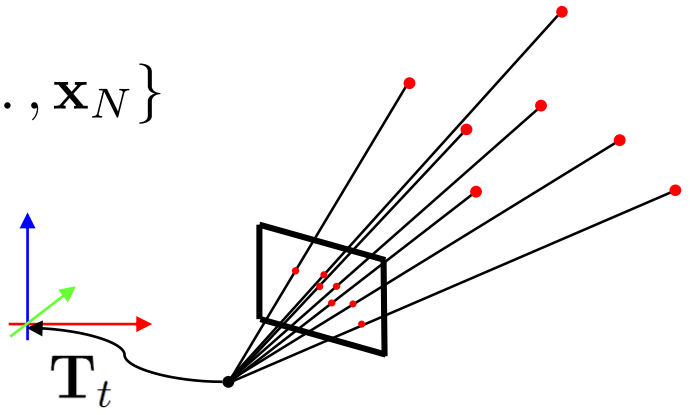
1. Extract and match keypoints between I_{k-1} and I_k
2. Compute relative pose $\mathbf{T}_k^{k-1}$ from essential matrix between I_k , I_{k-1}
3. Fine-tune pose estimate by minimizing reprojection error
4. Compute relative scale and rescale translation of $\mathbf{T}_k^{k-1}$
5. Aggregate camera pose by $\mathbf{T}_k = \mathbf{T}_{k-1}\mathbf{T}_k^{k-1}$

2D-to-3D Motion Estimation

- Given a local set of 3D points $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and corresponding image observations

  $$\mathcal{Y}_t = \{\mathbf{y}_{t,1}, \ldots, \mathbf{y}_{t,N}\}$$

  determine camera pose $\mathbf{T}_t$ within the local map



- Minimize least squares geometric reprojection error

$$E(\mathbf{T}_t) = \sum_{i=1}^{N} \left\| \mathbf{y}_{t,i} - \pi(\mathbf{T}_t^{-1}\mathbf{x}_i) \right\|_2^2$$

- A.k.a. Perspective-n-Points (PnP) problem, many approaches exist, f.e.
  - Direct linear transform (DLT)
  - EPnP (Lepetit et al., An accurate O(n) Solution to the PnP problem, IJCV 2009)
  - OPnP (Zheng et al., Revisiting the PnP Problem: A Fast, General and Optimal Solution, ICCV 2013)

Direct Linear Transform for PnP

- Goal: determine projection matrix $\mathbf{P} = (\mathbf{R}\ \mathbf{t}) \in \mathbb{R}^{3\times 4} = \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix}$
- Each 2D-to-3D point correspondence
  3D: $\widetilde{\mathbf{x}}_i = (x_i, y_i, z_i, w_i)^\top \in \mathbb{P}^3$ 2D: $\widetilde{\mathbf{y}}_i = (x'_i, y'_i, w'_i)^\top \in \mathbb{P}^2$
  gives two constraints

$$\begin{pmatrix} \mathbf{0} & -w'_i\widetilde{\mathbf{x}}_i^\top & y'_i\widetilde{\mathbf{x}}_i^\top \\ w'_i\widetilde{\mathbf{x}}_i^\top & \mathbf{0} & -x'_i\widetilde{\mathbf{x}}_i^\top \end{pmatrix} \begin{pmatrix} \mathbf{P}_1^\top \\ \mathbf{P}_2^\top \\ \mathbf{P}_3^\top \end{pmatrix} = \mathbf{0}$$

  through $\widetilde{\mathbf{y}}_i \times (\mathbf{P}\widetilde{\mathbf{x}}_i) = 0$

- Form linear system of equations $\mathbf{A}\mathbf{p} = \mathbf{0}$ with $\mathbf{p} := \begin{pmatrix} \mathbf{P}_1^\top \\ \mathbf{P}_2^\top \\ \mathbf{P}_3^\top \end{pmatrix} \in \mathbb{R}^9$ from $N \geq 6$ correspondences

- Solve for $\mathbf{p}$: determine unit singular vector of $\mathbf{A}$ corresponding to its smallest singular value

Algorithm: 2D-to-3D Visual Odometry

Input: image sequence $I_{0:t}$, camera calibration

Output: aggregated camera poses $\mathbf{T}_{0:t}$

Algorithm:

Initialize:

1. Extract and match keypoints between I_0 and I_1
2. Determine camera pose (essential matrix) and triangulate 3D keypoints X_1

For each current image I_k :

1. Extract and match keypoints between I_{k-1} and I_k
2. Compute camera pose $\mathbf{T}_k$ using PnP from 2D-to-3D matches
3. Triangulate all new keypoint matches between I_{k-1} and I_k and add them to the local map X_k

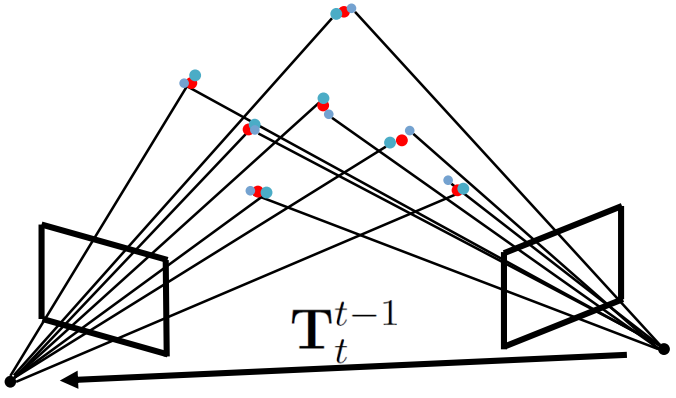
3D-to-3D Motion Estimation

- Given corresponding 3D points in two camera frames

  $\mathcal{X}_{t-1} = \{\mathbf{x}_{t-1,1}, \ldots, \mathbf{x}_{t-1,N}\}$

  $\mathcal{X}_{t} = \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,N}\}$

  determine relative camera pose $\mathbf{T}_t^{t-1}$



- Idea: determine rigid transformation that aligns the 3D points
- Geometric least squares error: $E\left(\mathbf{T}_t^{t-1}\right) = \sum_{i=1}^{N} \left\| \overline{\mathbf{x}}_{t-1,i} - \mathbf{T}_t^{t-1} \overline{\mathbf{x}}_{t,i} \right\|_2^2$
- Closed-form solutions available, f.e. Arun et al., 1987
- Applicable f.e. for calibrated stereo cameras (triangulation of 3D points) or RGB-D cameras (measured depth)

3D Rigid-Body Motion from 3D-to-3D Matches

- Arun et al., Least-squares fitting of two 3-d point sets, IEEE PAMI, 1987
- Corresponding 3D points, $N \geq 3$

$$\mathcal{X}_{t-1} = \{\mathbf{x}_{t-1,1}, \ldots, \mathbf{x}_{t-1,N}\} \qquad \mathcal{X}_t = \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,N}\}$$

- Determine means of 3D point sets

$$\boldsymbol{\mu}_{t-1} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_{t-1,i} \qquad \boldsymbol{\mu}_t = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_{t,i}$$

- Determine rotation from

$$\mathbf{A} = \sum_{i=1}^{N} \left(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}\right)\left(\mathbf{x}_t - \boldsymbol{\mu}_t\right)^\top \qquad \mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top \qquad \mathbf{R}_{t-1}^t = \mathbf{V}\mathbf{U}^\top$$

- Determine translation as $\mathbf{t}_{t-1}^t = \boldsymbol{\mu}_t - \mathbf{R}_{t-1}^t \boldsymbol{\mu}_{t-1}$

Algorithm: Stereo 3D-to-3D Visual Odometry

Input: stereo image sequence $I^l_{0:t}, I^r_{0:t}$, camera calibration (including known pose between stereo cameras)

Output: aggregated camera poses $\mathbf{T}_{0:t}$

Algorithm:

For each current stereo image I^l_k, I^r_k:

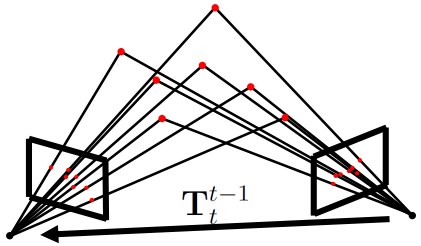
1. Extract and match keypoints between I^l_k and I^l_{k-1}
2. Triangulate 3D points X_k between I^l_k and I^r_k
3. Compute camera pose $\mathbf{T}^{k-1}_k$ from 3D-to-3D point matches X_k to X_{k-1}
4. Aggregate camera pose by $\mathbf{T}_k = \mathbf{T}_{k-1}\mathbf{T}^{k-1}_k$

Motion Estimation from Point Correspondences

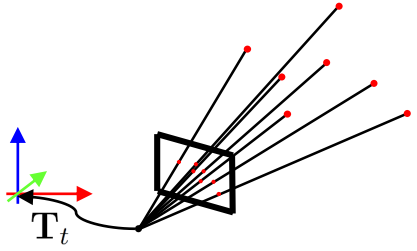
- **2D-to-2D**
  - Reproj. error:

$$E\left(\mathbf{T}_t^{t-1}, X\right) = \sum_{i=1}^{N} \left\|\bar{\mathbf{y}}_{t,i} - \pi\left(\bar{\mathbf{x}}_i\right)\right\|_2^2 + \left\|\bar{\mathbf{y}}_{t-1,i} - \pi\left(\mathbf{T}_t^{t-1}\bar{\mathbf{x}}_i\right)\right\|_2^2$$

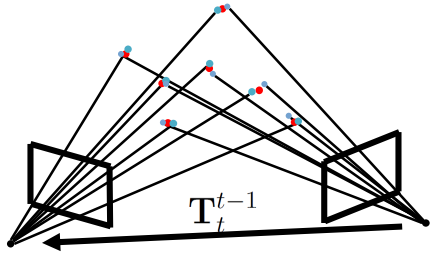
  - Linear algorithm: 8-point



- **2D-to-3D**
  - Reprojection error: $E(\mathbf{T}_t) = \sum_{i=1}^{N} \left\|\mathbf{y}_{t,i} - \pi(\mathbf{T}_t\bar{\mathbf{x}}_i)\right\|_2^2$
  - Linear algorithm: DLT PnP



- **3D-to-3D**
  - Reprojection error: $E\left(\mathbf{T}_t^{t-1}\right) = \sum_{i=1}^{N} \left\|\bar{\mathbf{x}}_{t-1,i} - \mathbf{T}_t^{t-1}\bar{\mathbf{x}}_{t,i}\right\|_2^2$
  - Linear algorithm: Arun‘s method



Thanks for your attention!