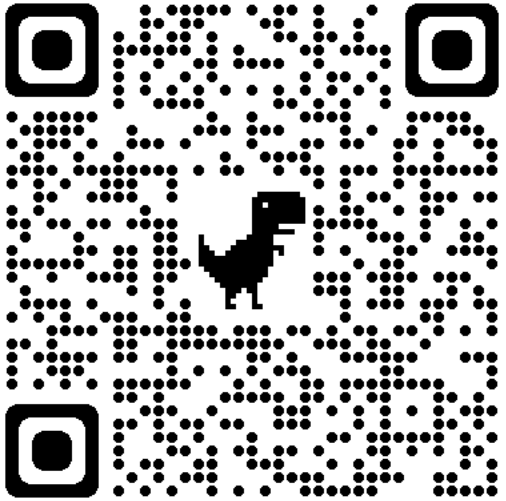
Lecture 8: Features 2

COMP 590/776: Computer Vision

Instructor: Soumyadip (Roni) Sengupta

TA: Mykhailo (Misha) Shvets



Course Website:
Scan Me!

Recap

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features
Step 2: match features
Step 3: align images
Step 4: blending images

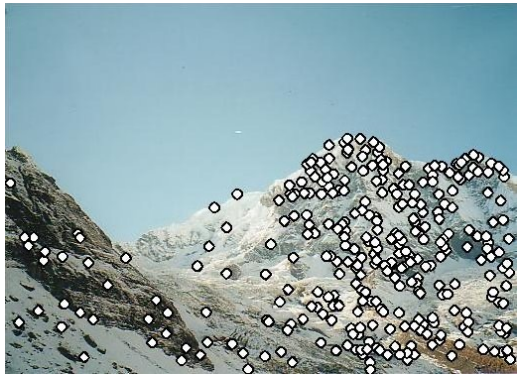
This Week

Next Week

Local features: main components

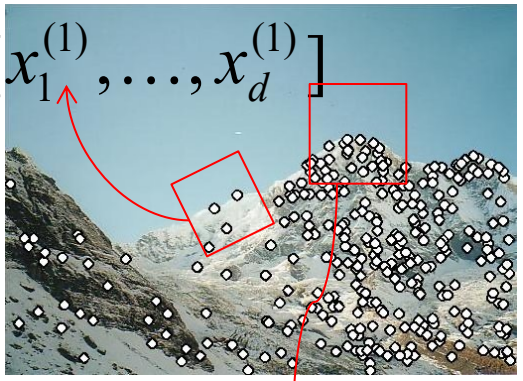
1) **Detection**: Identify the interest points

   e.g. corners



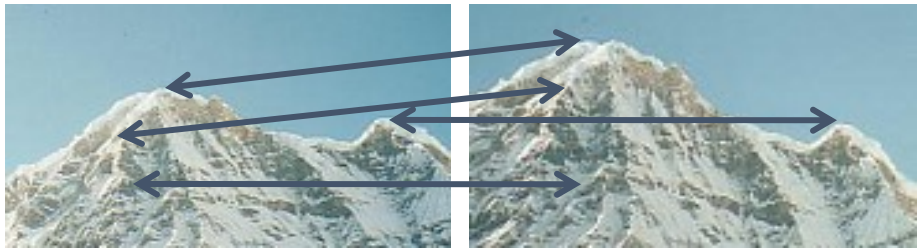
2) **Description**: Extract vector feature descriptor surrounding each interest point

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

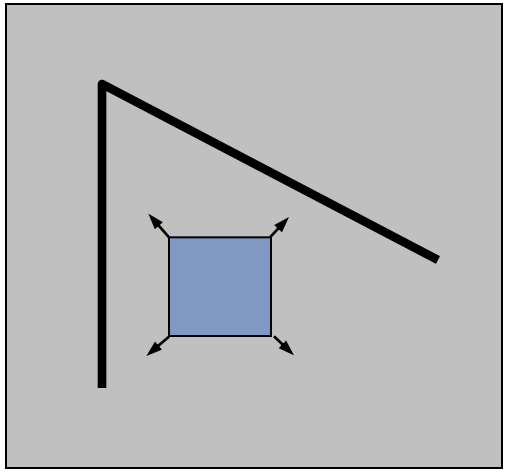
3) **Matching**: Determine correspondence between descriptors in two views



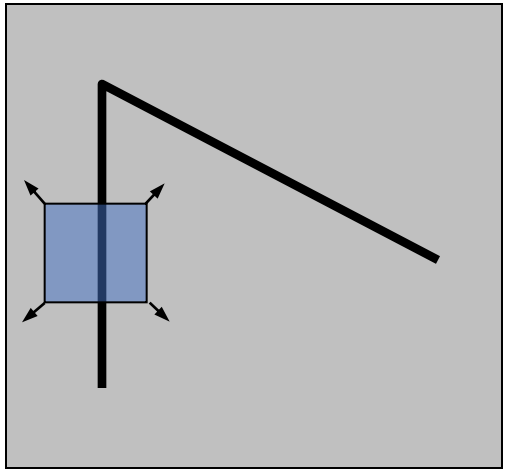
Credit: Kristen Grauman

How do we measure corner?

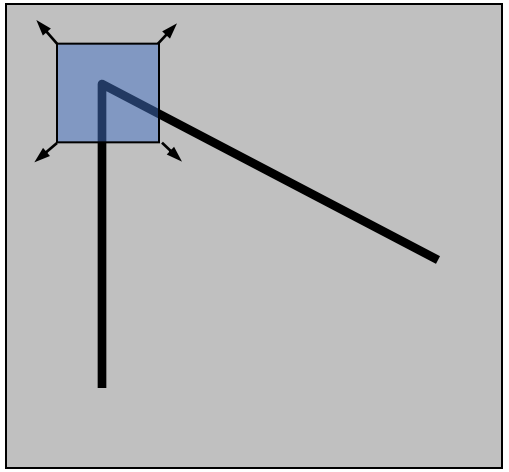
- Take a window W, and shift it in all directions by (u,v) pixels
- Corner = where shifting window in all directions causes significant change.



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

Credit: S. Seitz, D. Frolova, D. Simakov

Corner Detection: Mathematics

For every pixel (x,y) in an image consider a window W. Shift the window by (u,v) in every direction and measure the change in pixel intensities using:

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

Computing this with for loop is expensive.

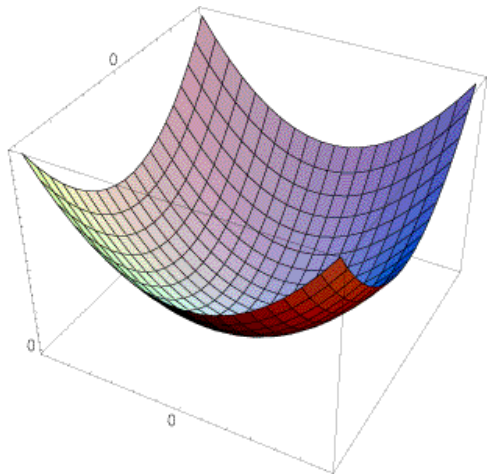
Let's assume (u,v) shift is relatively small, and use Taylor series approximation to obtain:

$$\begin{aligned} E(u,v) &\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2 \\ &\approx Au^2 + 2Buv + Cv^2 \end{aligned}$$

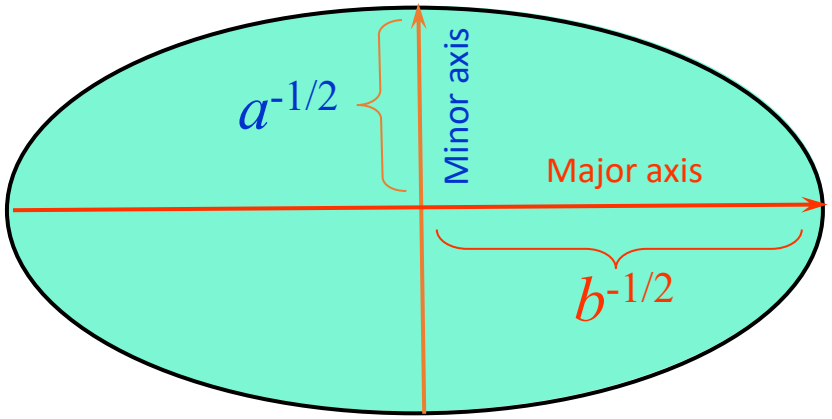
$$A = \sum_{(x,y)\in W} I_x^2 \quad B = \sum_{(x,y)\in W} I_x I_y \quad C = \sum_{(x,y)\in W} I_y^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

2nd moment matrix



Corner Detection: Understanding H



$$(u \quad v)\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = 1$$

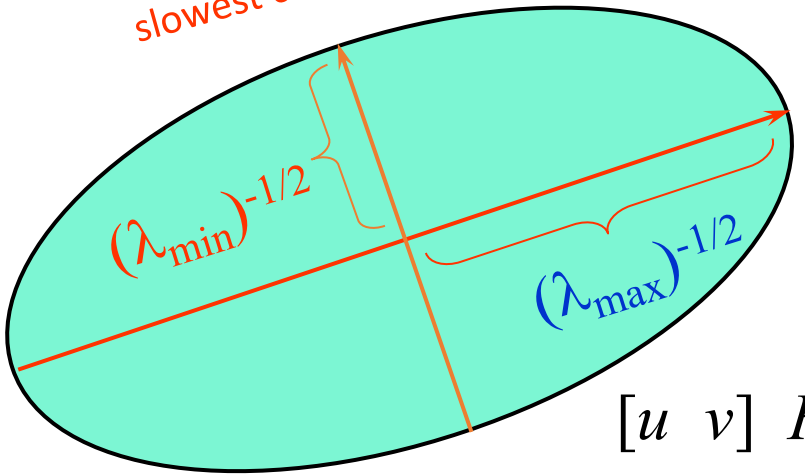
$$\frac{u^2}{(a^{-1/2})^2} + \frac{v^2}{(b^{-1/2})^2} = 1$$

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

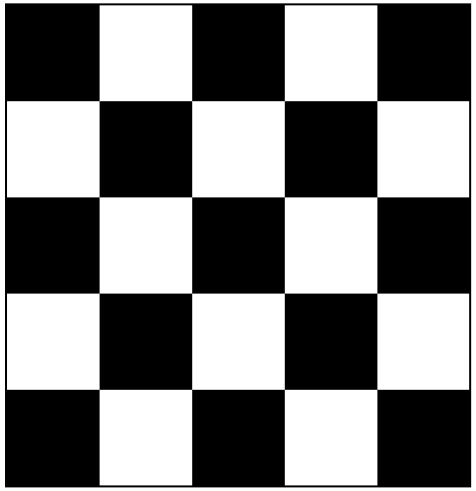
direction of the slowest change

λ_{max}, λ_{min} : eigenvalues of *H*

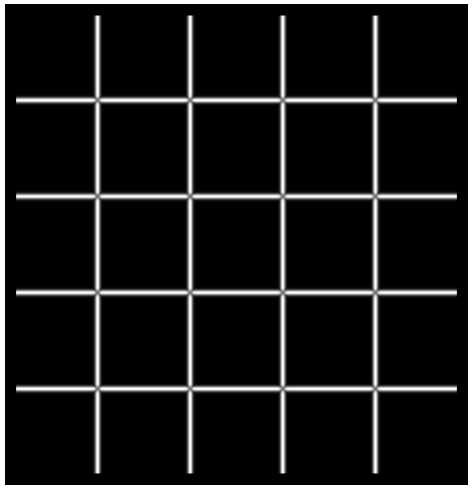
direction of the fastest change



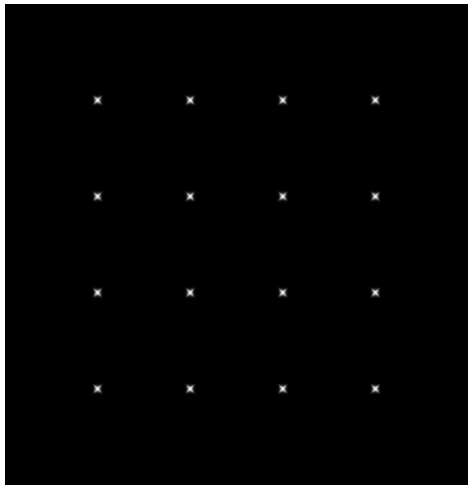
$$[u \quad v] \; H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



I



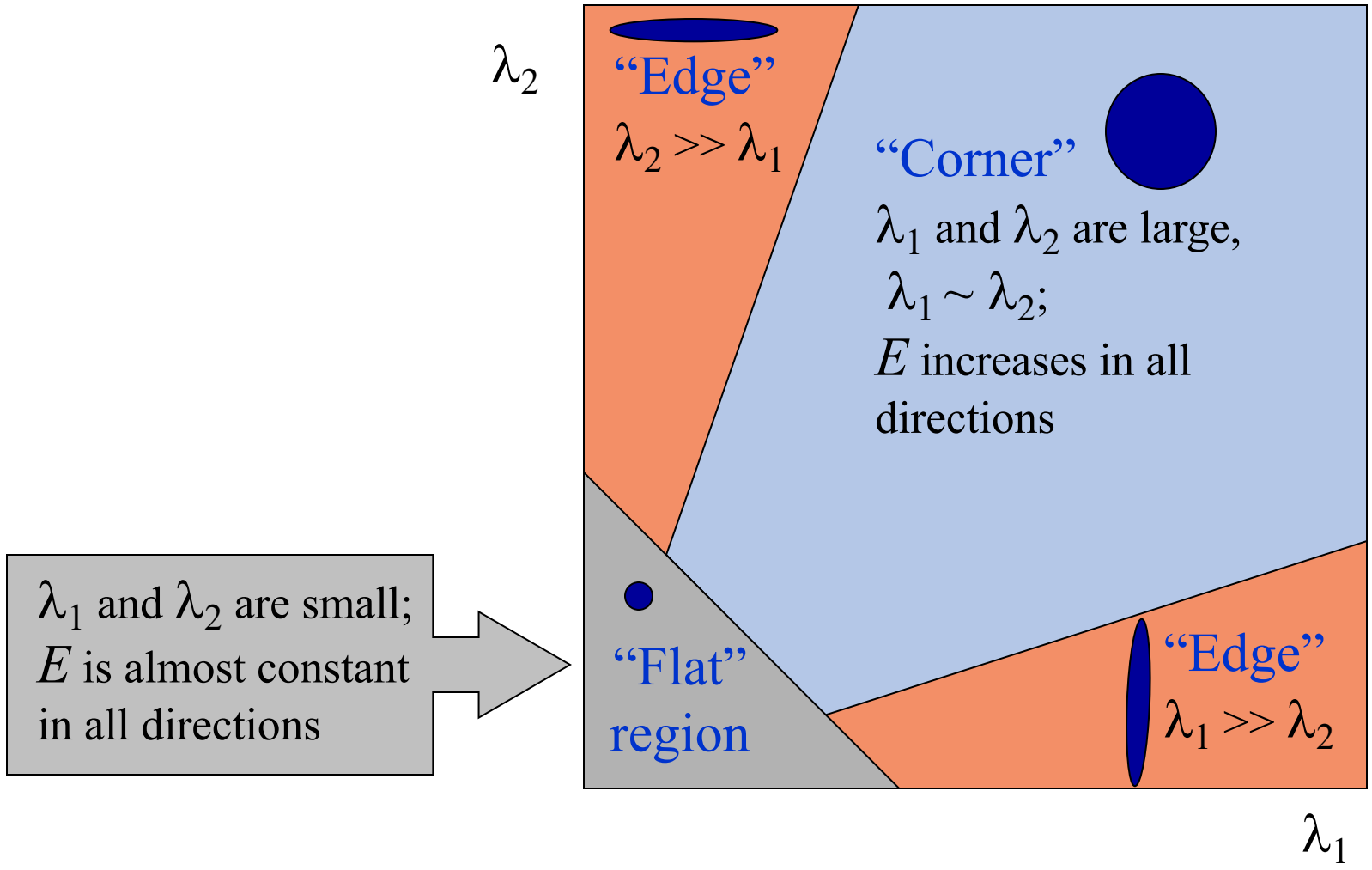
λ_{max}



λ_{min}

Interpreting the eigenvalues of H

Classification of image points using eigenvalues of *M*:



Harris Corner detector: Steps

1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix *H* in a Gaussian window around each pixel
3. Compute corner response function *f* or *R*

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

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

$$R = \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 = \det(M) - k \cdot \mathrm{tr}(M)^2$$

4. Threshold *f* or *R*
5. Find local maxima of response function (nonmaximum suppression)

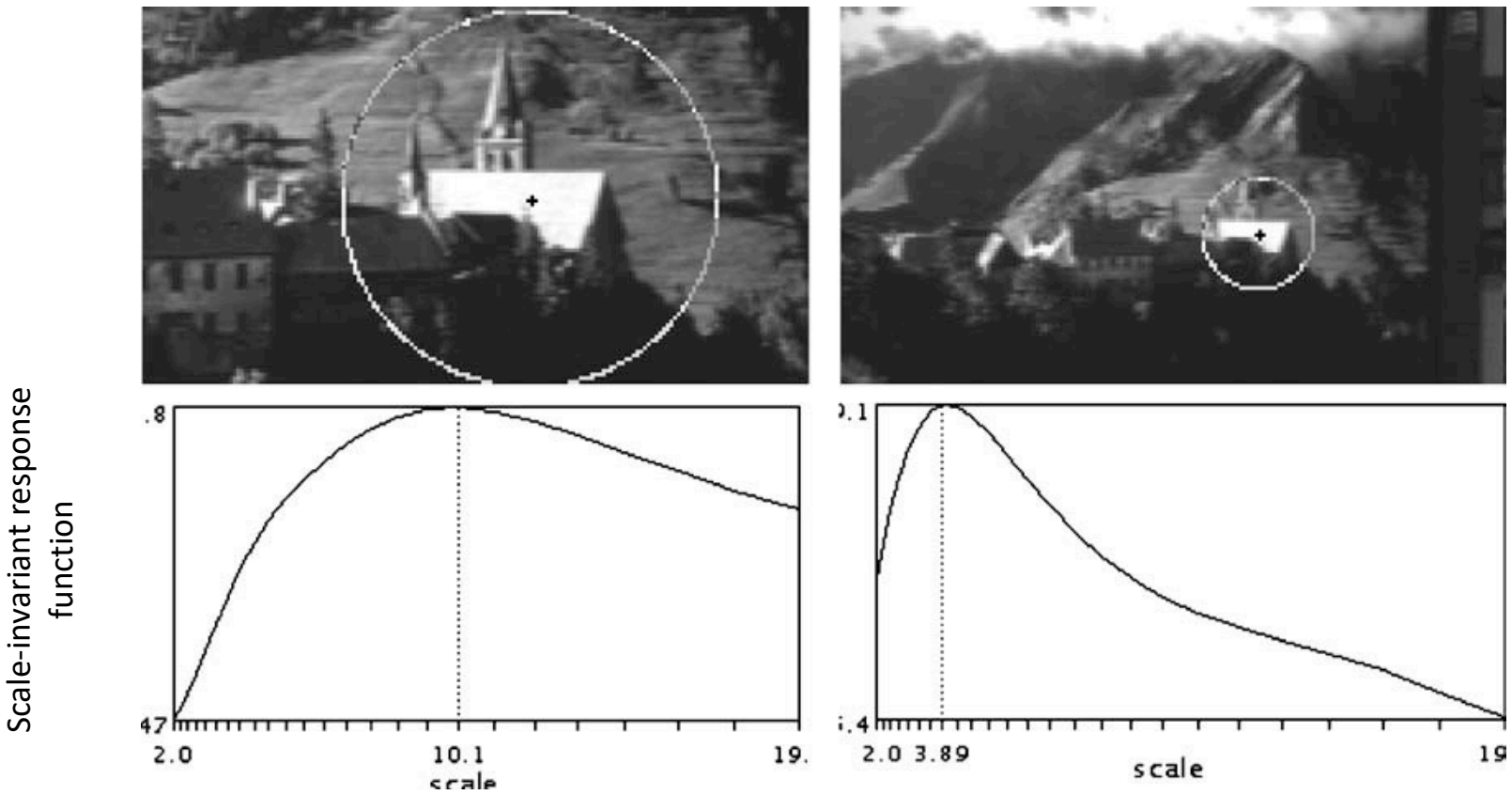
C.Harris and M.Stephens. "A Combined Corner and Edge Detector."
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

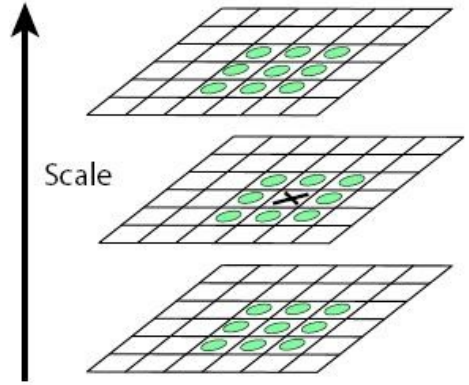
Properties of Harris: Invariance and equivariance

- We want corner locations to be *invariant* to photometric transformations and *equivariant* to geometric transformations
  - **Invariance:** image is transformed and corner locations do not change
  - **Equivariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

- Harris detector is equivariant to translation and rotation.
- Harris detector is somewhat invariant to intensity change (I' =a*I +b).
- Harris detector is NOT equivariant to scaling.

Keypoint detection with scale selection





"scale space"

Characteristic scale = scale at which the Harris operator *f*/R is maximum.

Approach: compute a *scale-invariant* response function over neighborhoods centered at each location (x, y) and a range of scales (σ), find *scale-space locations* (x, y, σ) where this function reaches a local maximum.

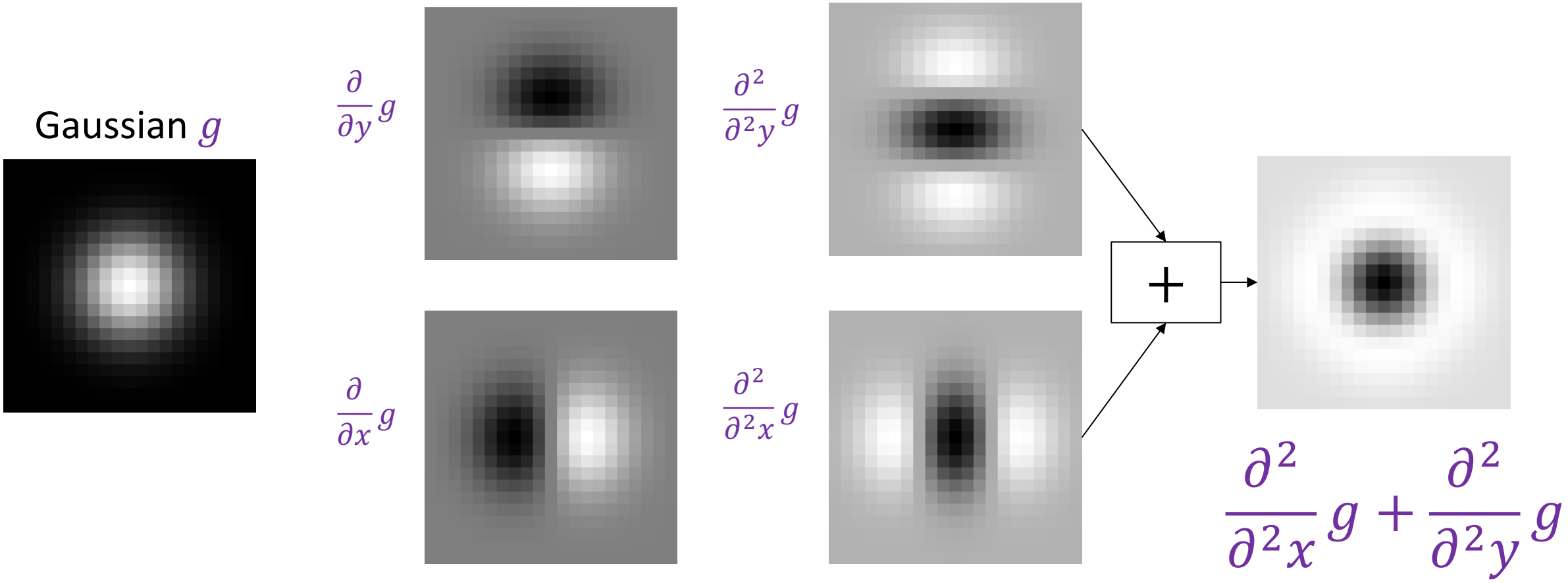
Today’s class

- SIFT detector
- SIFT descriptor
- Feature Matching
- Evaluating Results

Today’s class

- SIFT detector
- SIFT descriptor
- Feature Matching
- Evaluating Results

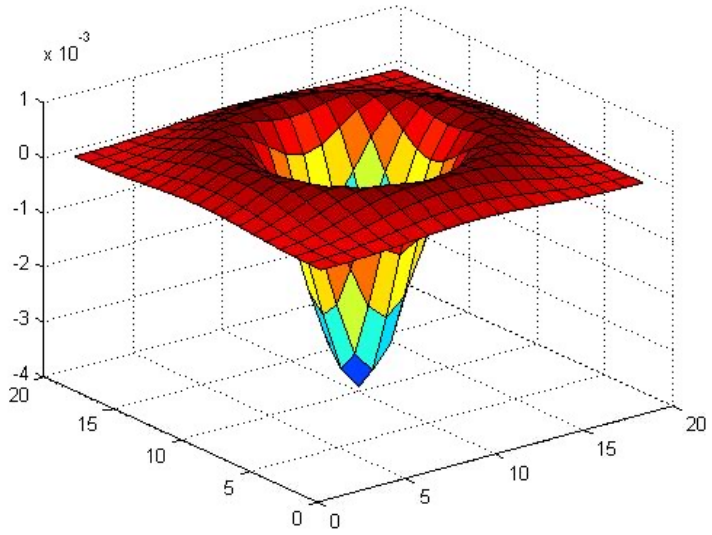
Laplacian of Gaussian

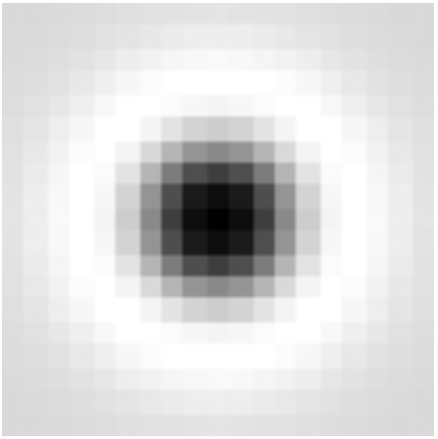


Source: J. Johnson and D. Fouhey

Scale-normalized Laplacian

- You need to multiply the LoG by σ^2 to make responses comparable across scales

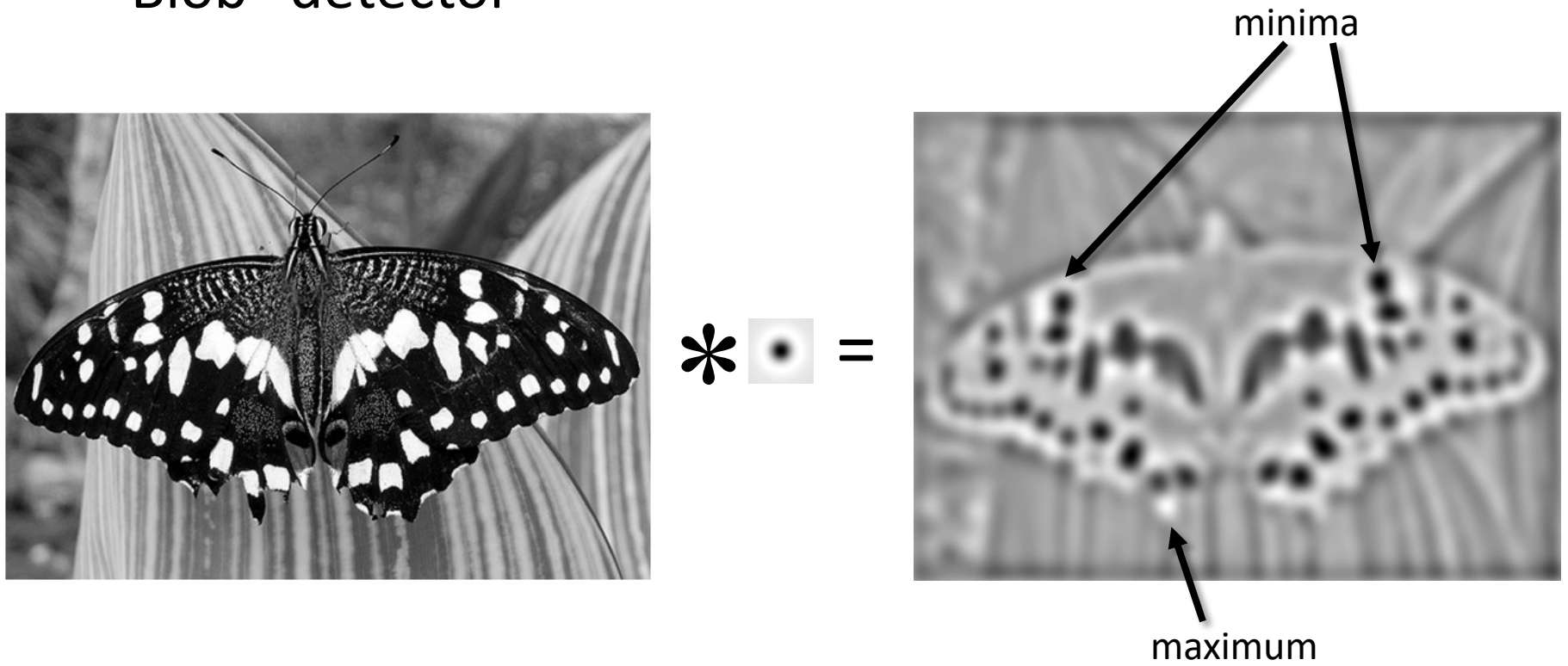




$$\nabla^2_{\text{norm}} = \sigma^2 \left(\frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial^2 y} g \right)$$

Laplacian of Gaussian

- “Blob” detector



- Find maxima *and minima* of LoG operator in space and scale

Approximating Laplacian of Gaussian

- Functions for determining scale $f = \text{Kernel} * \text{Image}$

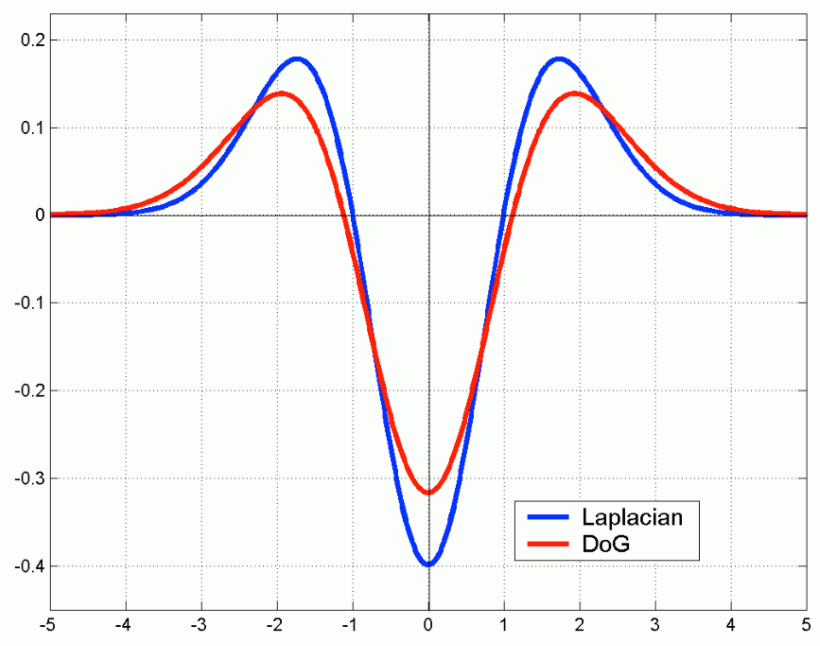
Kernels:

$L = \sigma^2 \left(G_{xx}(x, \right.$

(Laplacian)

$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$

(Difference of Gaussians)

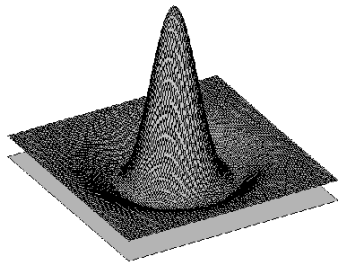


where Gaussian

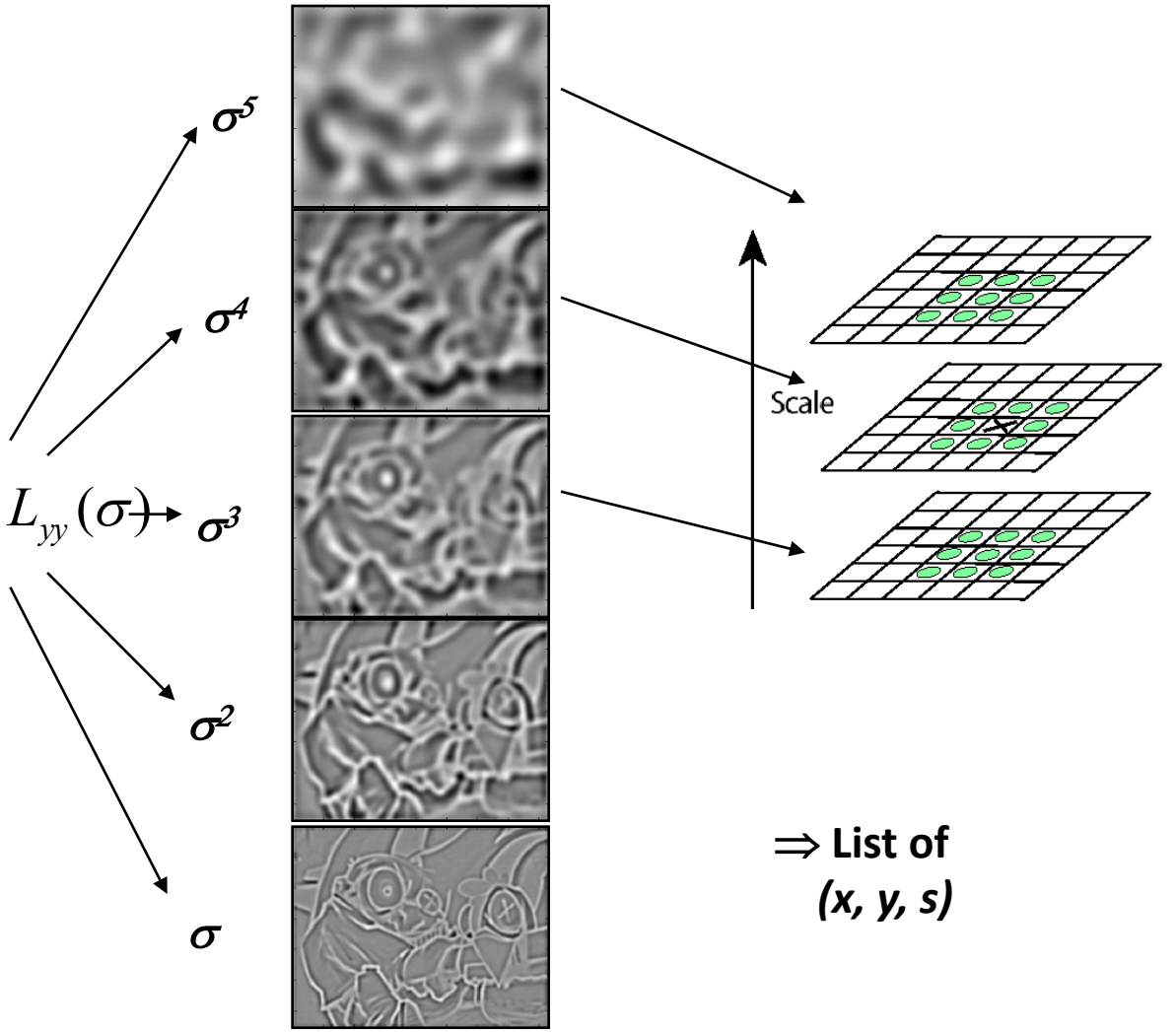
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Note: The LoG and DoG operators are both rotation equivariant

Find local maxima in 3D position-scale space

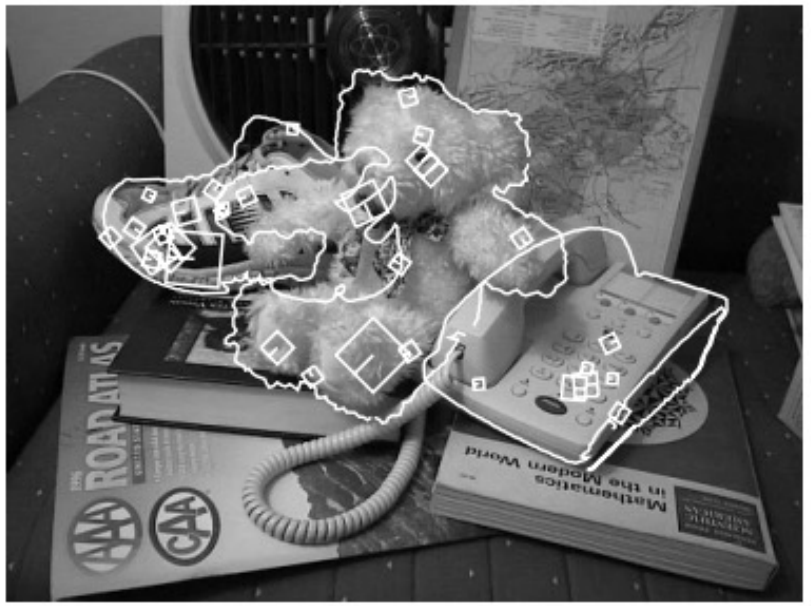


$$L_{xx}(\sigma) + L_{yy}(\sigma)$$



⇒ **List of**
(x, y, s)

SIFT: Scale-invariant feature transform

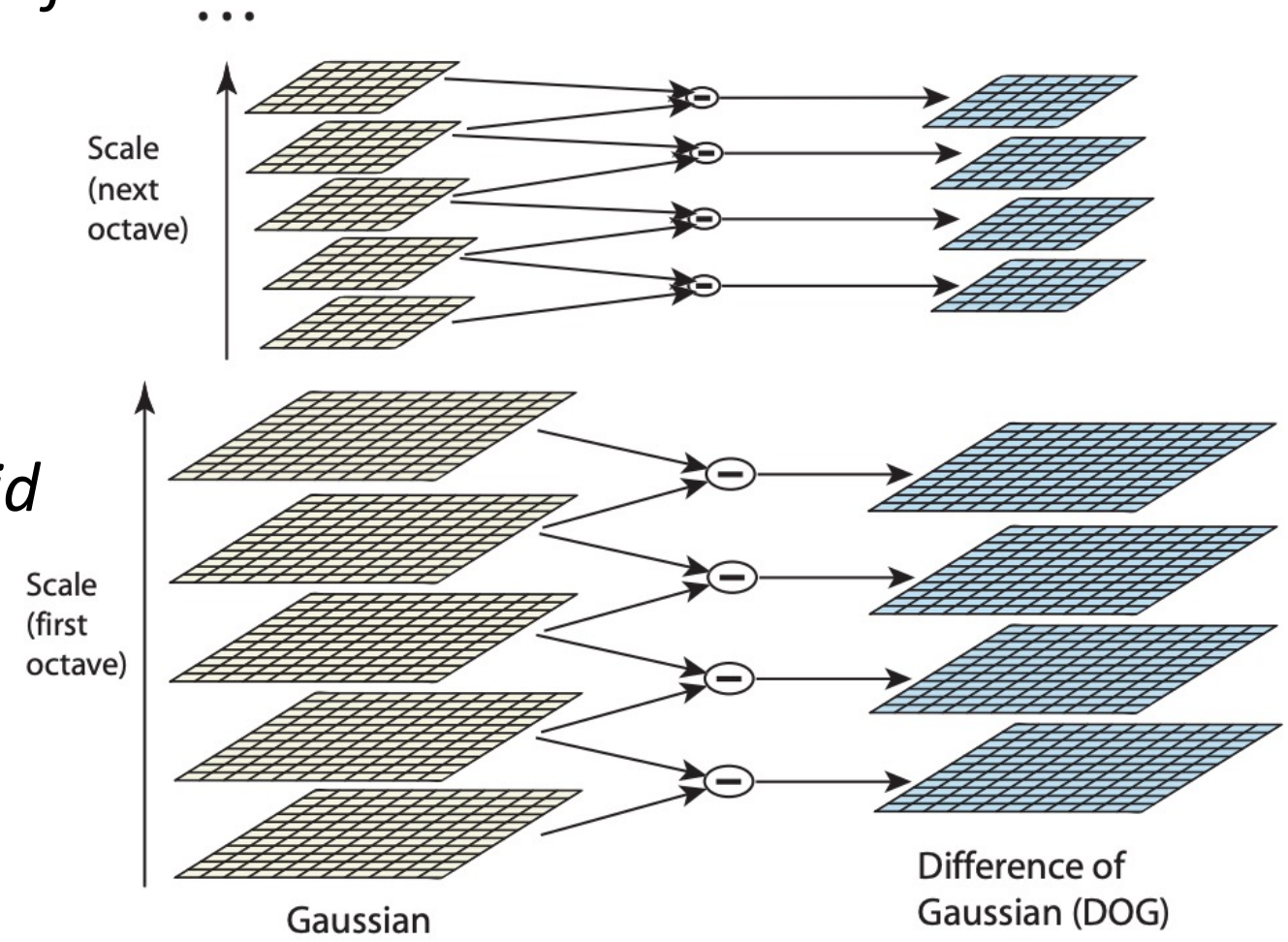


D. Lowe. Object recognition from local scale-invariant features. ICCV 1999

D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV* 60 (2), pp. 91-110, 2004

SIFT detector

- Approximate LoG with a *difference of Gaussians* (DoG)
  - Laplacian:
  - $\sigma^2\big(G_{xx}(x,y,\sigma) + G_{yy}(x,y,\sigma)\big)$
  - DoG:
  - $G(x,y,k\sigma) - G(x,y,\sigma)$
- Compute DoG via an *image pyramid*



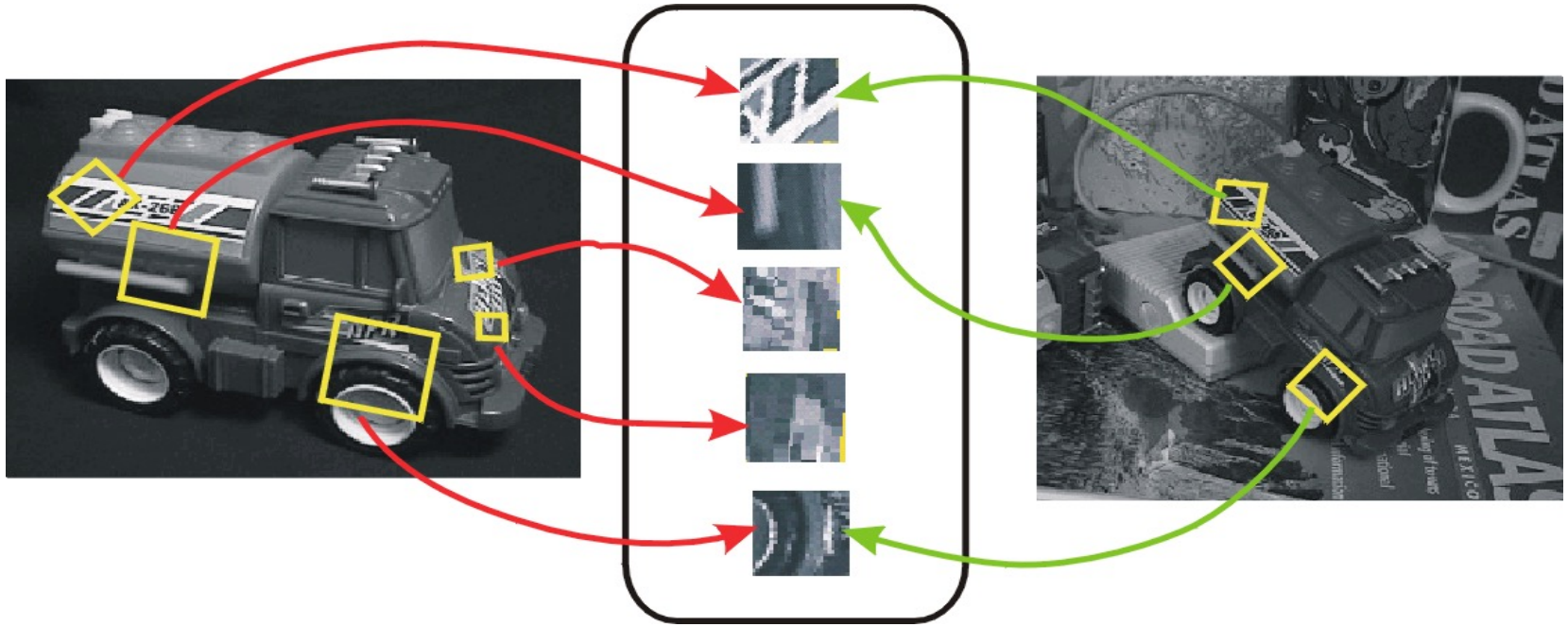
- In each Octave you progressively blur the image
- To go to next Octave you downsample the image by x2

Today’s class

- SIFT detector
- SIFT descriptor
- Feature Matching
- Evaluating Results

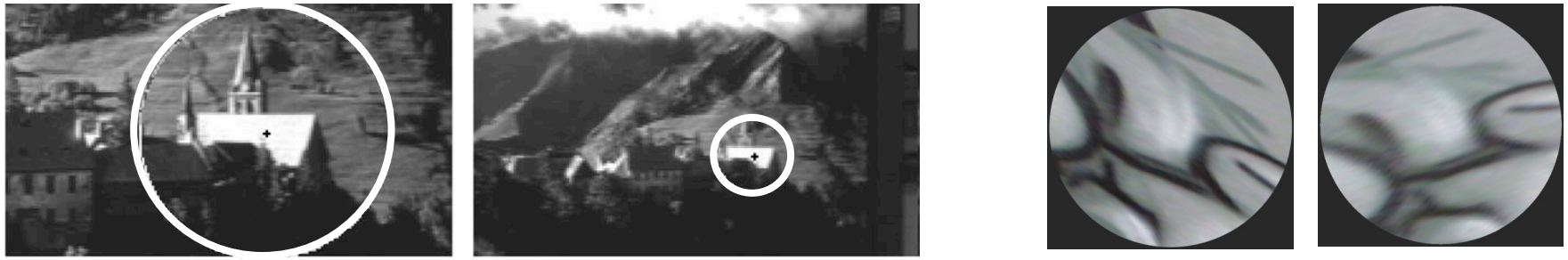
SIFT for matching

- The main goal of SIFT is to enable image matching in the presence of significant transformations
  - To recognize the same keypoint in multiple images, we need to match appearance descriptors or “signatures” in their neighborhoods
  - Descriptors that are *locally* invariant w.r.t. scale and rotation can handle a wide range of *global* transformations



SIFT for matching

- SIFT detector returns a characteristic scale that can be normalized out, but no characteristic orientation



Invariant descriptors

- We looked at invariant / equivariant **detectors**

- Most feature descriptors are also 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 transforms (some are fully affine invariant)
  - Limited illumination/contrast changes

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

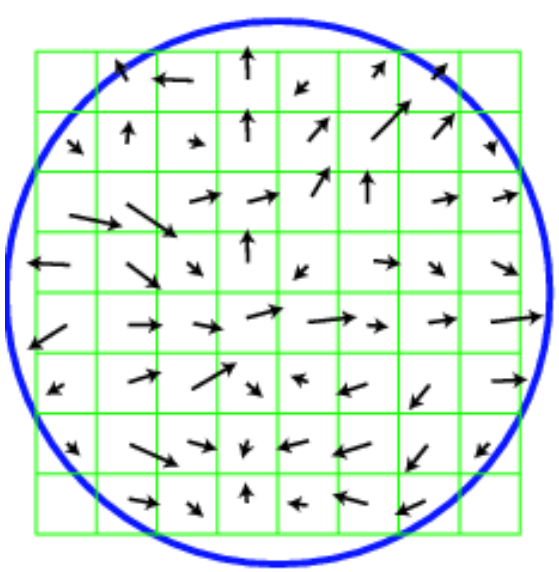
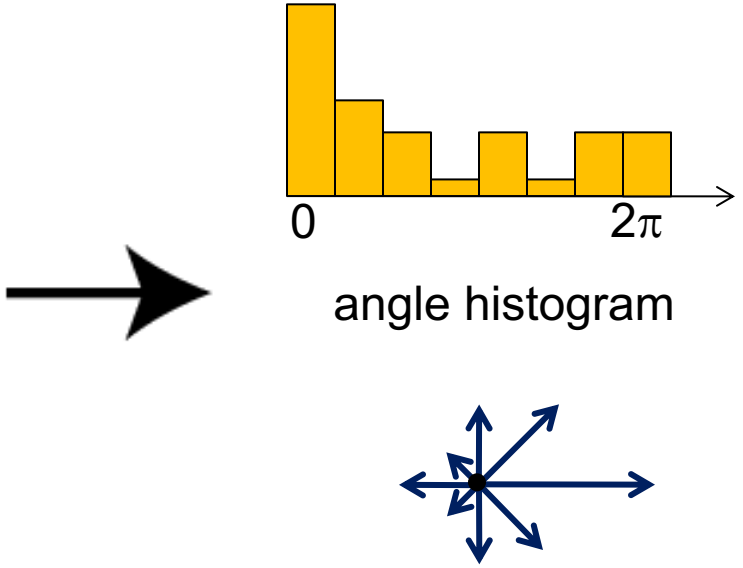


Image gradients

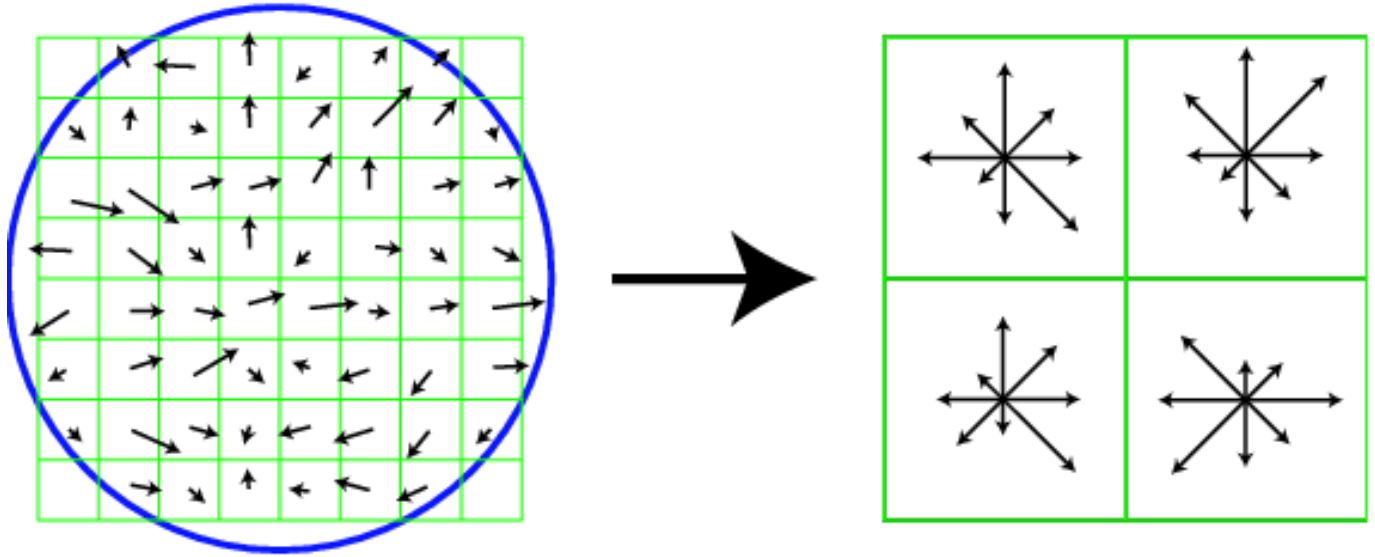


Adapted from slide by David Lowe

SIFT descriptor

Full version

- 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

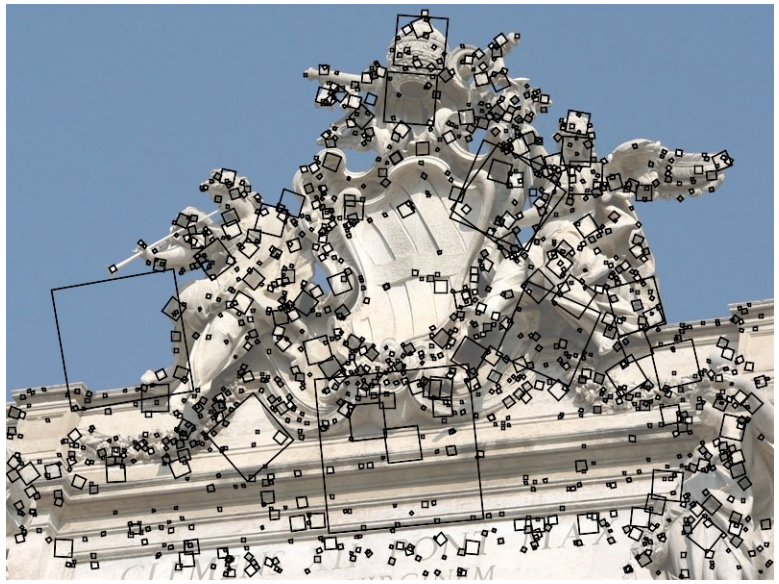


Inspiration: complex neurons in the primary visual cortex

Adapted from slide by David Lowe

SIFT detector: Example outputs

- Detected keypoints with characteristic scales and orientations:

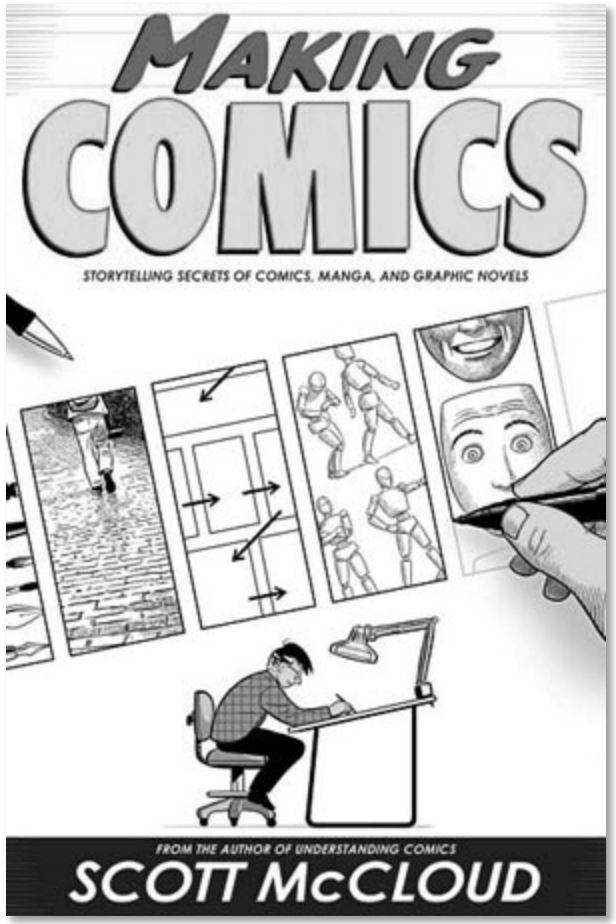


SIFT for matching

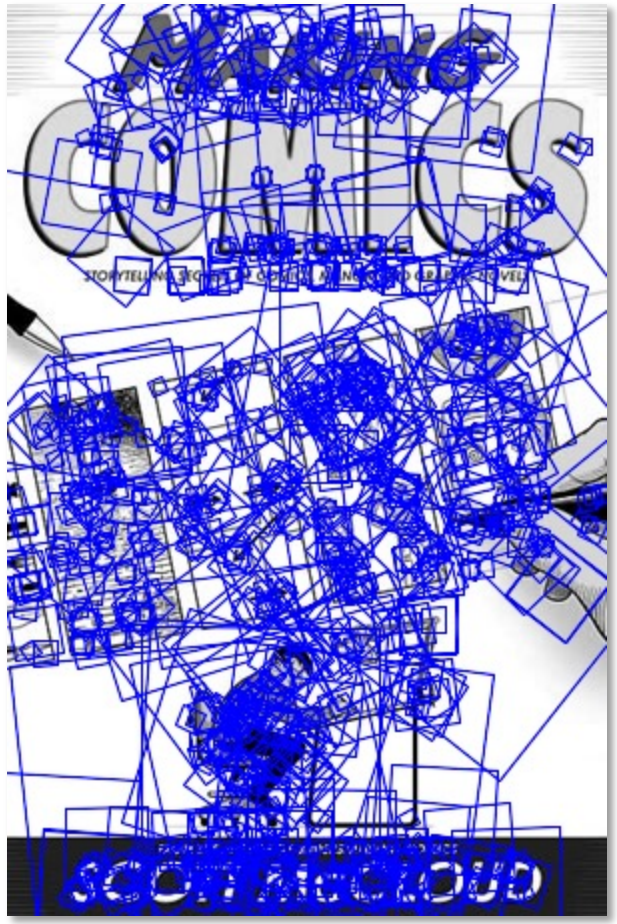
- Extraordinarily robust detection and description technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out-of-plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night
  - Fast and efficient—can run in real time
  - Lots of code available

Source: N. Snavely

SIFT Example



sift



868 SIFT features

Other descriptors

- HOG: Histogram of Gradients (HOG)
  - Dalal/Triggs
  - Sliding window, pedestrian detection

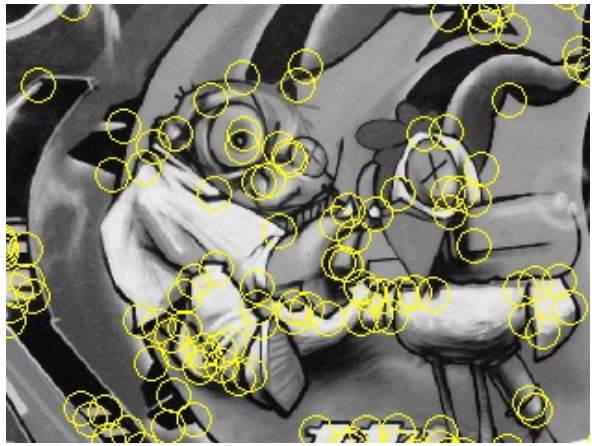
- FREAK: Fast Retina Keypoint
  - Perceptually motivated
  - Can run in real-time; used in Visual SLAM on-device

- LIFT: Learned Invariant Feature Transform
  - Learned via deep learning – along with many other recent features

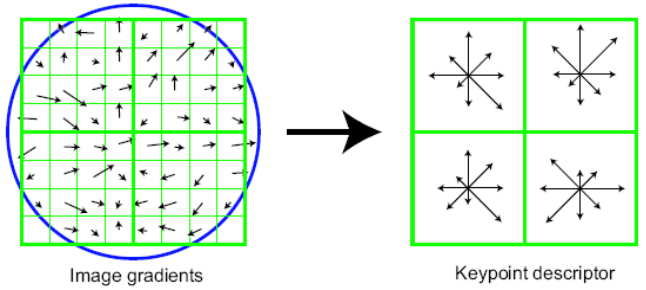
    https://arxiv.org/abs/1603.09114

Summary

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



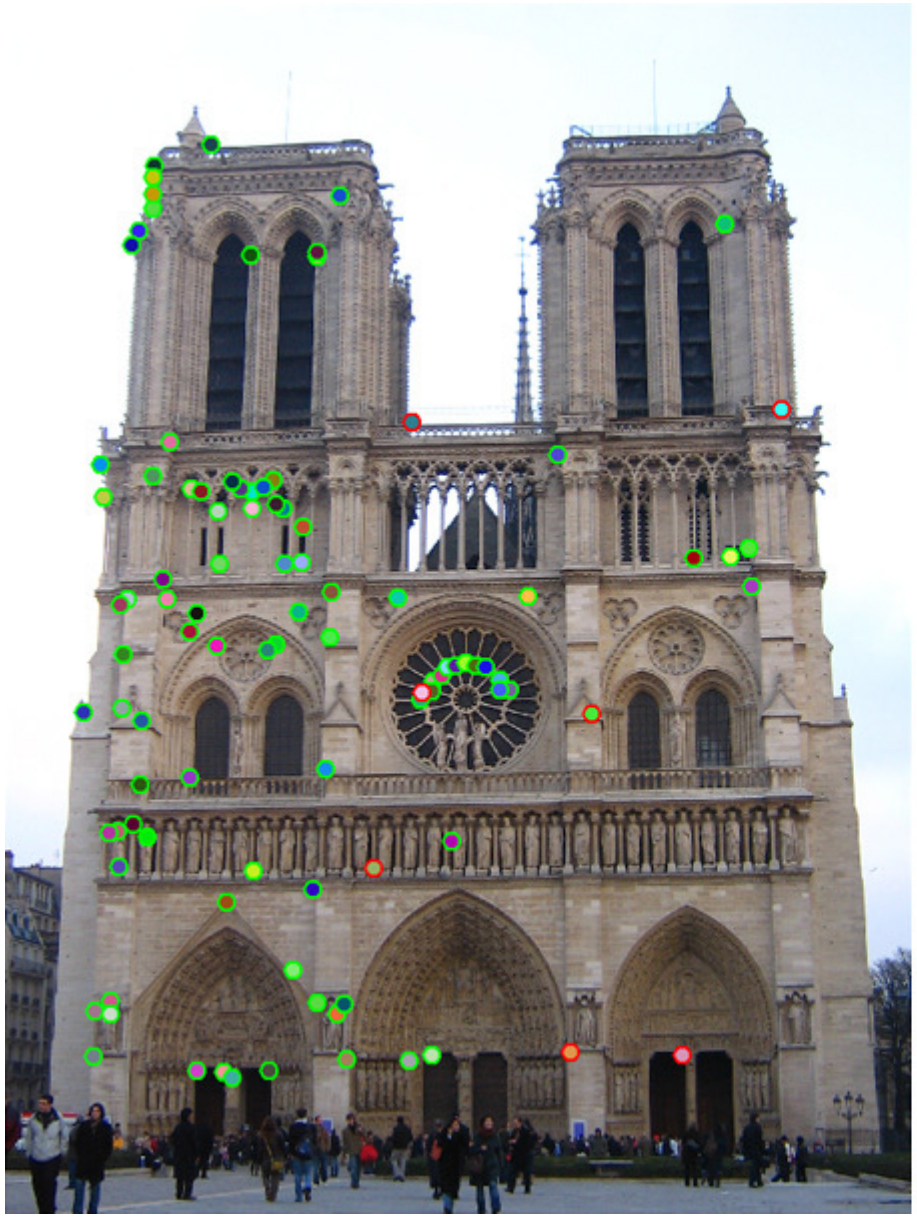
- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT and variants are typically good for stitching and recognition

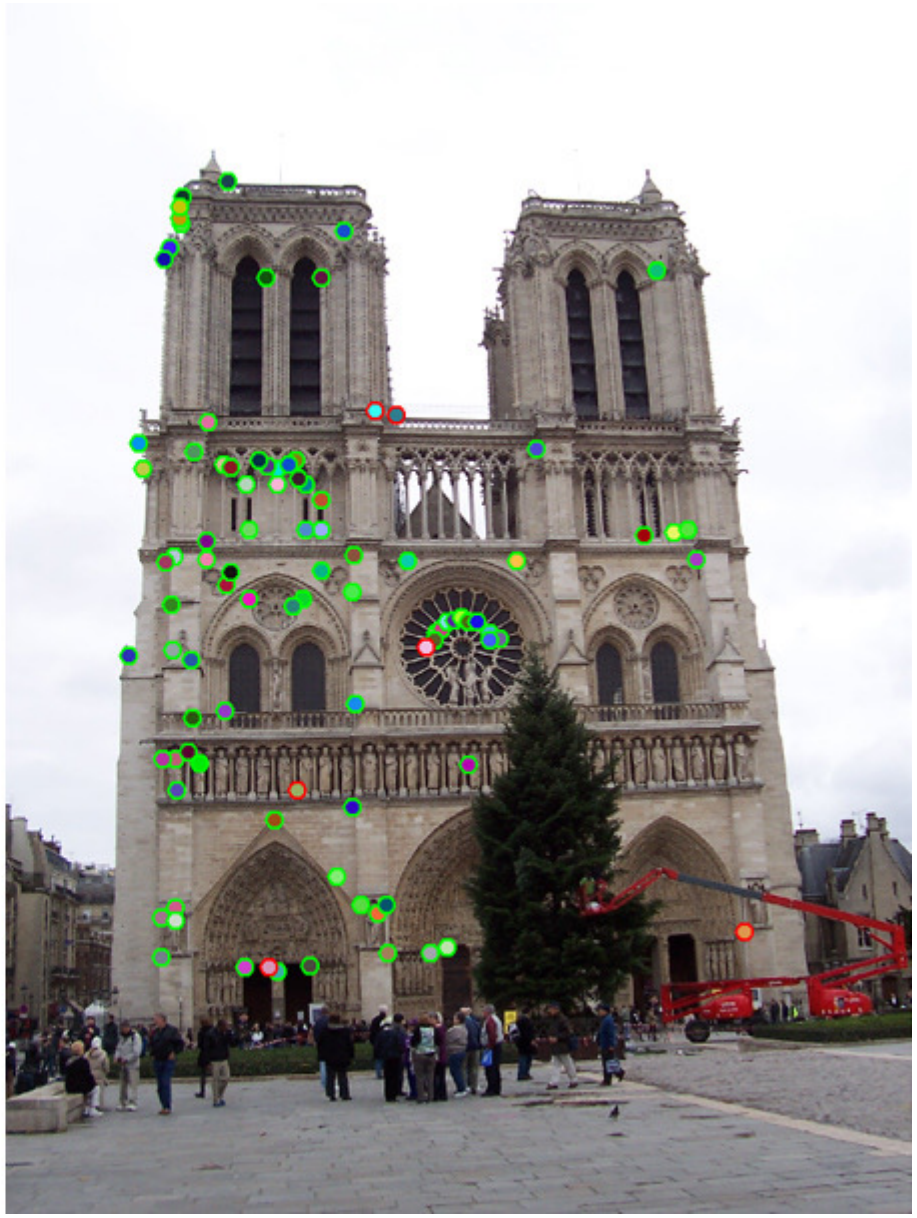


Today’s class

- SIFT detector
- SIFT descriptor
- Feature Matching
- Evaluating Results

Which features match?





Feature matching

Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
   - Any distance metric, *d(f1,f2)*, would work: L2, L1 loss are commonly used

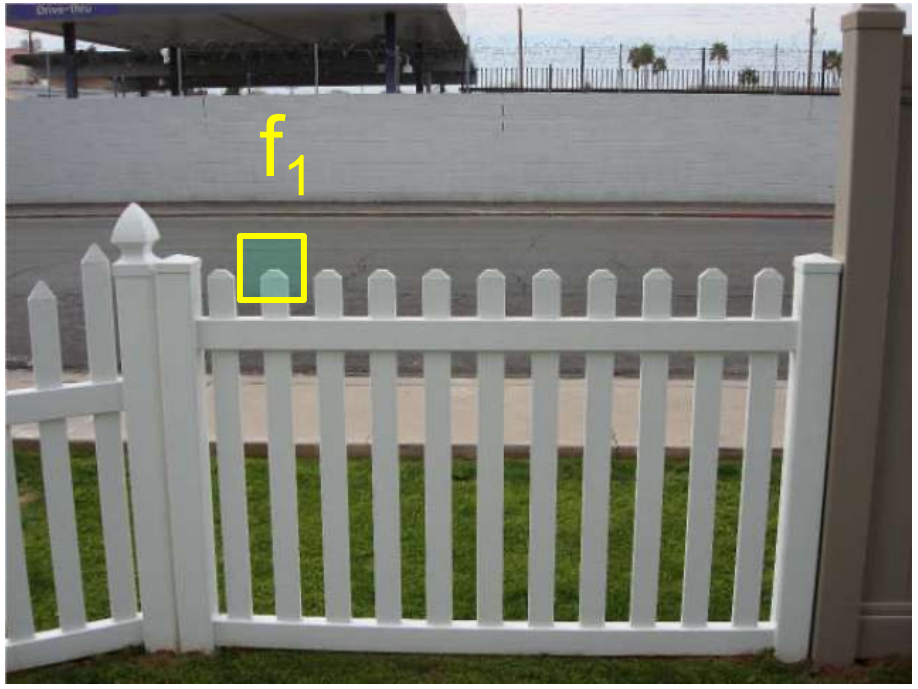
2. Test all the features in I_2, find the one with min distance

(OR)

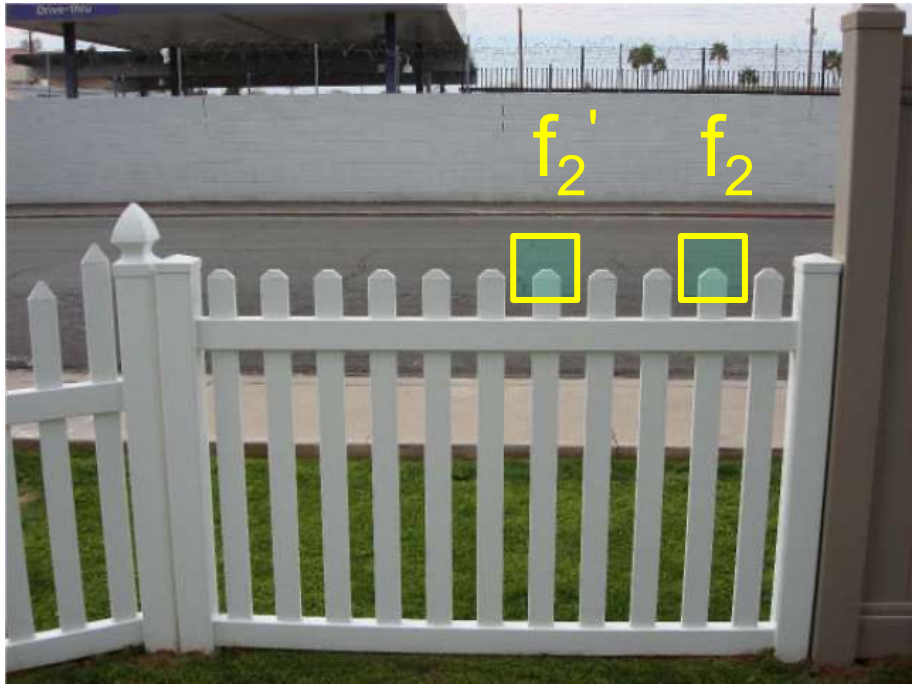
2. Test all the features in I_2, find top k matches

Feature distance: Ratio Test

- Often matches can be ambiguous. f1 can have similar distance to both f2 and f2’
- Ratio Test:
  - Keep top 2 match: f2, f2’
  - If d(f1,f2) < 0.75 * d(f1,f2’), then: match f1 with f2 and keep the point.
    - Else reject the match as ambiguous
  - If d(f1,f2) < Threshold, then: keep this as ‘strong’ match, else: ‘reject’



I_1

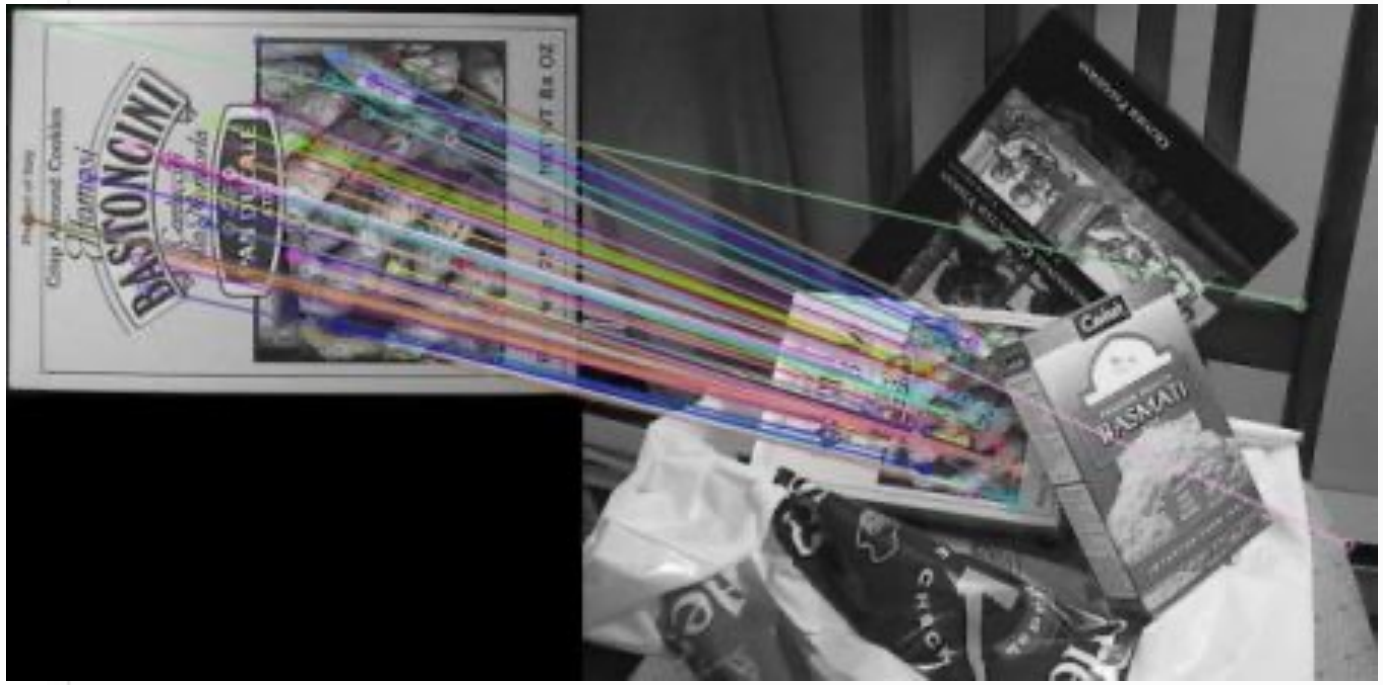


I_2

Image Matching in OpenCV

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img1 = cv2.imread('box.png',0)          # queryImage
img2 = cv2.imread('box_in_scene.png',0) # trainImage
```

Read Image



Compute SIFT

Feature matching

Ratio Test

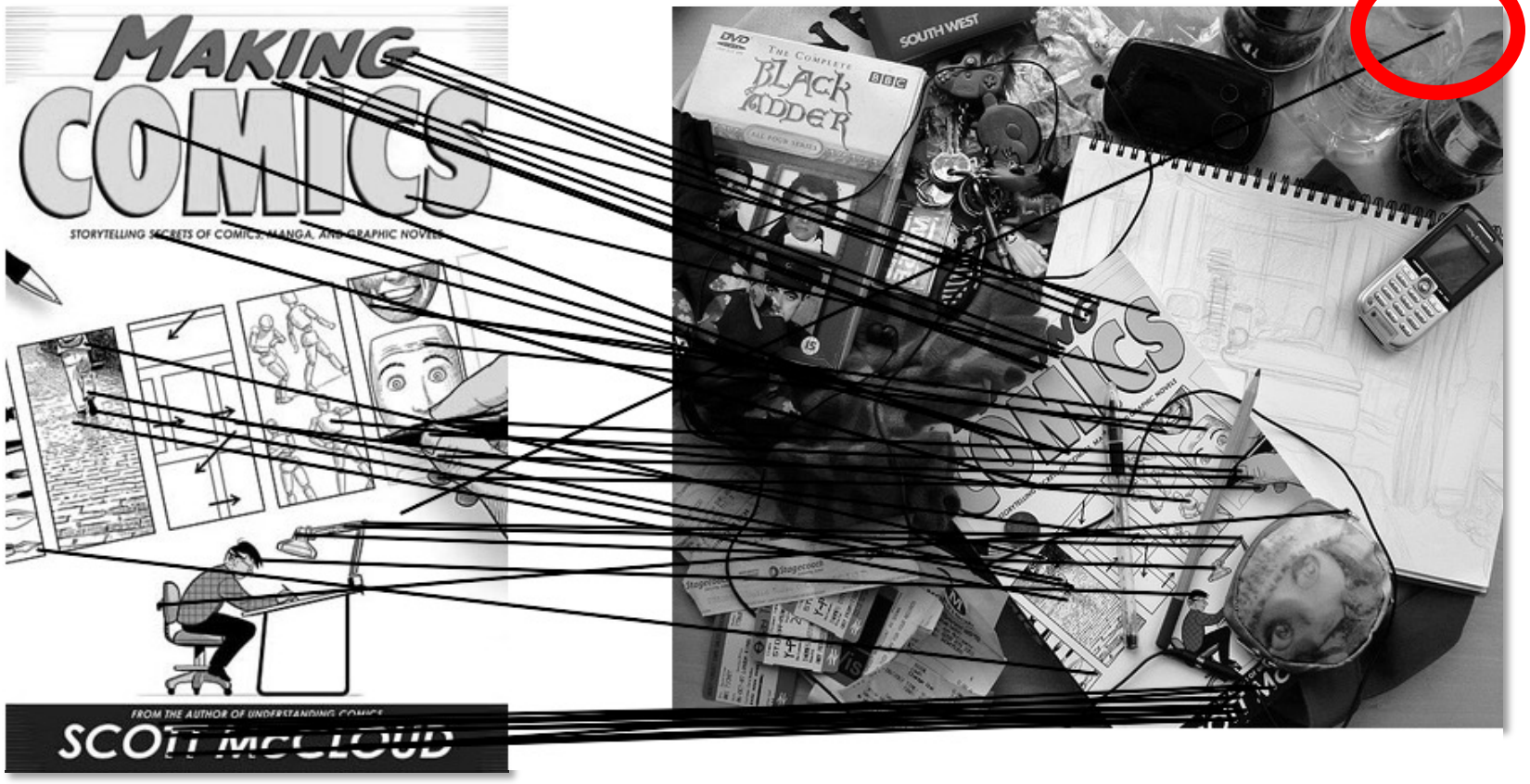
```
# cv2.drawMatchesKnn expects list of lists as matches.
img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,flags=2)

plt.imshow(img3),plt.show()
```

Visualization

Feature matching example

We'll deal with **outliers** later

RANSAC next week!



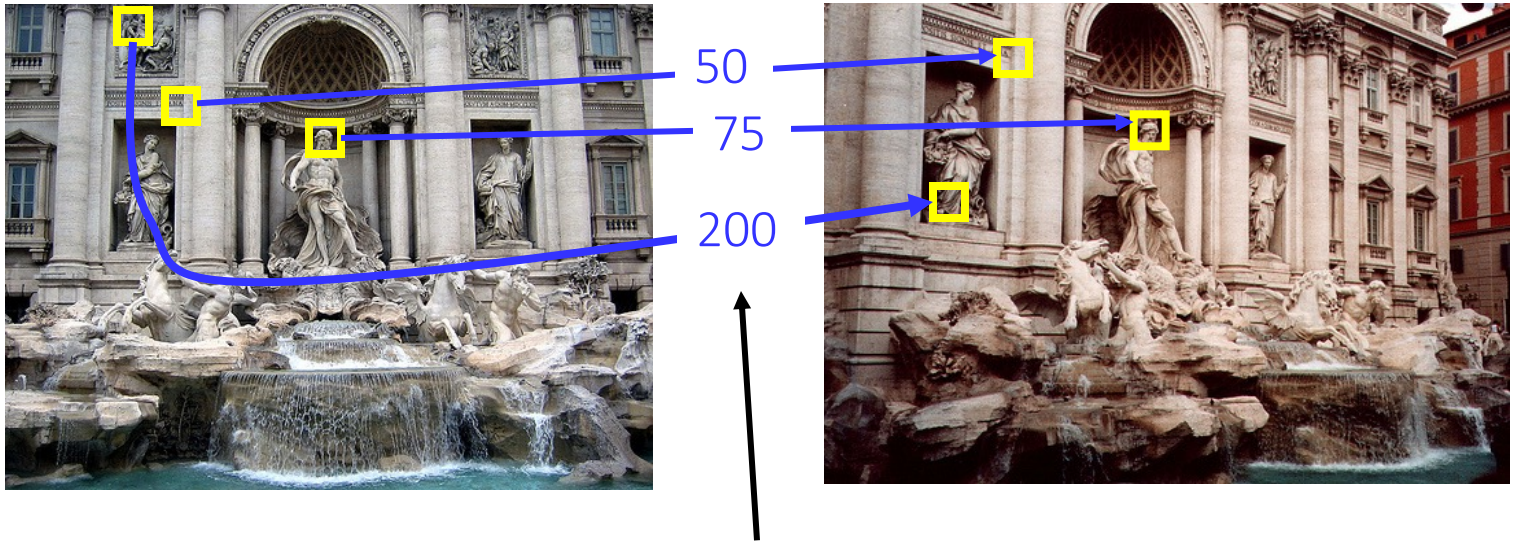
51 matches (thresholded by ratio score)

Today’s class

- SIFT detector
- SIFT descriptor
- Feature Matching
- Evaluating Results

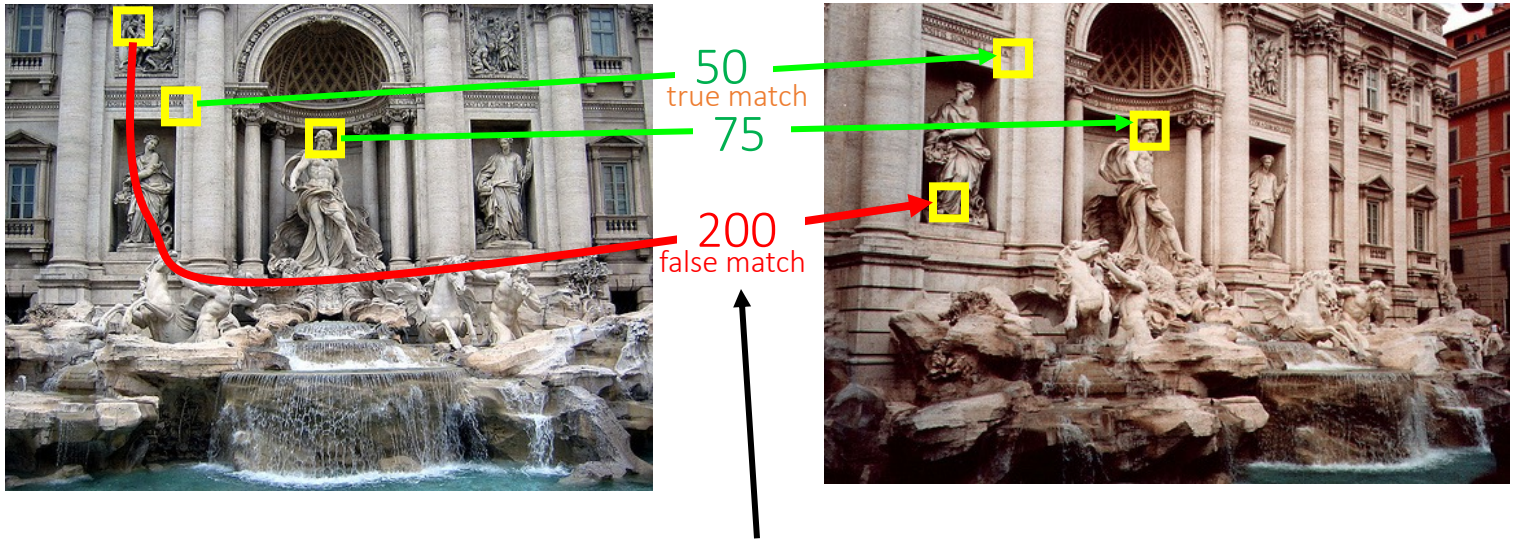
Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives

How can we measure the performance of a feature matcher?



We can choose distance threshold to decide if the match is ‘good’ or not.

The distance threshold affects performance

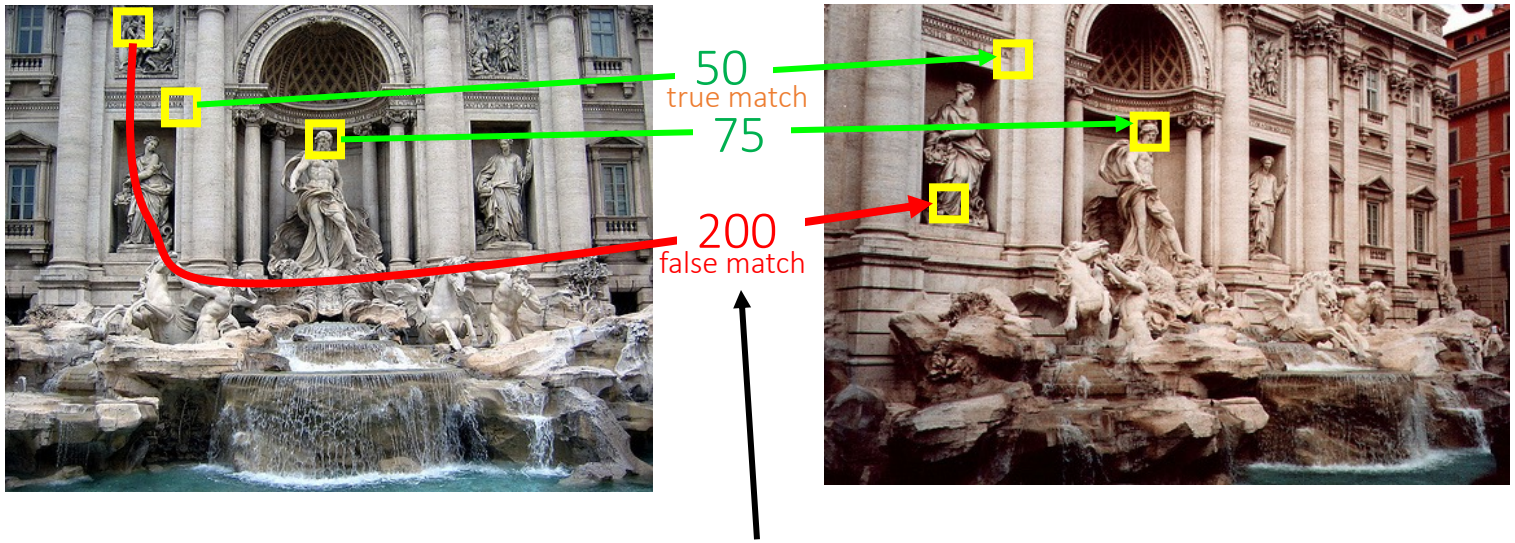
- True positives = # of detected matches that survive the threshold that are correct
- False positives = # of detected matches that survive the threshold that are incorrect

Example

- Suppose our matcher computes 1,000 matches between two images
  - 800 are correct matches, 200 are incorrect (according to an oracle that gives us ground truth matches)
  - A given threshold (e.g., ratio distance = 0.6) gives us 600 correct matches and 100 incorrect matches that survive the threshold
  - True positive rate = 600 / 800 = ¾
  - False positive rate = 100 / 200 = ½

True/false positives

How can we measure the performance of a feature matcher?

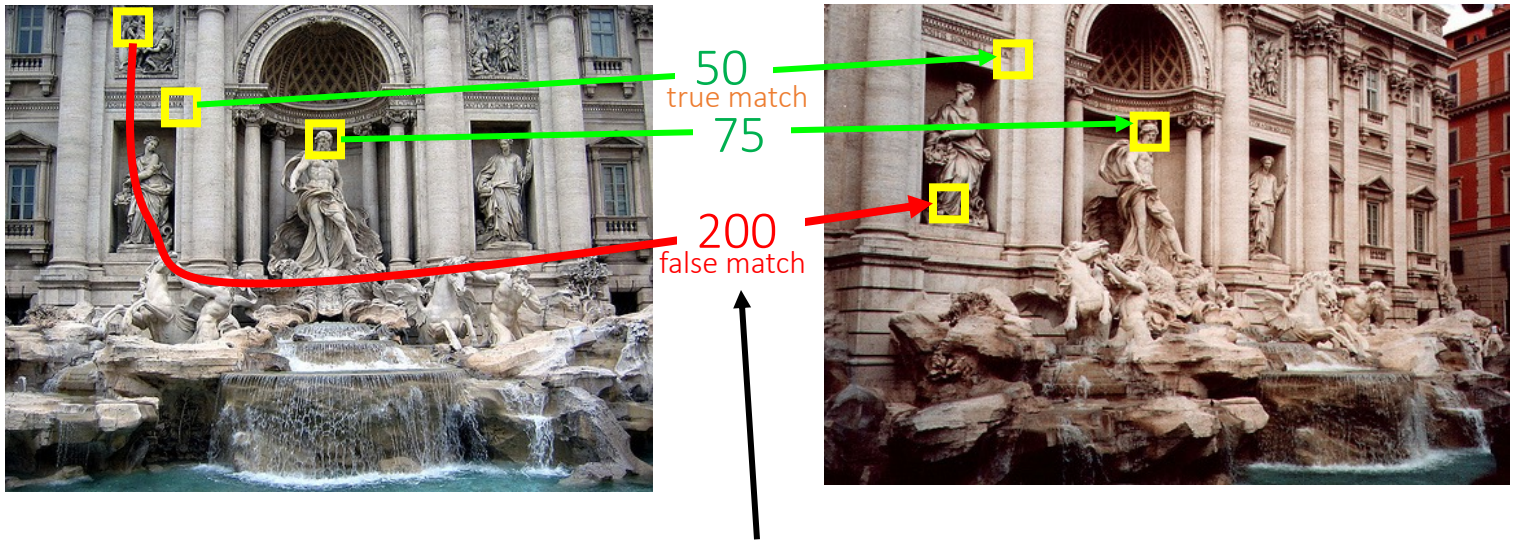


True positives = # of detected matches that survive the threshold that are correct
False positives = # of detected matches that survive the threshold that are incorrect

Suppose we want to maximize true positives. How do we set the threshold? (We keep all matches with distance below the threshold.)

True/false positives

How can we measure the performance of a feature matcher?



True positives = # of detected matches that survive the threshold that are correct
False positives = # of detected matches that survive the threshold that are incorrect

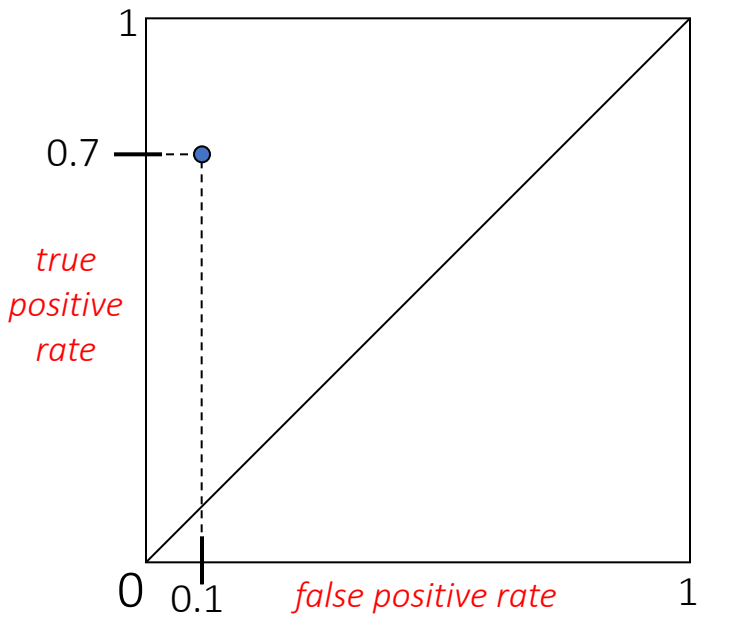
Suppose we want to minimize false positives. How do we set the threshold? (We keep all matches with distance below the threshold.)

Evaluating the results

How can we measure the performance of a feature matcher?

$$\frac{\text{\# true positives surviving threshold}}{\text{\# total correct matches (positives)}}$$

recall



$$\frac{\text{\# false positives surviving threshold}}{\text{\# total incorrect matches (negatives)}}$$

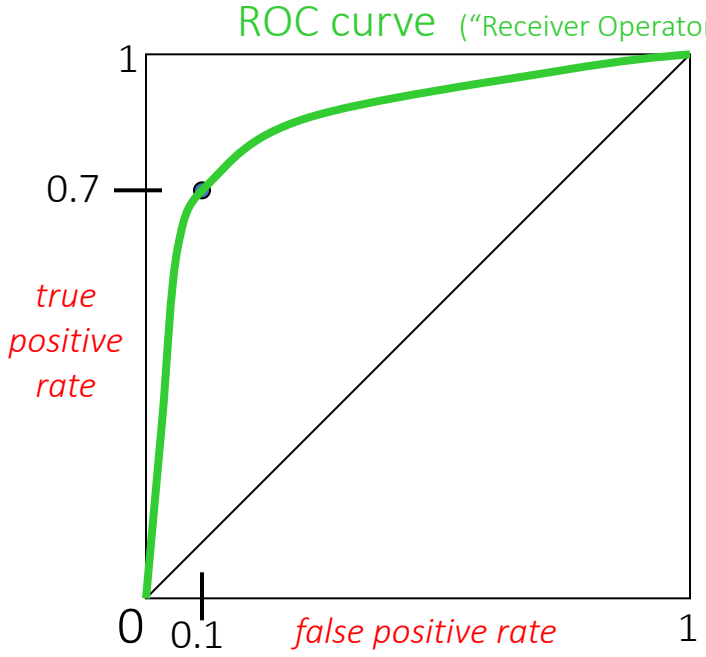
1 - *specificity*

Evaluating the results

How can we measure the performance of a feature matcher?

$$\frac{\text{\# true positives surviving threshold}}{\text{\# total correct matches (positives)}}$$

recall



$$\frac{\text{\# false positives surviving threshold}}{\text{\# total incorrect matches (negatives)}}$$

1 - *specificity*

Generated by choosing different threshold value

Single number: Area Under the Curve (AUC)

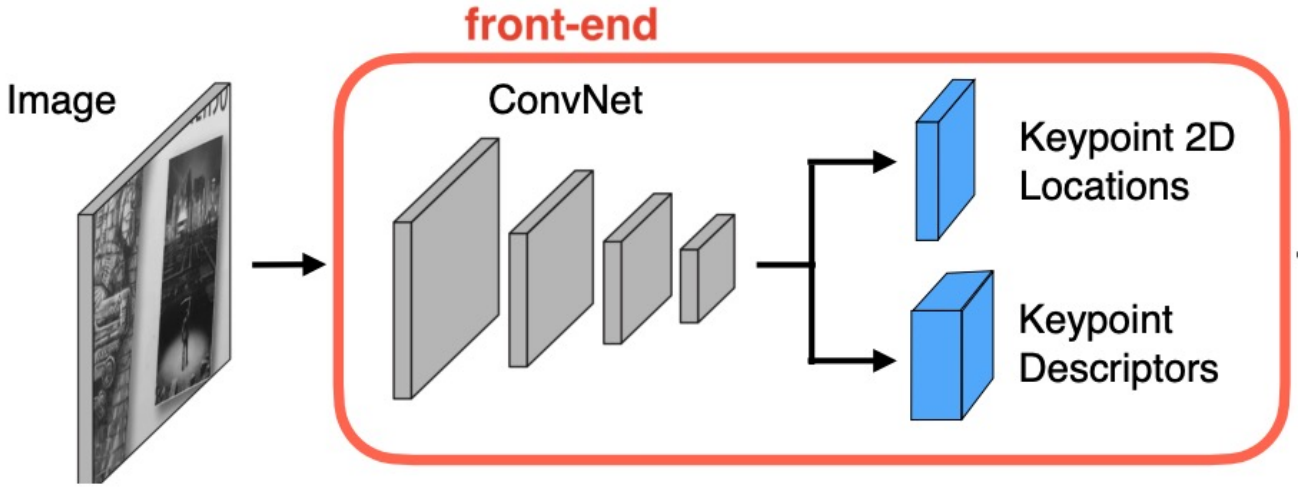
E.g. AUC = 0.87

1 is the best

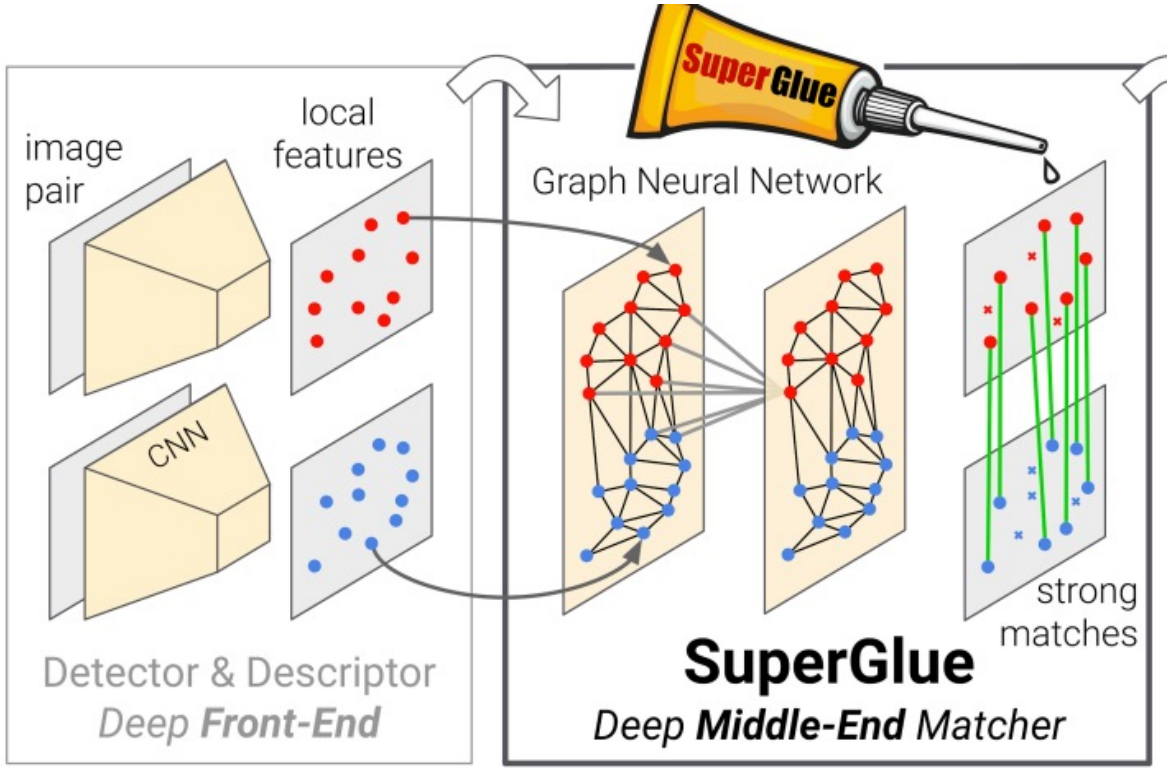
ROC curves – summary

- By thresholding the match distances at different thresholds, we can generate sets of matches with different true/false positive rates
- ROC curve is generated by computing rates at a set of threshold values swept through the full range of possible threshold
- Area under the ROC curve (AUC) summarizes the performance of a feature pipeline (higher AUC is better)
- We will come back to this in binary classification, face verification, object detection, image retrieval, etc.

Local features & matching in Deep Learning era

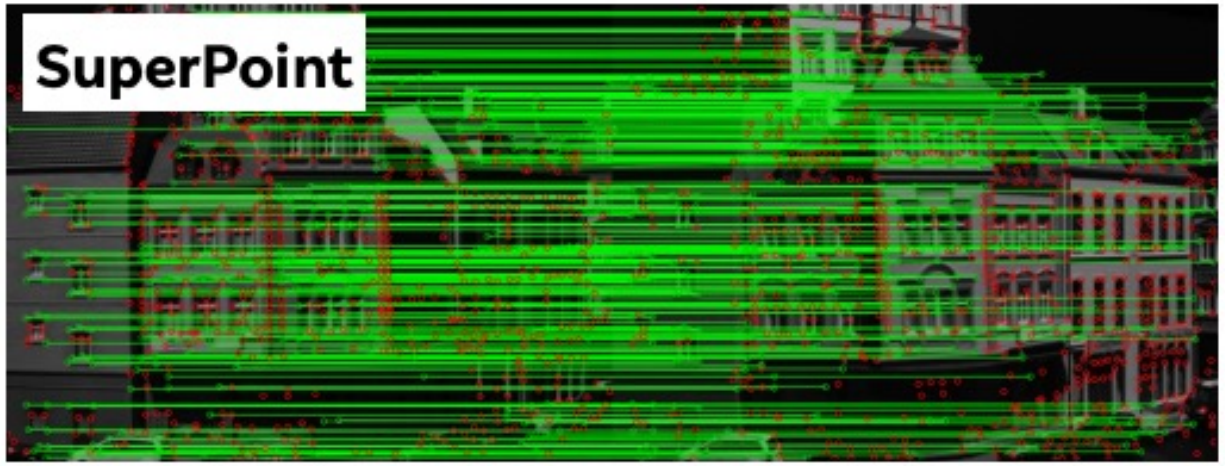


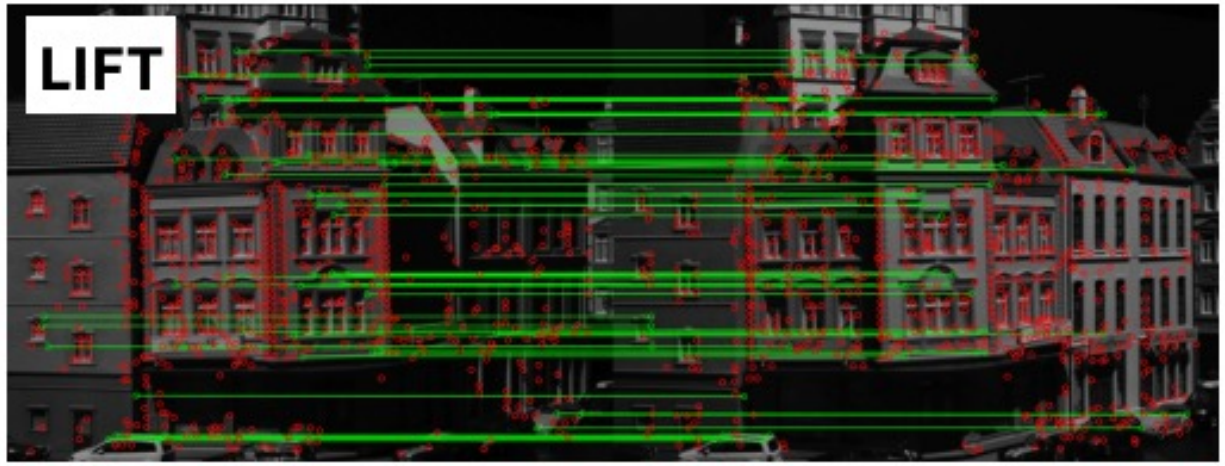
- Local Features = SuperPoint
- Train first on synthetic data
- Self-supervised learning on real data.

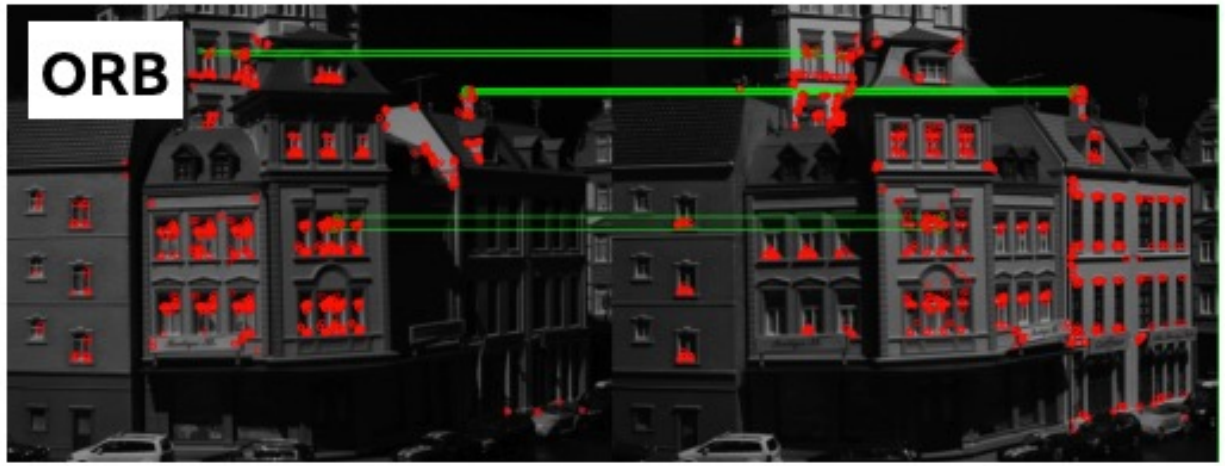


- Feature Matching = SuperGlue

Performance of SuperPoint

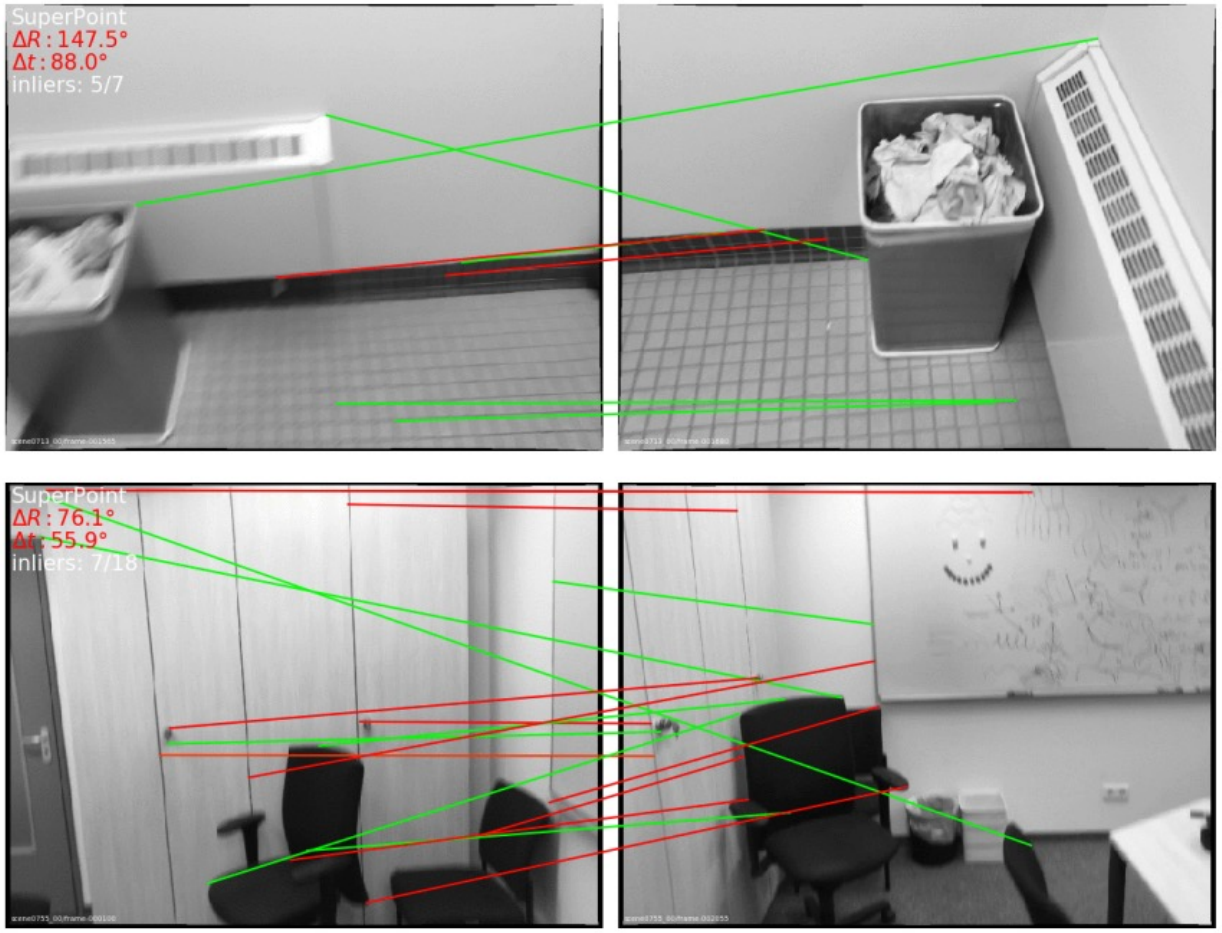




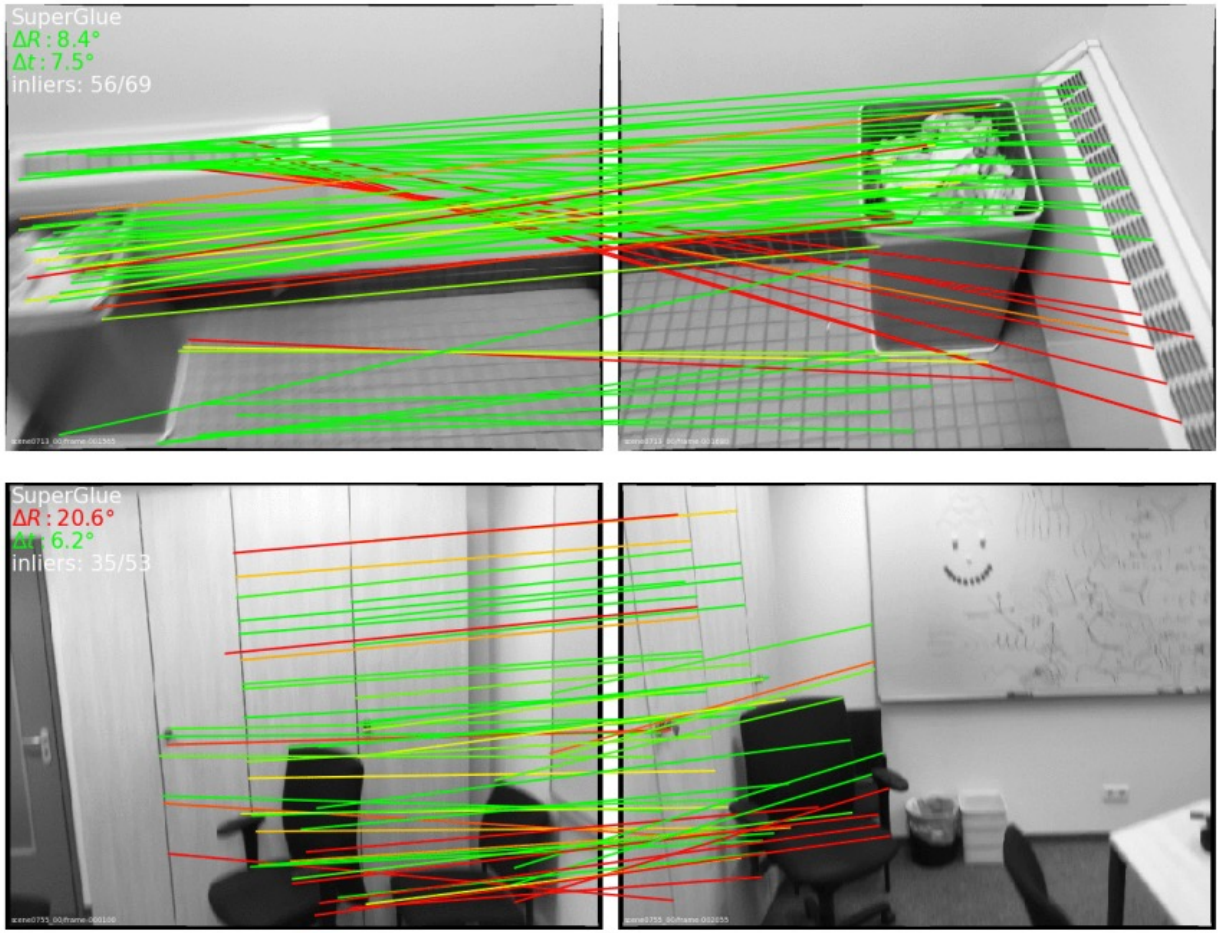


Performance of SuperGlue

SuperPoint + NN + heuristics



SuperPoint + **SuperGlue**



SuperGlue: more **correct matches** and fewer **mismatches**

Slide Credits

- CS5670, Introduction to Computer Vision, **Cornell Tech, by Noah Snavely.**
- CS 194-26/294-26: Intro to Computer Vision and Computational Photography, **UC Berkeley, by Alyosha Efros.**
- Fall 2022 CS 543/ECE 549: Computer Vision, UIUC, by **Svetlana Lazebnik.**