Computer Vision

(Summer Semester 2020)

Lecture 5, Part 2

Feature Descriptors (2)

Feature Descriptors and Matching

- SIFT: scale-invariant image descriptor
- Feature matching

Note: The core of these slides stems from the class CSCI 1430: “Introduction to Computer Vision” by James Tompkin, Fall 2017, Brown University.

Review: Basic Feature Detection Algorithm

1. Compute horizontal and vertical derivatives, *viz.* I_x and I_y

   $I_x = I \ast \mathrm{LoG}_x$ & $I_y = I \ast \mathrm{LoG}_y$

2. Compute the moment matrix **M** $M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

3. Convolve each of these images with a larger gaussian

4. Compute thresholds to find the featureness (eg: cornerness) scores

5. Find local maxima above a threshold and report them as features (eg: MSER)

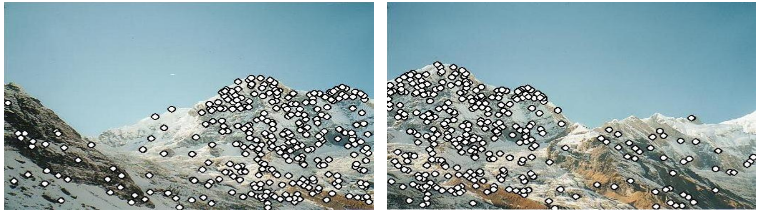
Local Image Descriptors
(Szeliski 4.1)

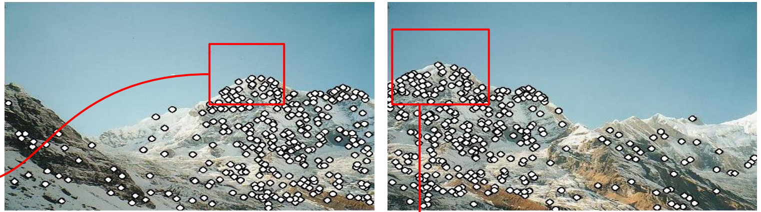
Links
- https://www.robots.ox.ac.uk/~vgg/research/affine/
- Distinctive Image Features from Scale-Invariant Keypoints
- https://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

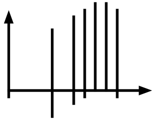
Local features: main components

1) Detection:
Find a set of distinctive key points.

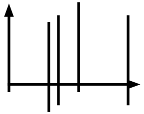
2) Description:
Extract feature descriptor around each interest point as vector.





$\mathbf{x}_1$ 

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

$\mathbf{x}_2$ 

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

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

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

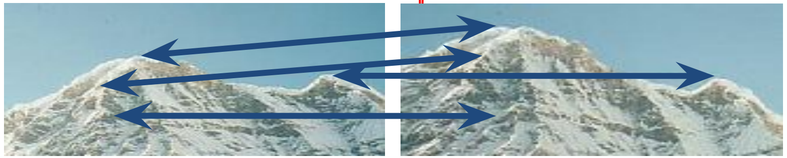
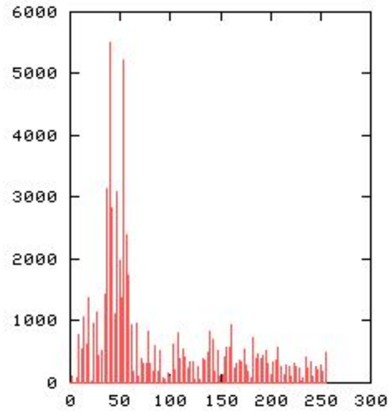
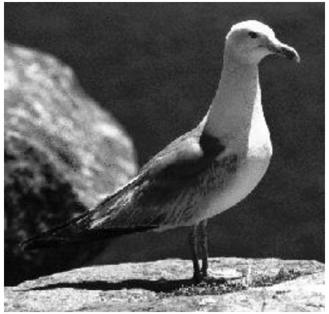


Image Representations: Histograms





Global histogram to represent distribution of features

- Color, texture, depth, oriented gradients…

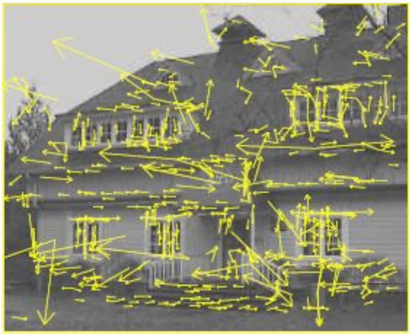
Local histogram per detected point

SIFT – Lowe IJCV 2004

- Compute local histograms of oriented gradients
- Find Difference of Gaussian scale-space extrema (for target scale)
- Post-processing
  - Position interpolation
  - Discard low-contrast points
  - Eliminate points along edges
- Orientation estimation
- Descriptor extraction
  - Motivation: We want some sensitivity to spatial layout, but not too much, so blocks of histograms give us that.

SIFT Descriptor Extraction

- Given a keypoint with scale and orientation:
- Pick scale-space image which most closely matches estimated scale
- Resample image to match orientation OR
- Subtract detector orientation from vector to give invariance to general image rotation.



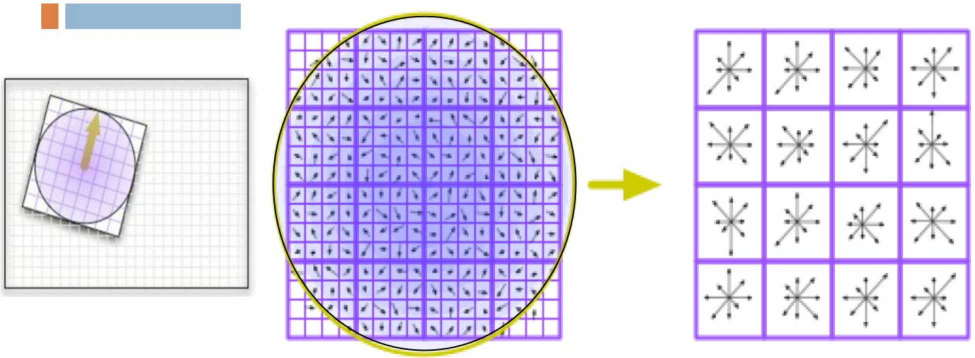
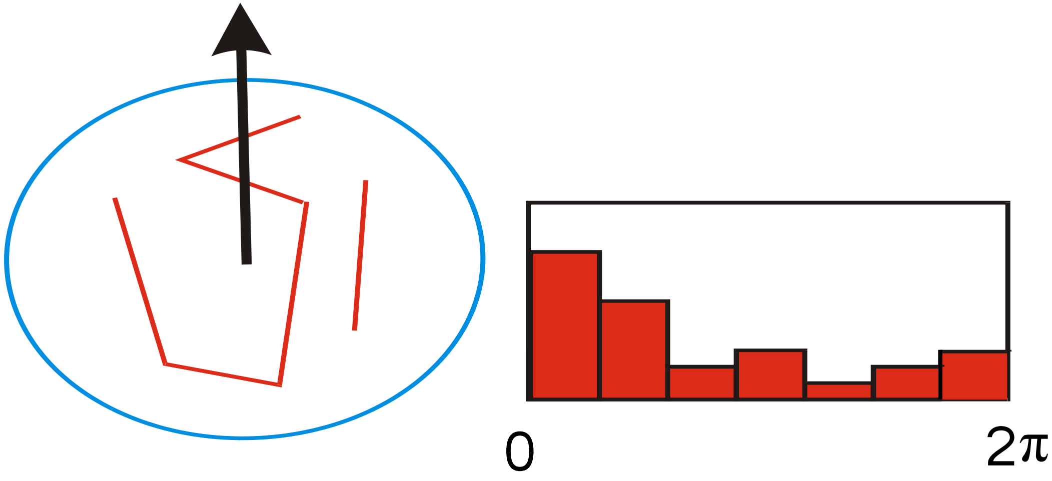


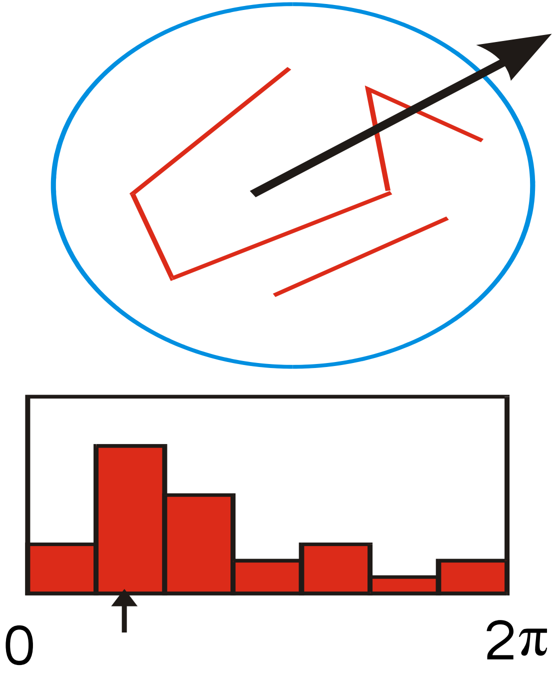
Image gradients

Keypoint descriptor

SIFT Orientation Normalization

- Compute orientation histogram
- Select dominant orientation Θ
- Normalize: rotate to fixed orientation

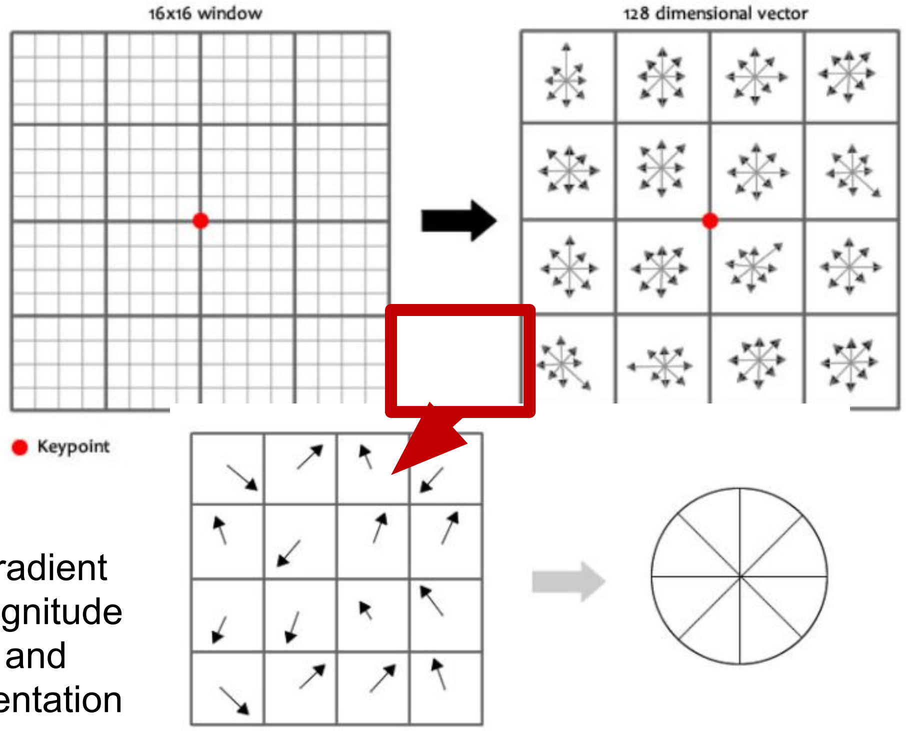




D.Lowe

SIFT Descriptor Extraction

- Given a keypoint with scale and orientation



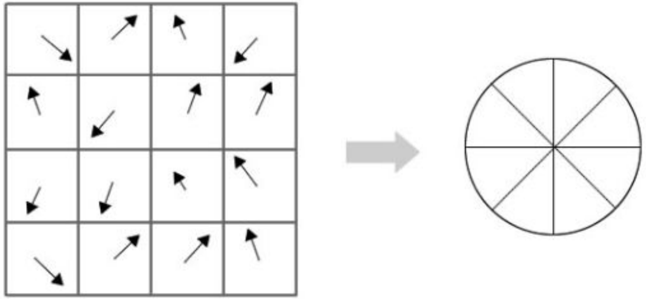
Gradient magnitude and orientation

8 bin 'histogram' - add magnitude amounts!

SIFT Descriptor Extraction

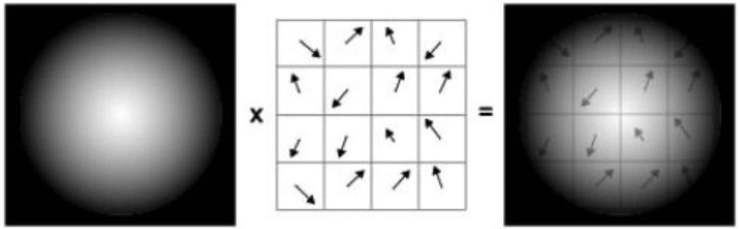
- Within each 4x4 window

Gradient magnitude and orientation



8 bin 'histogram' - add magnitude amounts!

Weight magnitude that is added to 'histogram' by Gaussian



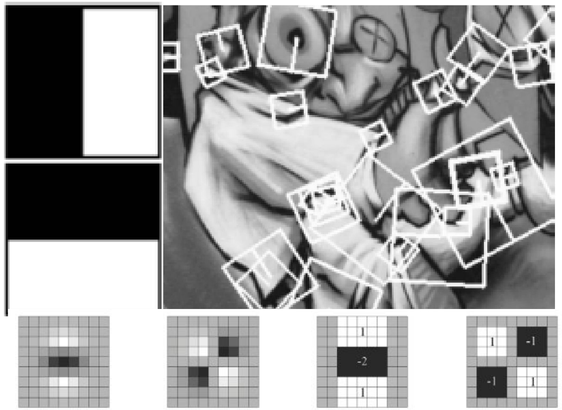
SIFT Descriptor Extraction

- Extract 8 x 16 values 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
  - Clamp all vector values > 0.2 to 0.2.
    - Robust to “non-linear illumination effects”
    - Image value saturation / specular highlights
  - Renormalize

Implementation Notes

- Efficient Implementation
  - Filter using oriented kernels based on directions of histogram bins.
  - Called ‘steerable filters’
- Sources
  - tutorial: http://aishack.in/tutorials/sift-scale-invariant-feature-transform-features/
  - Lowe’s original paper: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

Local Descriptors: SURF



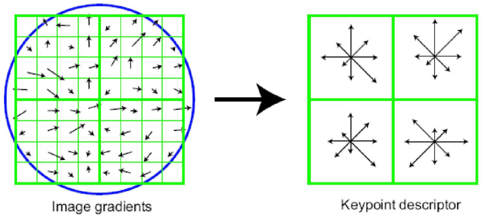
- Fast approximation of SIFT idea
  - Efficient computation by 2D box filters & integral images
    ⇒ 6 times faster than SIFT
  - Equivalent quality for object identification

- GPU implementation available
  - Feature extraction @ 200Hz
    (detector + descriptor, 640×480 img)
  - http://www.vision.ee.ethz.ch/~surf

[Bay, ECCV'06], [Cornelis, CVGPU'08]

Review: Local Descriptors

- Most features can be thought of as templates, histograms (counts), or combinations
- The ideal descriptor should be
  - Robust and Distinctive
  - Compact and Efficient
- Most available descriptors focus on edge/gradient information
  - Capture texture information
  - Color rarely used



Available at a web site near you…

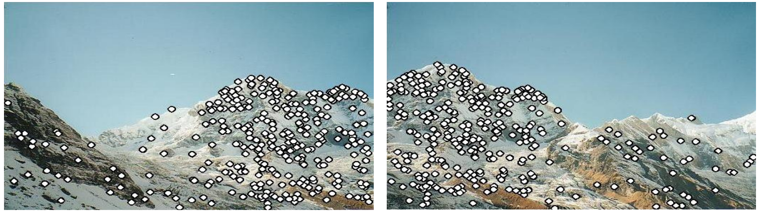
- Many local feature detectors have executables available online:
  - http://www.robots.ox.ac.uk/~vgg/research/affine
  - http://www.cs.ubc.ca/~lowe/keypoints/
  - http://www.vision.ee.ethz.ch/~surf

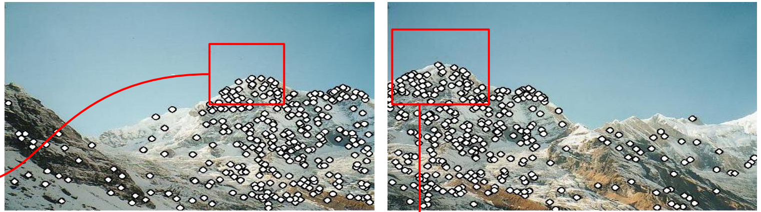
Feature Matching
(Szeliski 4.1)

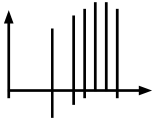
Local features: main components

1) Detection:
   Find a set of distinctive key points.

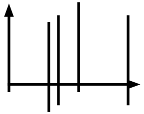
2) Description:
   Extract feature descriptor around each interest point as vector.





$\mathbf{x}_1$ 

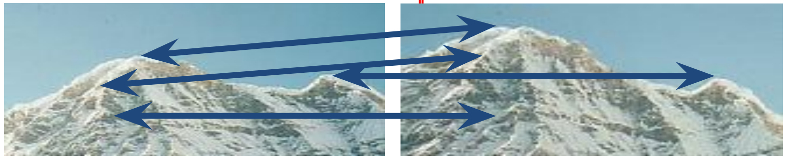
$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

$\mathbf{x}_2$ 

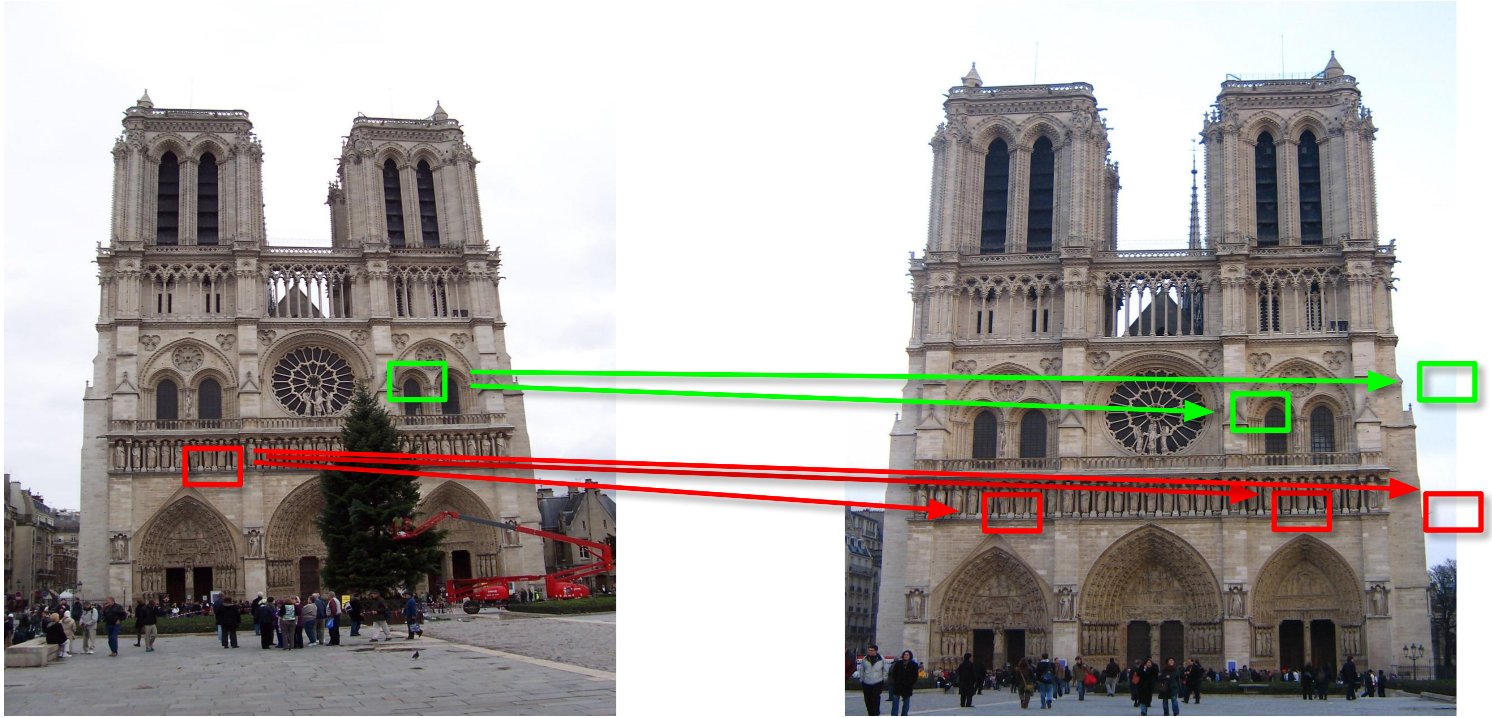
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

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

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



How do we decide which features match?



Distance: 0.34, 0.30, 0.40

Distance: 0.61, 1.22

Feature Matching

- Criteria:
  - 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.

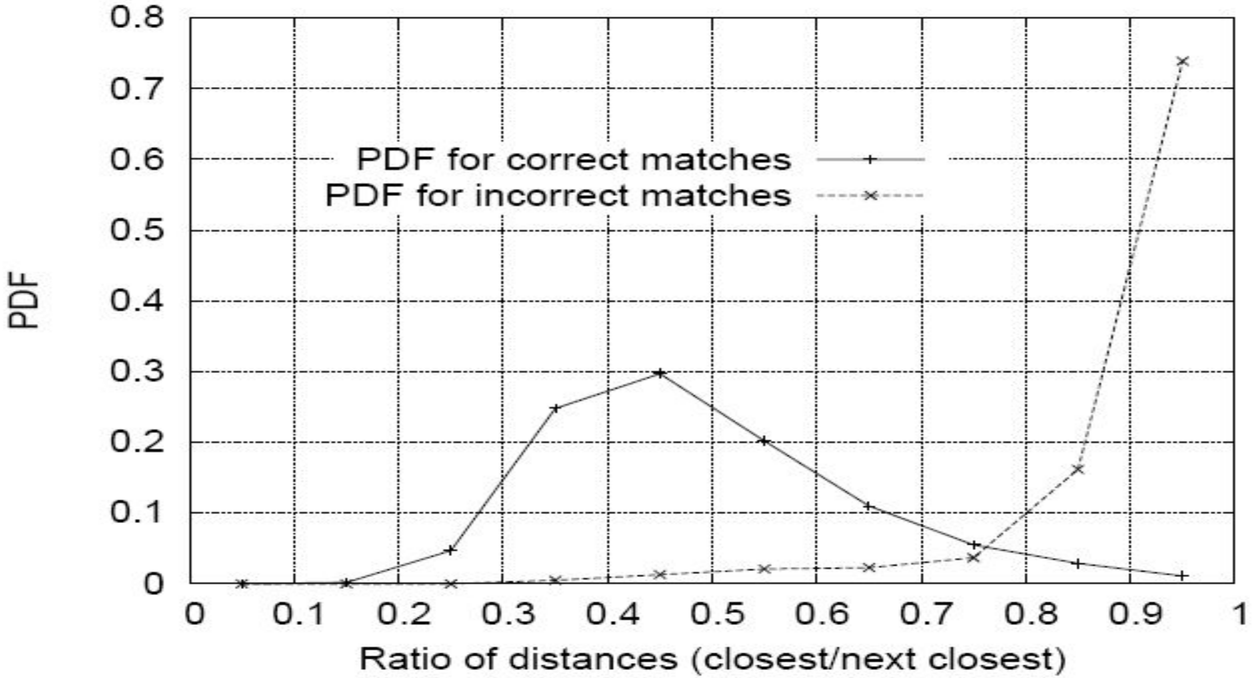
- If NN1 ≈ NN2, ratio $\frac{NN1}{NN2}$ will be ≈ 1 -> matches too close.
- As NN1 << NN2, ratio $\frac{NN1}{NN2}$ tends to 0.

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
- 40,000 keypoints with hand-labeled ground truth



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