Announcements / Reminders

- Homework 1 is due on 09/29
  - If you are sending me a question, please cc the TA
  - One of us will respond to you faster!

- Project Proposal is due on 10/03 (see Blackboard)
  - written collaboratively by the group
  - submitted individually by each student

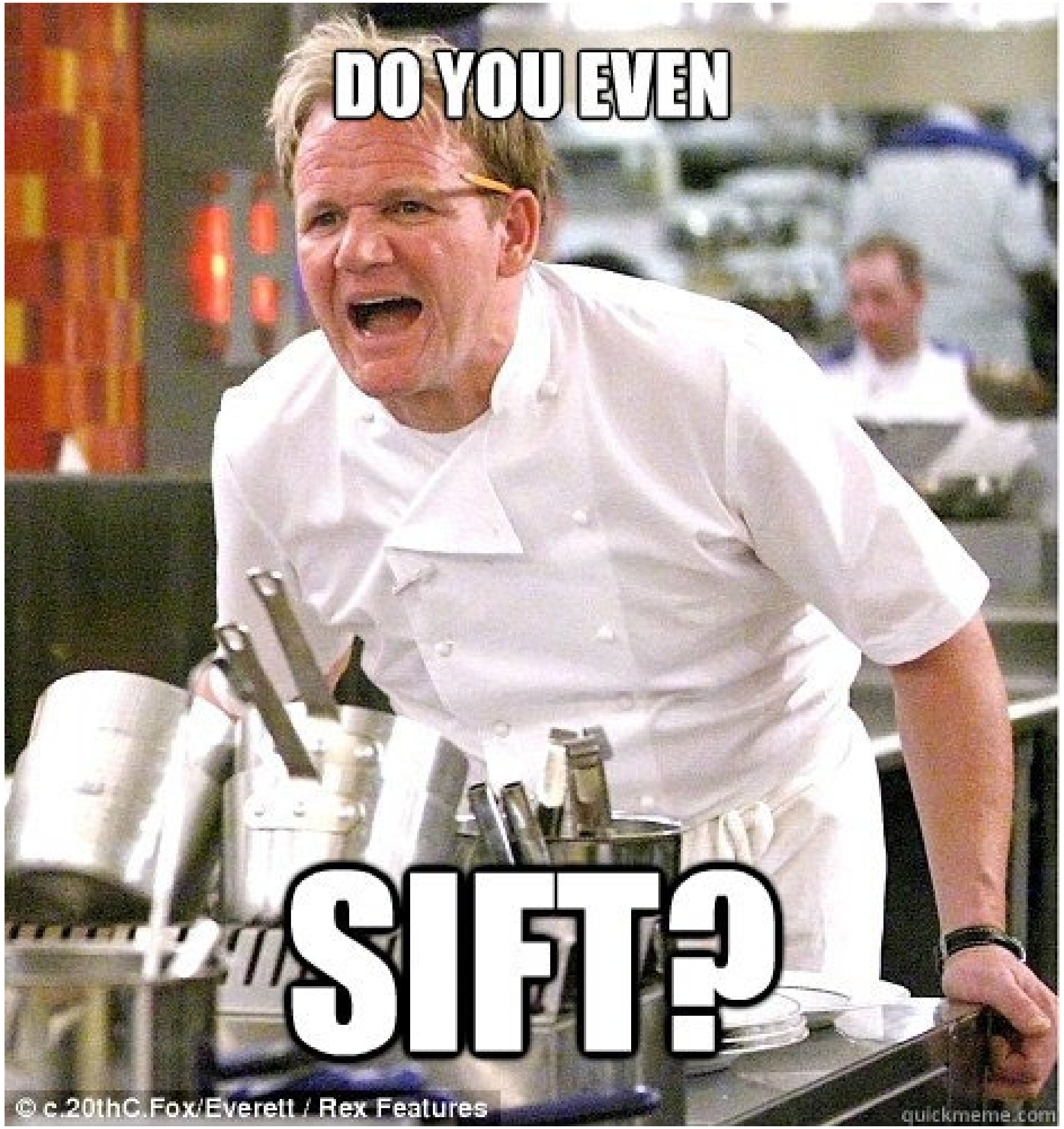
- You are *highly* encouraged to choose your own topic

- On Wednesday (09/24) we will release a set of seed ideas

CMSC 472/672

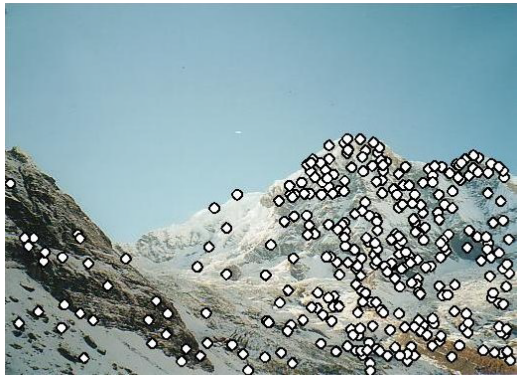
Lecture 6 Addendum

Image Features III



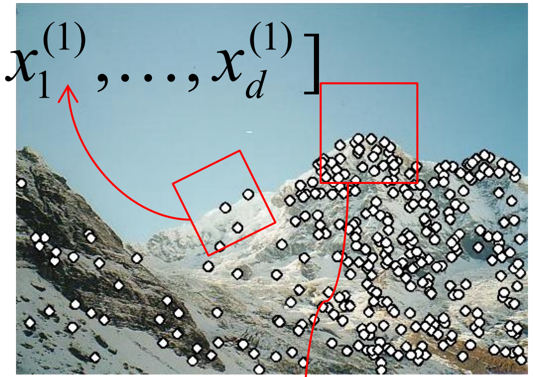
Features: Main Components

1. **DETECTION**
   Identify "interest points"



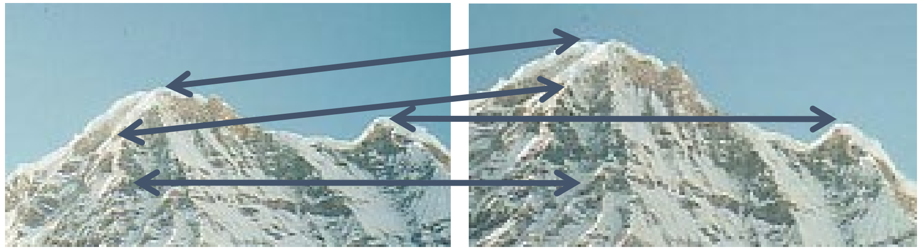
2. **DESCRIPTION**
   Extract "feature descriptor" vectors surrounding each interest point

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



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

3. **MATCHING**
   Determine correspondence between descriptors in two views



Slide Credit: Kristen Grauman

Invariance and Discriminability

- **Invariance:**
  - Descriptor shouldn’t change even if image is transformed

- **Discriminability:**
  - Descriptor should be highly unique for each point

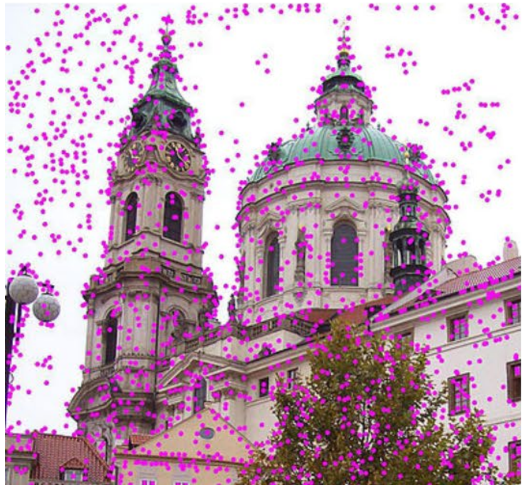
Invariant descriptors

- 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

Classical Feature Detector+Descriptor: SIFT



SIFT

(Scale Invariant Feature Transform)

SIFT

(Scale Invariant Feature Transform)

SIFT describes both a **detector** and **descriptor**

1. Multi-scale extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Scale Invariant Feature Transform

- 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
- Shift the bins so that the biggest one is first

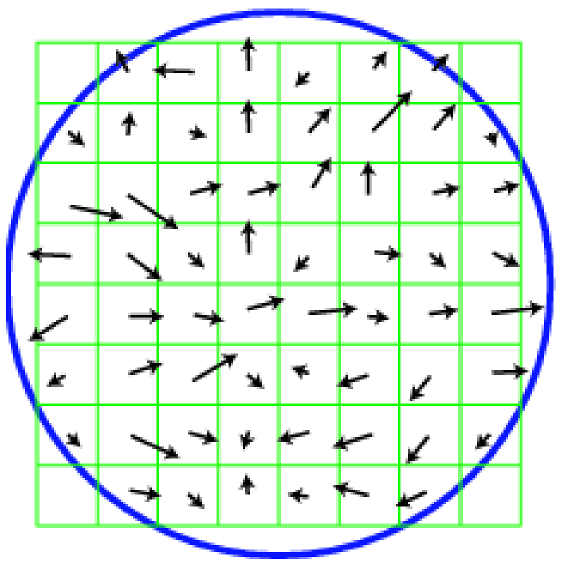
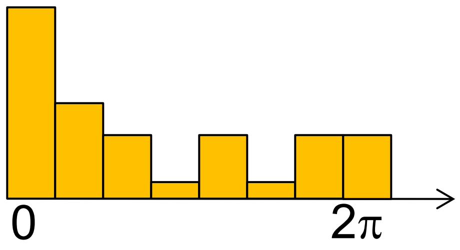
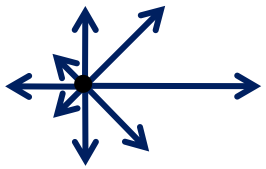


Image gradients



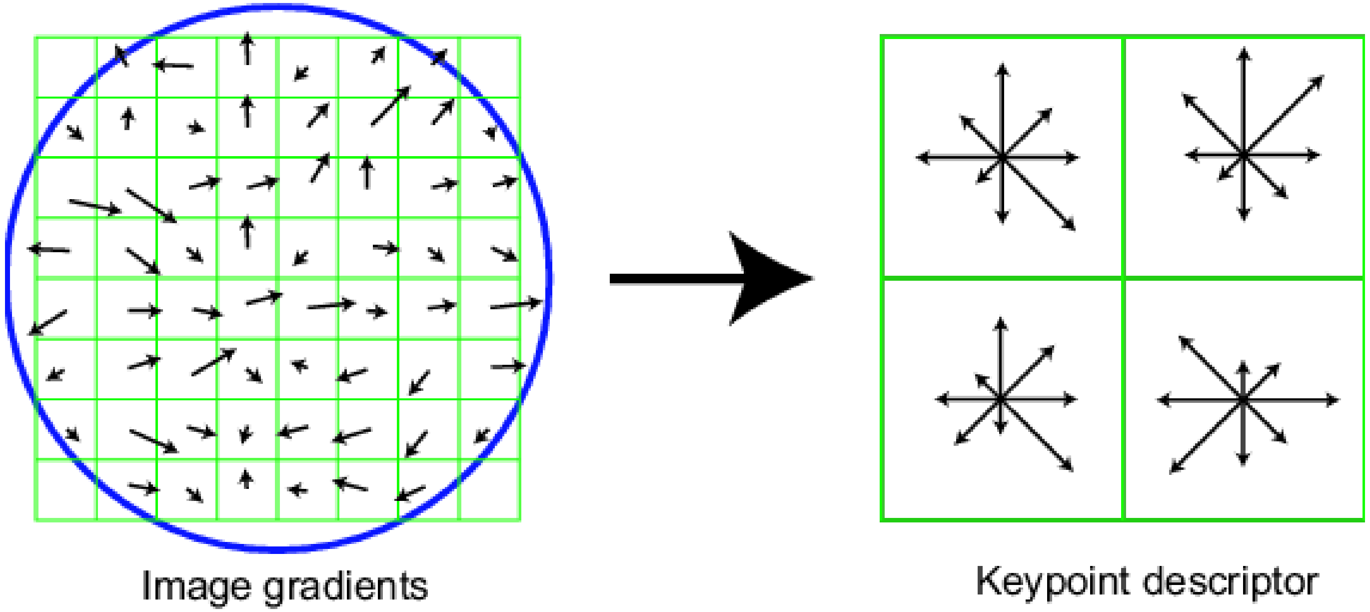
angle histogram



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



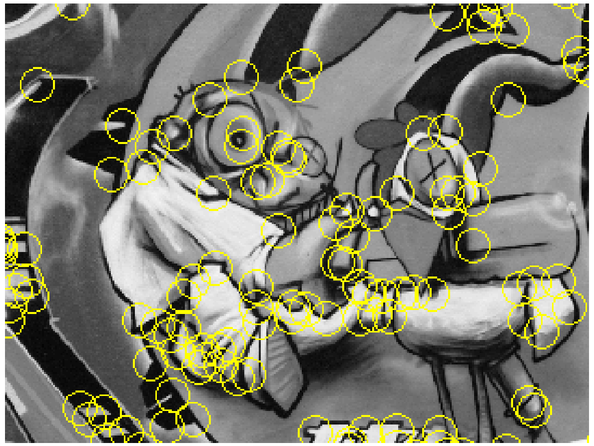
Properties of SIFT

Extraordinarily robust matching technique

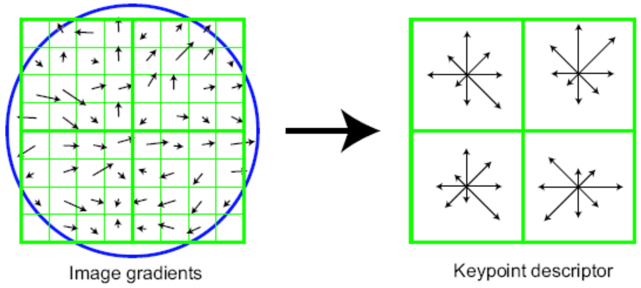
- Can handle changes in viewpoint (up to about 60 degree out of plane rotation)
- Can handle significant changes in illumination (sometimes even day vs. night (below))
- Pretty fast—hard to make real-time, but can run in <1s for moderate image sizes
- Lots of code available

Feature Detection and Description

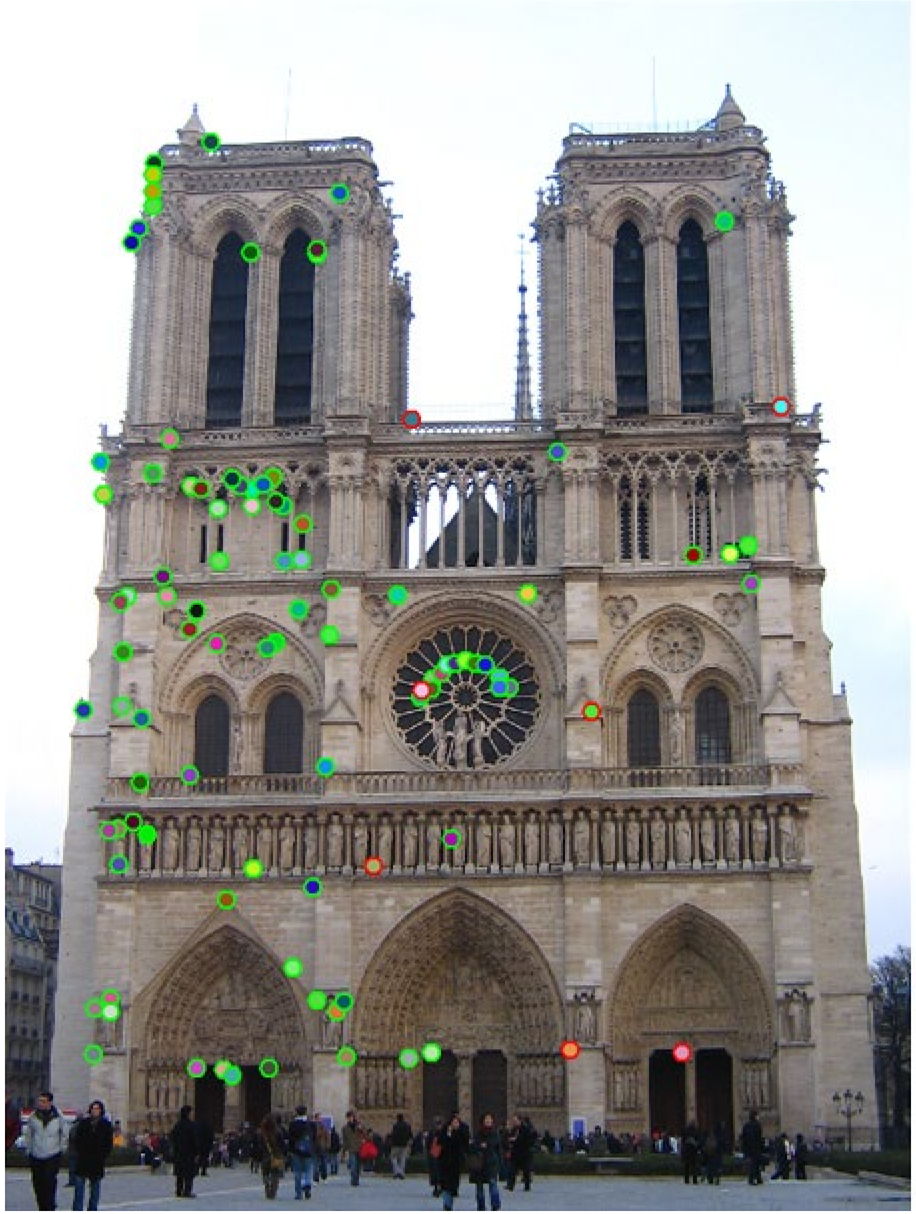
- Feature detection: repeatable and distinctive
  - Corners, blobs
  - Harris, DoG

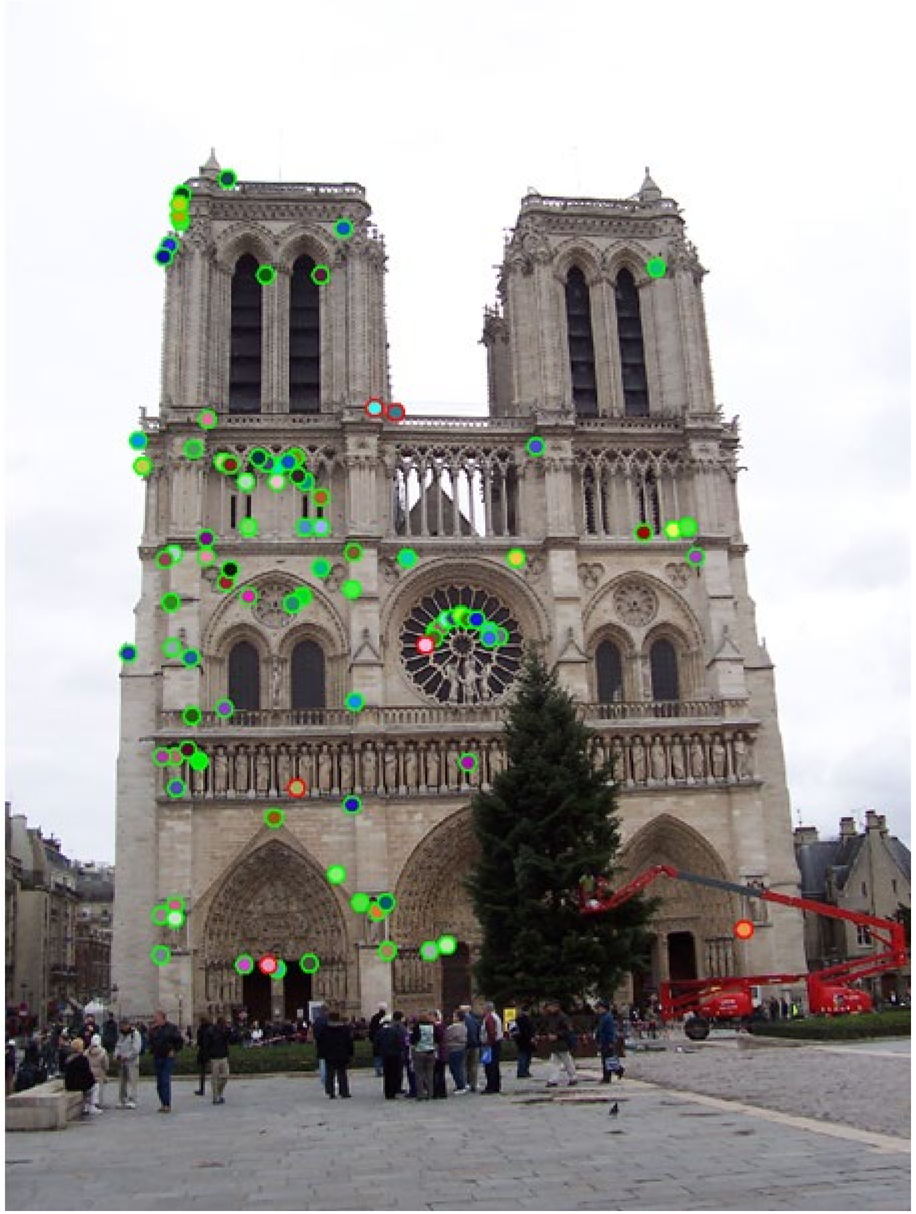


- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT and variants are typically good for stitching and recognition
  - But, need not stick to one



Which features match?





Feature Matching: Problem Statement

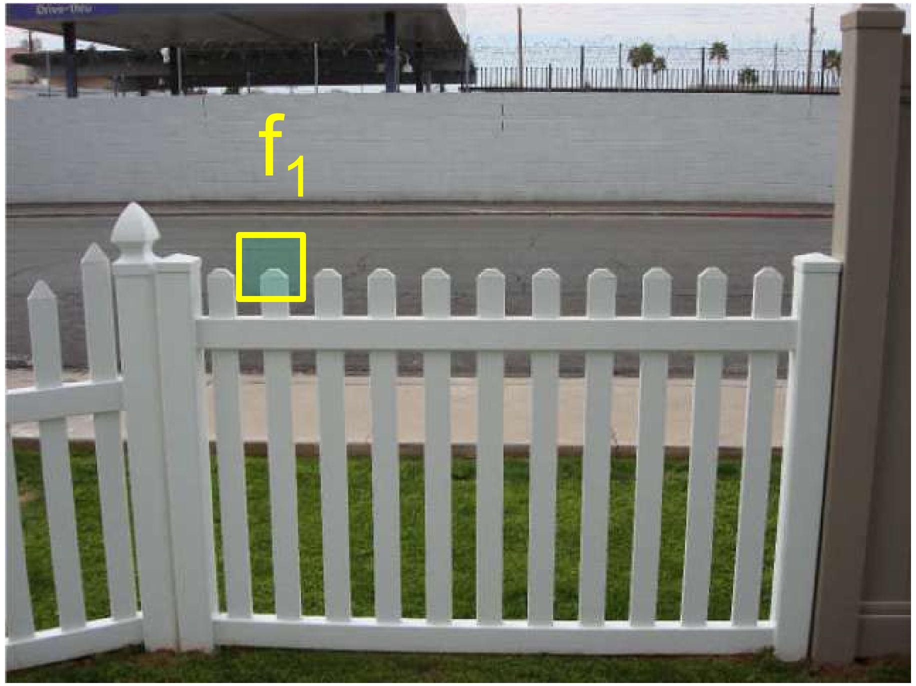
Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

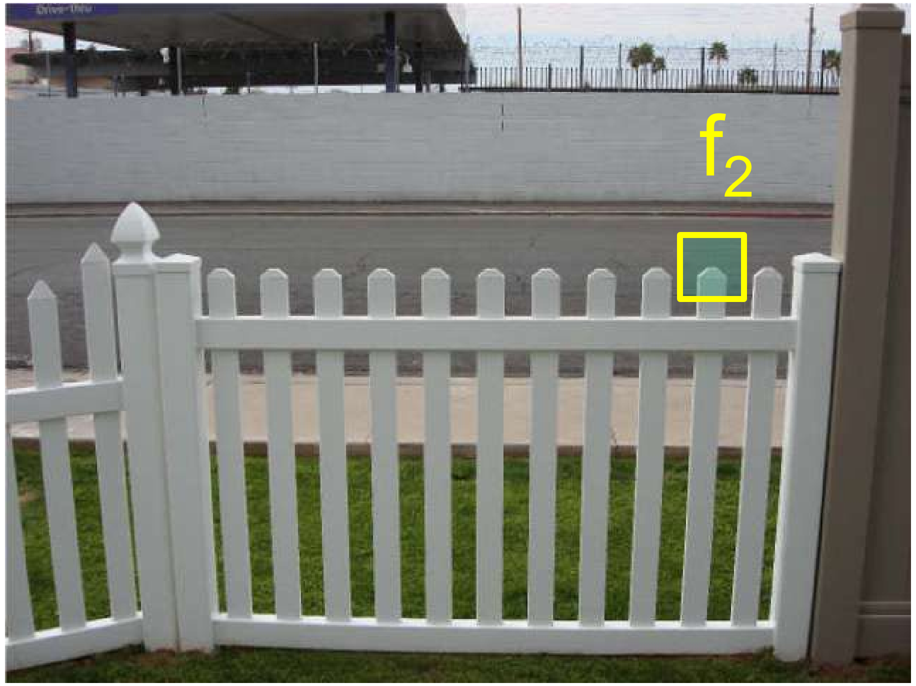
Feature distance

How to define the difference between two features f_1, f_2?

- Simple approach: L_2 distance, $\| f_1 - f_2 \|$
- can lead to small distances for ambiguous (incorrect) matches



I_1



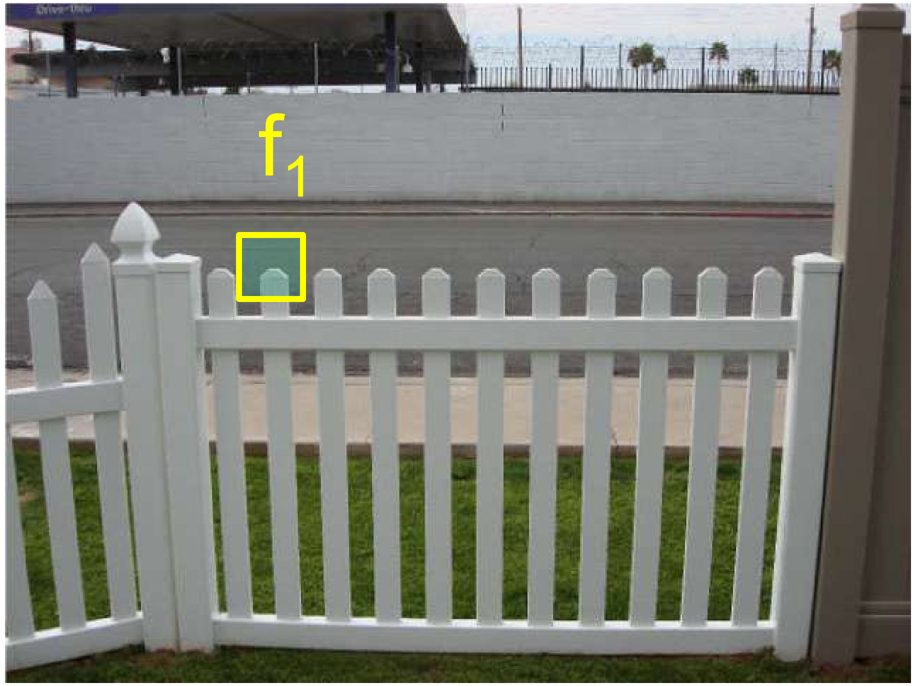
I_2

Feature distance

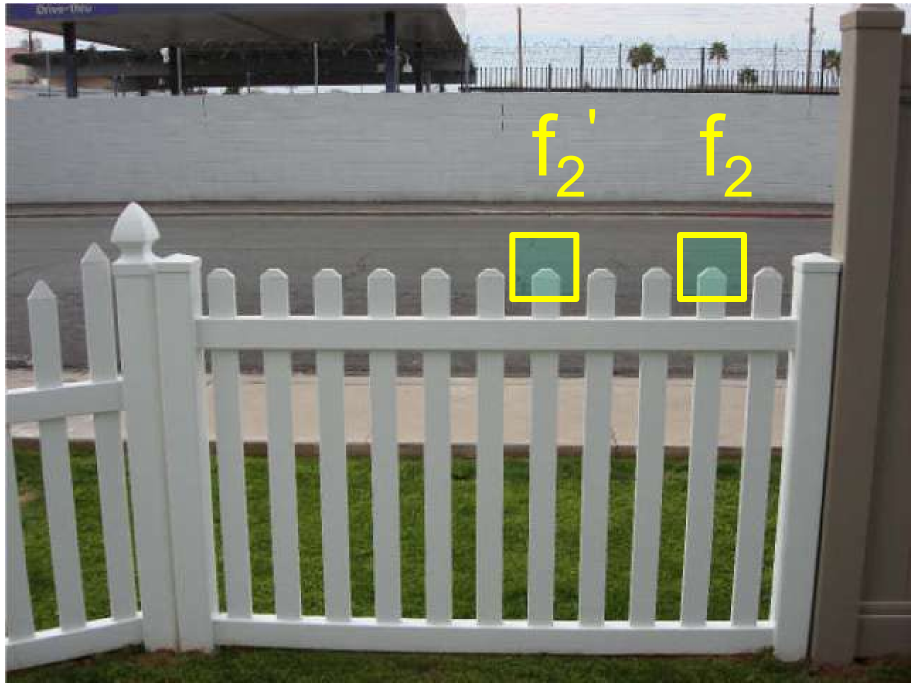
How to define the difference between two features f_1, f_2?

Better approach: ratio distance $= \dfrac{\| f_1 - f_2 \|}{\| f_1 - f_2' \|}$

- f_2 is the best SSD match to f_1 in I_2
- f_2' is the 2nd best SSD match to f_1 in I_2
- gives large values for ambiguous matches



I_1



I_2

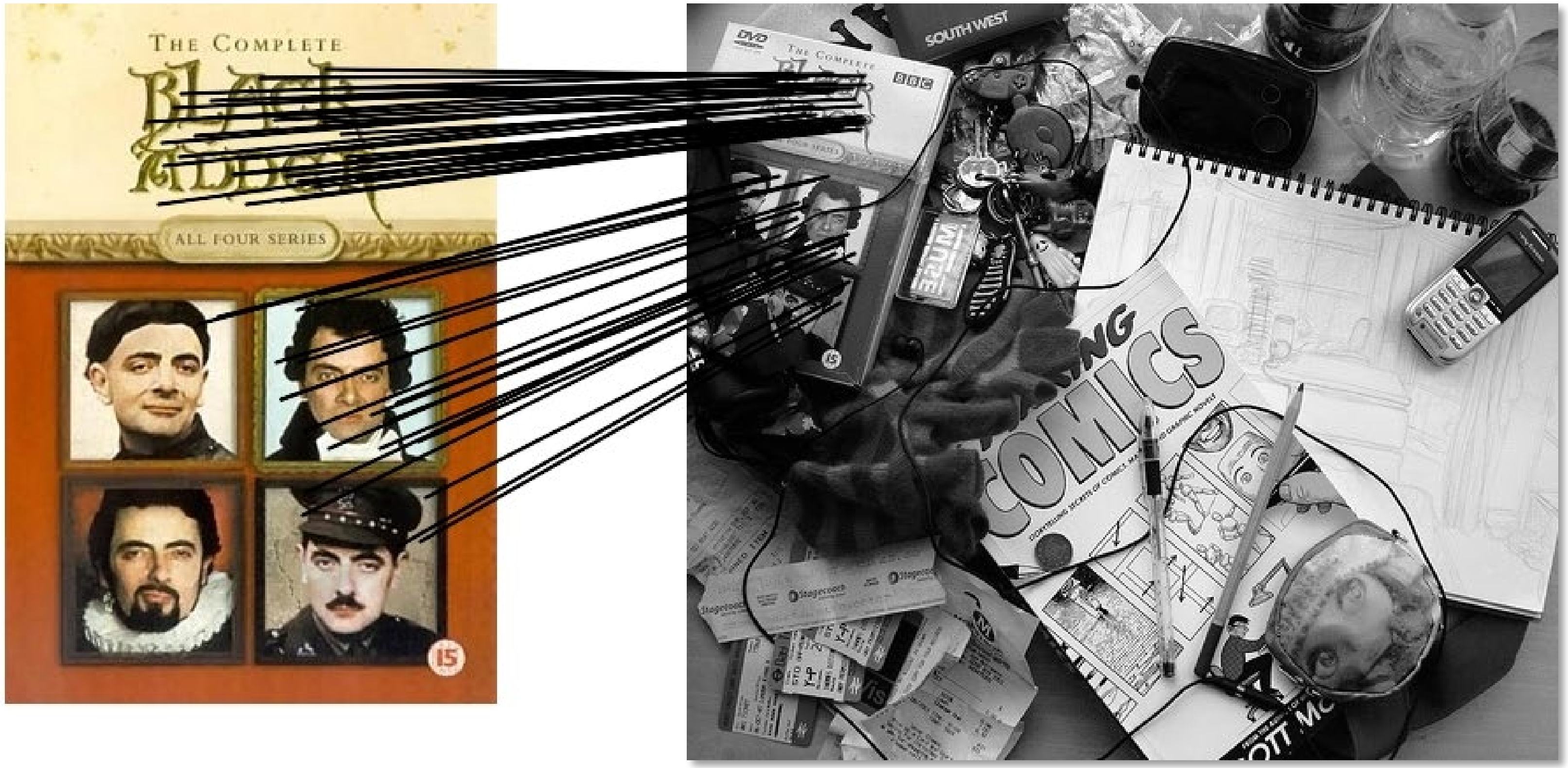
Feature Selection

Each “match” (i.e. pair of features) has a ratio score associated with it.

A high ratio score indicates more ambiguity (i.e. 1^{st} best and 2^{nd} best matches have identical distances)

Solution: use a threshold and only select the matches ***below*** the threshold.

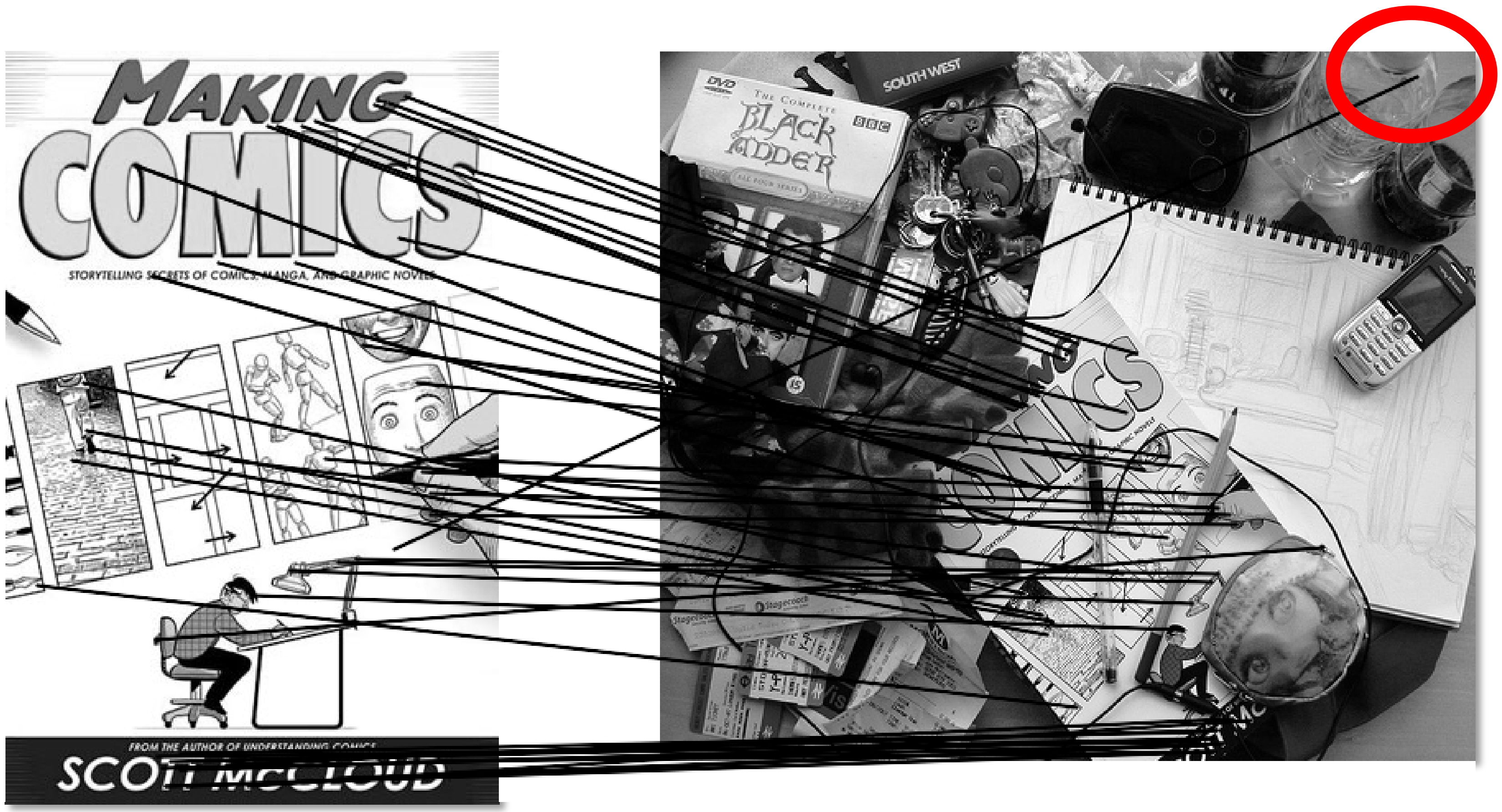
Feature matching example



58 matches (thresholded by ratio score)

Feature matching example

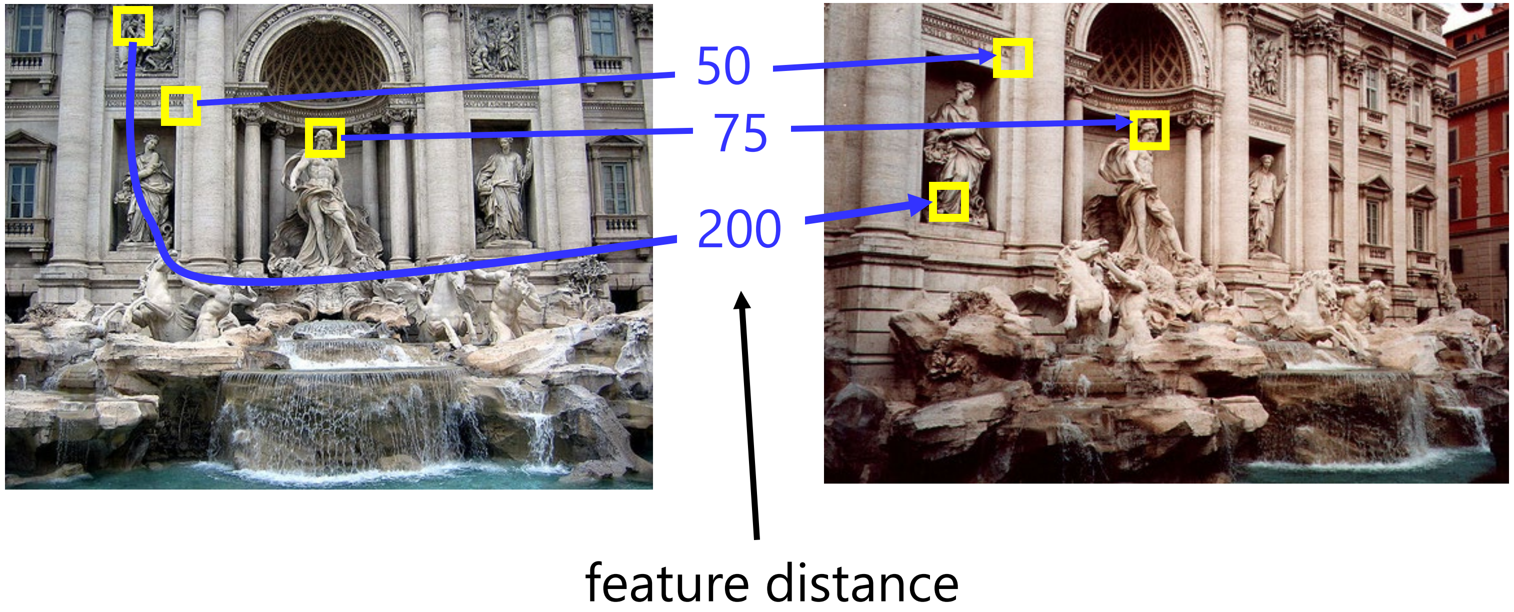
We'll deal with
outliers later



51 matches (thresholded by ratio score)

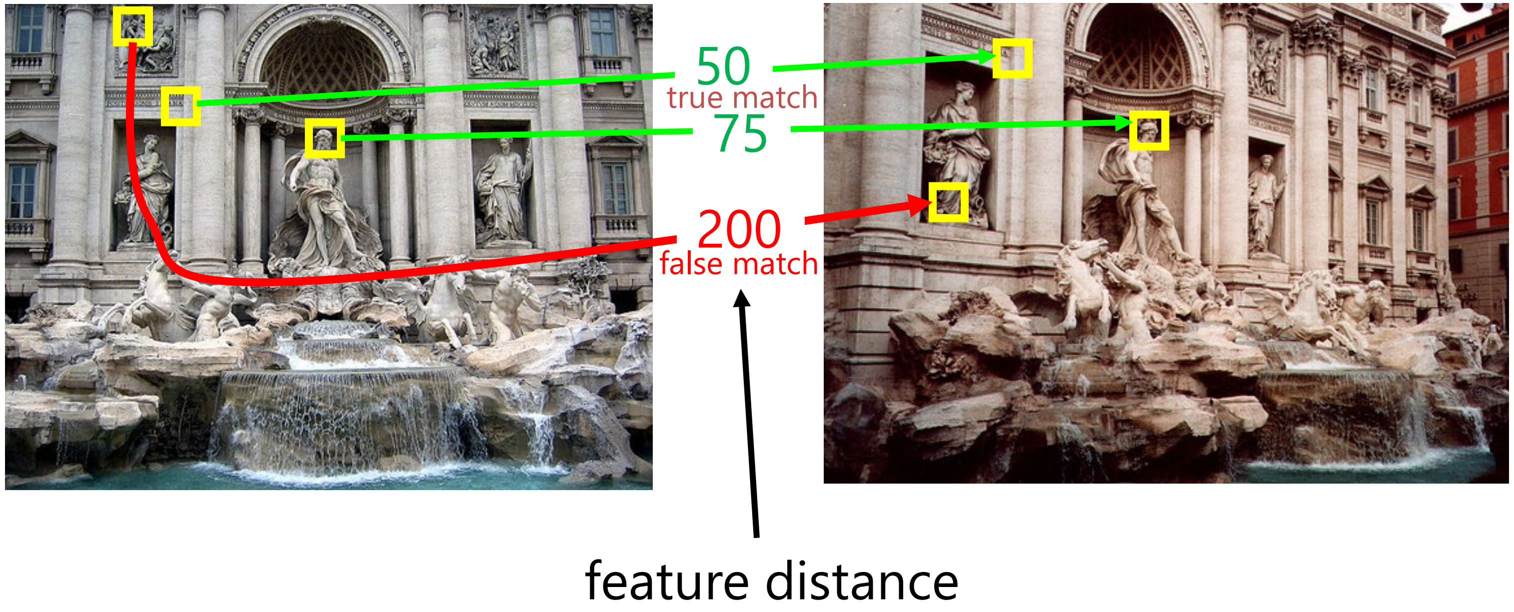
Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives

How can we measure the performance of a feature matcher?

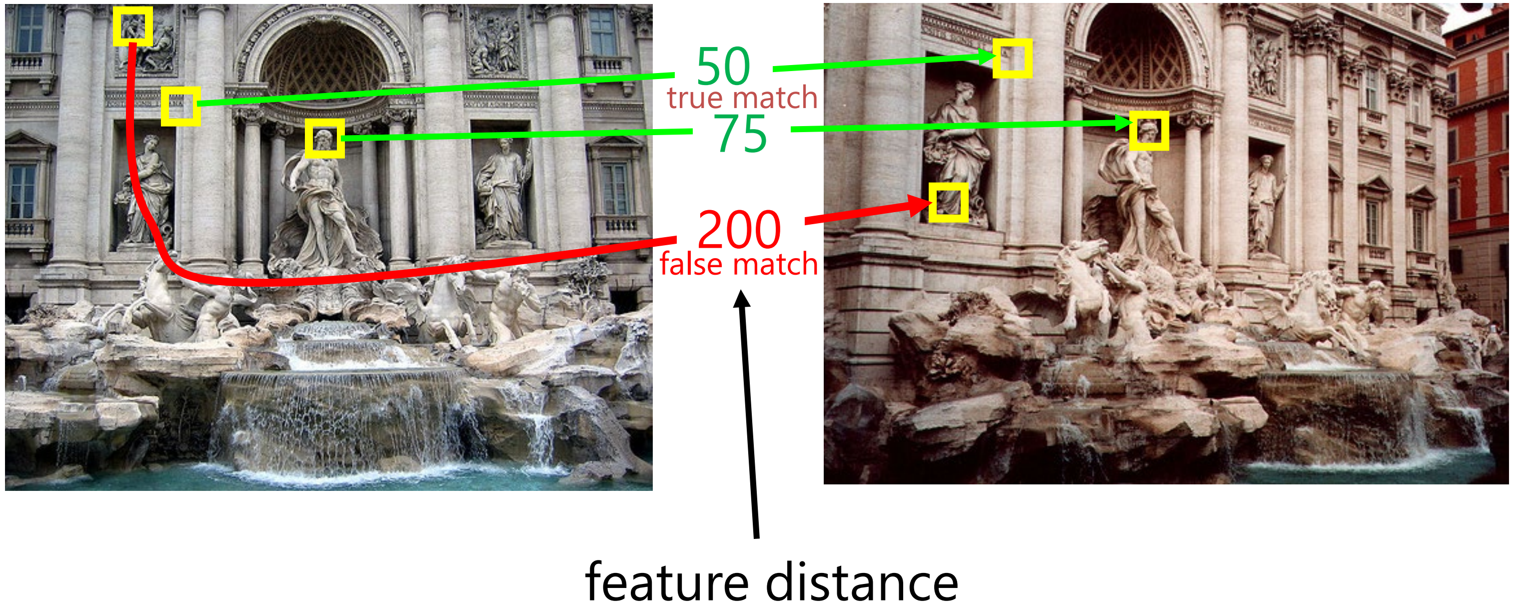


The distance threshold affects performance

- True positives = # of correctly detected matches that survive the threshold
- False positives = # of incorrectly detected matches that survive the threshold

True/false positives

How can we measure the performance of a feature matcher?



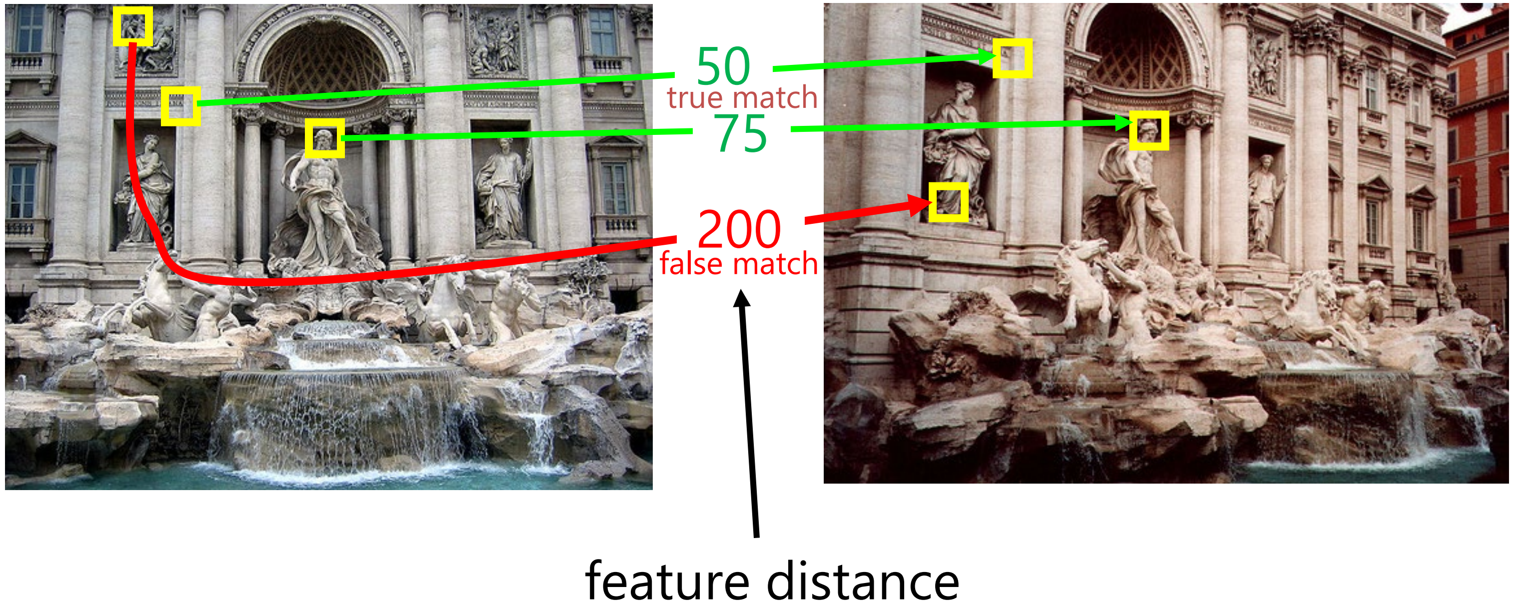
Suppose we want to **maximize true positives**.

How do we set the threshold?

(Note: we keep all matches with distance below the threshold.)

True/false positives

How can we measure the performance of a feature matcher?



Suppose we want to **minimize false positives**.

How do we set the threshold?

(Note: we keep all matches with distance below the threshold.)

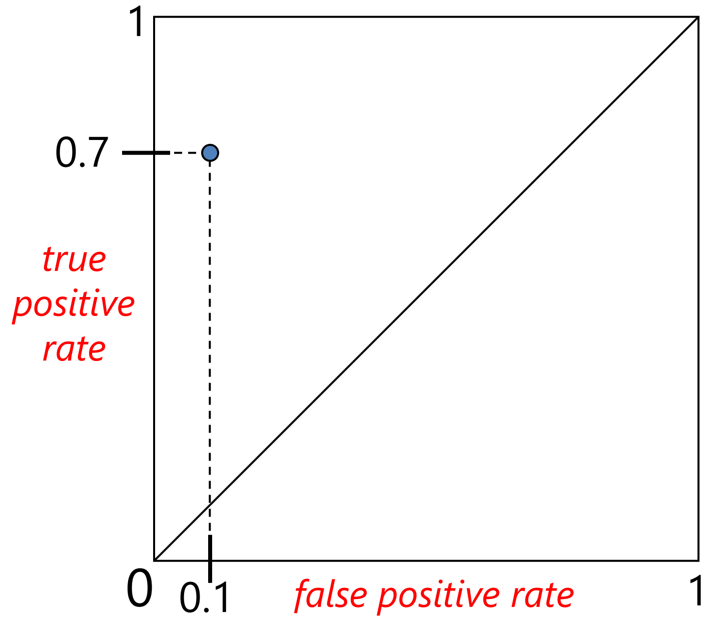
Example

- 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

Evaluating the results

How can we measure the performance of a feature matcher?

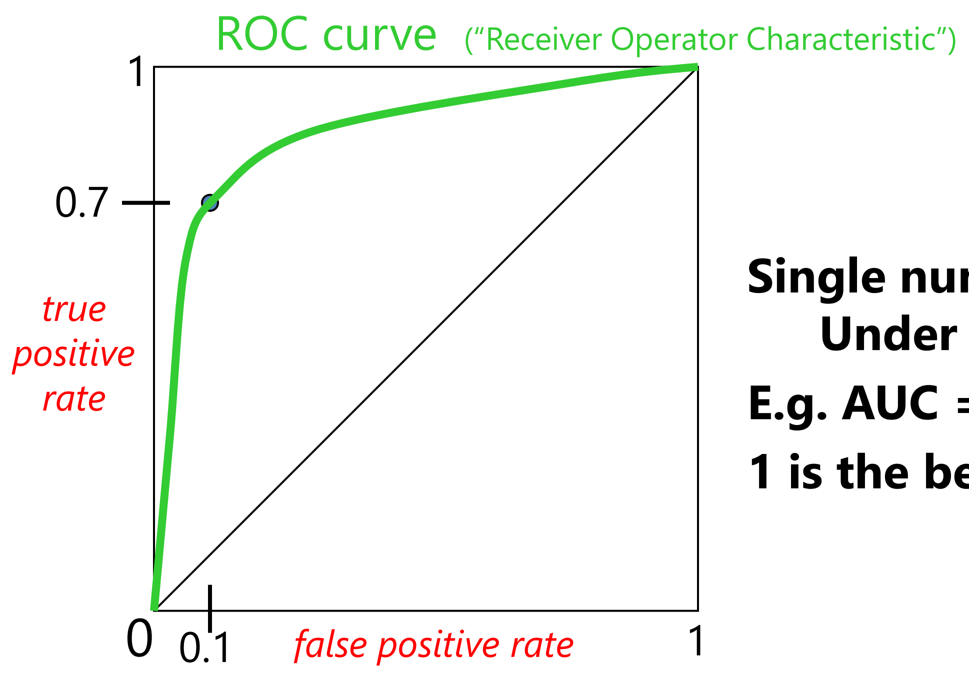
$$\text{recall} = \frac{\text{\# true positives surviving threshold}}{\text{\# total correct matches (positives)}}$$



$$1 - \textit{specificity} = \frac{\text{\# false positives surviving threshold}}{\text{\# total incorrect matches (negatives)}}$$

Evaluating the results

How can we measure the performance of a feature matcher?



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

$$1 - \textit{specificity} = \frac{\text{\# false positives surviving threshold}}{\text{\# total incorrect matches (negatives)}}$$

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)

Lots of applications

Features are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

Feature Matching is Useful for …

Object instance recognition

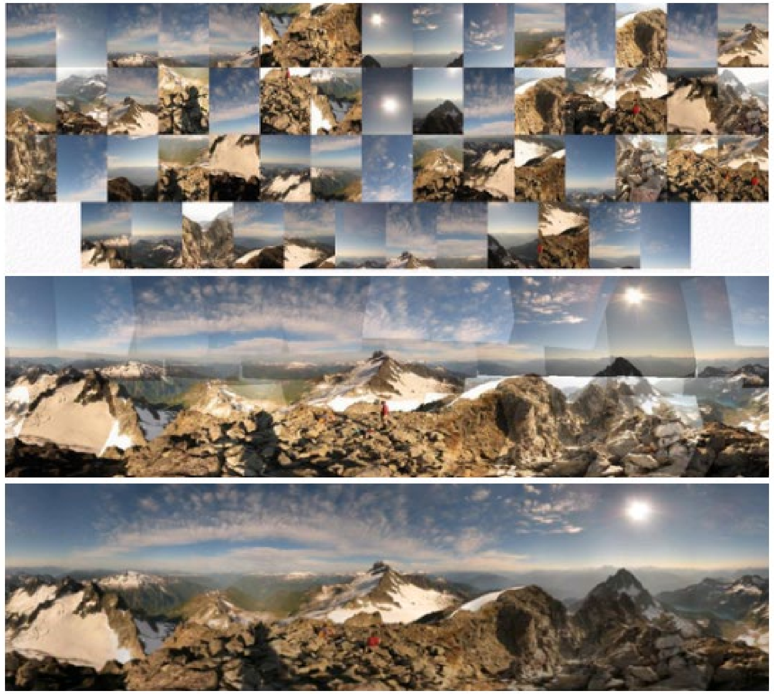
Schmid and Mohr 1997

Sivic and Zisserman, 2003

Rothganger et al. 2003

Lowe 2002

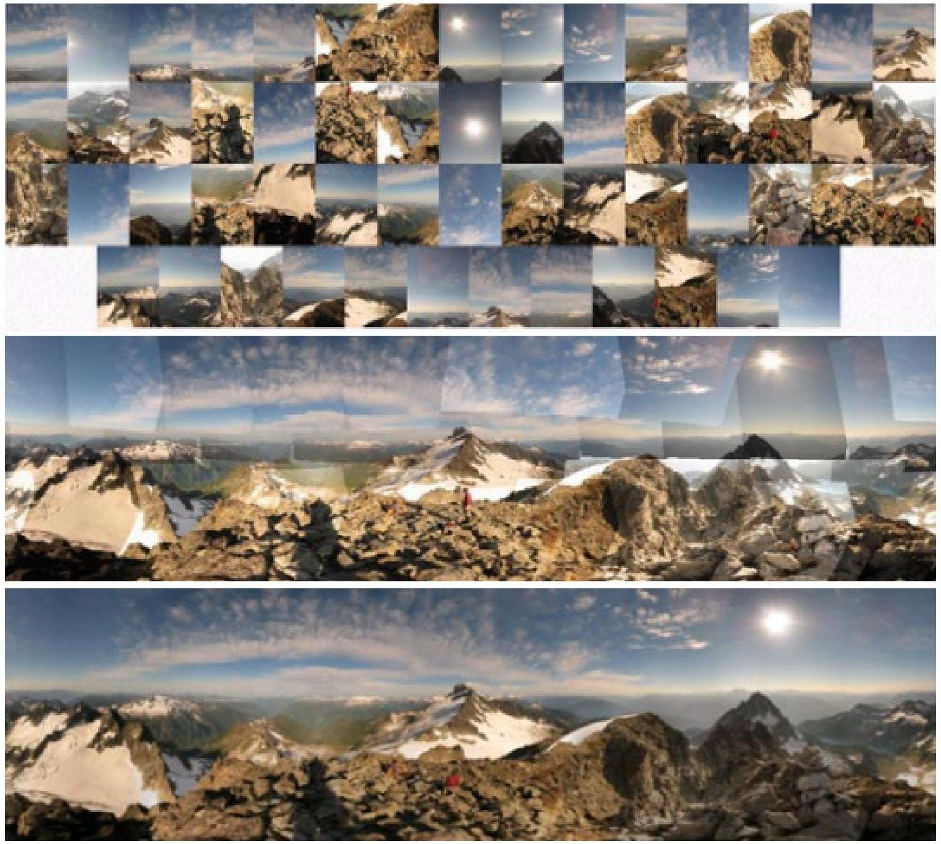
Image mosaicing



Feature Matching is Useful for …

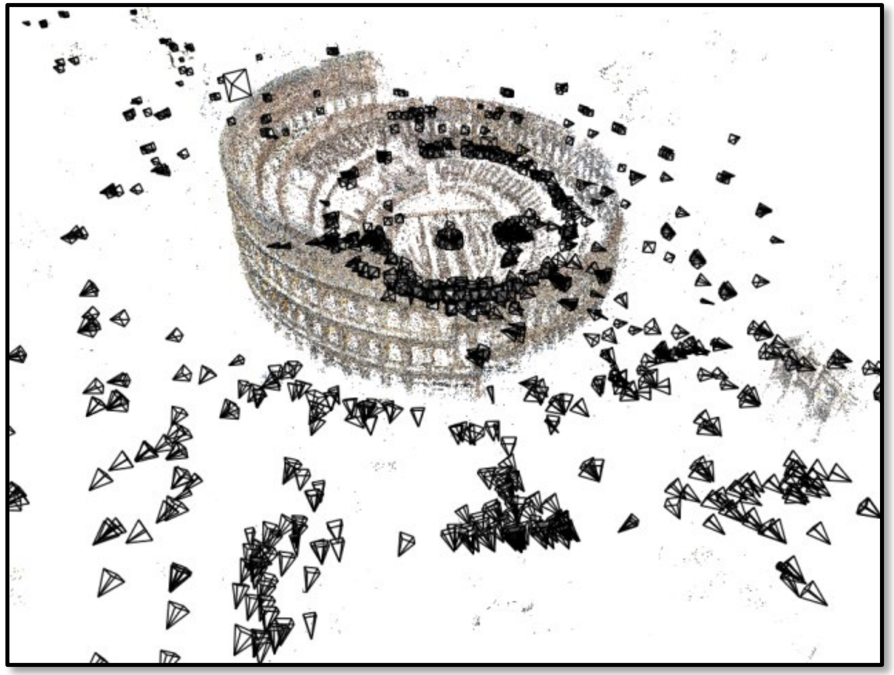
NEXT HOMEWORK !!!

Image mosaicing



3D Reconstruction

Internet Photos ("Colosseum")



Reconstructed 3D cameras and points

Augmented Reality

Now,

The Good Stuff
You've All Been Waiting For …