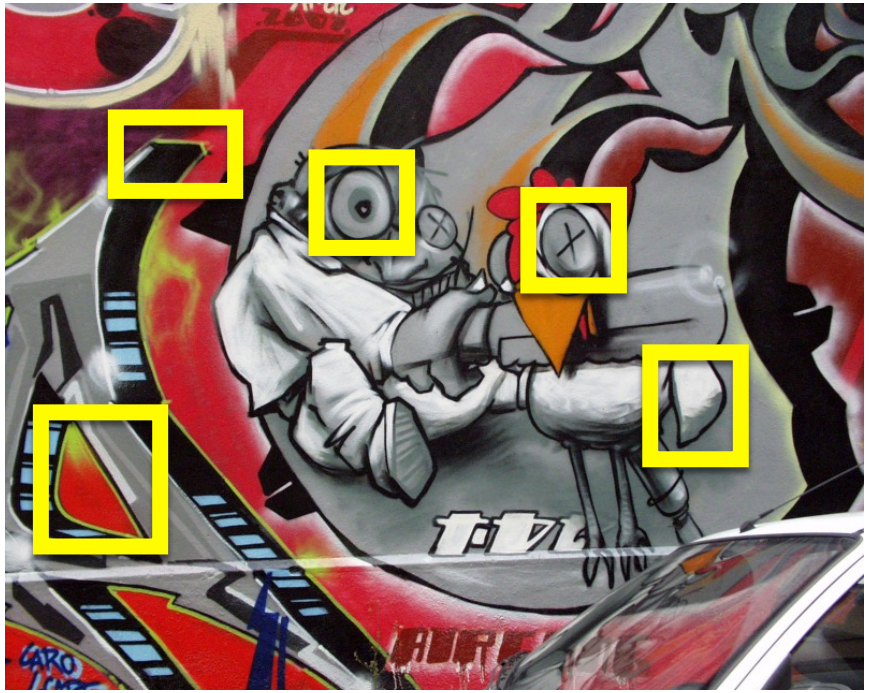
Feature Descriptors and Matching

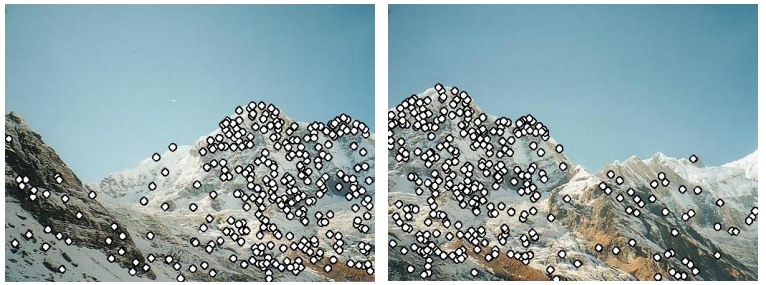


If we know where the good features are, how do we match them?

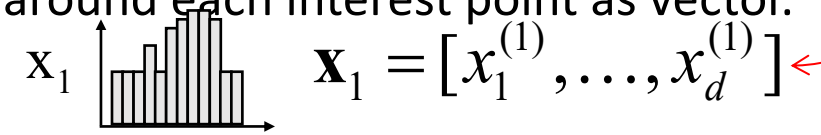
Designing feature descriptors

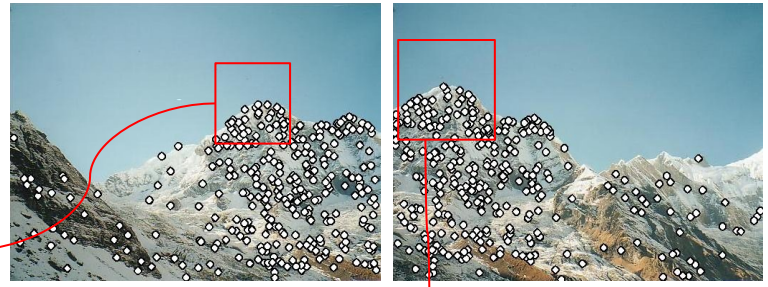
Local Feature Matching

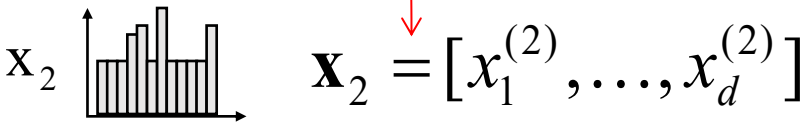
1) Detection:
Find a set of distinctive features (e.g., key points)



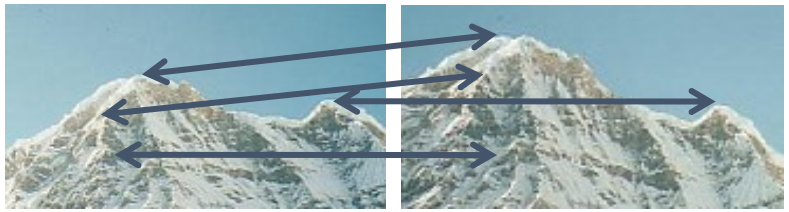
2) Description:
Extract feature descriptor around each interest point as vector.







3) Matching:
Compute distance between feature vectors to find correspondence.



What is the best descriptor for an image feature?

Photometric transformations

Objects will appear at different scales, translation and rotation

Geometric transformations

Global histogram to represent distribution of features

- How ‘well exposed’ a photo is

What about a local histogram per detected point?

Possible features:

- Templates
  - Intensity, gradients, etc.
- Histograms
  - Color, texture, SIFT descriptors, etc.

Image patch

Just use the pixel values of the patch

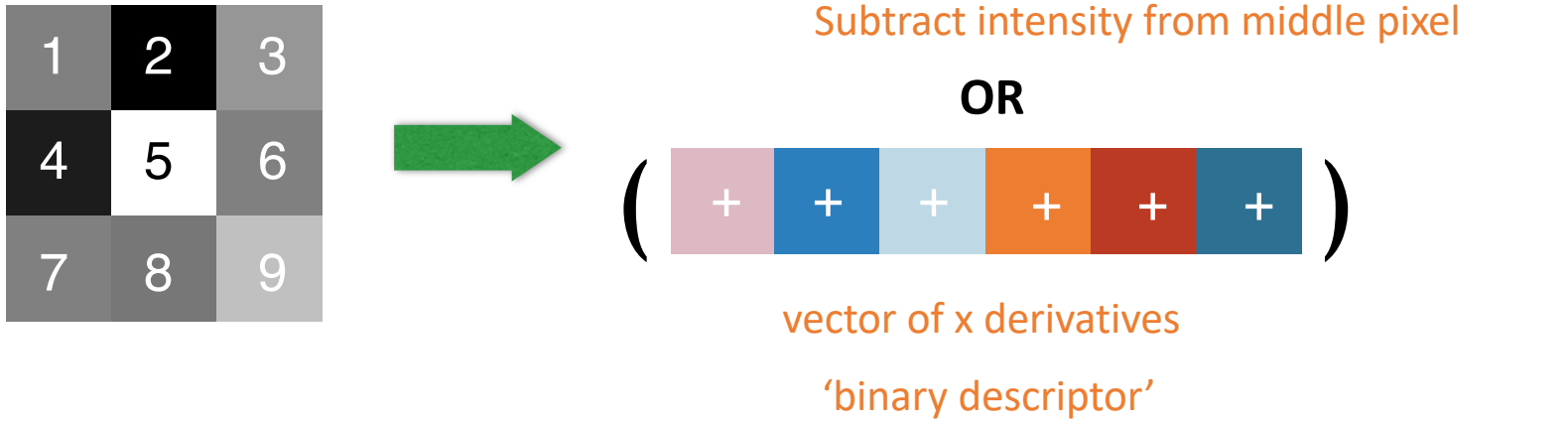
Perfectly fine if geometry and appearance is unchanged (a.k.a. template matching)

What are the problems?

How can you be less sensitive to absolute intensity values?

Image gradients

Use pixel differences



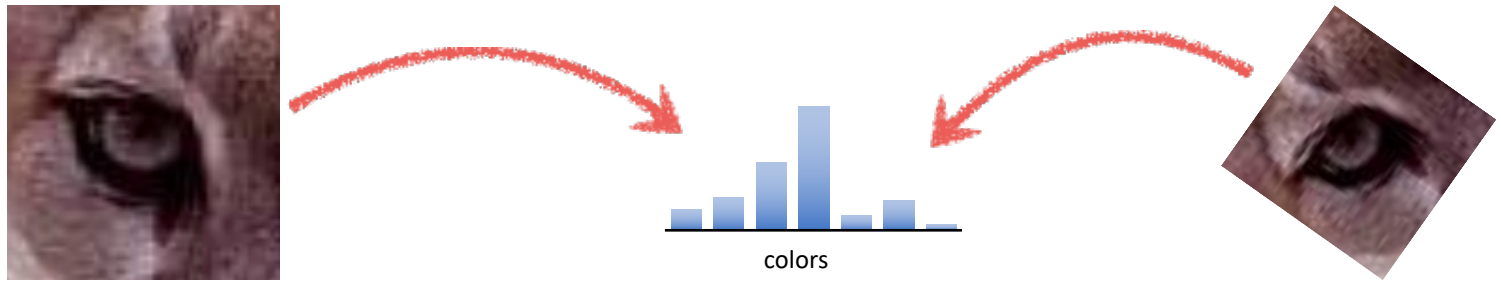
Feature is invariant to absolute intensity values

What are the problems?

How can you be less sensitive to deformations?

Color histogram

Count the colors in the image using a histogram

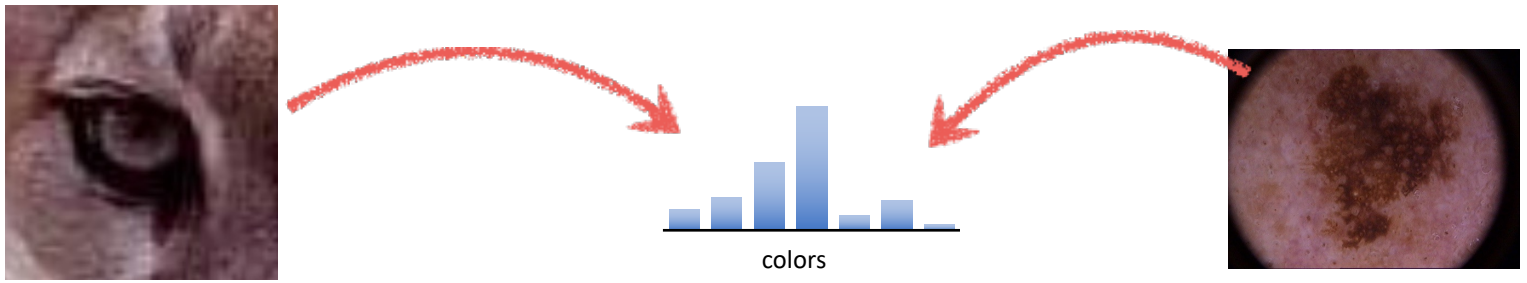


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram

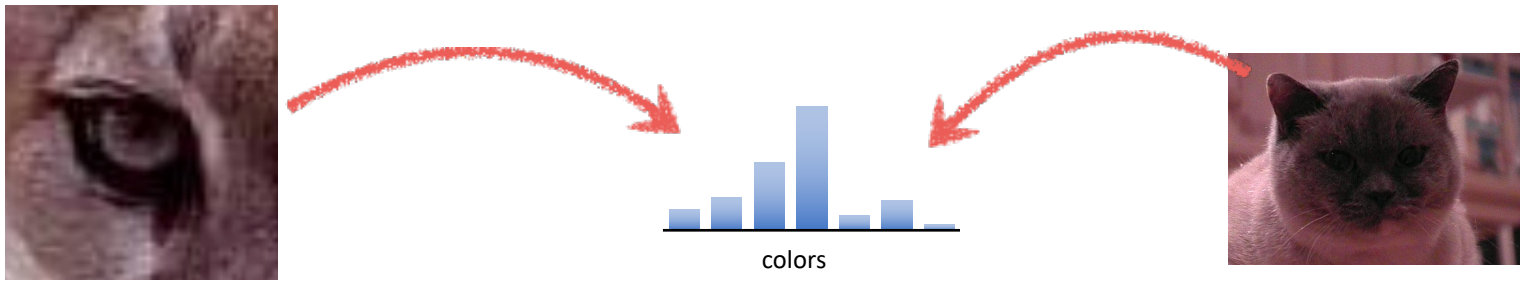


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram



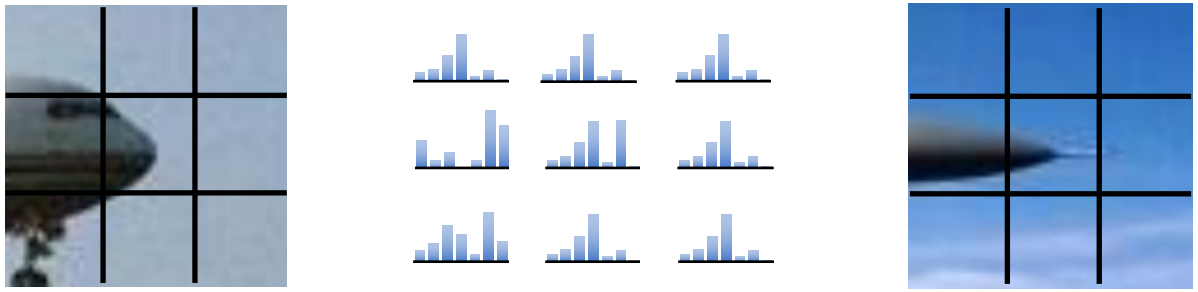
Invariant to changes in scale and rotation

What are the problems?

How can you be more sensitive to spatial layout?

Spatial histograms

Compute histograms over spatial ‘cells’



Retains rough spatial layout

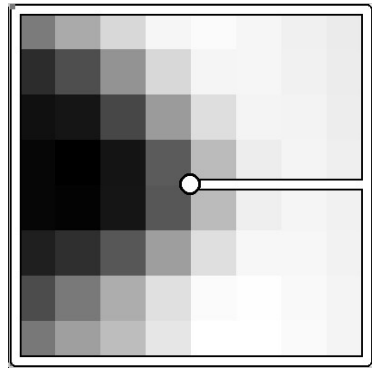
Some invariance to deformations

What are the problems?

How can you be completely invariant to rotation?

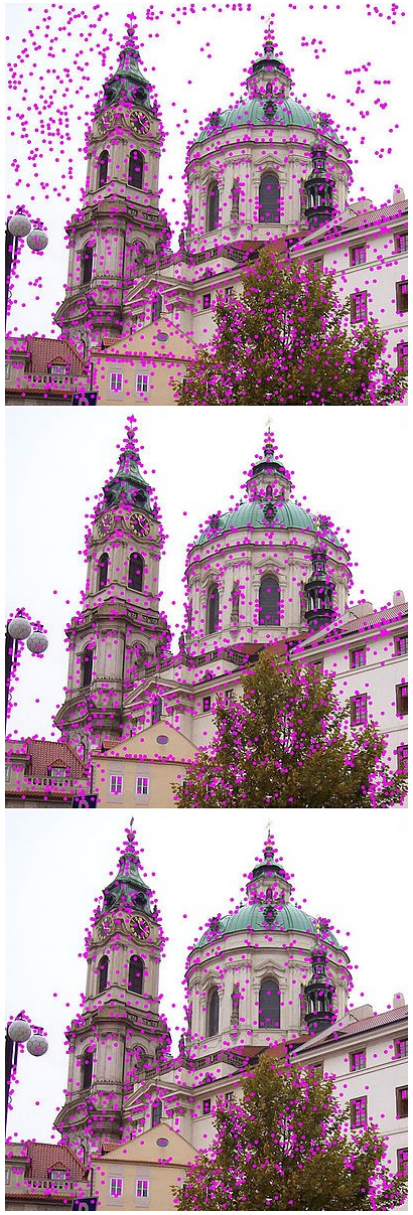
Orientation normalization

Use the dominant image gradient direction to normalize the orientation of the patch



Save the orientation angle θ along with (x, y, s)

What are the problems?



SIFT
(Scale Invariant Feature Transform)

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

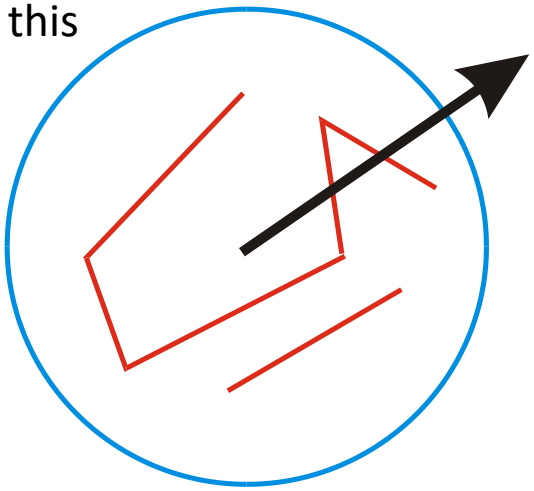
1. Multi-scale extrema detection
   - Find Harris corners scale-space extrema as feature point locations
2. Keypoint localization
   - Corner post-processing
3. Orientation assignment
4. Keypoint descriptor

At a high level, we want to compute a histogram around each key point using local information.

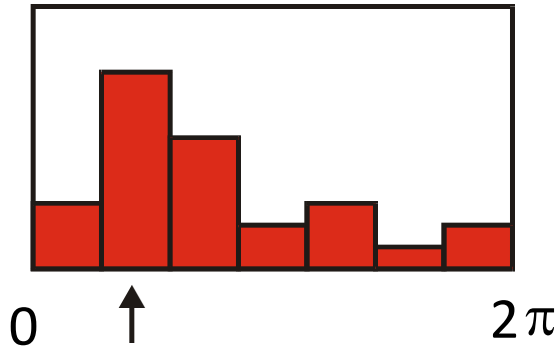
Orientation Estimation

- Compute gradient orientation histogram
- Select dominant orientation θ

For average gradient direction (black vector), sum all the gradients in the local window. It is known as the 'dominant orientation' of this neighborhood.

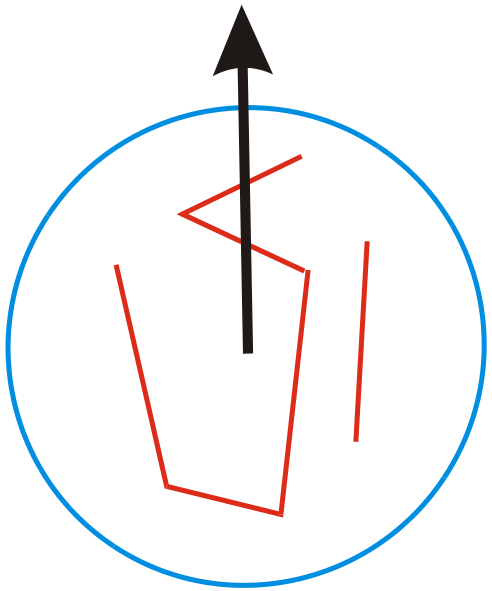


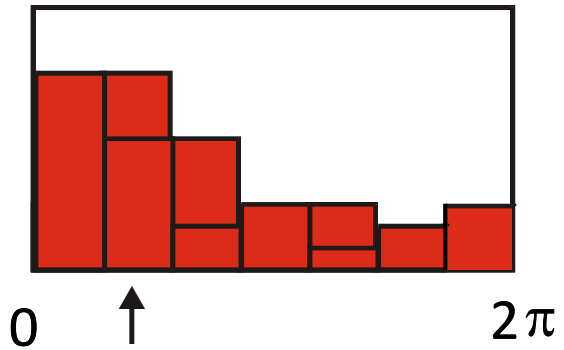
For histogram, we bin the possible range of gradient orientations (0 to 2pi radians) into 8 discrete number of bins and count the number of gradients that have those corresponding orientations.



Orientation Normalization

- Compute orientation histogram
- Select dominant orientation θ
- Normalize: rotate to fixed orientation. It makes the key point features invariant to rotation





Orientation assignment

For a keypoint, **L** is the **Gaussian-smoothed** image with the closest scale,

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

x-derivative y-derivative

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Detection process returns

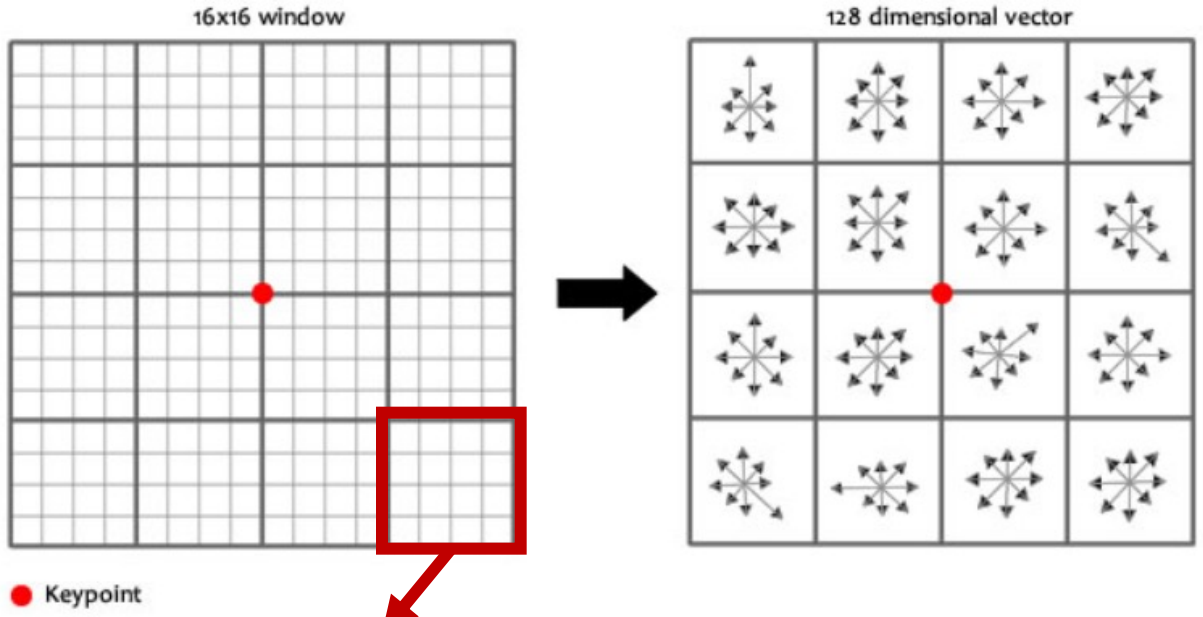
$$\{x, y, \sigma, \theta\}$$

location scale orientation

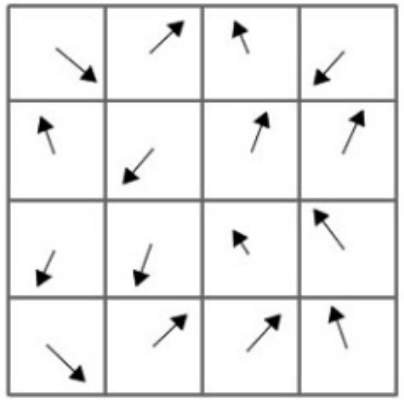
SIFT Descriptor Extraction

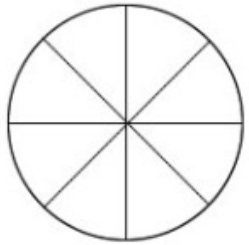
- Given a keypoint with scale and orientation

After rotation, scale the window according to discovered scale.

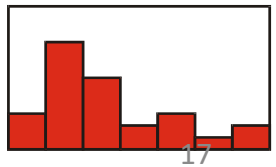


Gradient magnitude and orientation





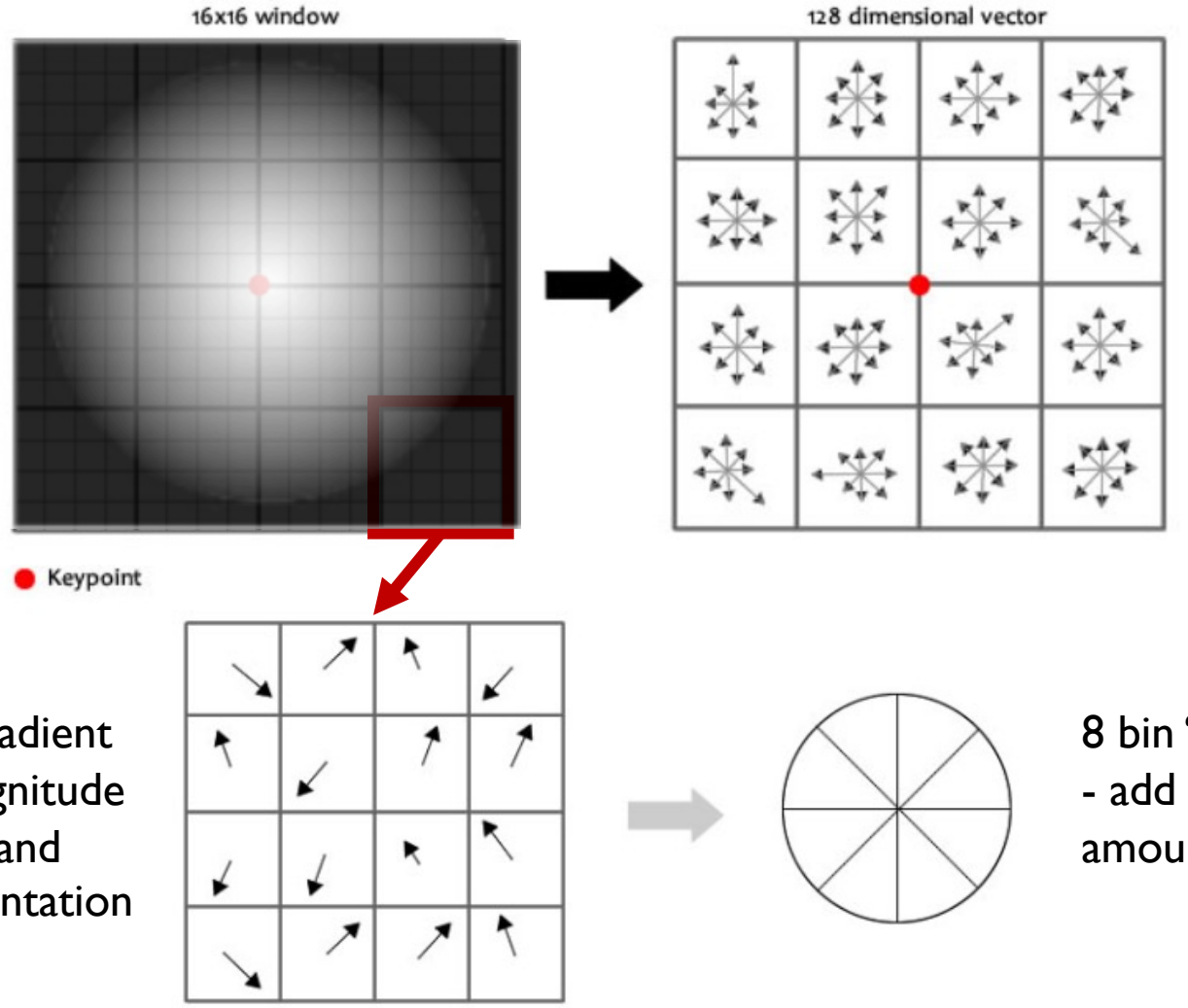
8 bin 'histogram' - add magnitude amounts!



SIFT Descriptor Extraction

Weight 16x16 grid by Gaussian to add location robustness and reduce effect of outer regions

σ = half window width



Closer points will matter more in the final descriptor when doing this.

Gradient magnitude and orientation

8 bin 'histogram' - add magnitude amounts!

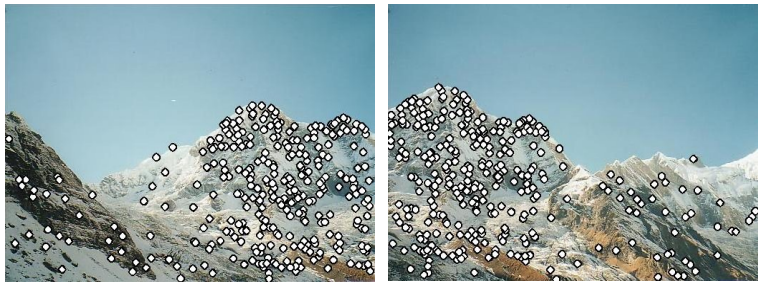
SIFT Descriptor Extraction

- Concatenate 4 x 4 8-bin histograms into 128-dim vector
- Illumination invariance:
  - Working in gradient space, so robust to $I = I + b$
  - Normalize vector to [0…1]
    - Robust to $I = \alpha I$ brightness changes
  - Clip all values in the vector > 0.2 to 0.2.
    - Robust to “non-linear illumination effects”
    - Image value saturation / specular highlights
  - Renormalize

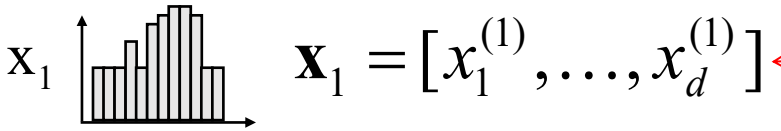
- Non-linear illumination changes can also occur due to camera saturation or illumination changes. It results in large change in relative magnitudes for some gradients, but are less likely to affect the gradient orientations.
- Thus, we clip the values before normalization to unit length. This means that matching the magnitudes for large gradients is no longer as important, and that the distribution of orientations has greater emphasis.
- The value of 0.2 was determined experimentally using images containing differing illuminations for the several 3D objects.

Local Feature Matching

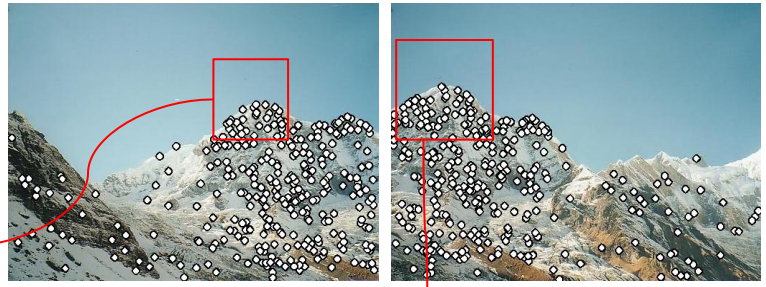
1) Detection:
Find a set of distinctive features (e.g., key points)

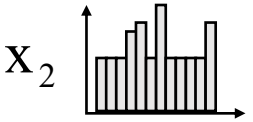


2) Description:
Extract feature descriptor around each interest point as vector.



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

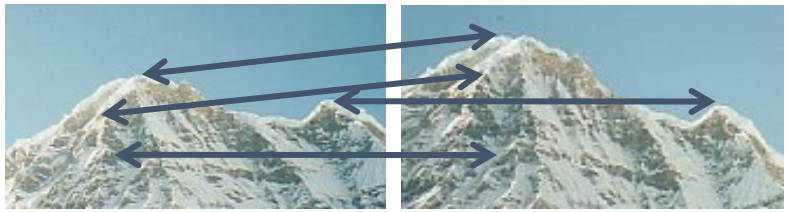




$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) Matching:
Compute distance between feature vectors to find correspondence.

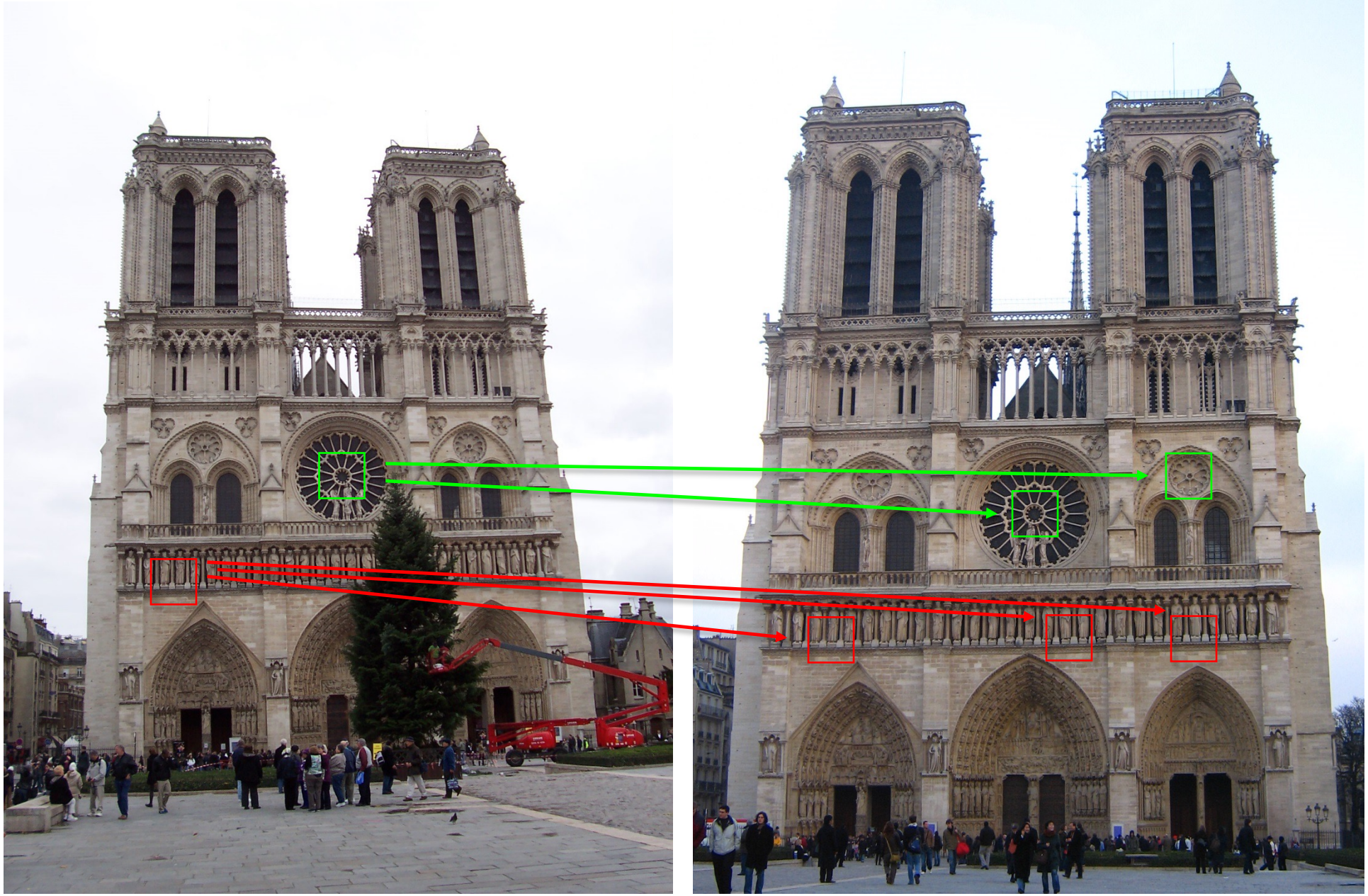
$$d(\mathbf{x}_1, \mathbf{x}_2) < T$$



Find SIFT descriptors of both images.

We have a match between key points when their feature vectors are similar enough.

Some points correspond better than others, so we will also discuss a way to rank correspondences



Distance: 0.34, 0.30, 0.40

Distance: 0.61, 1.22

Euclidean Distance vs. Cosine Similarity

- Euclidean $\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$

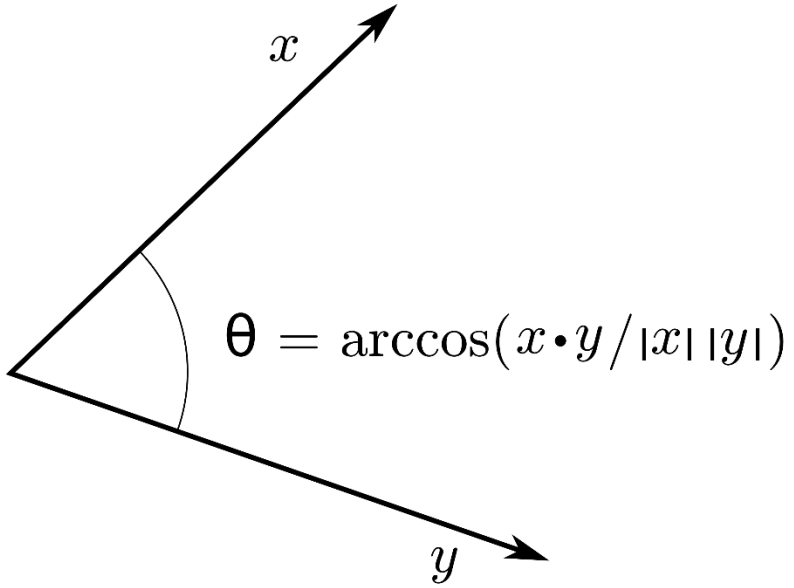
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

$$\|\mathbf{q} - \mathbf{p}\| = \sqrt{(\mathbf{q} - \mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})}.$$

- Cosine similarity:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \cos\theta$$

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2}$$



Feature Matching

- Criteria 1:
  - Compute distance in feature space, e.g., Euclidean distance between 128-dim SIFT descriptors
  - Match point to lowest distance (nearest neighbor)

- Problems:
  - Does everything have a match?

Feature Matching

- Criteria 2:
  - Compute distance in feature space, e.g., Euclidean distance between 128-dim SIFT descriptors
  - Match point to lowest distance (nearest neighbor)
  - Ignore anything higher than threshold (no match!)

- Problems:
  - Threshold is hard to pick
  - Non-distinctive features could have lots of close matches, only one of which is correct

Nearest Neighbor Distance Ratio

Compare distance of closest (NN1) and second-closest (NN2) feature vector neighbor.

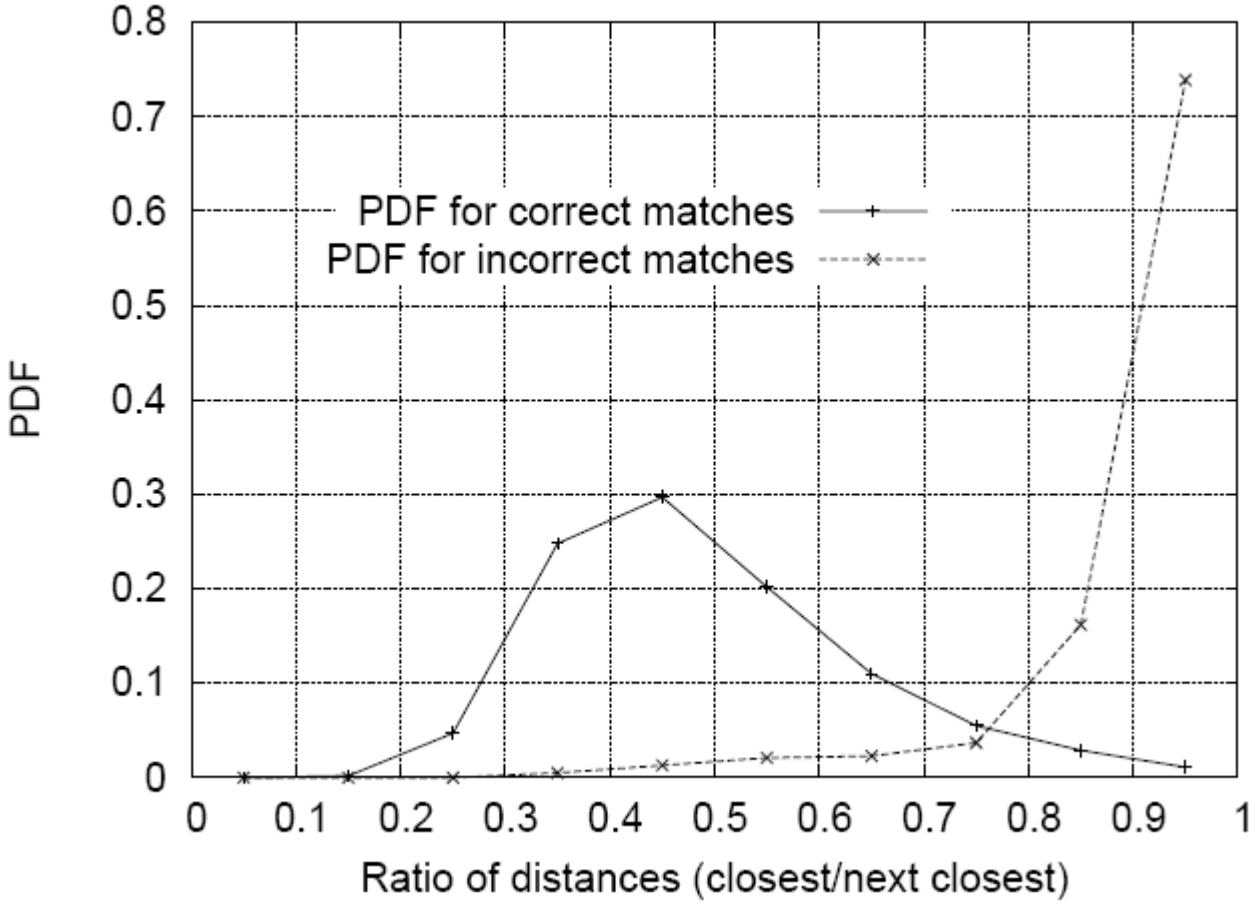
- If NN1 ≈ NN2, ratio $\frac{NN1}{NN2}$ will be ≈ 1 -> matches too close.
- As NN1 << NN2, ratio $\frac{NN1}{NN2}$ tends to 0.

Sorting by this ratio puts matches in order of confidence.

Threshold ratio – but how to choose?

Nearest Neighbor Distance Ratio

- Lowe computed a probability distribution functions of ratios
- 40,



Lowe IJCV 2004

Ratio threshold depends on your application's view on the trade-off between the number of false positives and true positives!

Demo

- https://demo.ipol.im/demo/82/