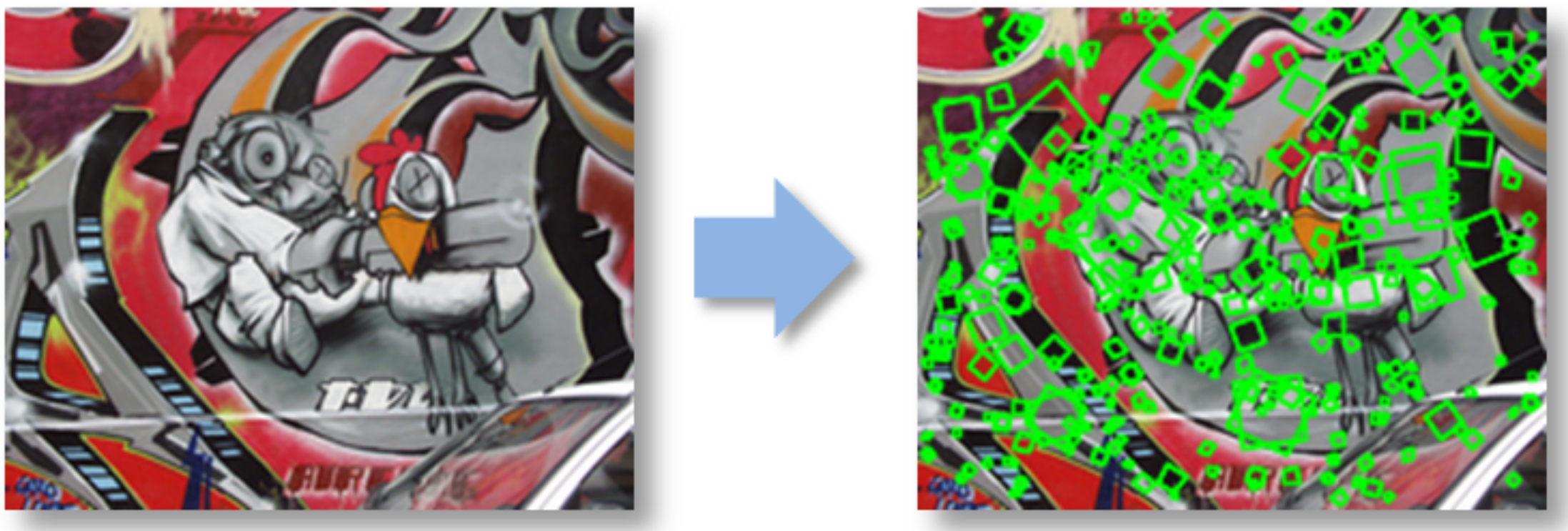
CSCI 497P/597P: Computer Vision



Lecture 10: Image Features

Feature Descriptors

Feature Matching

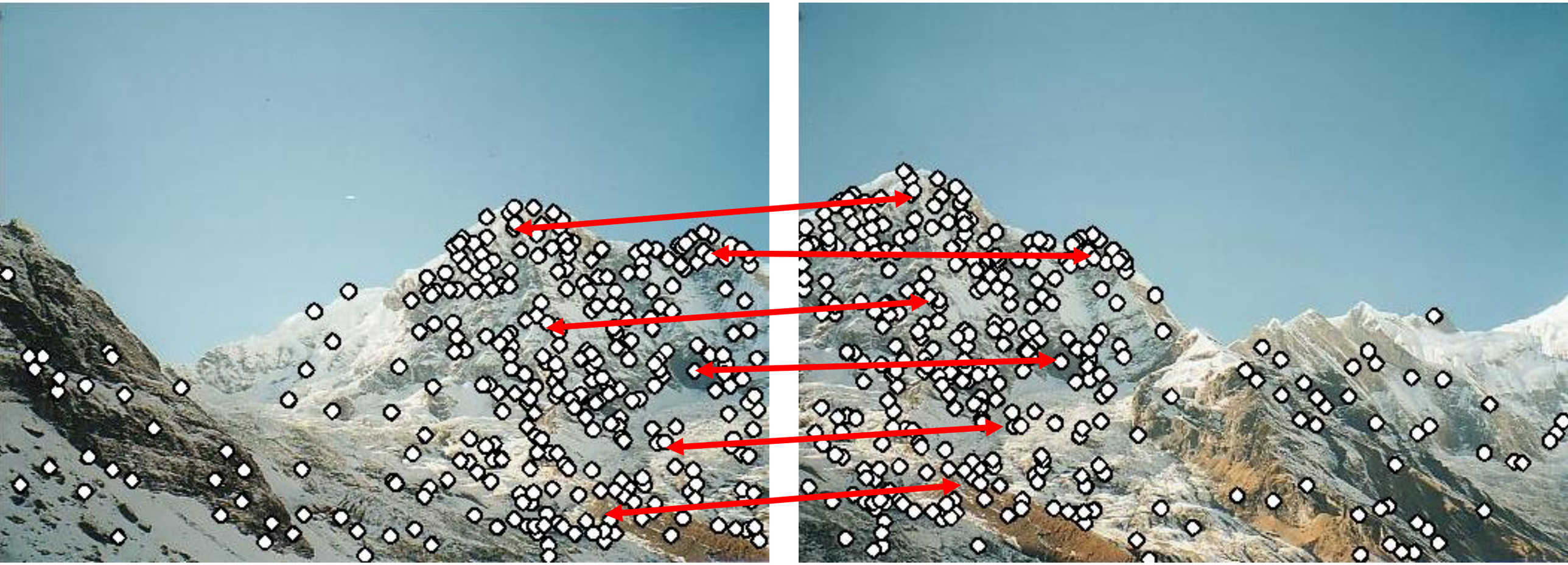
Announcements

- HW2 out. You will be prepared to answer:
  - Problem 1 - now.
  - 3.1-3 - by the end of Tuesday
  - 3.4-5 - by the end of Wednesday
  - 2 and 3.6 - by the end of Friday
  - It's due a week from Wednesday (10/21)

Goals

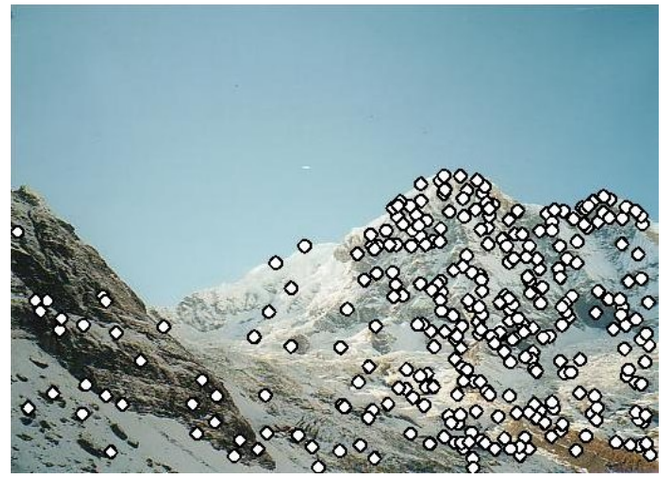
- Understand the concept of invariance as pertains to feature detectors and feature descriptors
- Know the how and why of the MOPS feature descriptor
- Understand the gist of the SIFT descriptor
- Know how and why to **match** features using:
  - The SSD metric
  - The ratio test

Running motivational example: Panorama Stitching



Features - Overview

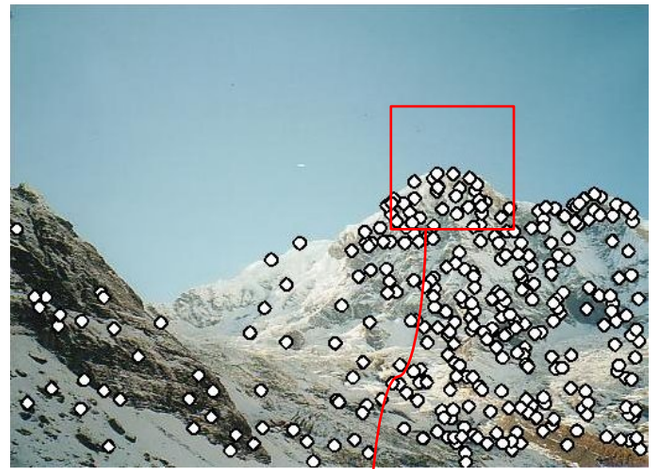
1. **Detect**



detect *unique* points

e.g., using the Harris corner detdctor

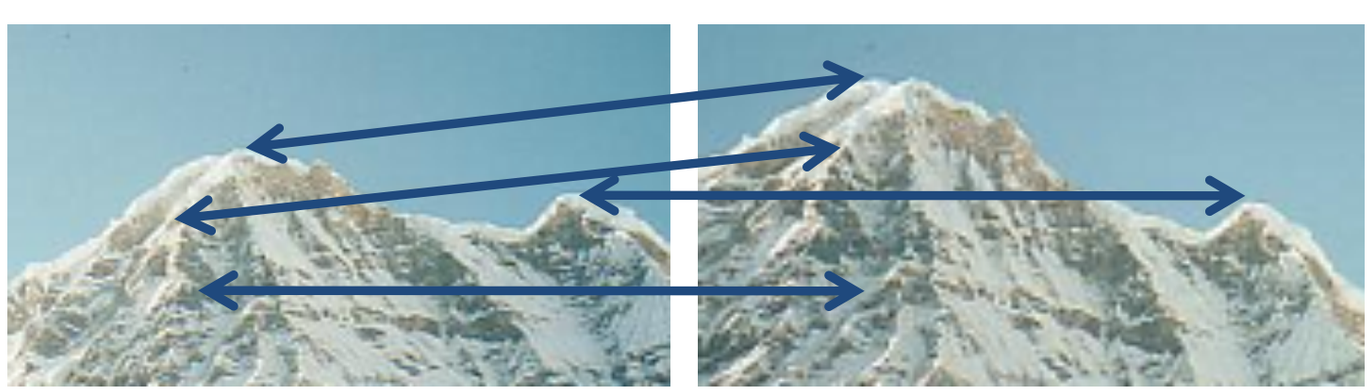
2. Describe



describe using *invariant* representation

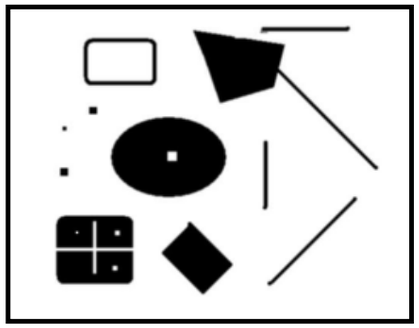
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3. Match



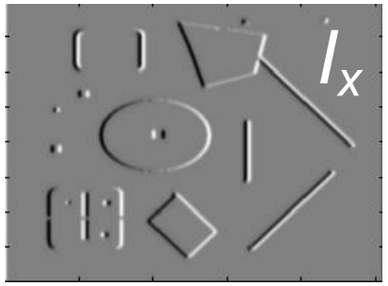
match them *robustly*

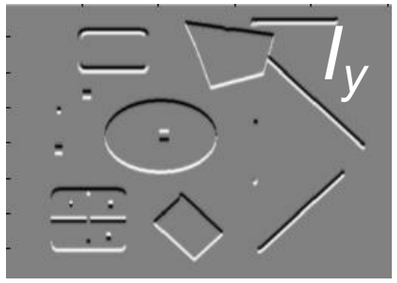
Harris Corners: TL;DM



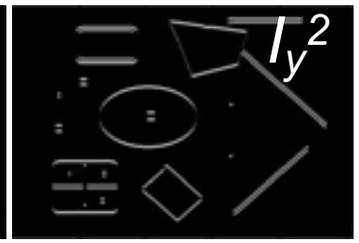
Algorithm:

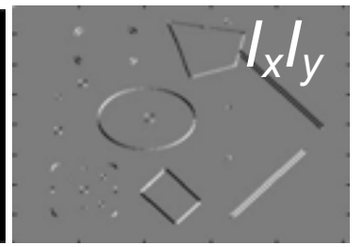
1. Use Sobel filter to estimate gradients I_x, I_y



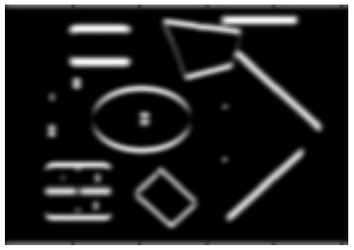


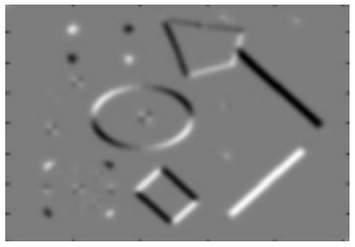
2. Compute I_x^2, I_y^2, I_xI_y





3. Filter each with a K x K mean filter

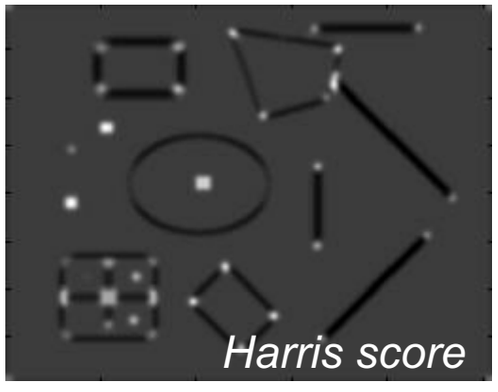




$$\sum_{x,y\in W} I_x^2 \qquad \sum_{x,y\in W} I_y^2 \qquad \sum_{x,y\in W} I_xI_y$$

4. Approximate smallest eigenvalue as:
   det(H) / tr(H)

5. Threshold



6. Maximum filter

Two desirable properties:

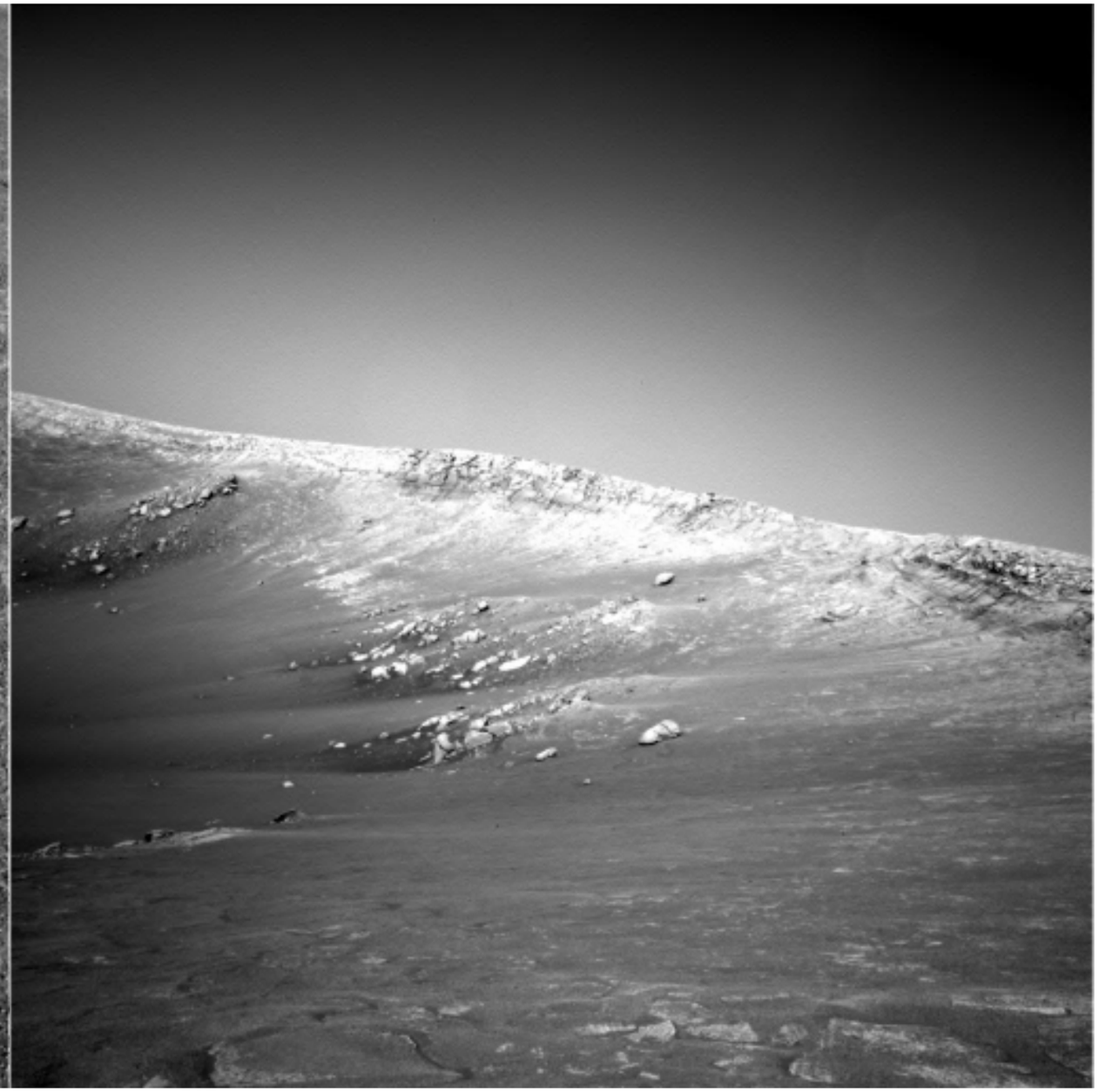
- Uniqueness: features **shouldn't** match if they're from different points in the scene.

- Invariance: features **should** match if they do come from the same point in the scene.

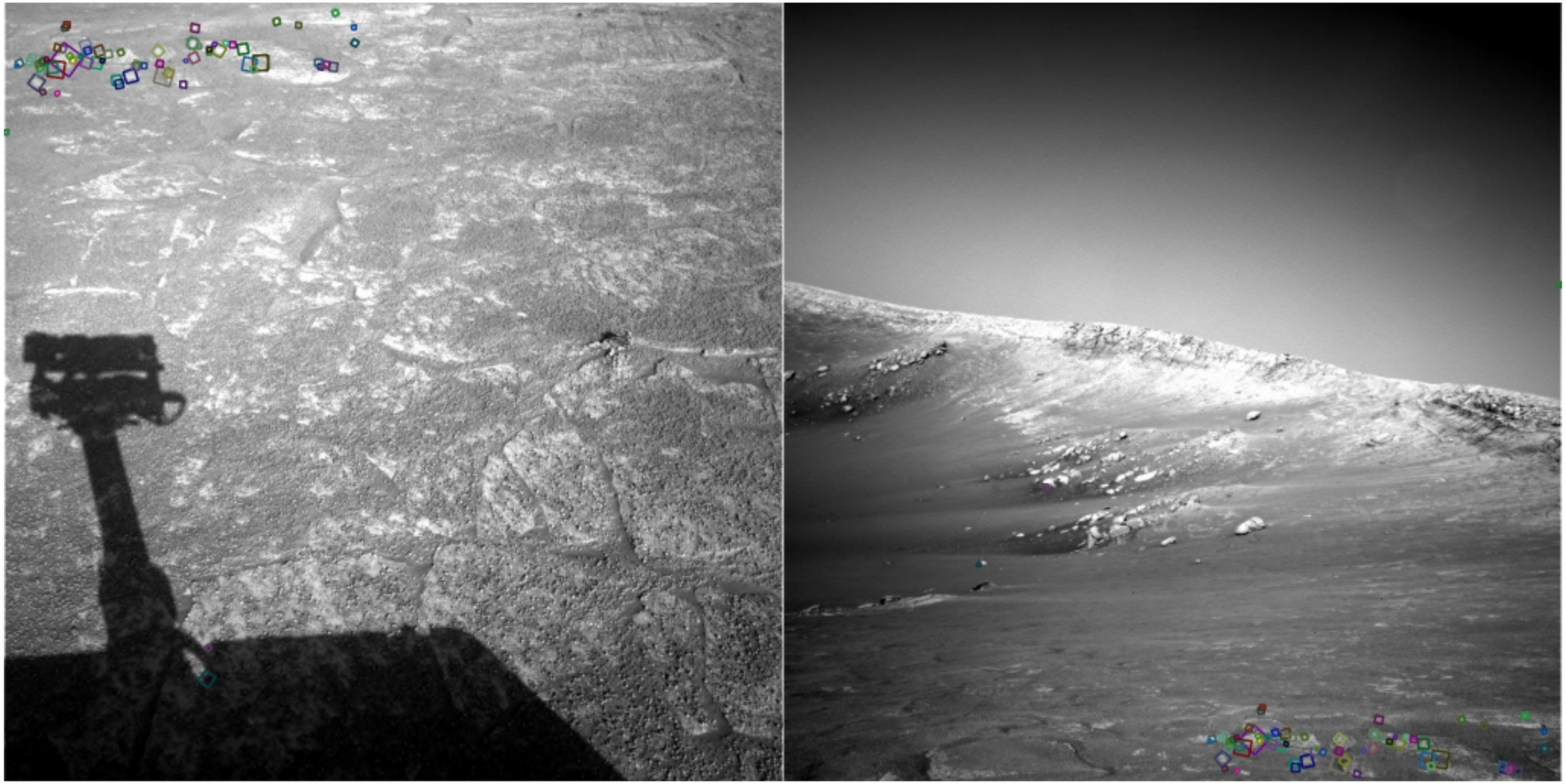
Invariance

Invariance: Hard Mode

Invariance: Mars Mode



Invariance: Mars Mode



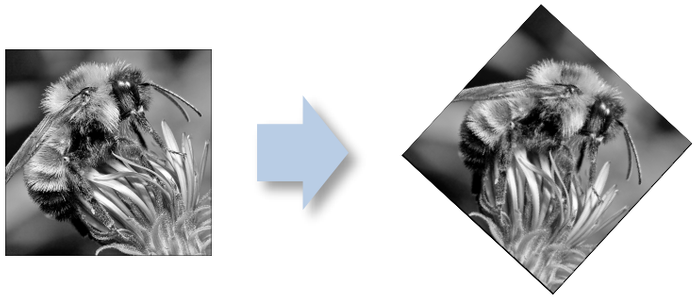
Invariance

Suppose we're comparing two images of the same scene.
What kinds of transformations could relate the two images if:

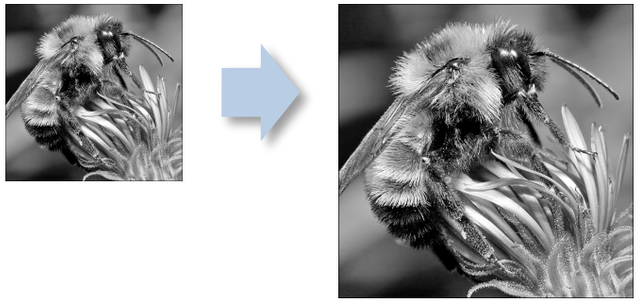
- They are part of a panorama sequence?
- They were taken at different times of day?
- They were taken by different cameras?
- They were taken from different viewpoints?

Desirable Invariances

- Geometric transformations



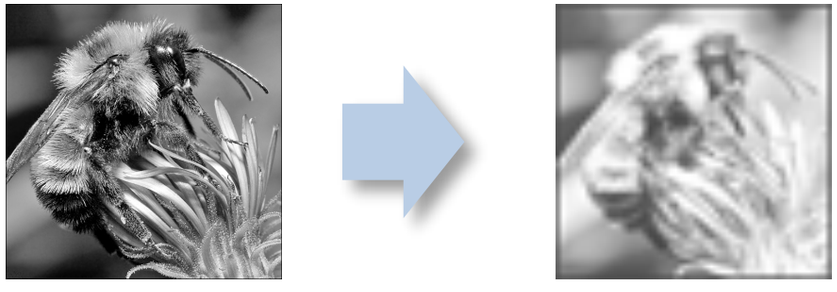
  - Rotation, Translation, Scale



  - (some others, more on this later)

- Photometric transformations:

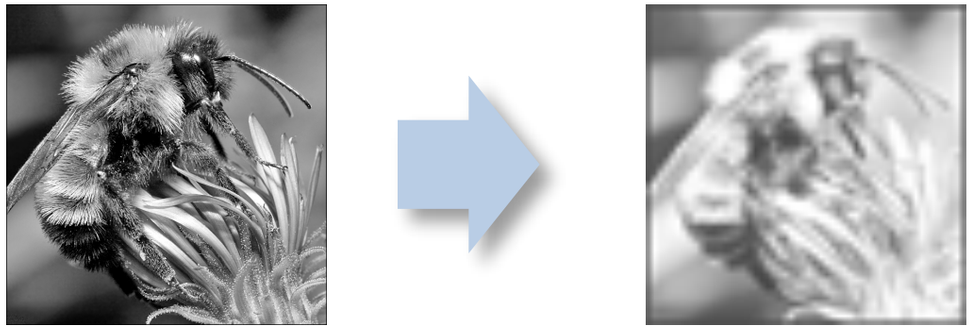
  - brightness shift and scale



  - contrast

Harris detector: invariance

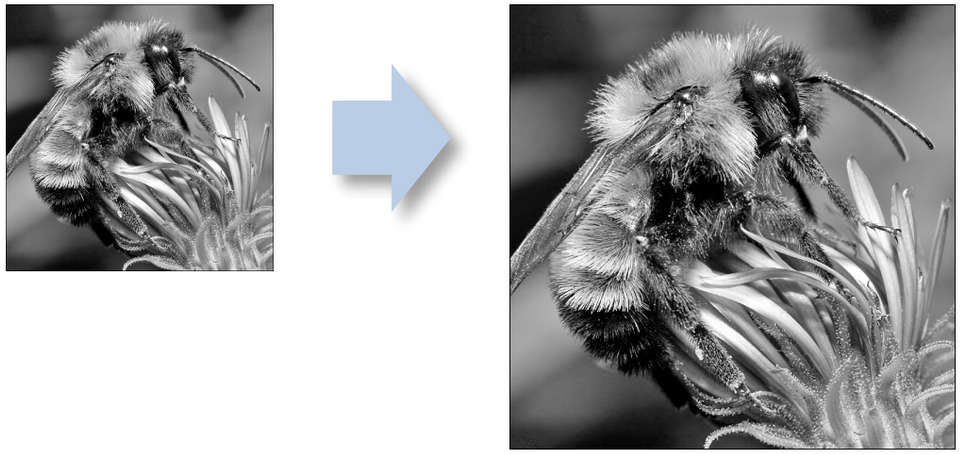
Is the Harris detector invariant to intensity shift?



$$I' = I + 20$$

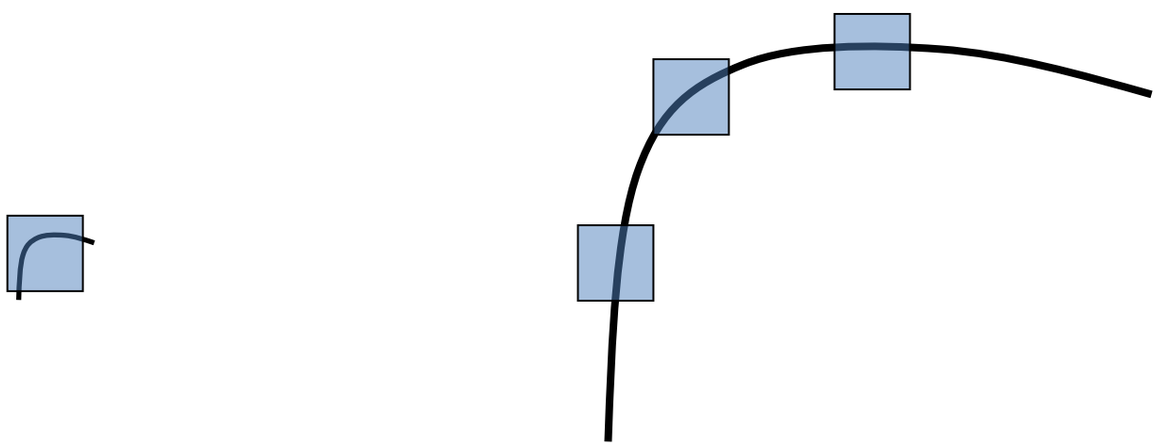
Harris detector: invariance

Is the Harris detector invariant to scaling?



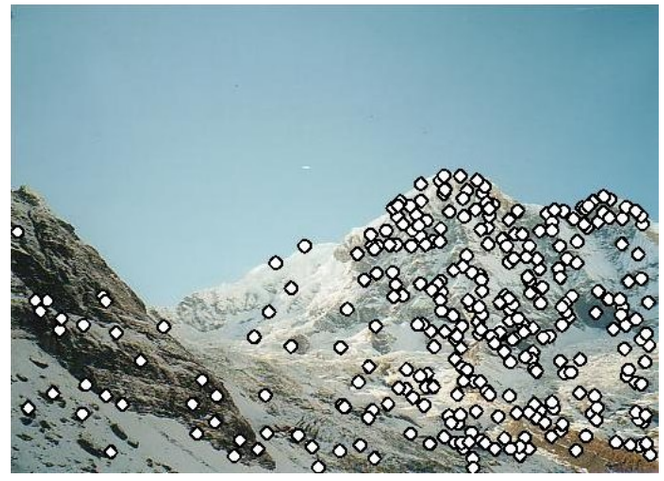
Harris detector: invariance

- Invariant to scaling?



Features - Overview

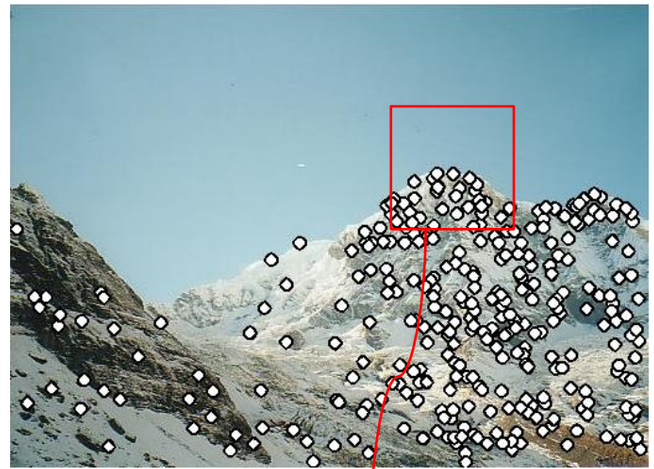
1. Detect



detect *unique* points

e.g., using the Harris corner detdctor

2. **Describe**

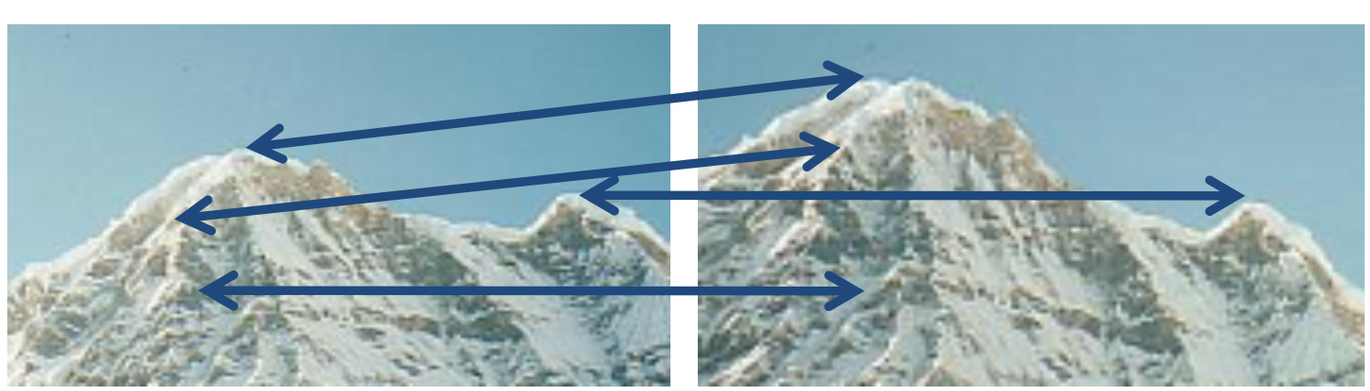


describe using *invariant* representation

e.g., using the MOPS descriptor

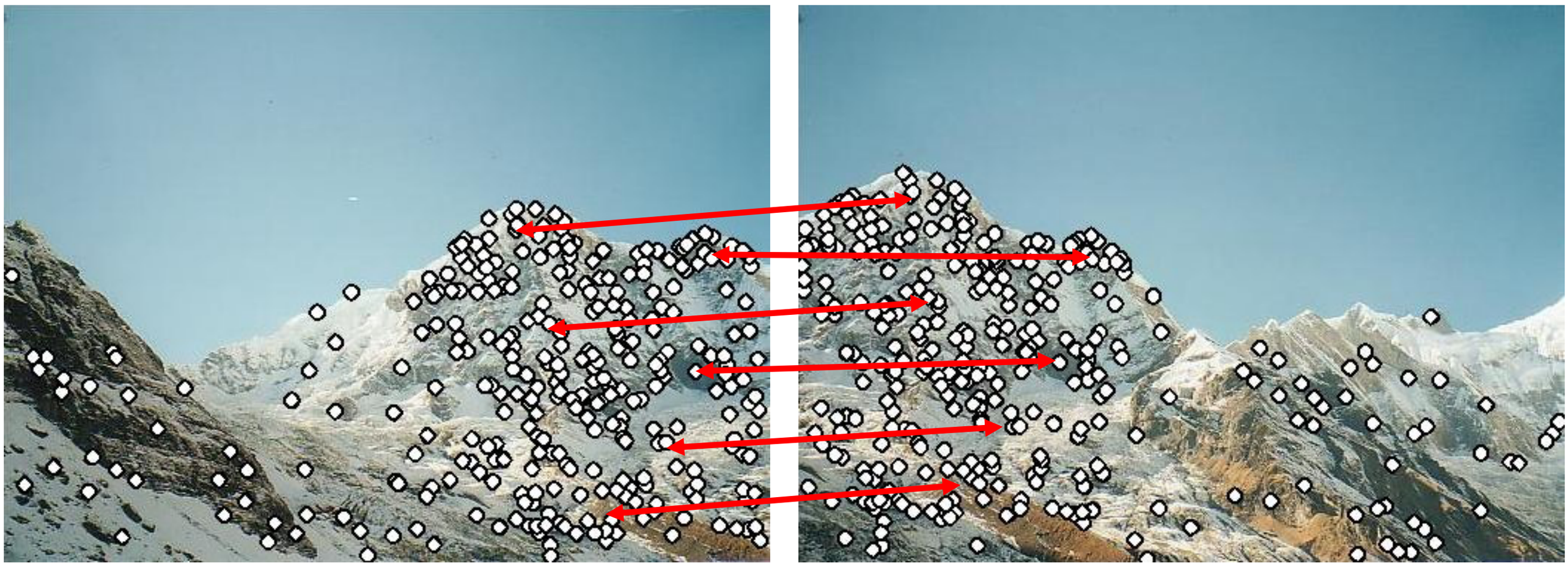
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3. Match



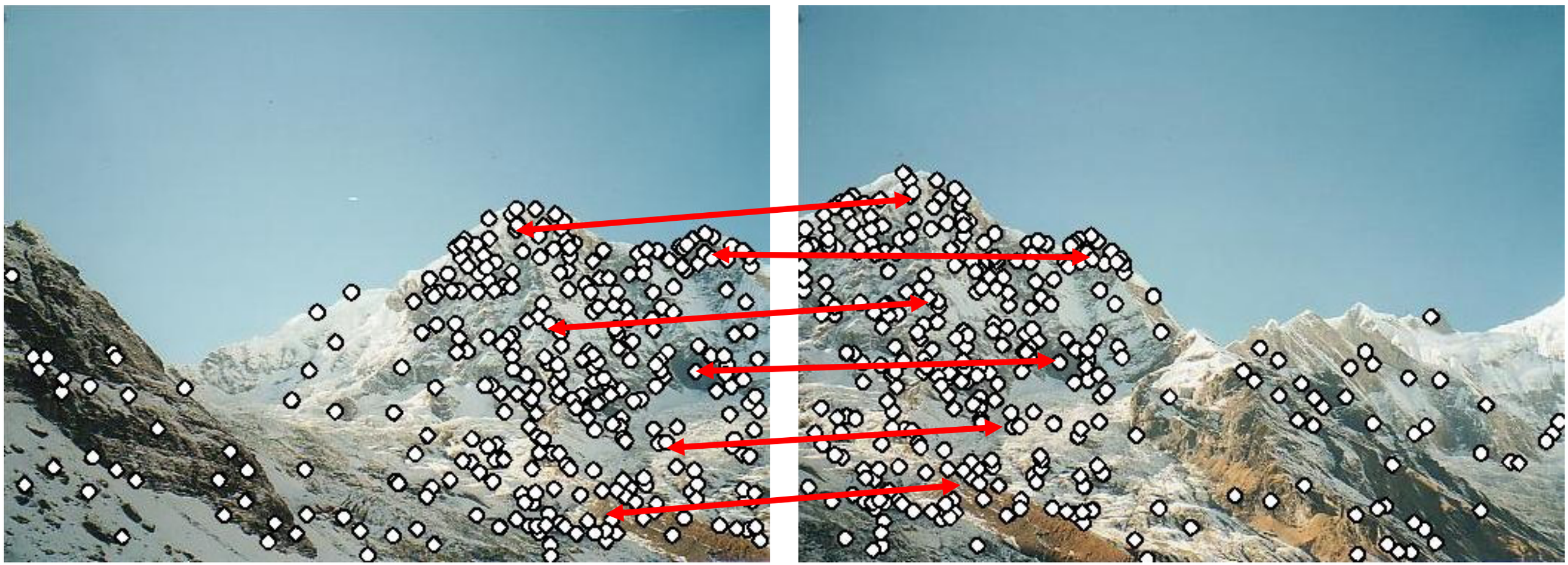
match them *robustly*

Feature Descriptors



The detector gives me the (x, y) coordinates of a unique-looking point. How should I compare it to other unique-looking points?

Feature Descriptors



Simple starting point: window of pixels around the point.

Feature Descriptors

Starting with a window of pixels, let's add invariances:

- Brightness (scale and/or shift - "*affine* invariance")

Feature Descriptors

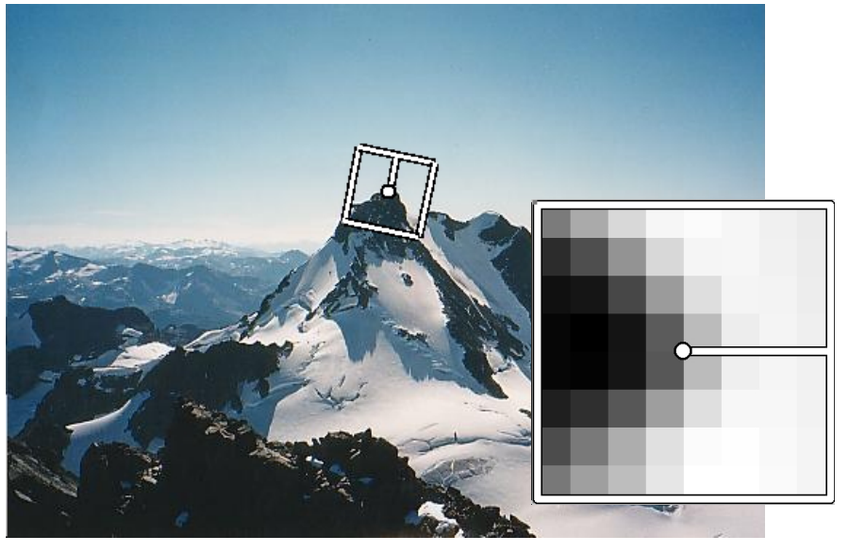
Starting with a window of pixels, let's add invariances:

- Brightness (scale and/or shift - "*affine* invariance")
- Rotation

Feature Descriptors

Starting with a window of pixels, let's add invariances:

- Brightness (scale and/or shift - "*affine* invariance")
- Rotation



Feature Descriptors

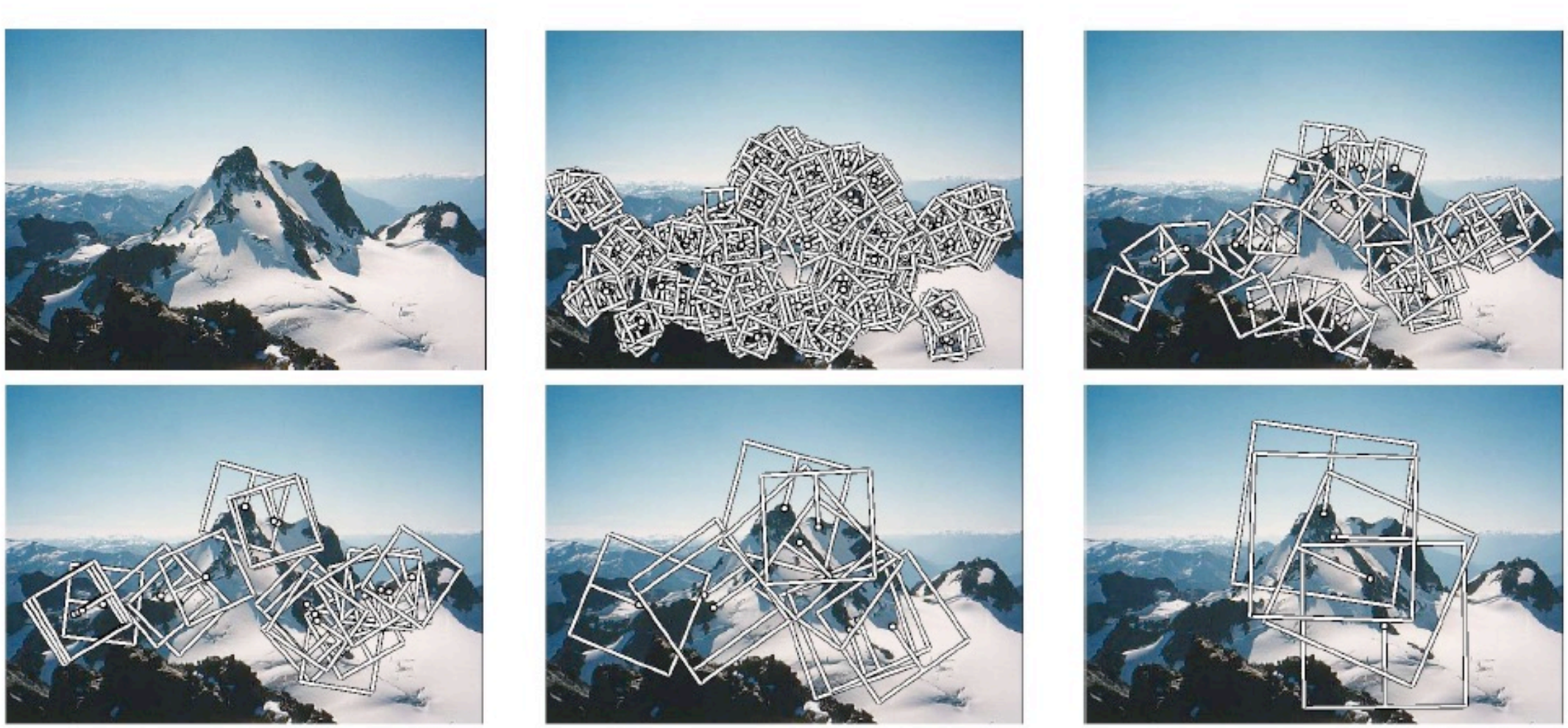
Starting with a window of pixels, let's add invariances:

- Scale

Feature Descriptors

Starting with a window of pixels, let's add invariances:

- Scale



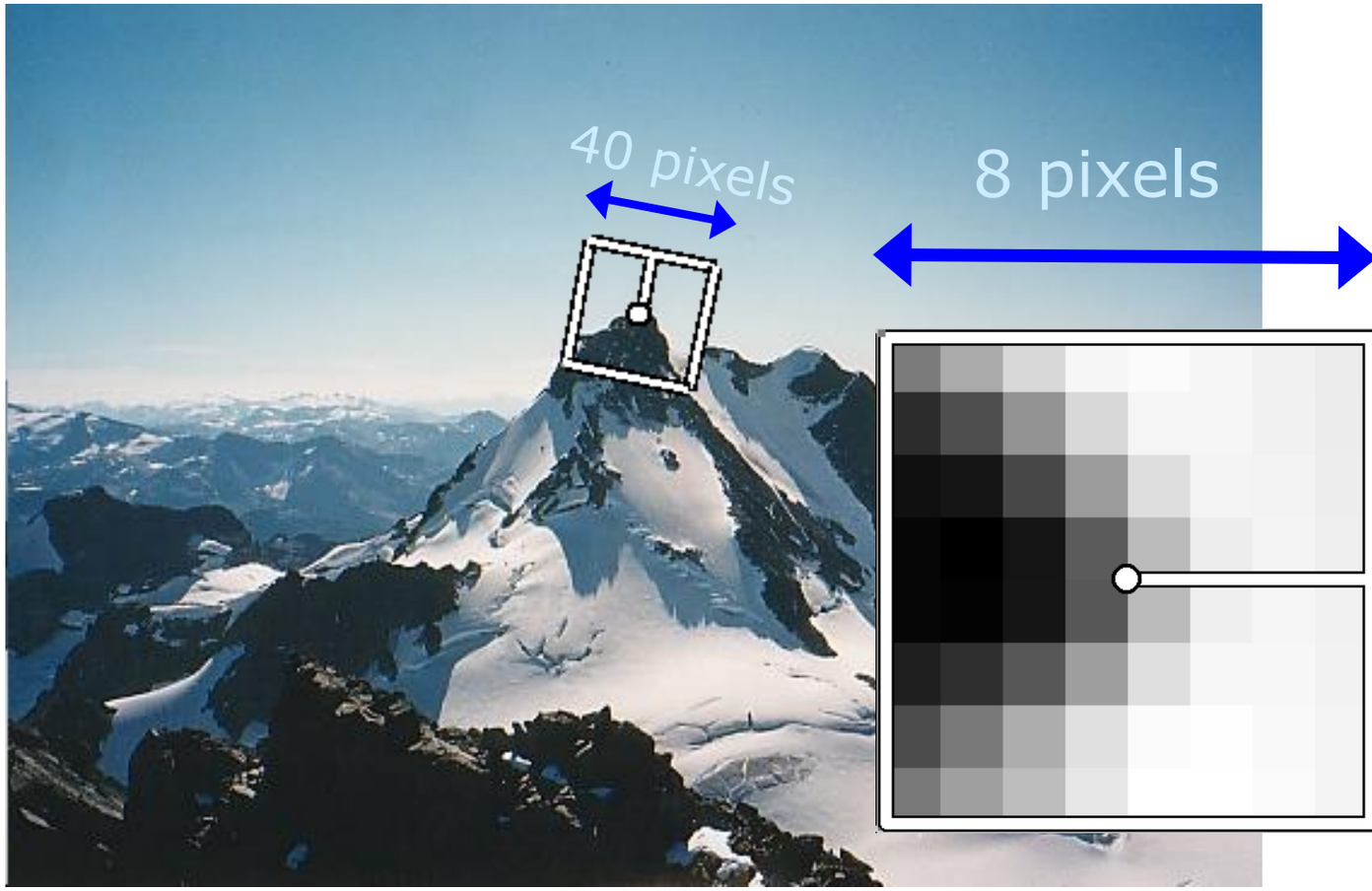
Feature Descriptors

Starting with a window of pixels, let's add invariances:

- Scale

Multiscale Oriented PatcheS: The MOPS Descriptor

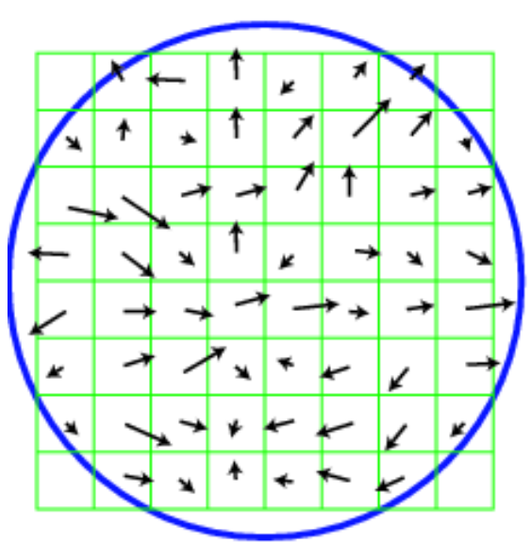
- Scale to 1/5 size
- Rotate to horizontal
- Normalize intensity:
  - subtract mean
  - divide by standard dev

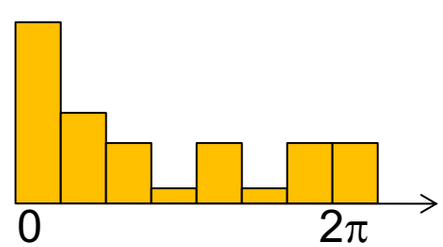


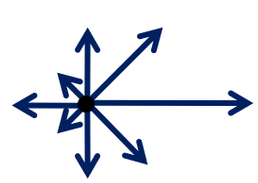
- Run it on a Gaussian pyramid

Fancy, industrial-strength feature descriptors: SIFT

- Take a 16x16 window
- Compute edge orientation at each pixel
- Discard weak edges
- Create a histogram of remaining edge orientations



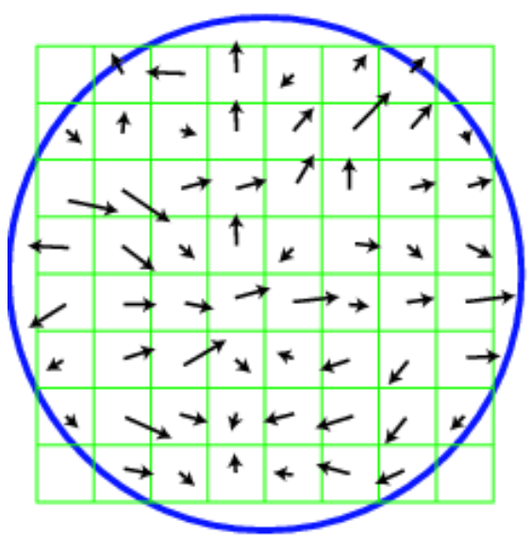


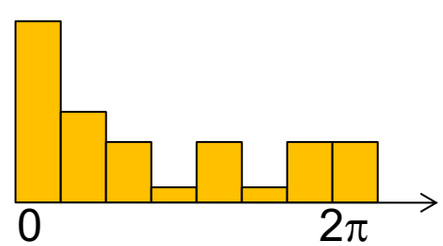


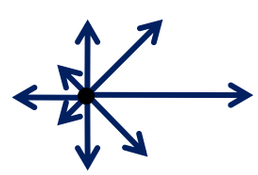
Scale Invariant Feature Transform: SIFT

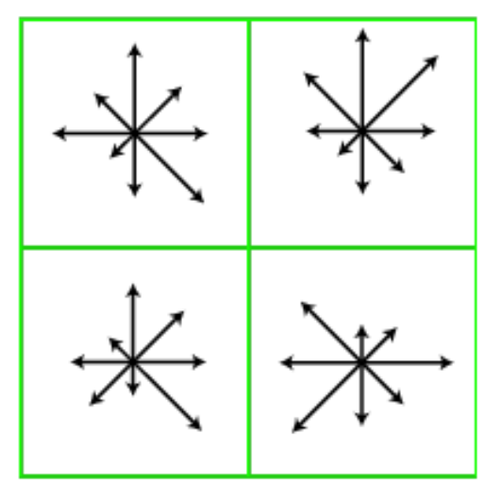
Real-deal, industrial-strength feature descriptors.

- Take a 16x16 window
- Compute edge orientation at each pixel
- Discard weak edges
- Create a histogram of remaining edge orientations
- Actually do this for each of 4 quadrants of the window





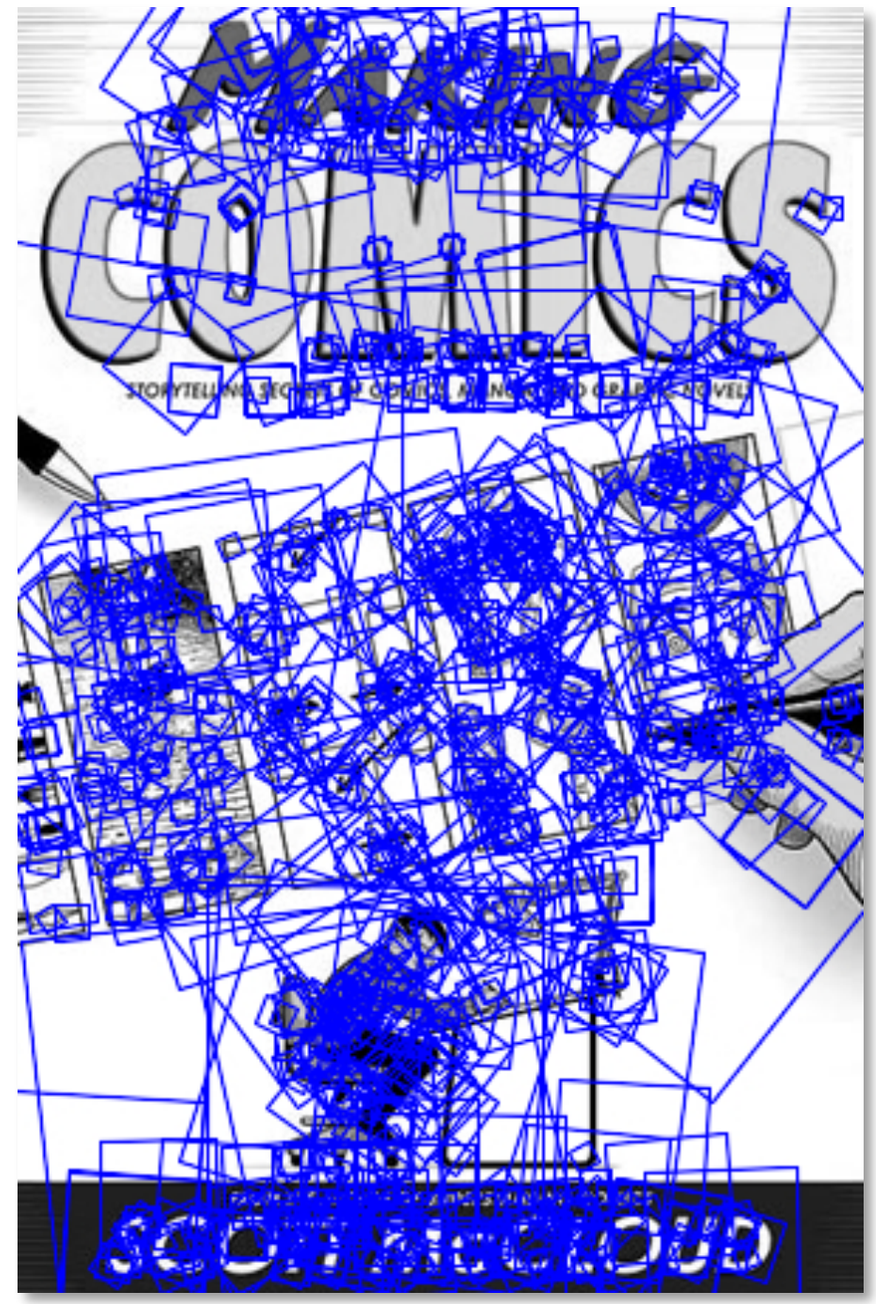




4 histograms - unroll into vector

SIFT: Example

sift



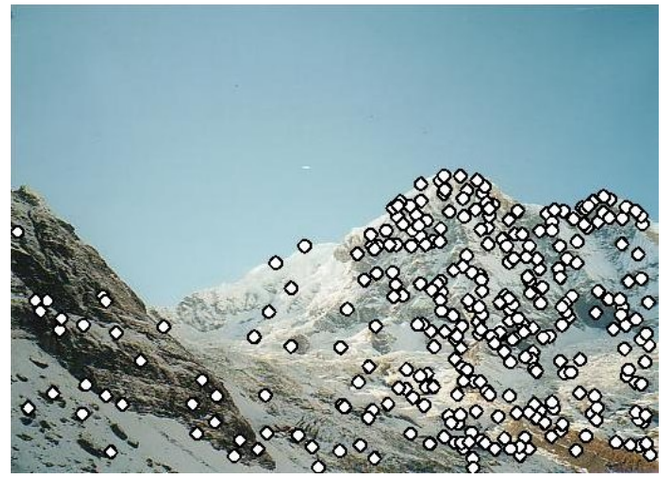
SIFT: Properties

Remarkably invariant (in practice, if not theory) to:

- Viewpoint, illumination, rotation, scale (via pyramid)

Features - Overview

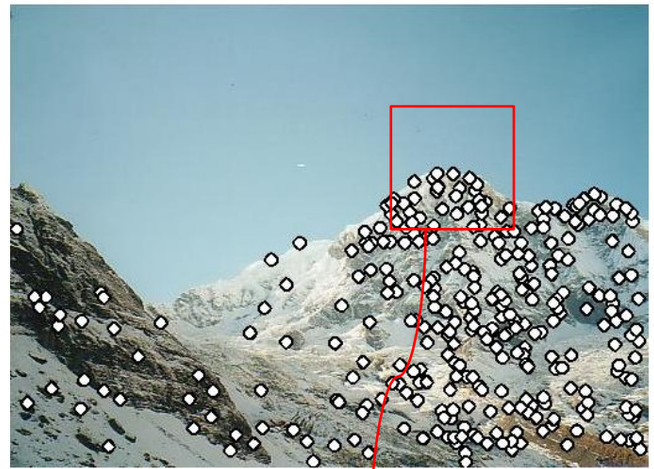
1. Detect



detect *unique* points

e.g., using the Harris corner detdctor

2. Describe

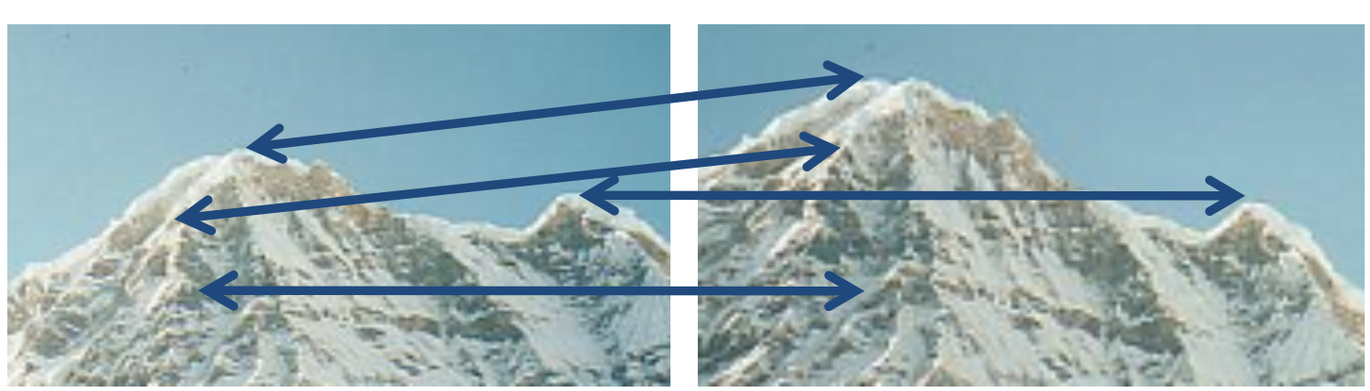


describe using *invariant* representation

e.g., using the MOPS descriptor

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

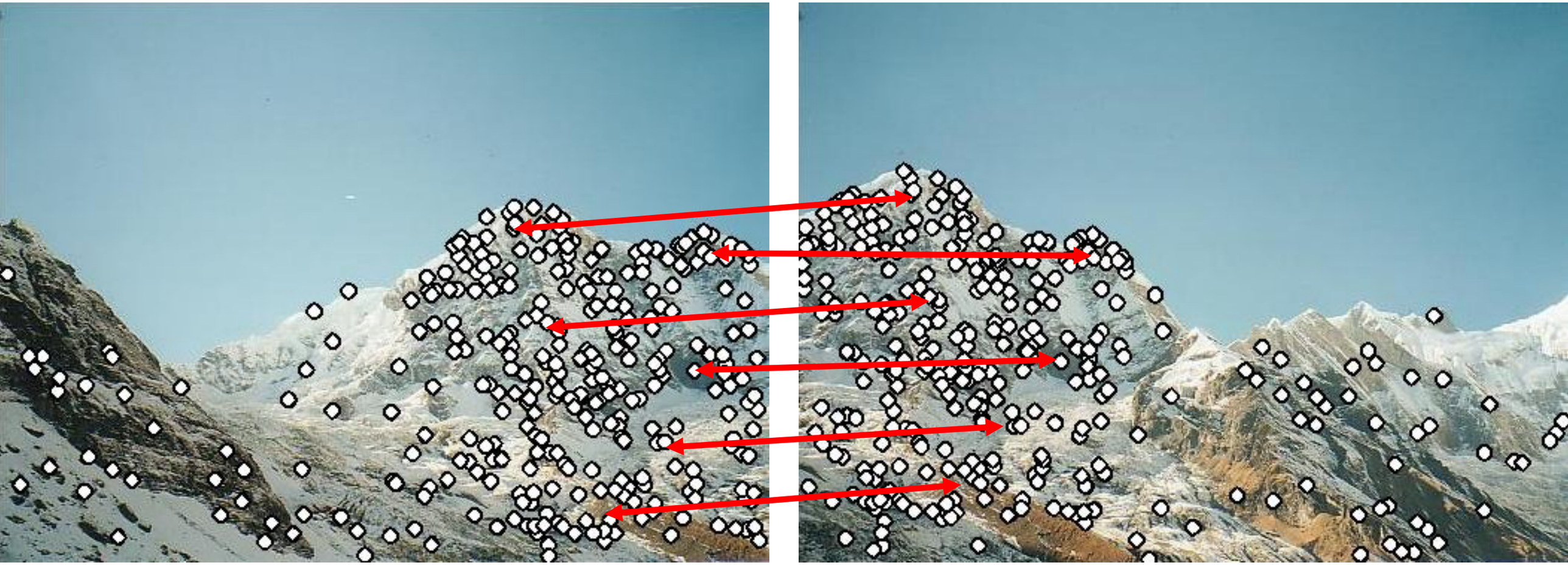
3. Match



match them *robustly*

e.g., using the ratio distance

3. Match Features



At this point, a "feature" has:

1. A keypoint: its position in one of the images
2. A descriptor: a vector of numbers that 'captures' its characteristics

Feature Matching

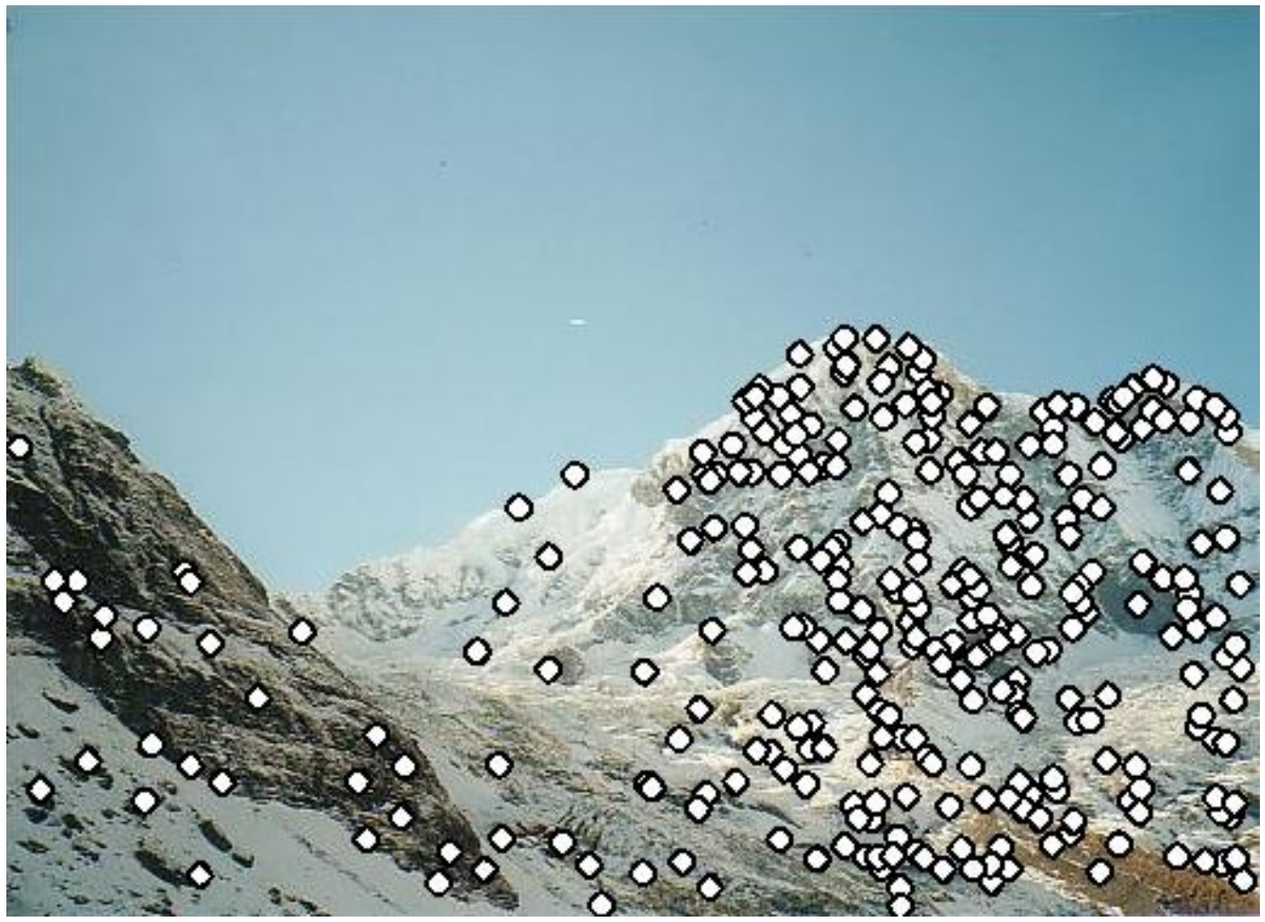


Image 1 features:

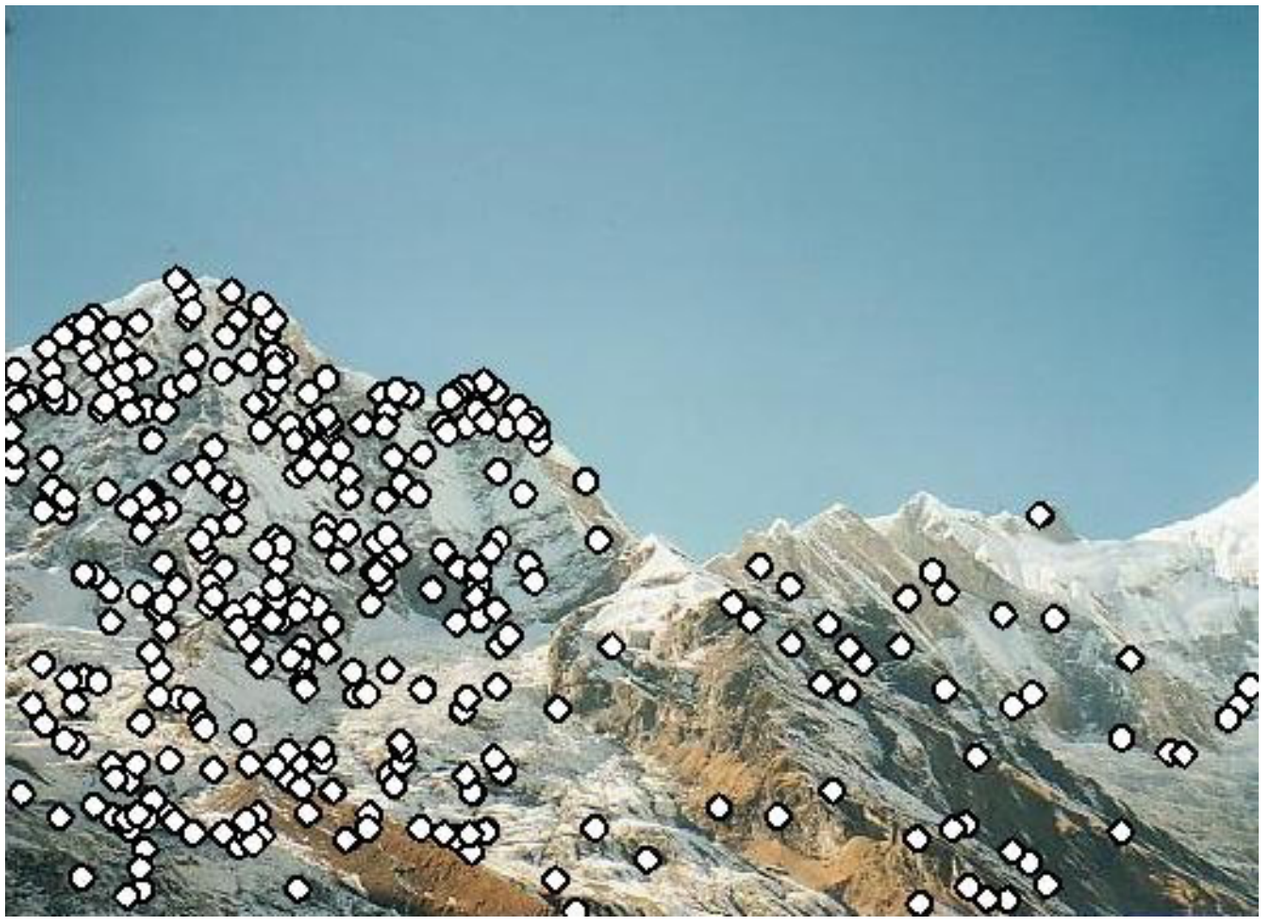


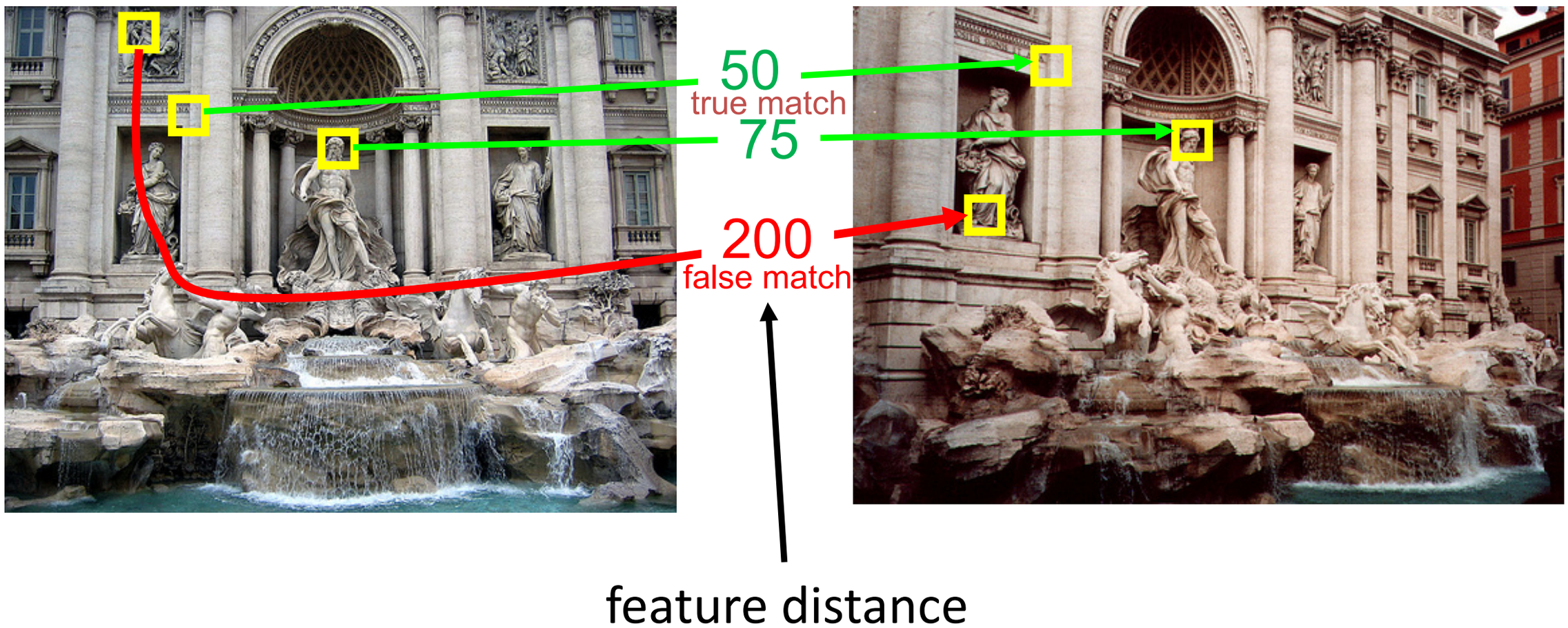
Image 2 features:

Matching Algorithm

Take 0

```
F1 = detect_describe(img1)

F2 = detect_describe(img2)

for f1 in F1:

    find f2 that minimizes d(f1, f2)

    add (f1, f2) to matches
```

But what if we're wrong?



- Answer #1: Threshold on match score

Matching Algorithm

Take 1

```
F1 = detect_describe(img1)

F2 = detect_describe(img2)

for f1 in F1:

    find f2 that minimizes d(f1, f2)

    if d(f1, f2) < T

        add (f1, f2) to matches
```

Distance Metrics

What should we use for **d** in **d**(f1, f2)?

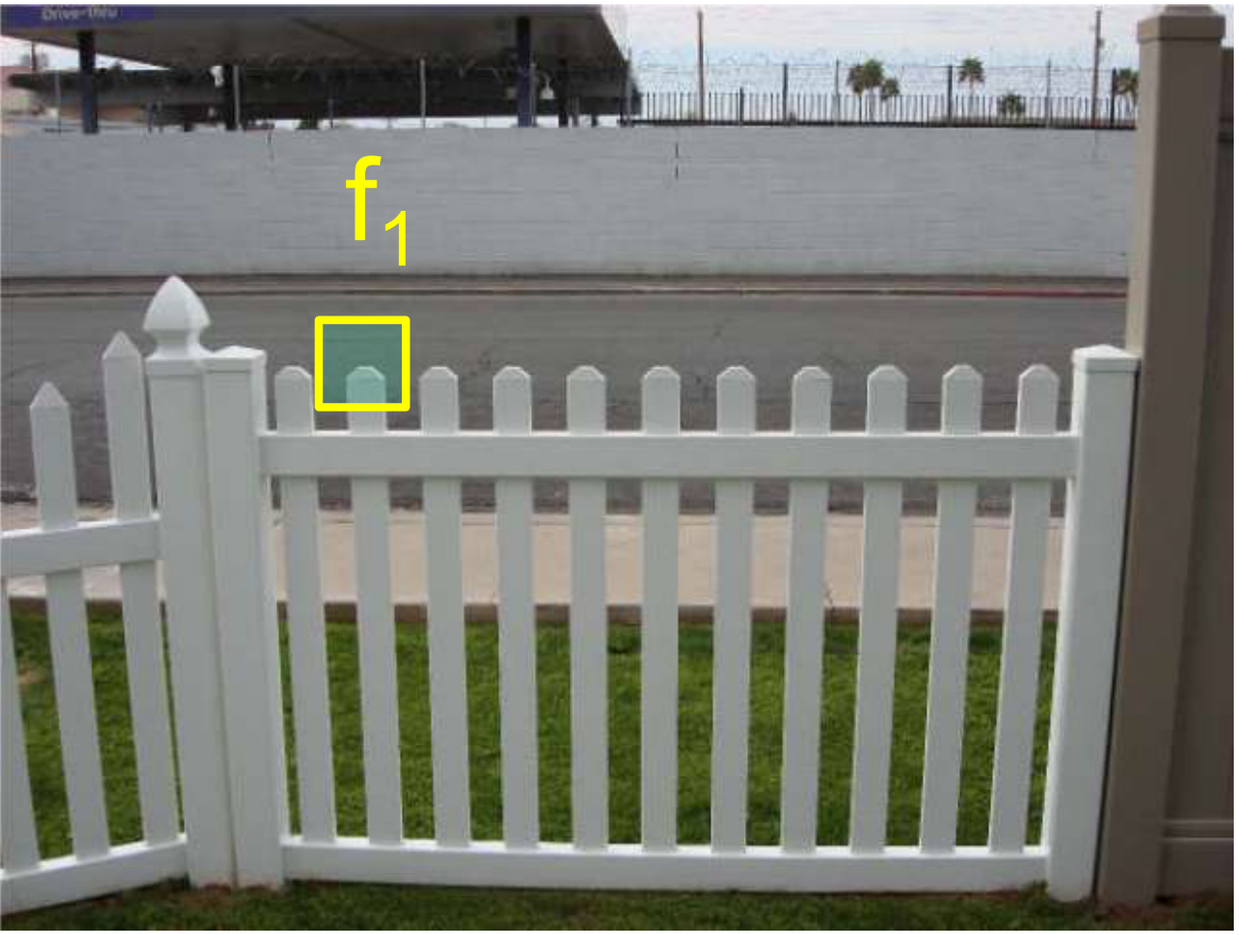
Efficiency Considerations

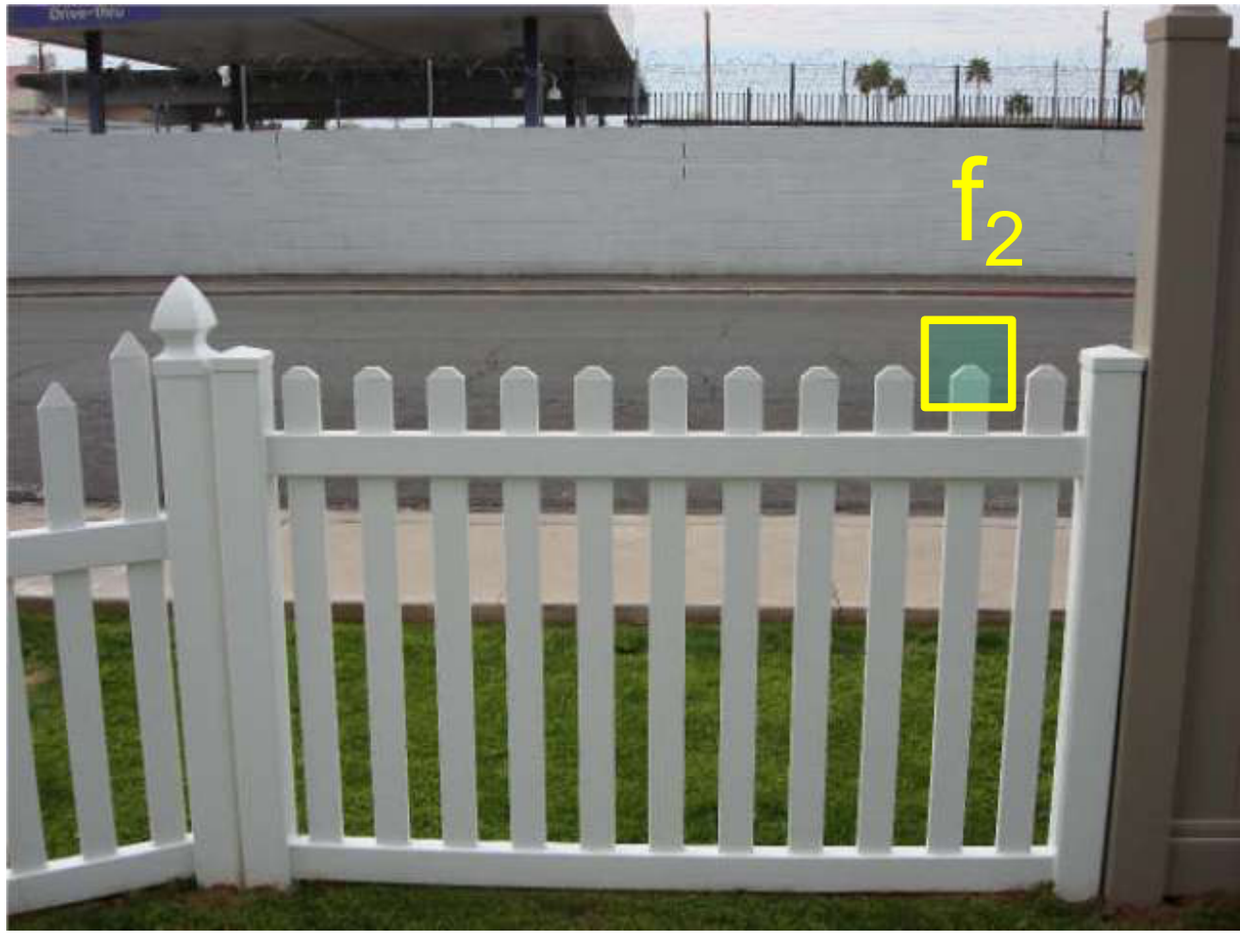
```
F1 = detect_describe(img1)

F2 = detect_describe(img2)

for f1 in F1:

    find f2 that minimizes d(f1, f2)

    if d(f1, f2) < T

        add (f1, f2) to matches
```

1. Basic: `scipy.spatial.distance.cdist` is nicely optimized
2. Fancy: spatial data structures to do $< mn$ comparisons

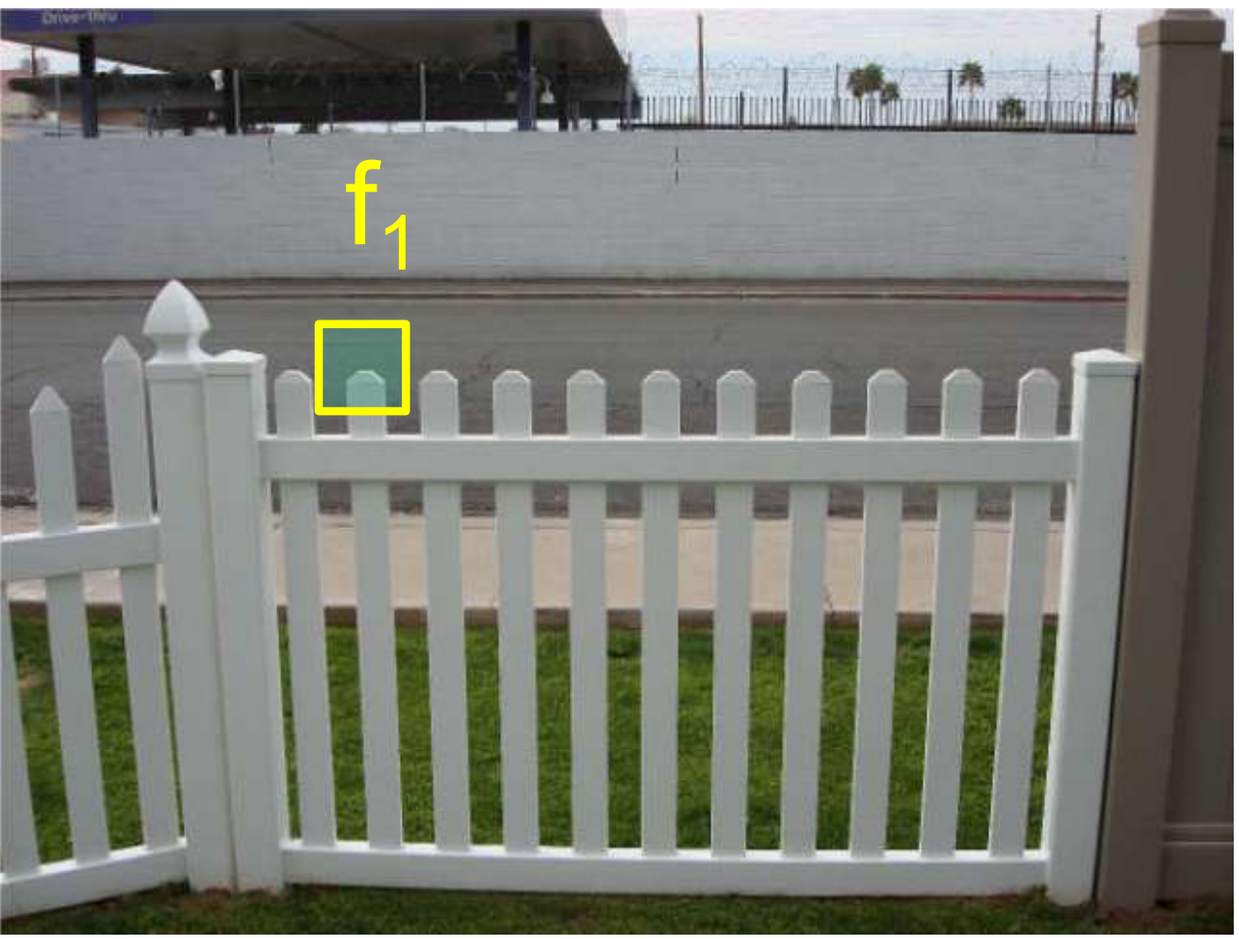
A problem with SSD

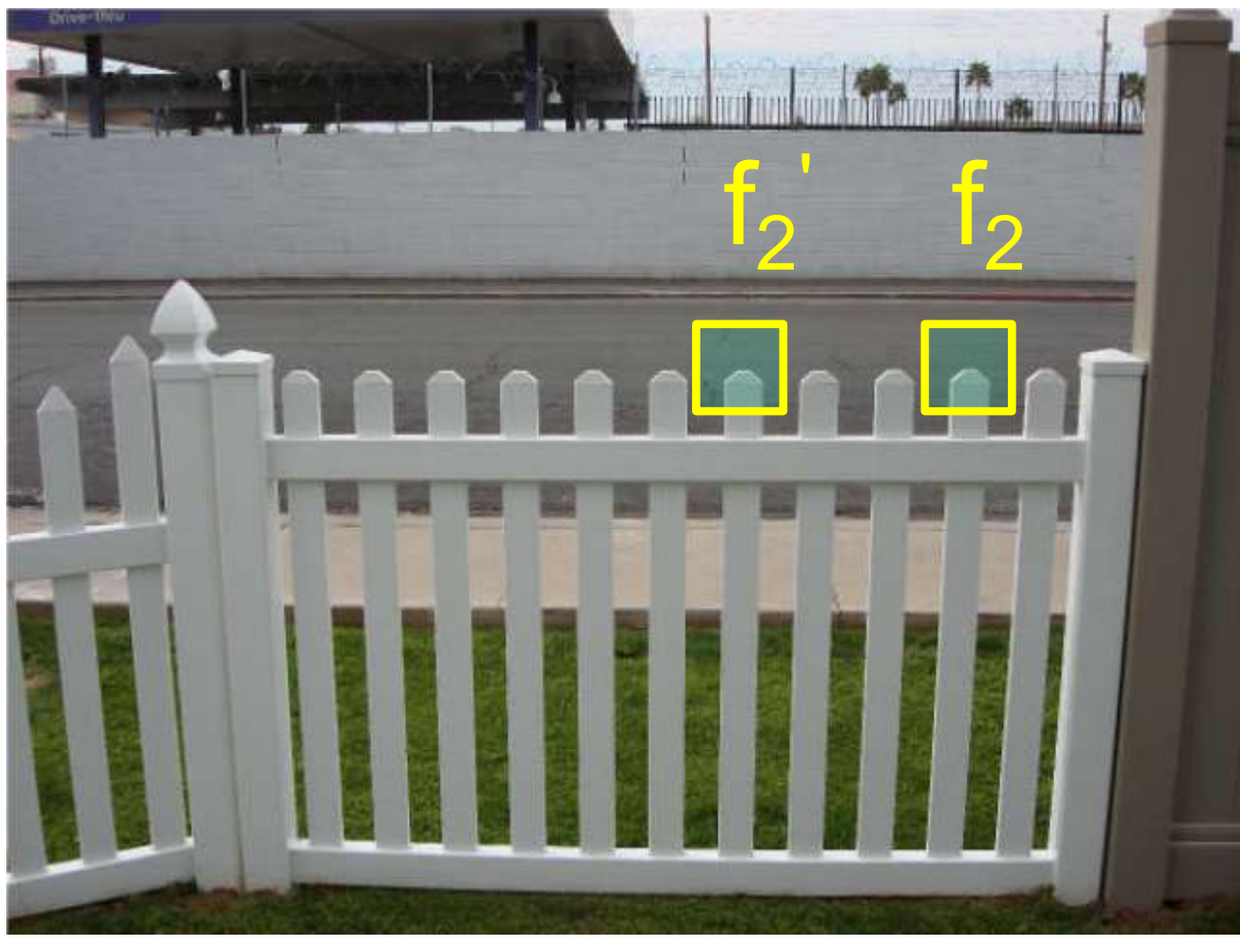
A problem with SSD



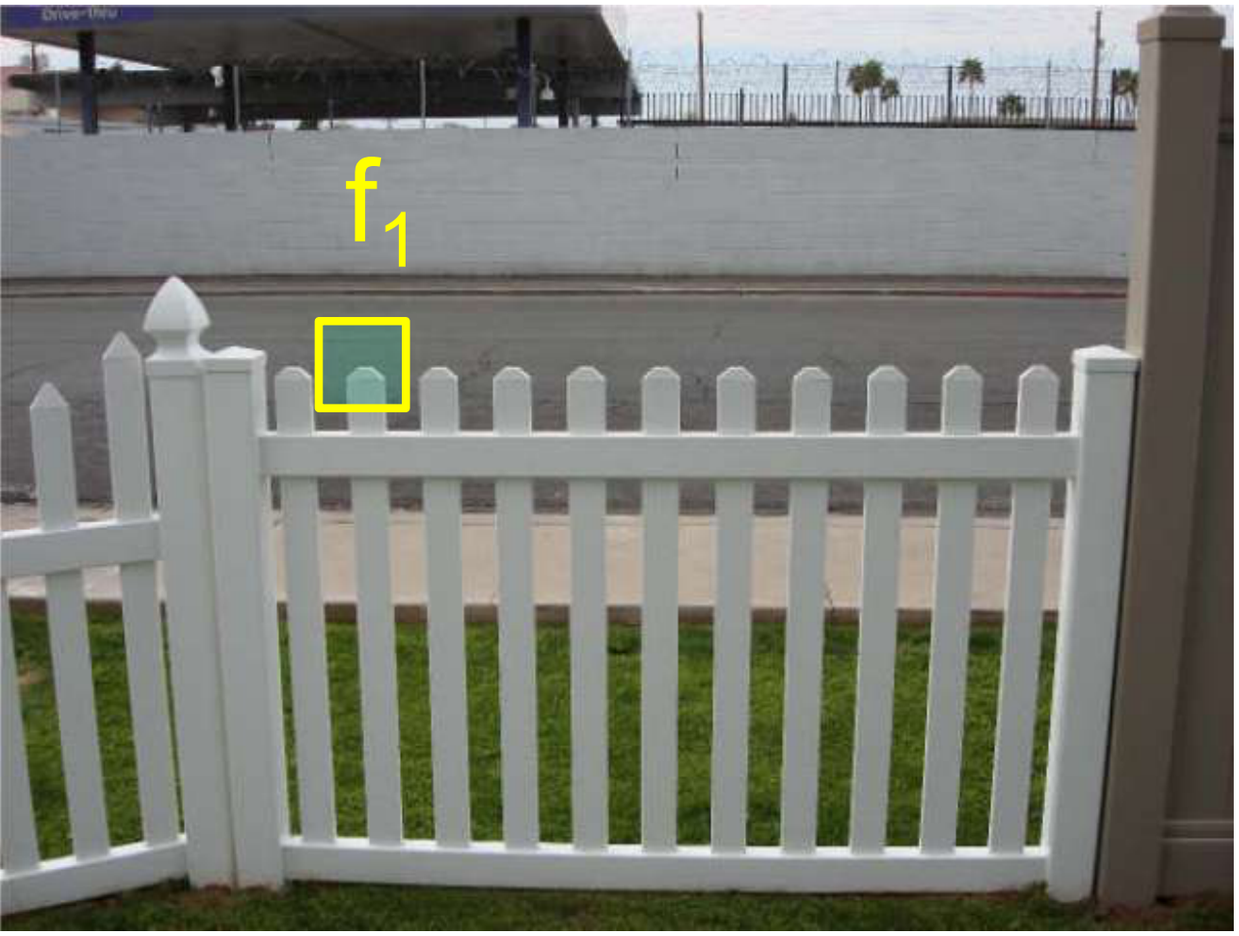


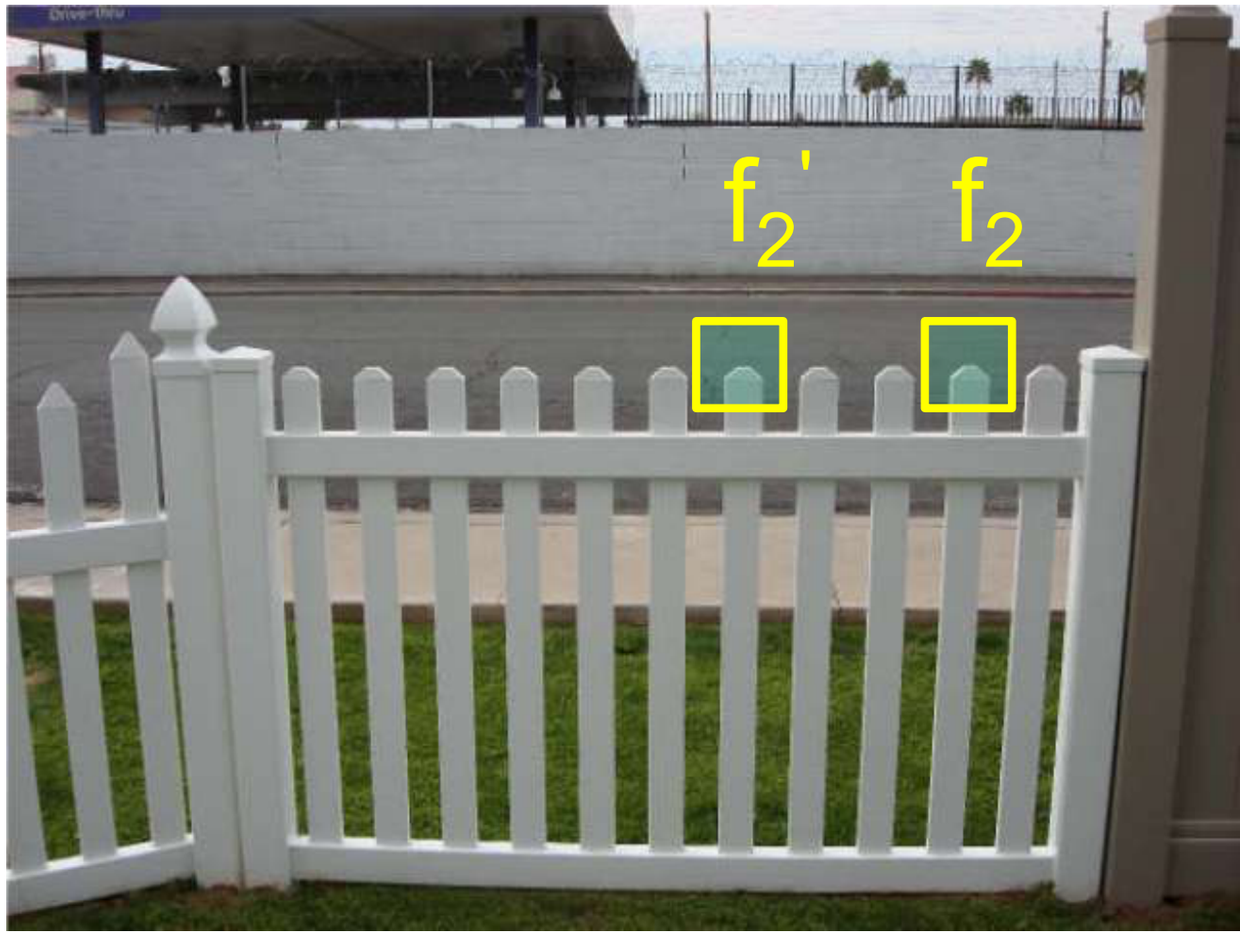
A problem with SSD





Ratio test - Intuition





We want a metric that gives small distance when features are

- like each other (according to SSD), but
- not like any others

Ratio Test, aka Ratio Distance

Notice: f_1's *closest match* is still f_2, but the *distance between them* is different.

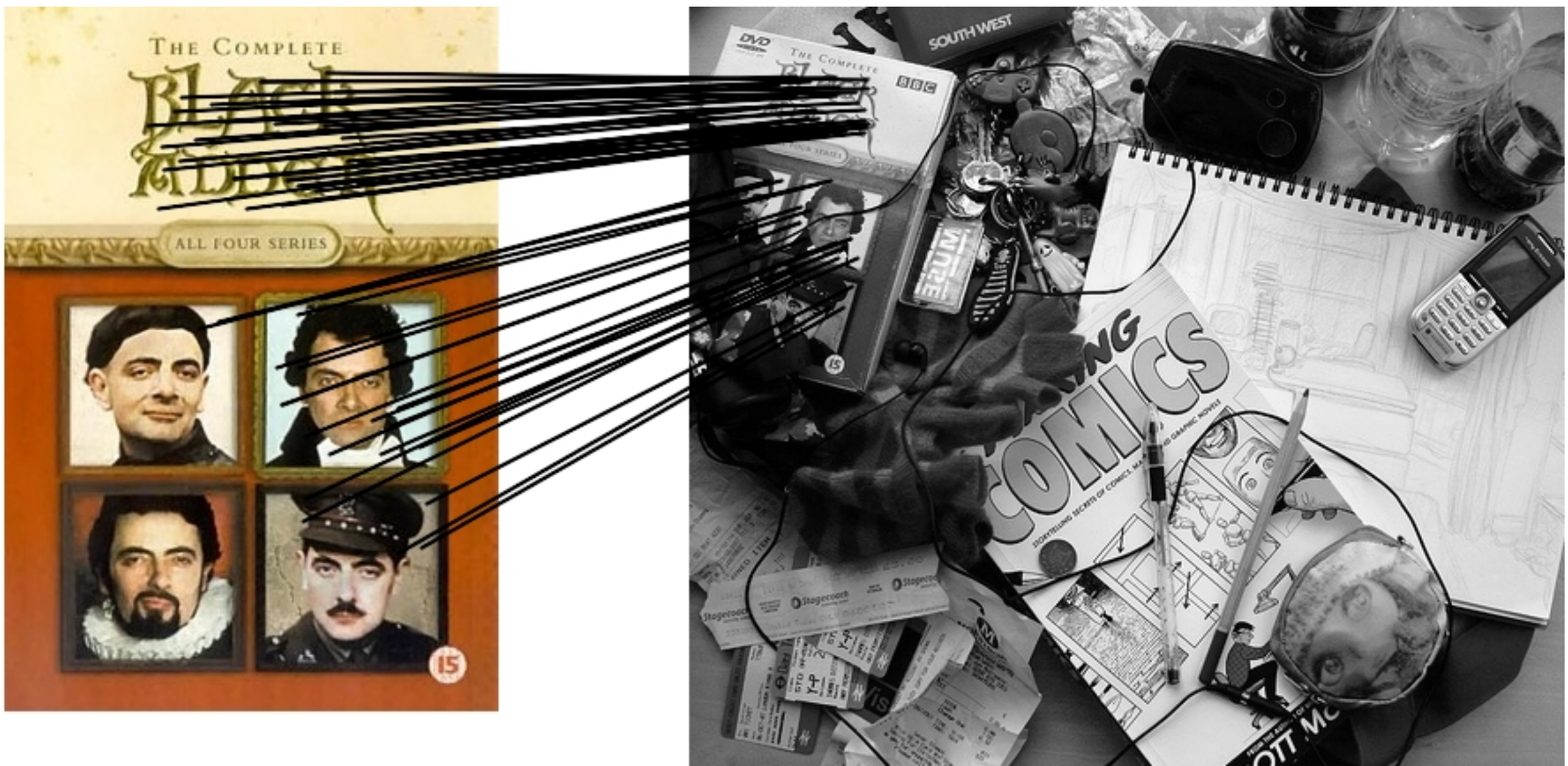
Matching Algorithm

Take 2

```
F1 = detect_describe(img1)

F2 = detect_describe(img2)

for f1 in F1:

    f2 = closest match according to SSD

    f2' = second-closest match according to SSD

    if SSD(f1,f2) / SSD(f1, f2') < T

        add (f1, f2) to matches
```

But what if we're wrong?

Decreasing the threshold T gives fewer **false positives** but also fewer **true positives**.



But what if we're wrong?

Many good matches, but still a few **outliers**.