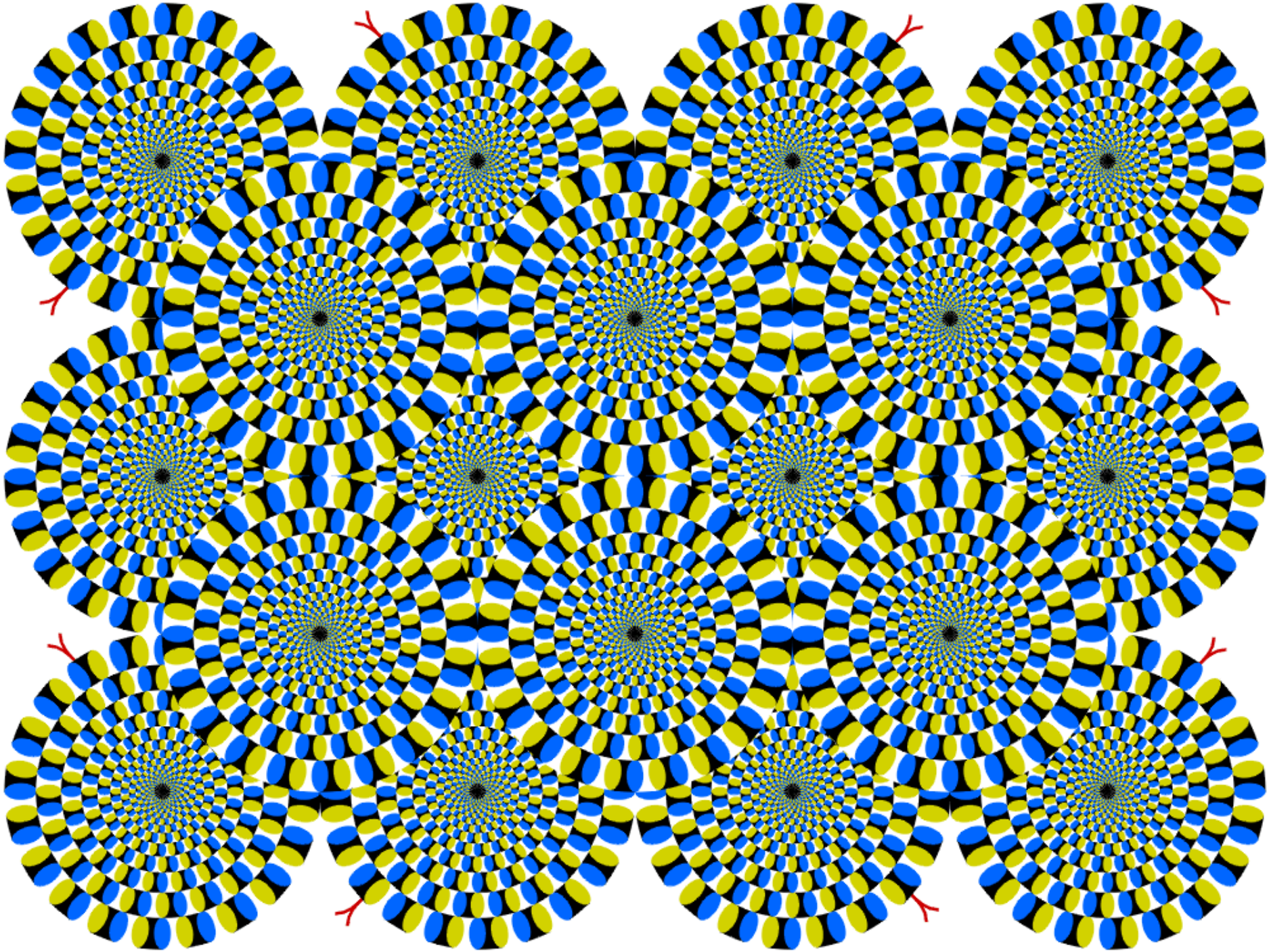
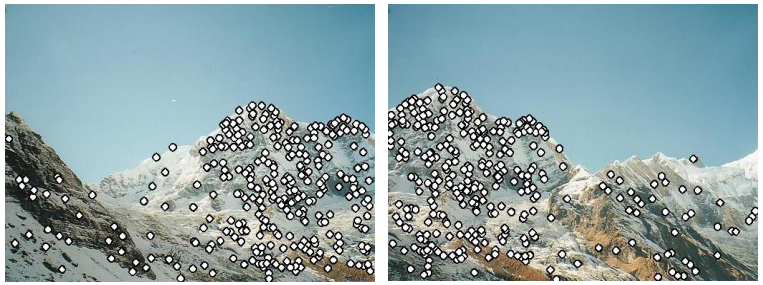
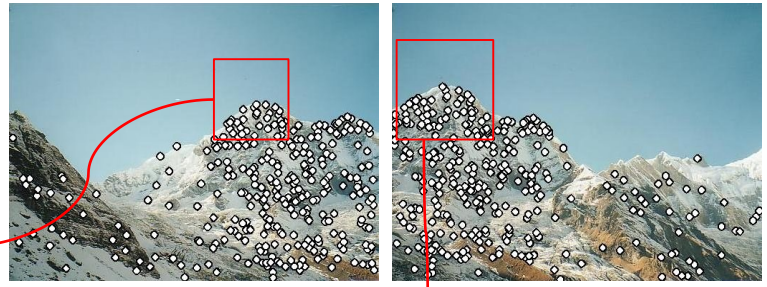


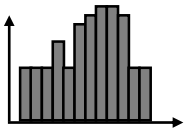
Local features: main components

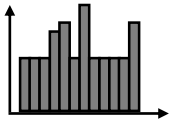
1) Detection:
Find a set of distinctive key points.



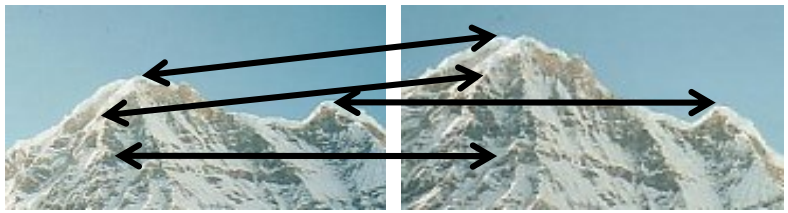
2) Description:
Extract feature descriptor around each interest point as vector.



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

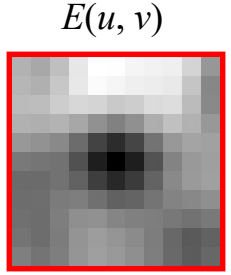
x_2  $\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

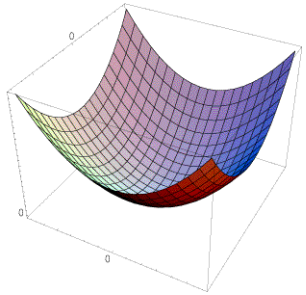
3) Matching:
Compute distance between feature vectors to find correspondence.



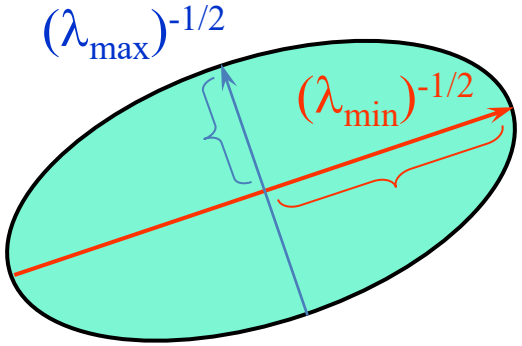
$$d(\mathrm{x}_1, \mathrm{x}_2) < T$$

Review: Harris corner detector





- Approximate distinctiveness by local auto-correlation.
- Approximate local auto-correlation by second moment matrix **M**.
- Distinctiveness (or cornerness) relates to the eigenvalues of **M**.
- Instead of computing eigenvalues directly, we can use determinant and trace of **M**.



$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Canny edge detector

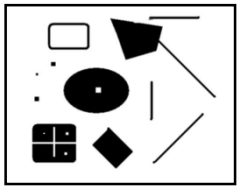
1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” to single pixel width
4. ‘Hysteresis’ Thresholding:
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
   - ‘Follow’ edges starting from strong edge pixels
     - Connected components (Szeliski 3.3.4)

- MATLAB: edge(image, ‘canny’)

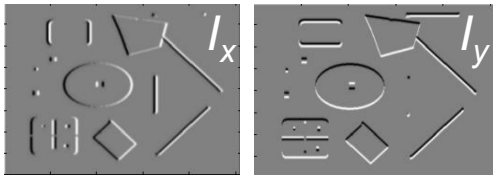
Source: D. Lowe, L. Fei-Fei

Harris Detector [Harris88]

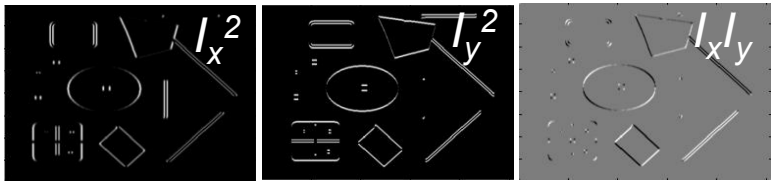
$$M(\sigma_I,\sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$



1. Image derivatives (optionally, blur first)



2. Square of derivatives

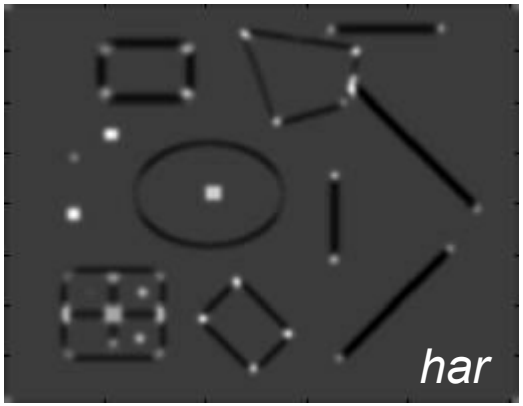


3. Gaussian filter $g(\sigma_I)$

$$\det M = \lambda_1 \lambda_2$$
$$\text{trace}\, M = \lambda_1 + \lambda_2$$

4. Cornerness function – both eigenvalues are strong

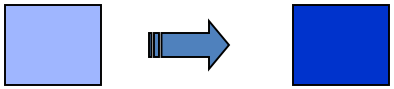
$$har = \det[M(\sigma_I,\sigma_D)] - \alpha[\text{trace}(M(\sigma_I,\sigma_D))^2]$$
$$= g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$



5. Non-maxima suppression

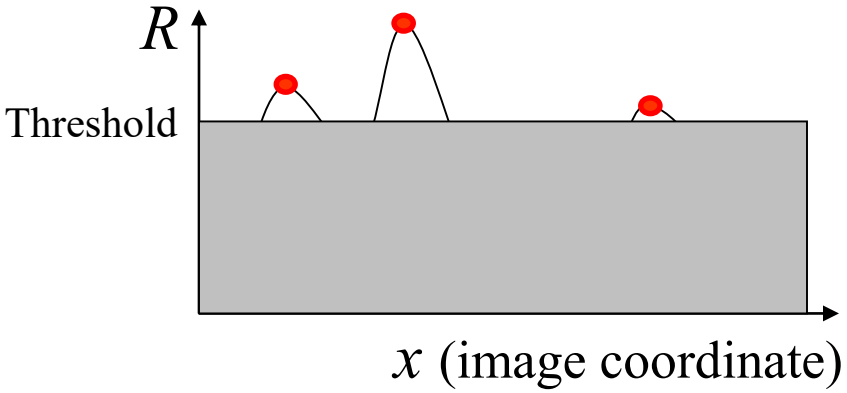
HOW INVARIANT ARE HARRIS CORNERS?

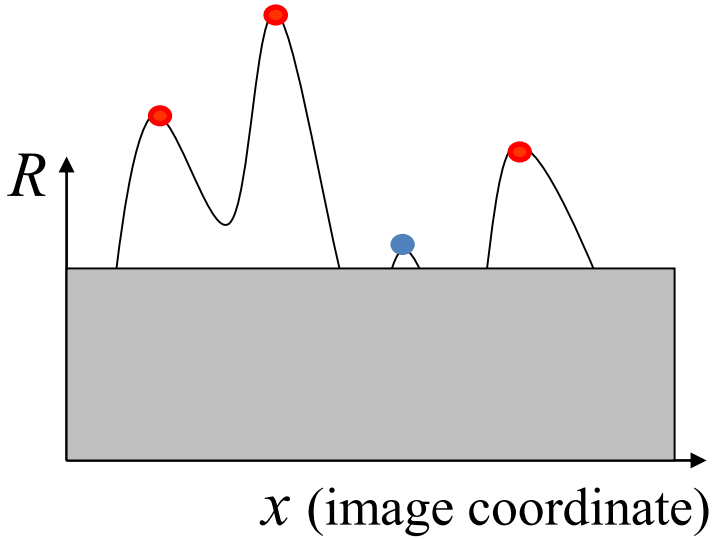
Affine intensity change



$$I \rightarrow a\,I + b$$

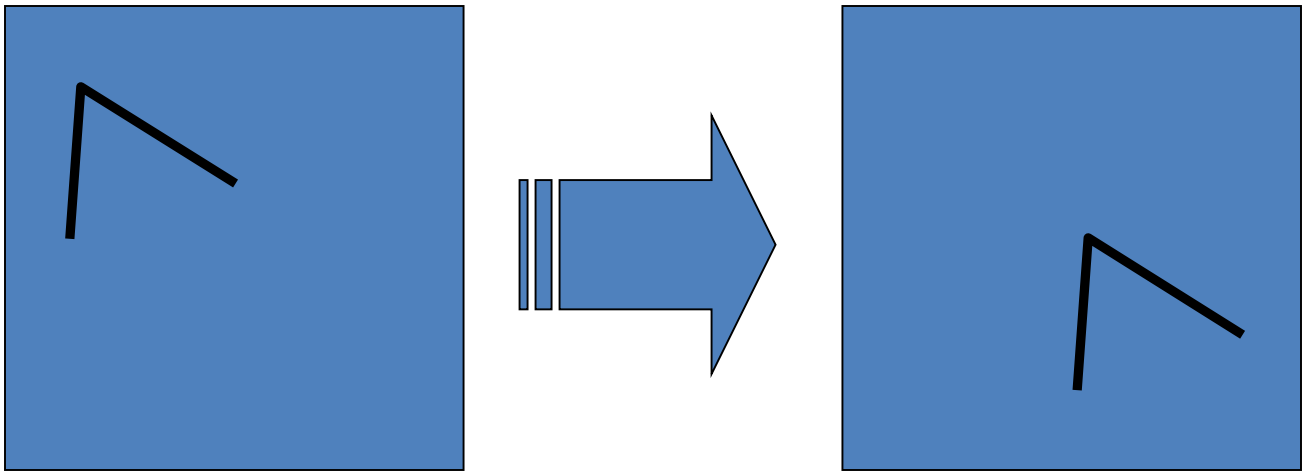
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\,I$





Partially invariant to affine intensity change

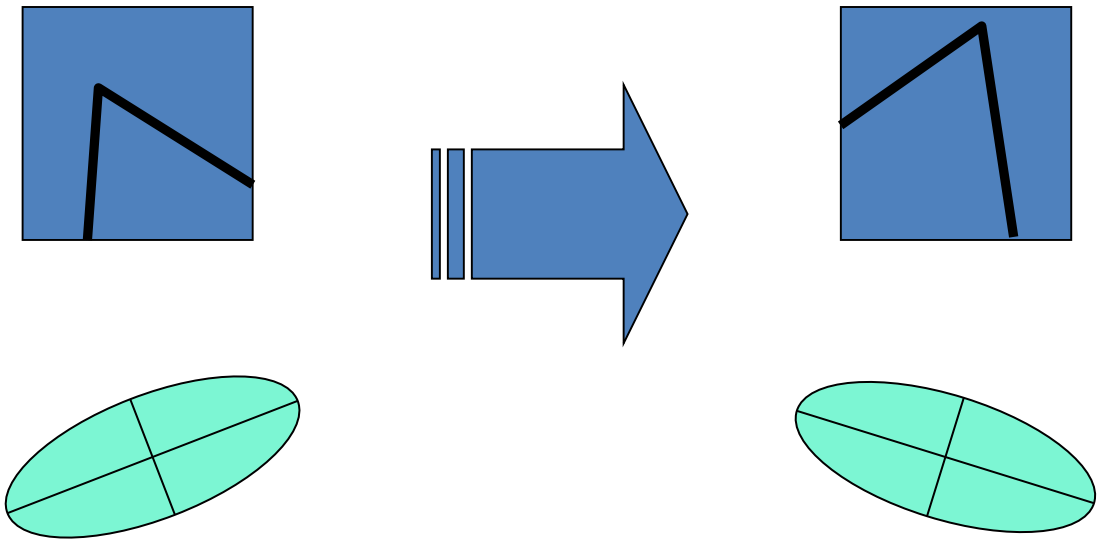
Image translation



- Derivatives and window function are shift-invariant.

Corner location is covariant w.r.t. translation

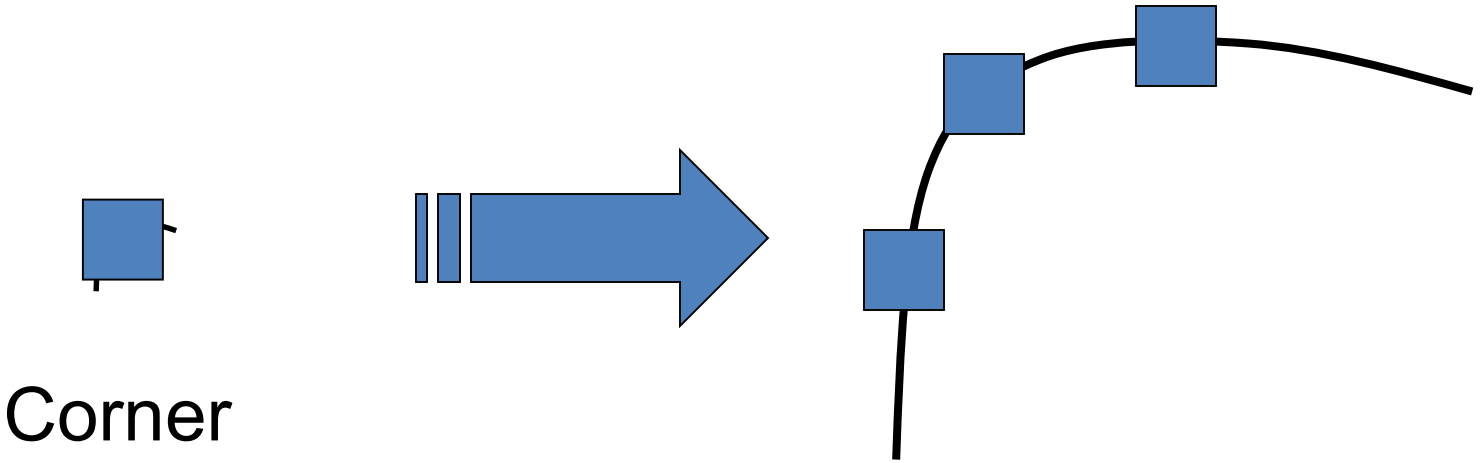
Image rotation



Second moment ellipse rotates but its shape (i.e., eigenvalues) remains the same.

Corner location is covariant w.r.t. rotation

Scaling



Corner

All points will be classified as edges

Corner location is not covariant to scaling!

WHAT IS THE ‘SCALE’ OF A FEATURE POINT?

Automatic Scale Selection

$$f(I_{i_1 \ldots i_m}(x,\sigma)) \;=\; f(I_{i_1 \ldots i_m}(x',\sigma'))$$

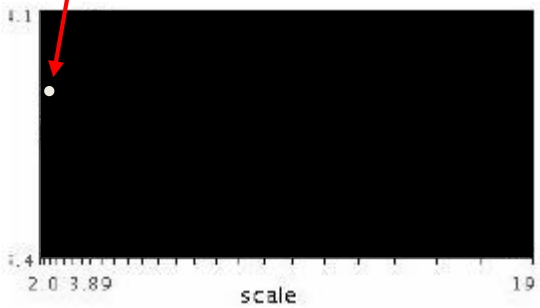
How to find patch sizes at which *f* response is equal?

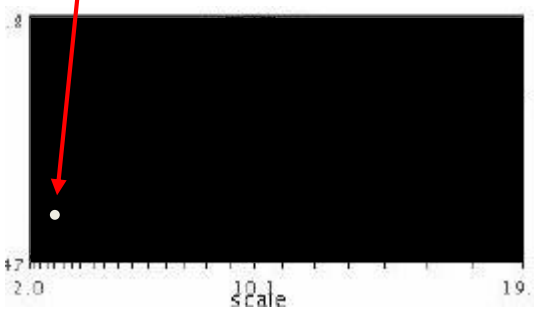
What is a good *f* ?

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Response of some function *f*





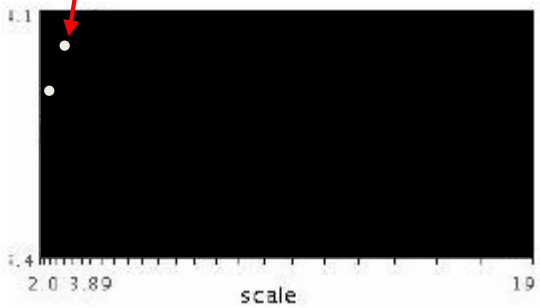
$f(I_{i_1 \ldots i_m}(x,\sigma))$

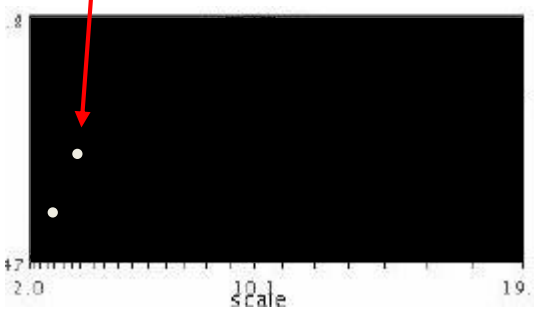
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Response of some function *f*



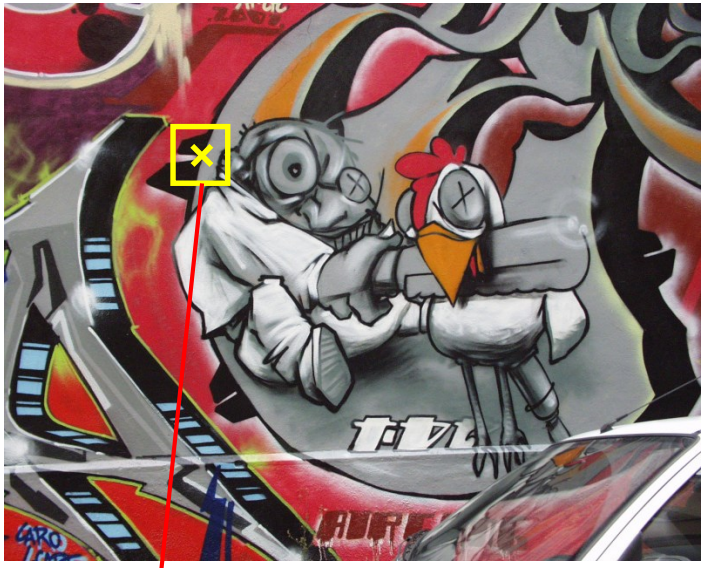


$f(I_{i_1 \ldots i_m}(x,\sigma))$

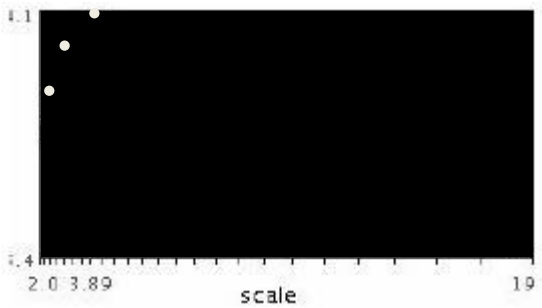
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

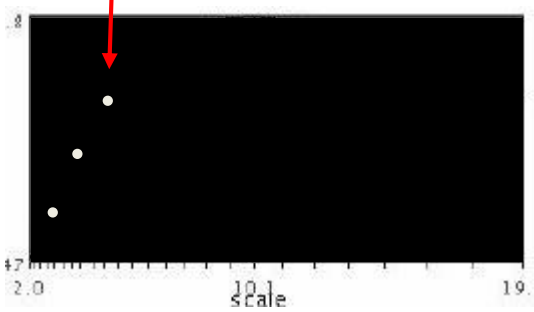
Automatic Scale Selection

- Function responses for increasing scale (scale signature)



Response of some function *f*



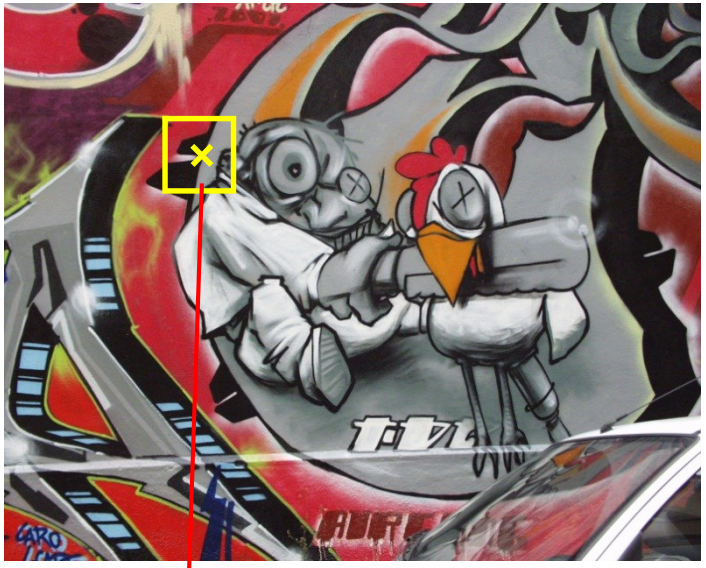


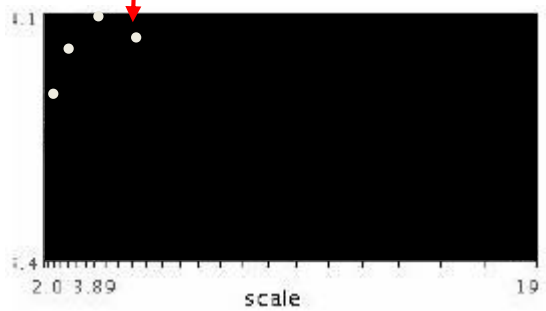
$f(I_{i_1 \ldots i_m}(x,\sigma))$

$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Automatic Scale Selection

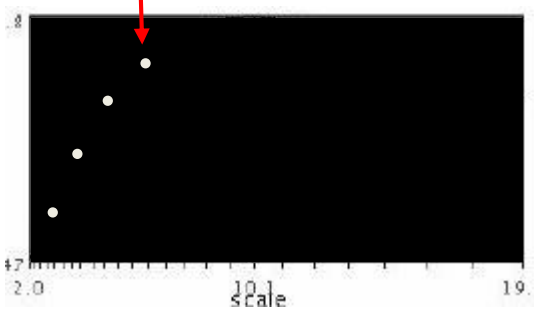
- Function responses for increasing scale (scale signature)





Response of some function *f*

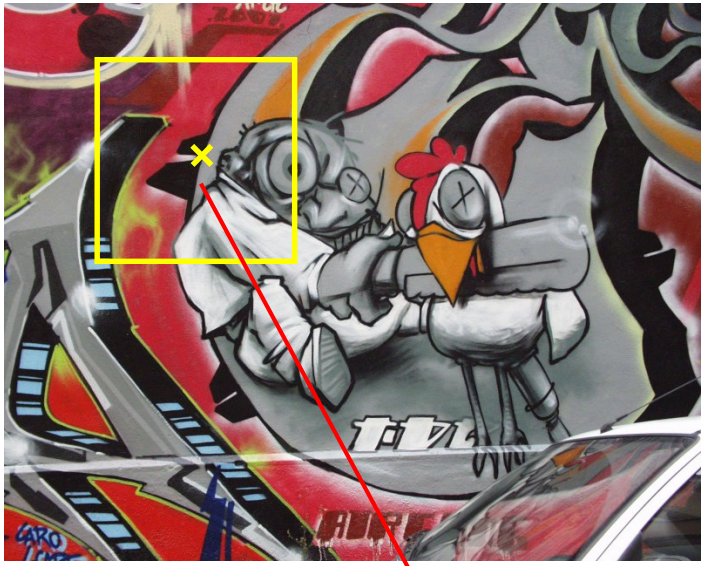
$f(I_{i_1 \ldots i_m}(x,\sigma))$

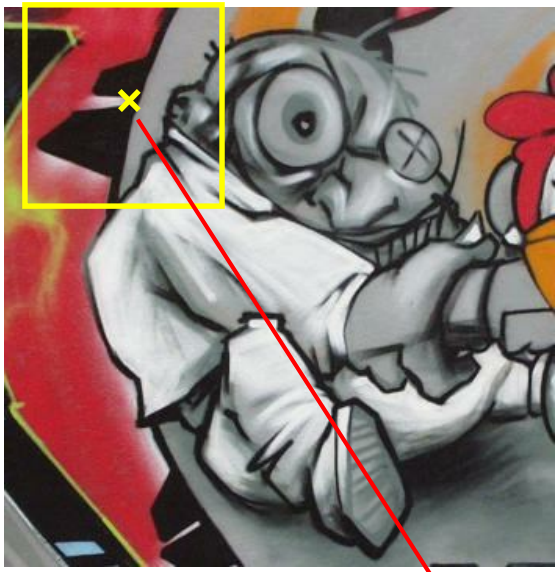


$f(I_{i_1 \ldots i_m}(x',\sigma'))$

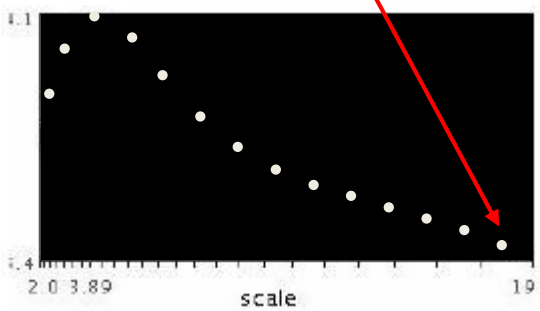
Automatic Scale Selection

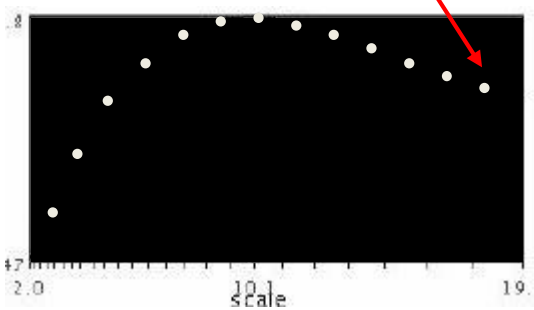
- Function responses for increasing scale (scale signature)





Response of some function *f*



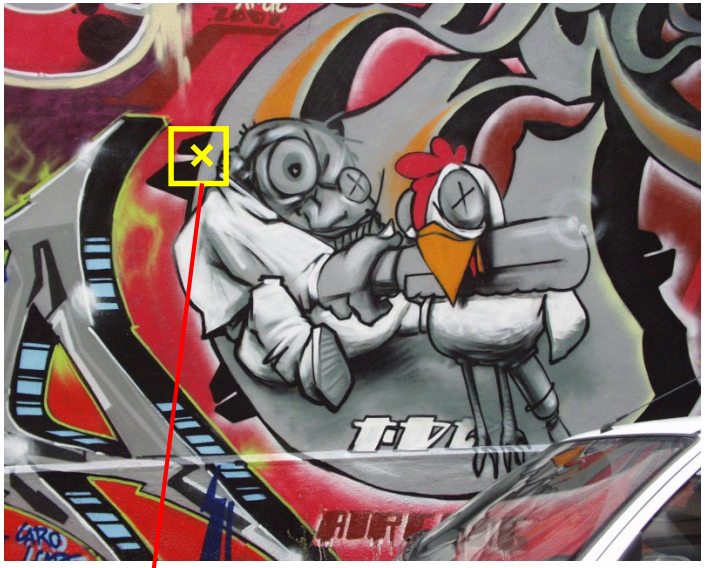


$f(I_{i_1 \ldots i_m}(x,\sigma))$

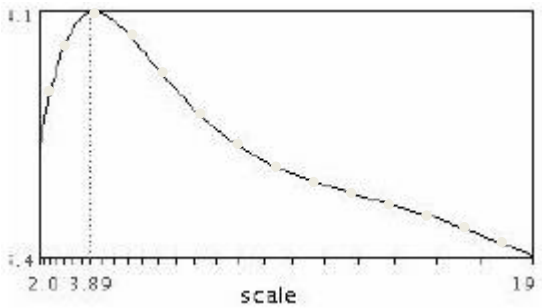
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

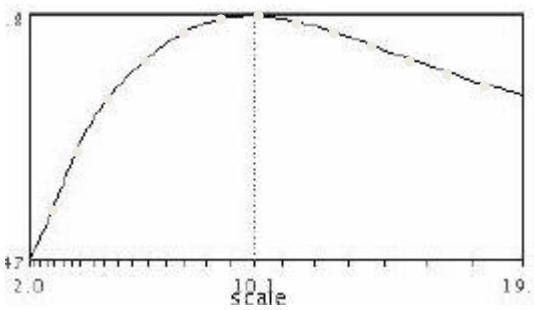
Automatic Scale Selection

- Function responses for increasing scale (scale signature)



Response of some function *f*

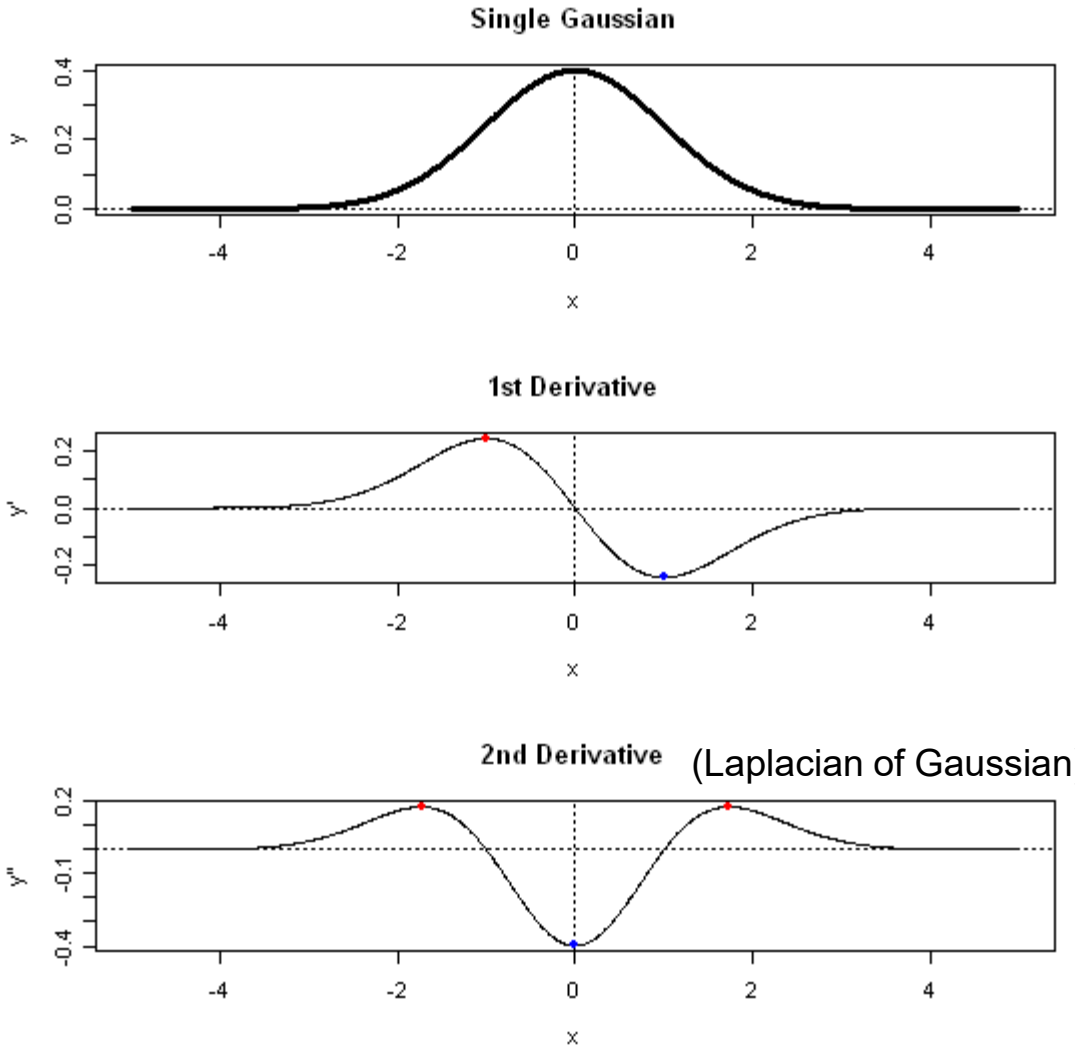




$f(I_{i_1 \ldots i_m}(x,\sigma))$

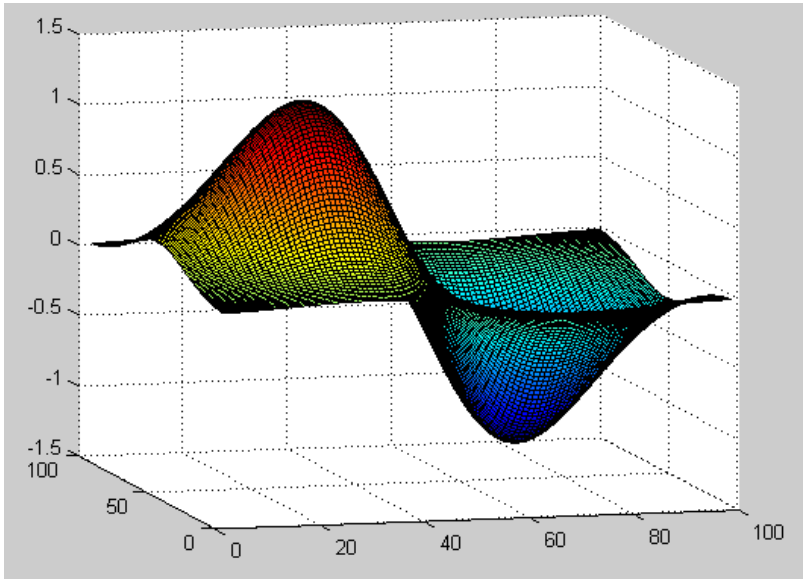
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

What Is A Useful Signature Function *f* ?



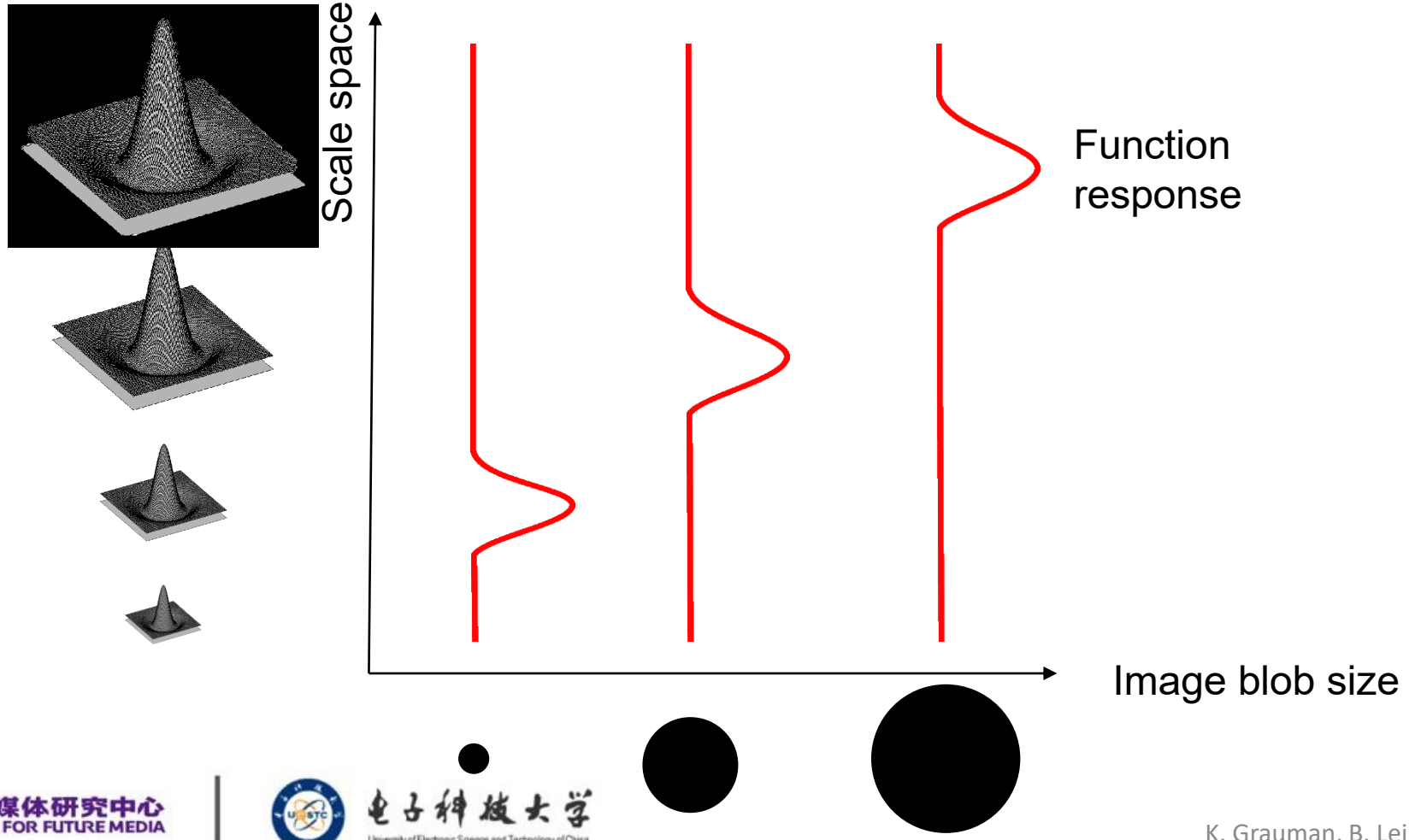
(Laplacian of Gaussian)

1st Derivative of Gaussian



What Is A Useful Signature Function *f* ?

- "Blob" detector is common for corners
  - - Laplacian (2nd derivative) of Gaussian (LoG)



Canny edge detector

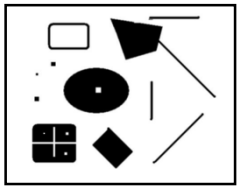
1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” to single pixel width
4. ‘Hysteresis’ Thresholding:
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
   - ‘Follow’ edges starting from strong edge pixels
     - Connected components (Szeliski 3.3.4)

- MATLAB: edge(image, ‘canny’)

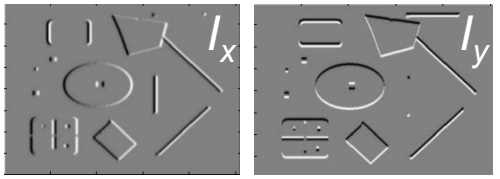
Source: D. Lowe, L. Fei-Fei

Harris Detector [Harris88]

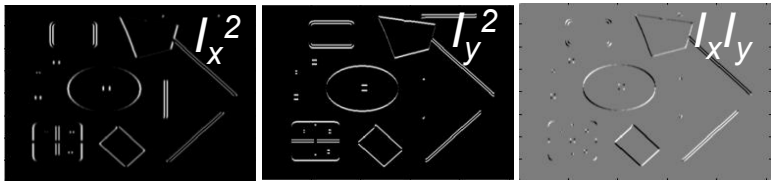
$$M(\sigma_I,\sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$



1. Image derivatives (optionally, blur first)



2. Square of derivatives

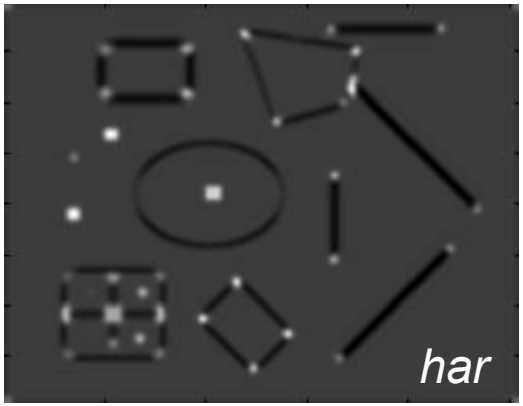


3. Gaussian filter $g(\sigma_I)$

$$\det M = \lambda_1 \lambda_2$$
$$\text{trace}\, M = \lambda_1 + \lambda_2$$

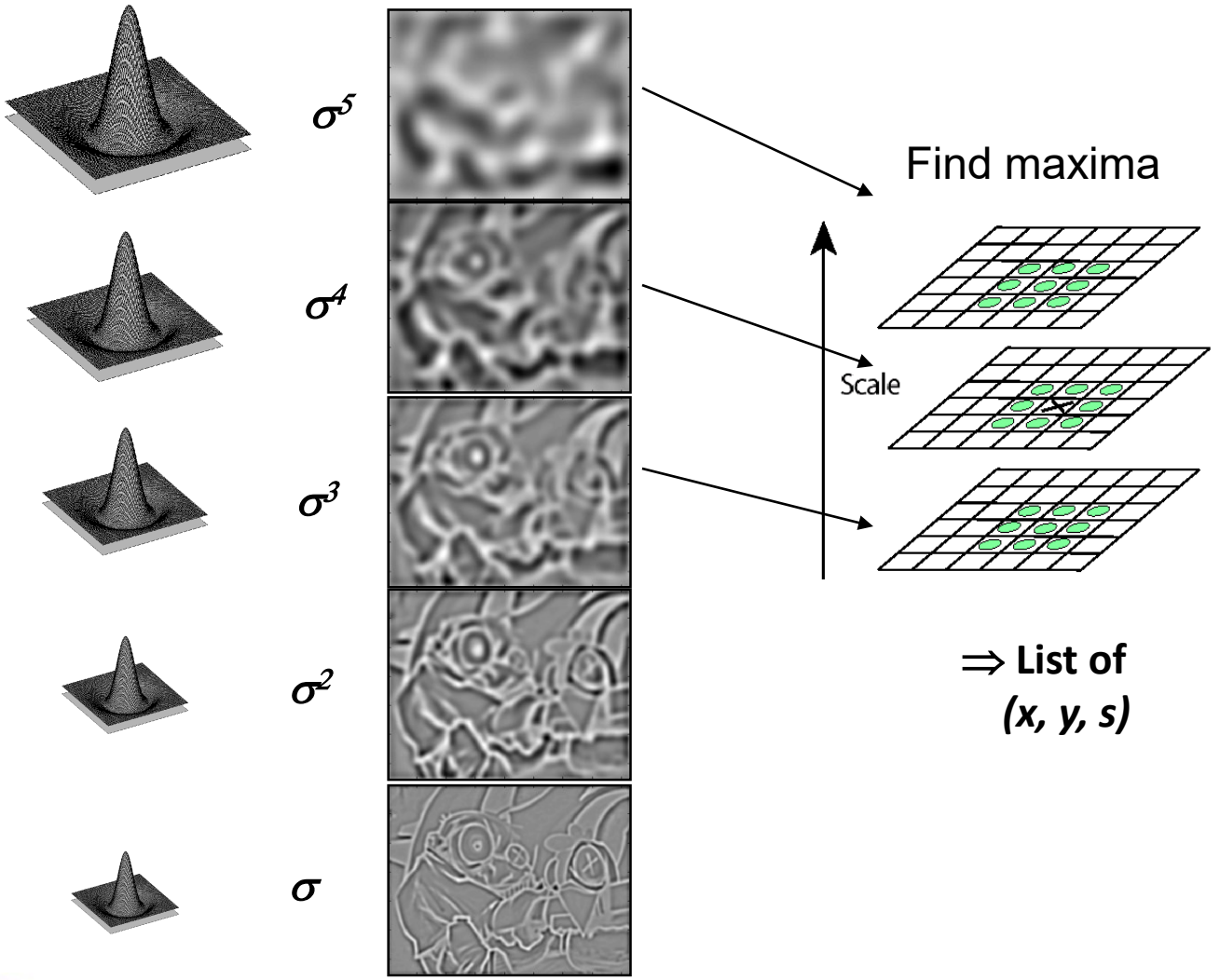
4. Cornerness function – both eigenvalues are strong

$$har = \det[M(\sigma_I,\sigma_D)] - \alpha[\text{trace}(M(\sigma_I,\sigma_D))^2]$$
$$= g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$



5. Non-maxima suppression

Find local maxima in position-scale space



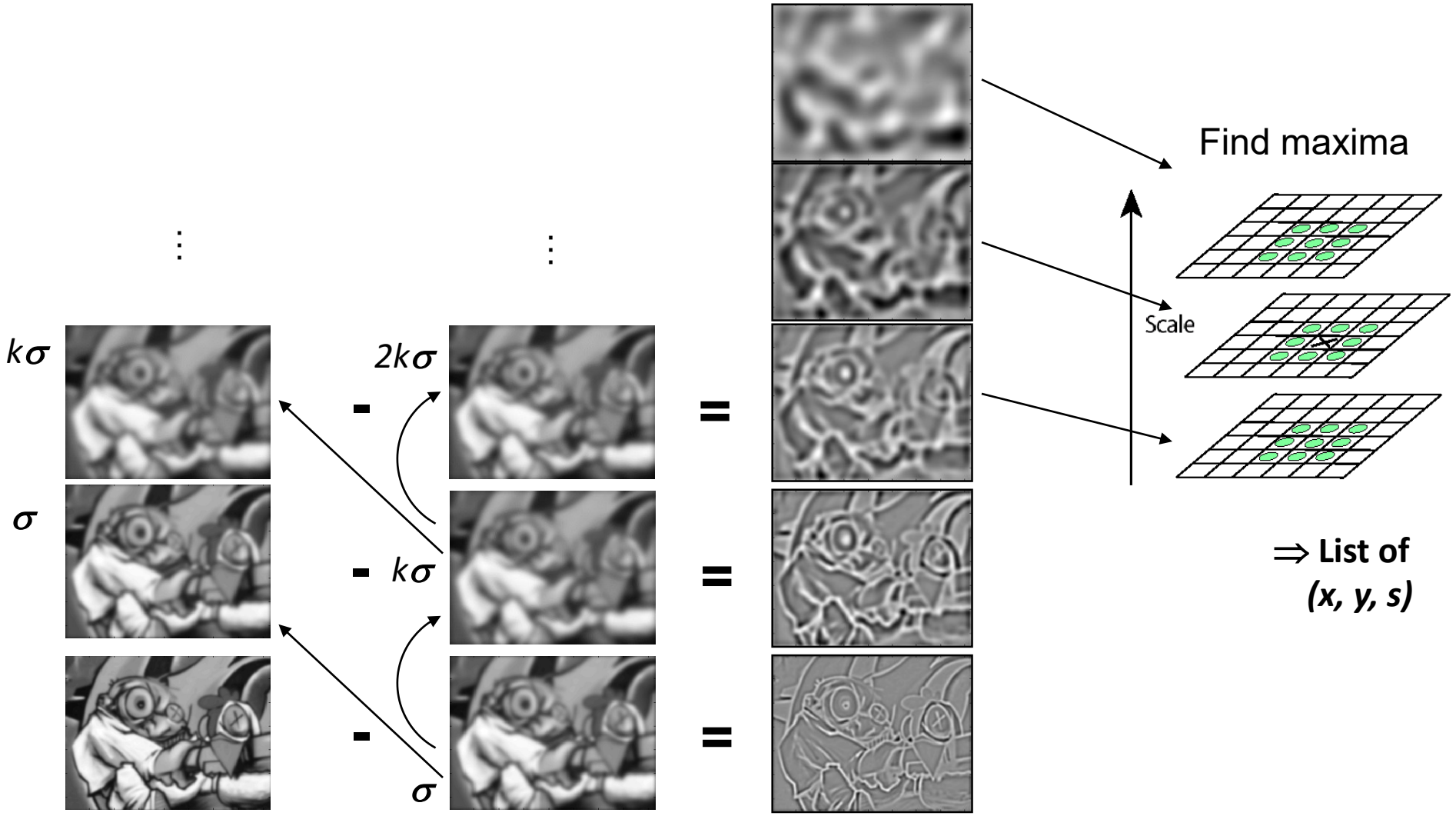
K. Grauman, B. Leibe

Alternative approach

Approximate LoG with Difference-of-Gaussian (DoG).

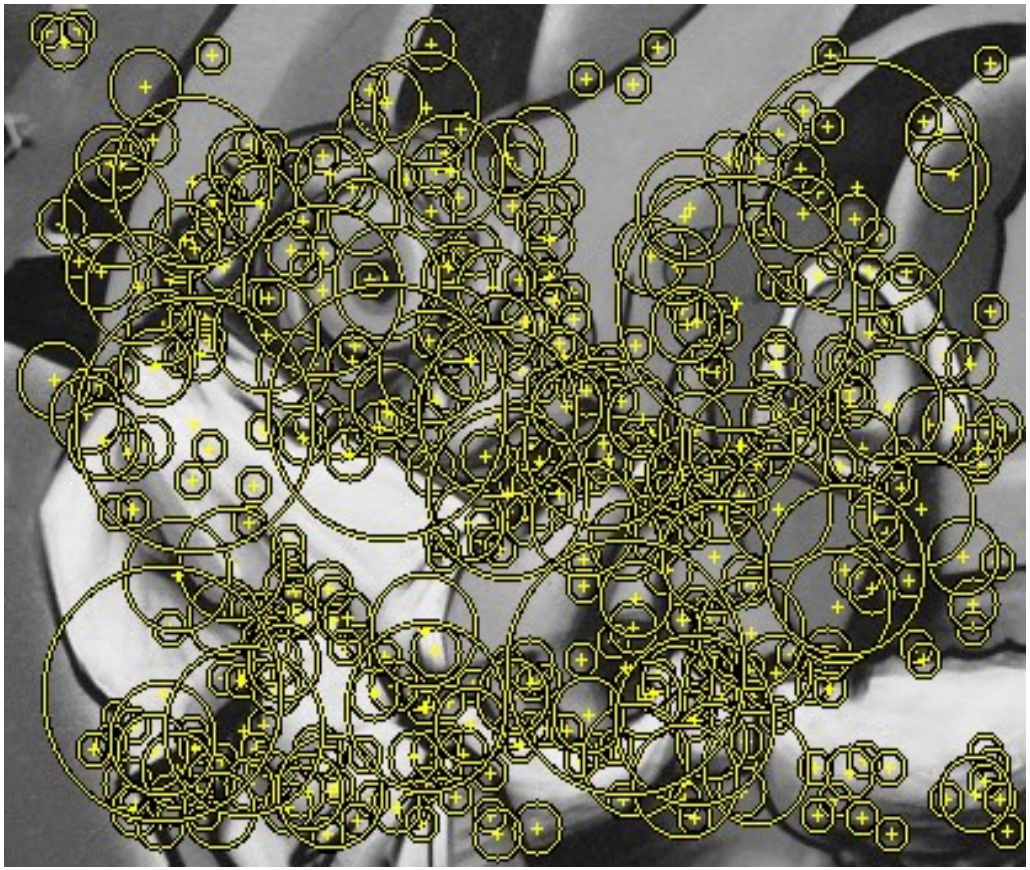
1. Blur image with σ Gaussian kernel
2. Blur image with $k\sigma$ Gaussian kernel
3. Subtract 2. from 1.

Find local maxima in position-scale space of DoG



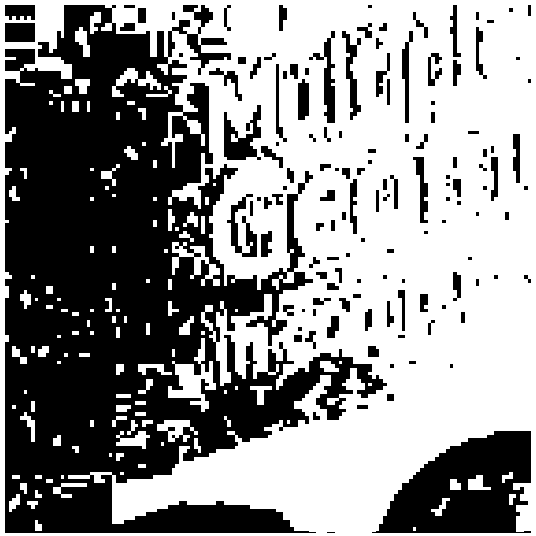
Results: Difference-of-Gaussian

- Larger circles = larger scale
- Descriptors with maximal scale response

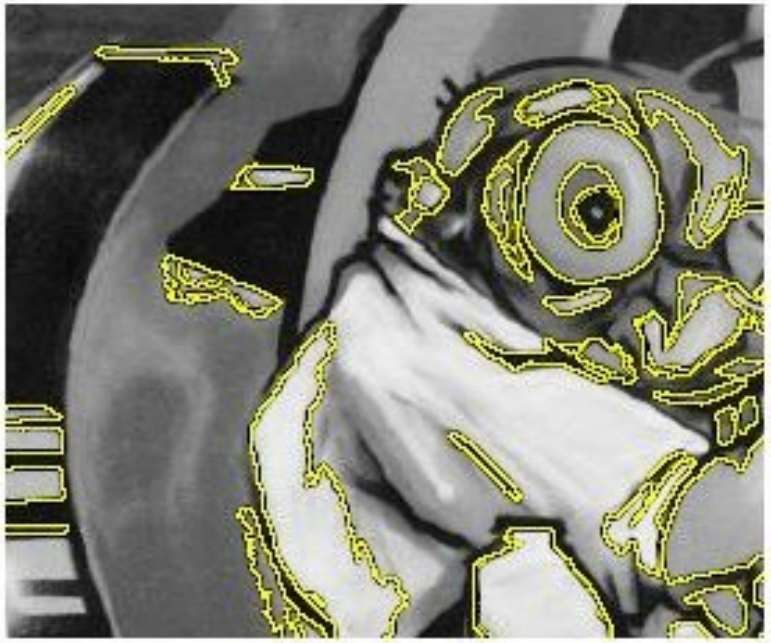


Maximally Stable Extremal Regions [Matas '02]

- Based on Watershed （分水岭） segmentation algorithm
- Select regions that stay stable over a large parameter range

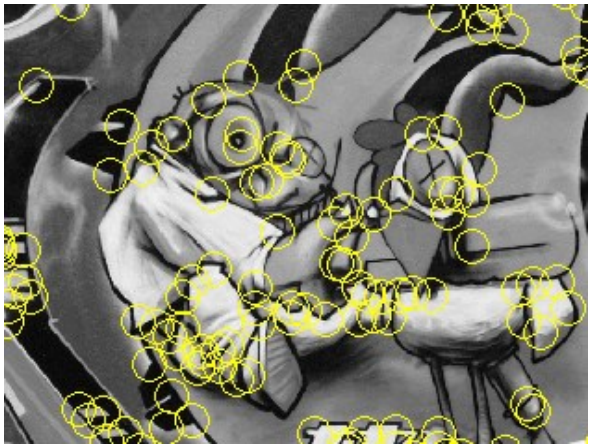


Example Results: MSER



Review: Interest points

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



(a) Gray scale input image

(b) Detected MSERs

Review: Choosing an interest point detector

- Why choose?
  - Collect more points with more detectors, for more possible matches
- What do you want it for?
  - Precise localization in x-y: Harris
  - Good localization in scale: Difference of Gaussian
  - Flexible region shape: MSER
- Best choice often application dependent
  - Harris-/Hessian-Laplace/DoG work well for many natural categories
  - MSER works well for buildings and printed things
- There have been extensive evaluations/comparisons
  - [Mikolajczyk et al., IJCV'05, PAMI'05]
  - All detectors/descriptors shown here work well

Local Image Descriptors

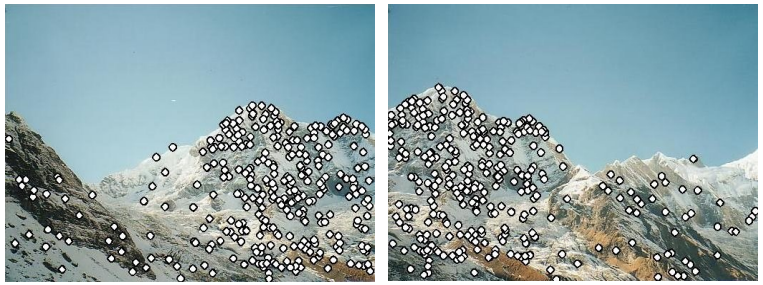
Read Szeliski 4.1

Acknowledgment: Many slides from James Hays, Derek Hoiem and Grauman & Leibe 2008 AAAI Tutorial

Local features: main components

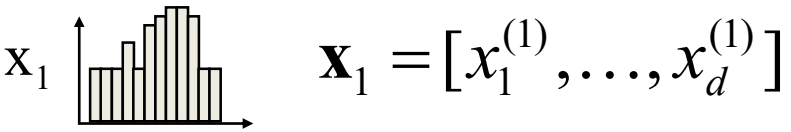
1) Detection:

Find a set of distinctive key points.

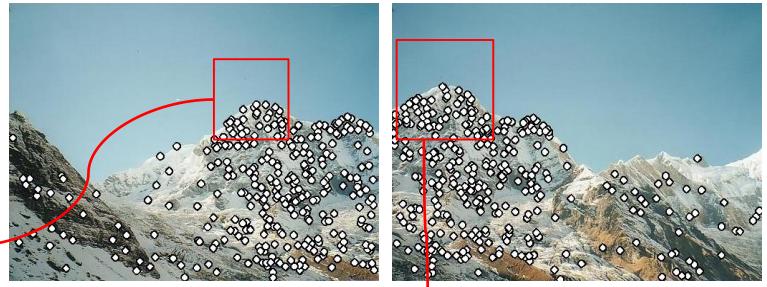


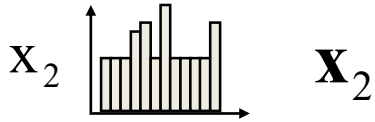
2) Description:

Extract feature descriptor around each interest point as vector.



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





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

3) Matching:

Compute distance between feature vectors to find correspondence.

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

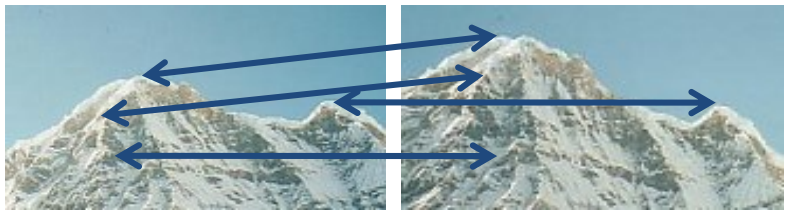
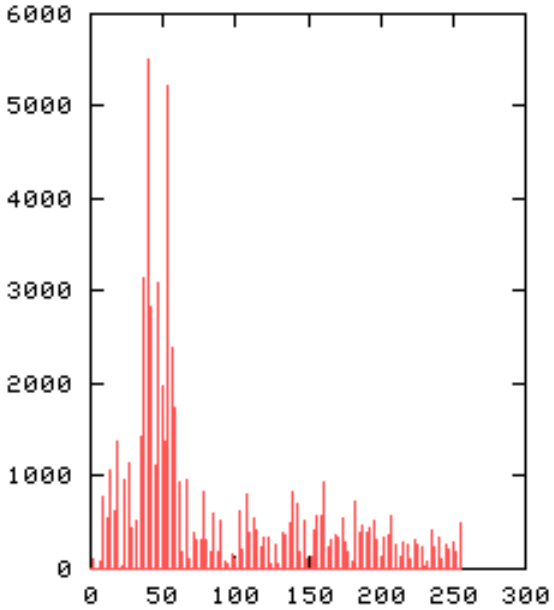
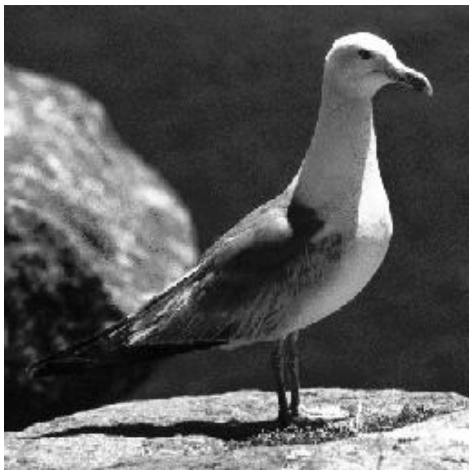


Image Representations: Histograms





Global histogram to represent distribution of features

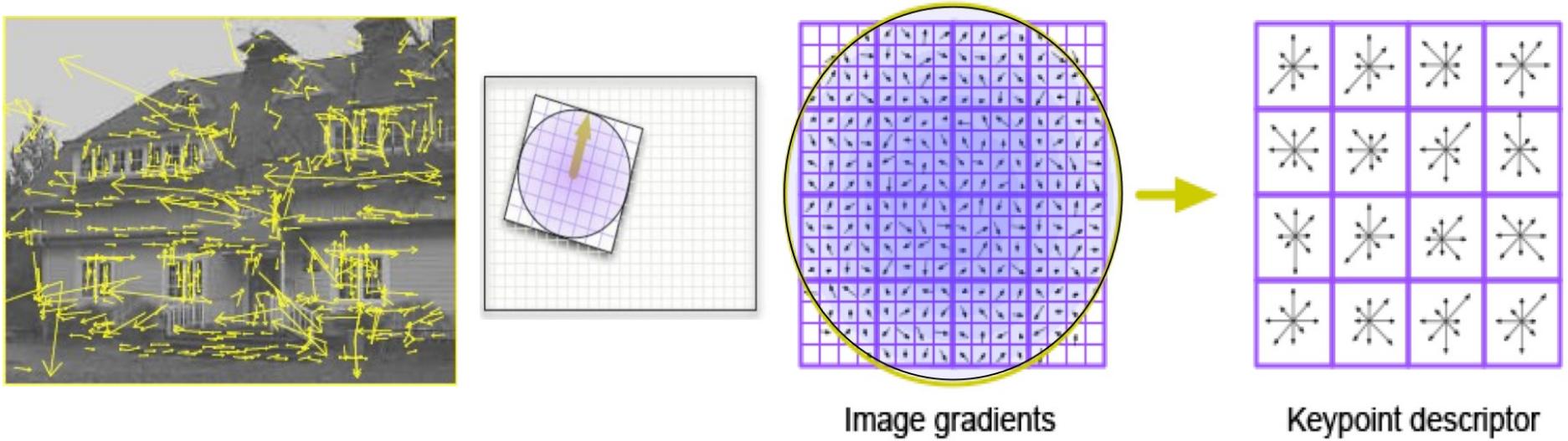
- Color, texture, depth, …

Local histogram per detected point

Images from Dave Kauchak

For what things do we compute histograms?

- Texture
- Local histograms of oriented gradients
- SIFT: Scale Invariant Feature Transform
  - Extremely popular (40k citations)



SIFT – Lowe IJCV 2004

SIFT descriptor formation

- Compute on local 16 x 16 window around detection.
- Rotate and scale window according to discovered orientation Θ and scale σ (gain invariance).
- Compute gradients weighted by a Gaussian of variance half the window (for smooth falloff).

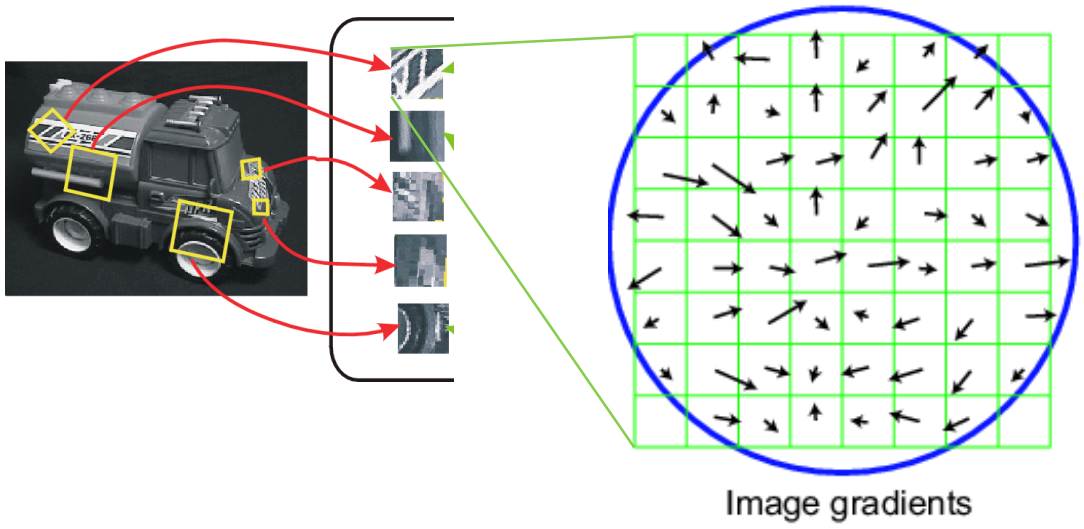
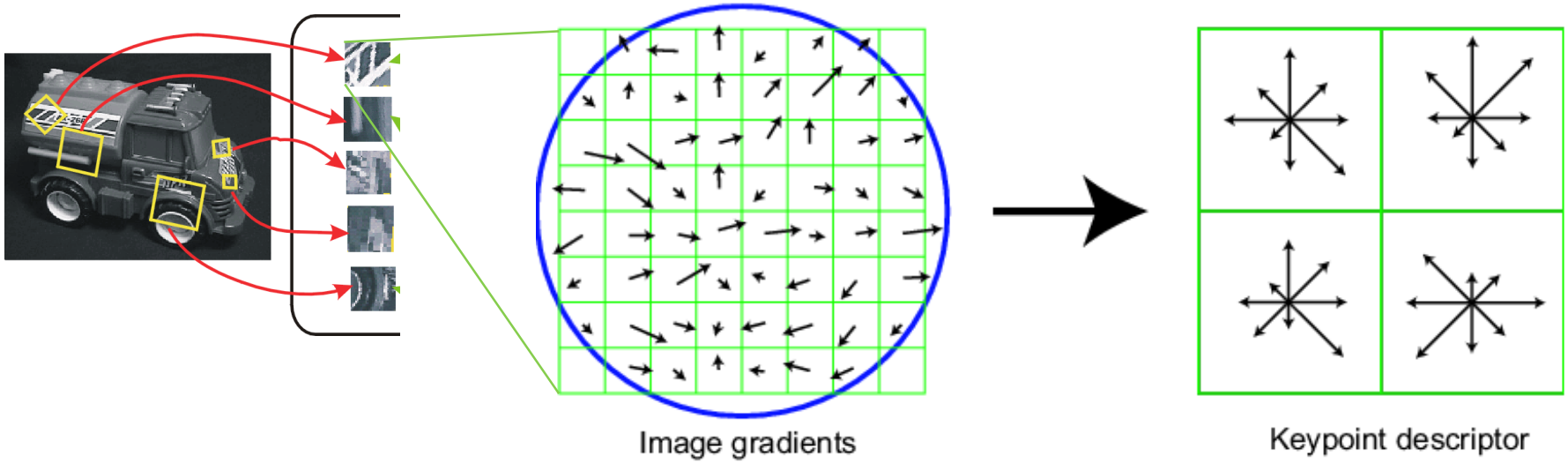


Image gradients

Actually 16x16, only showing 8x8

SIFT vector formation

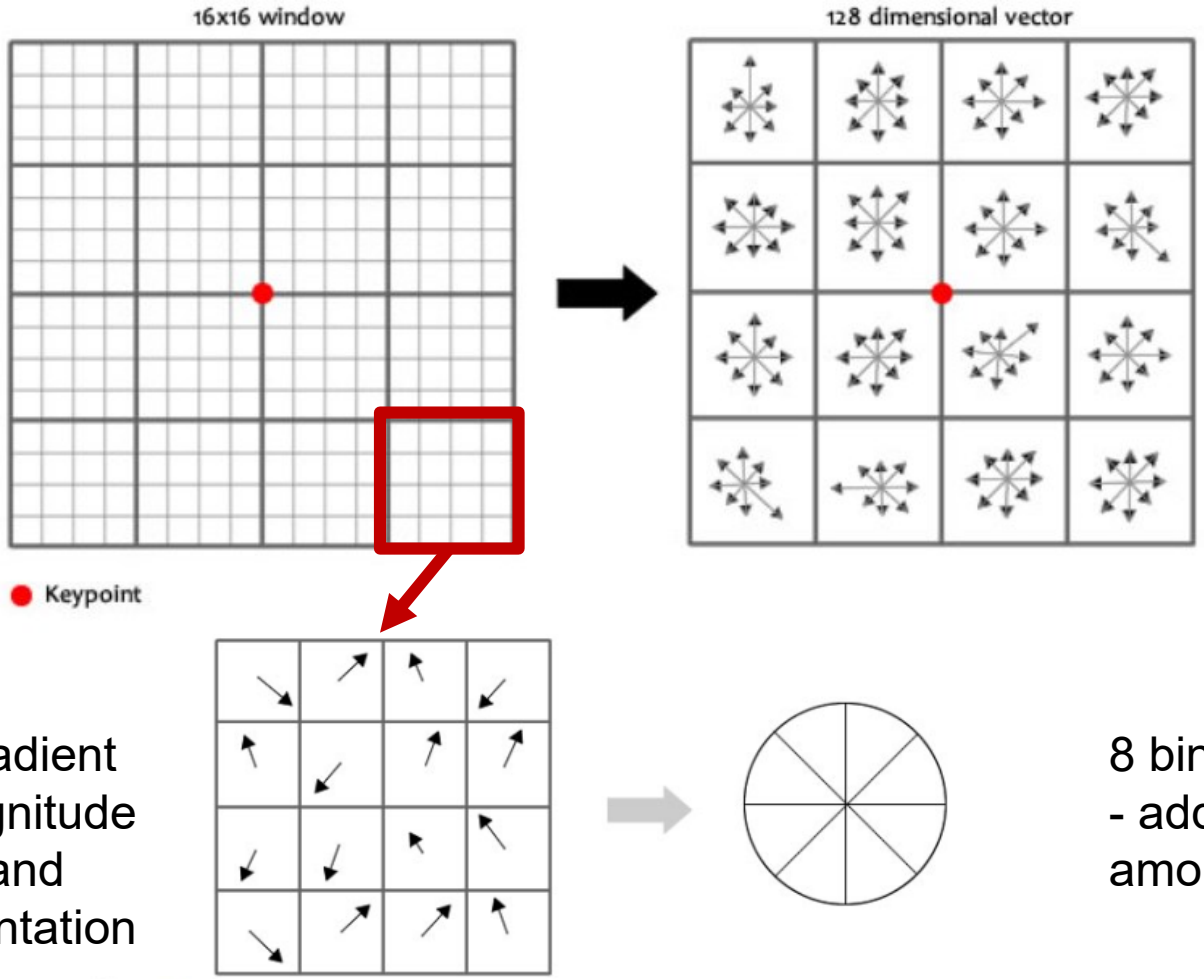
- 4x4 array of gradient orientation histograms weighted by gradient magnitude.
- Bin into 8 orientations x 4x4 array = 128 dimensions.



Showing only 2x2 here but is 4x4

SIFT Descriptor Extraction

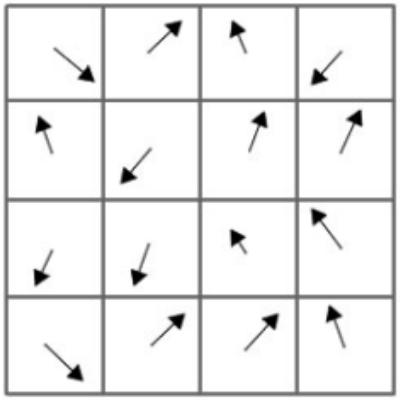
- Given a keypoint with scale and orientation

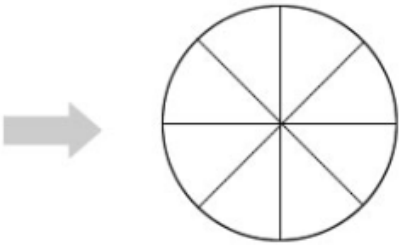


SIFT Descriptor Extraction

- Within each 4x4 window

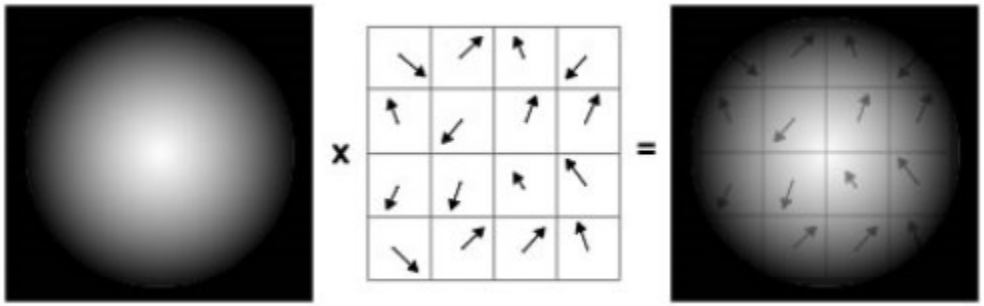
Gradient magnitude and orientation





8 bin ‘histogram’ - add magnitude amounts!

Weight magnitude that is added to ‘histogram’ by Gaussian

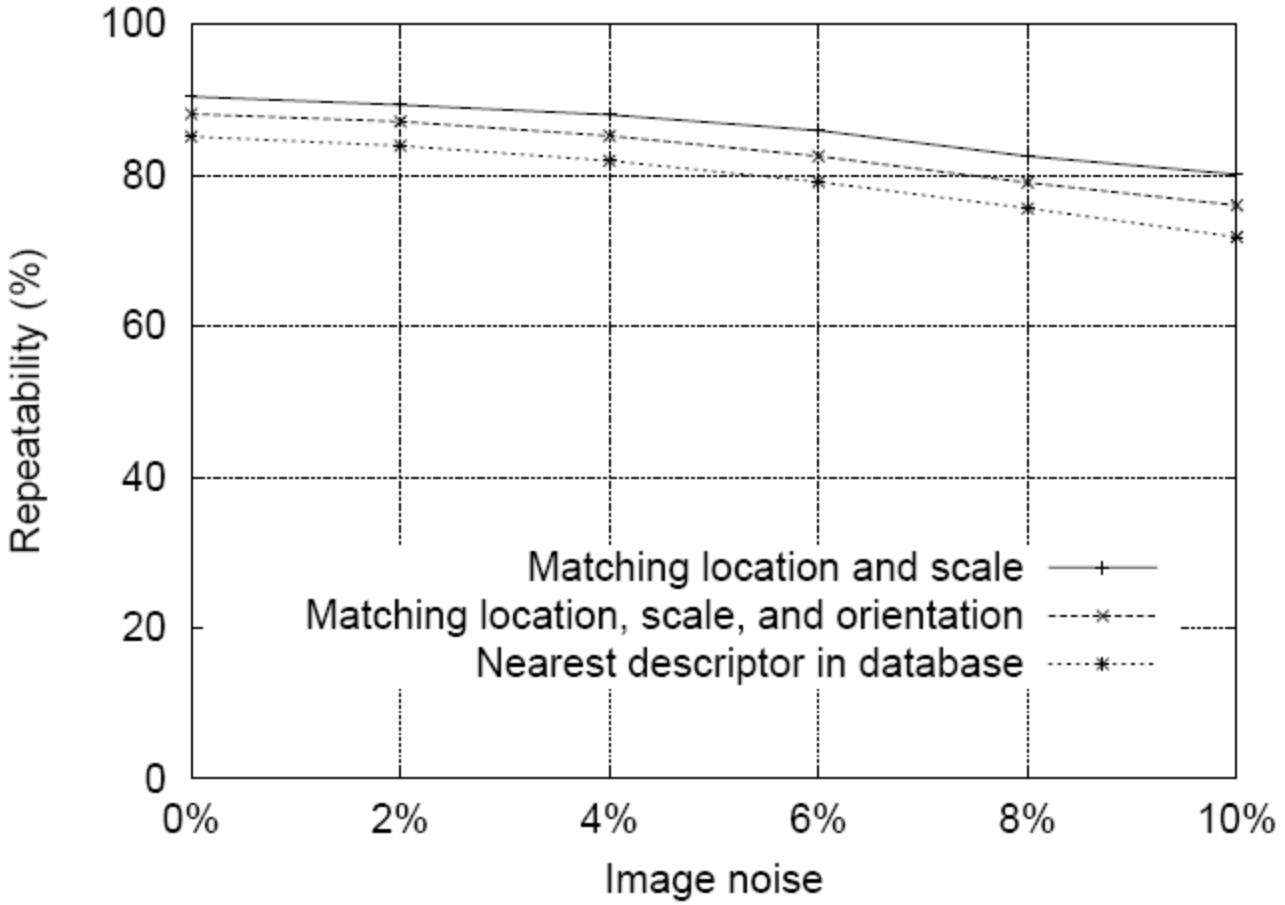


SIFT Descriptor Extraction

- 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

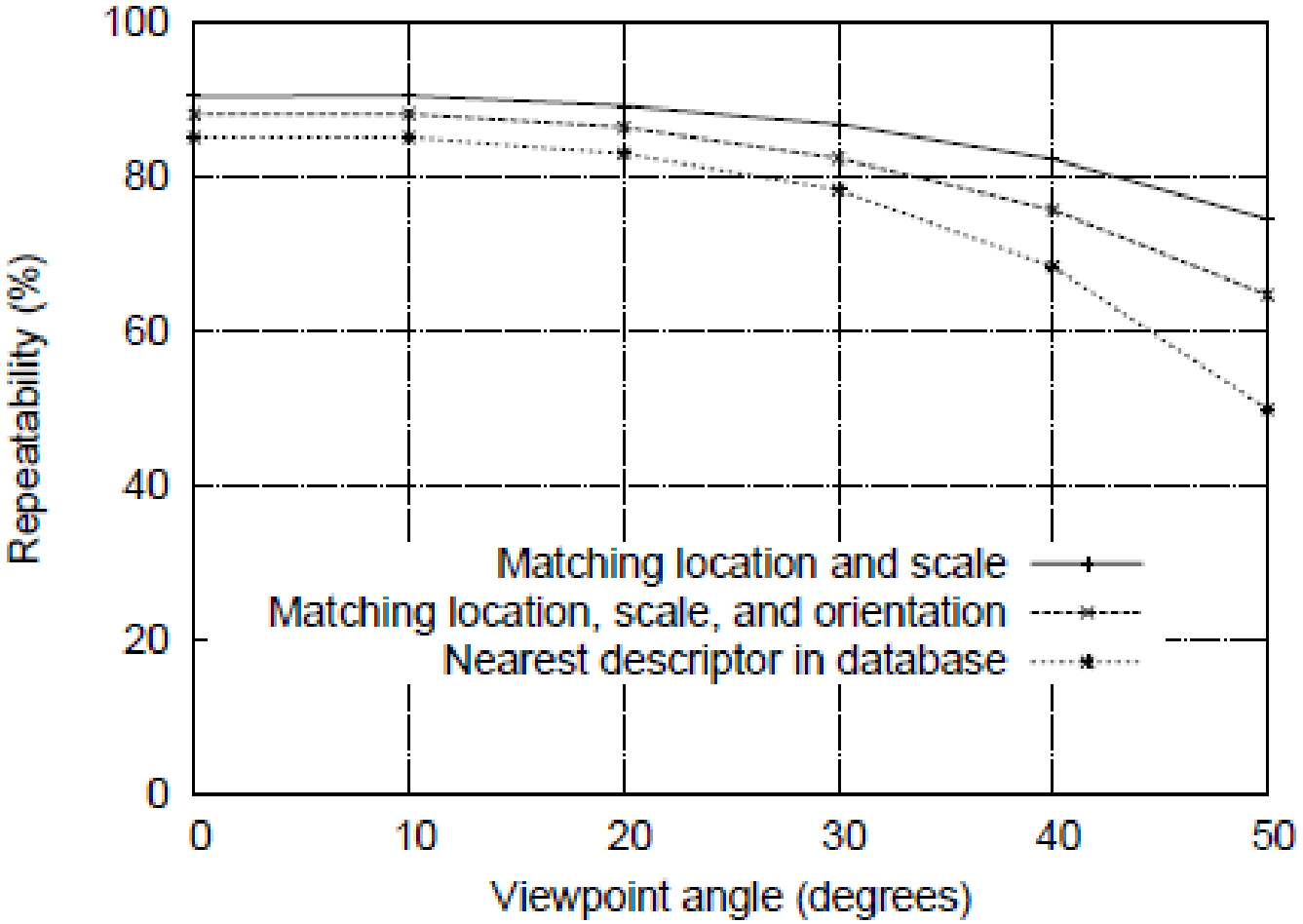
HOW GOOD IS SIFT?

SIFT Repeatability



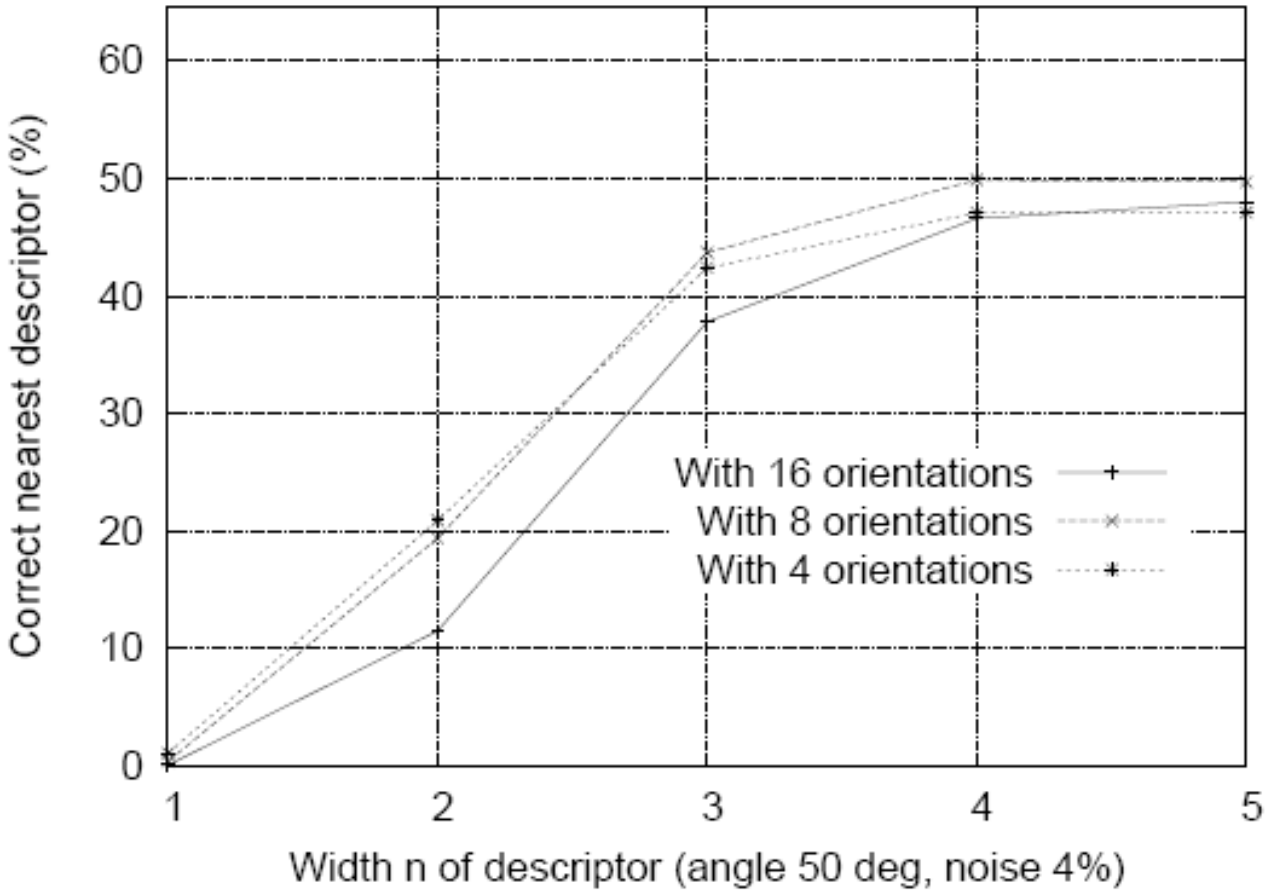
Lowe IJCV 2004

SIFT Repeatability



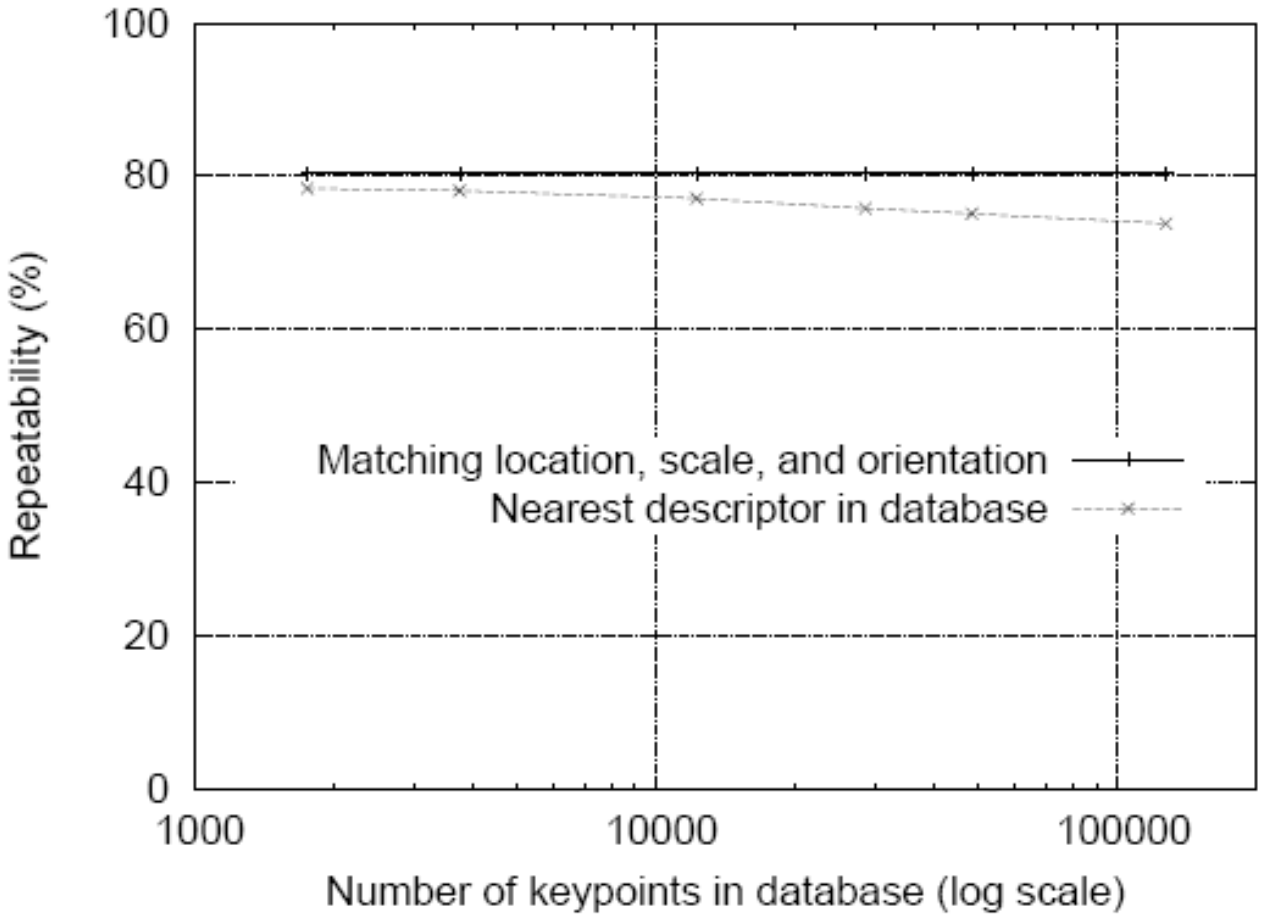
Lowe IJCV 2004

SIFT Repeatability

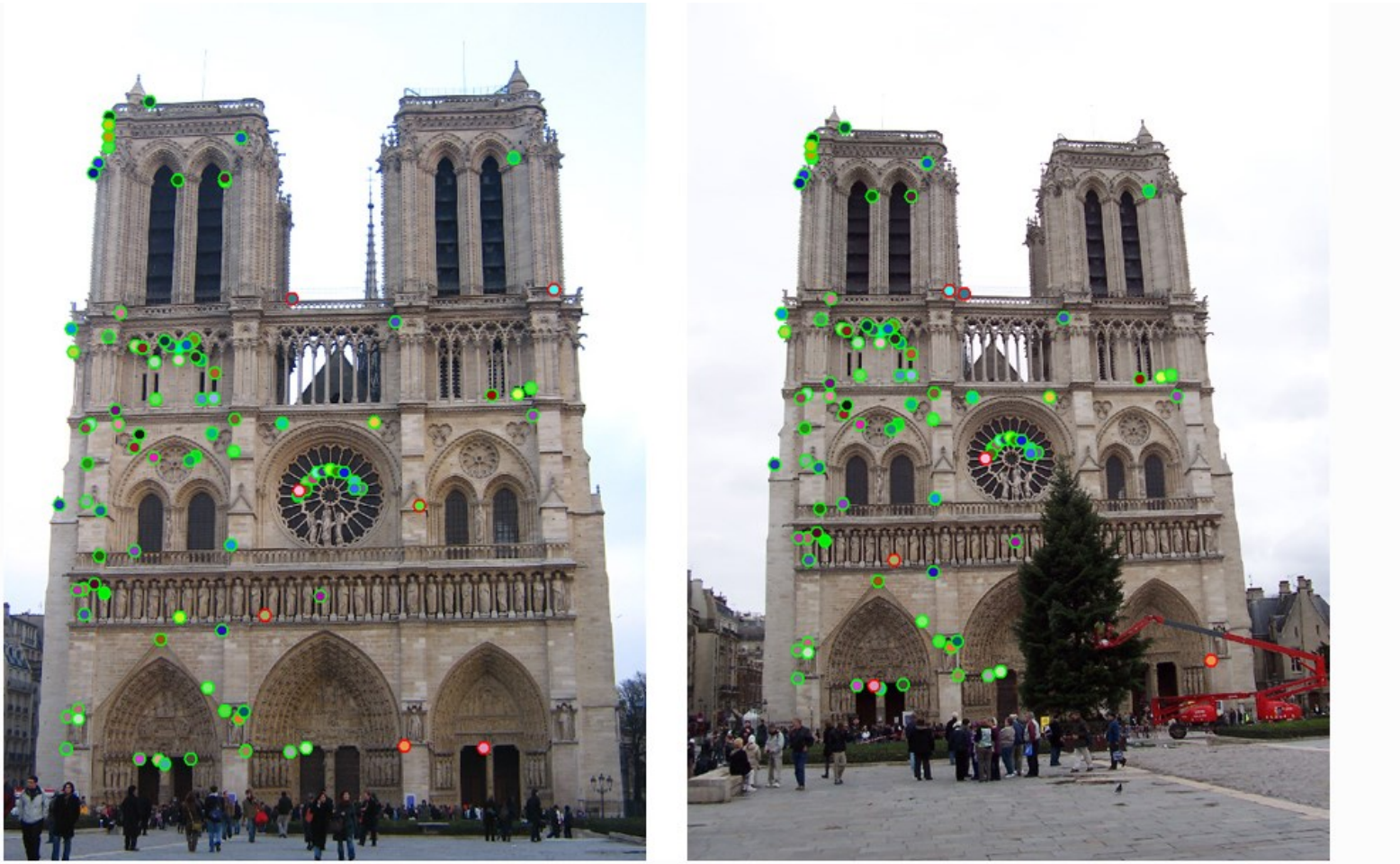


Lowe IJCV 2004

SIFT Repeatability



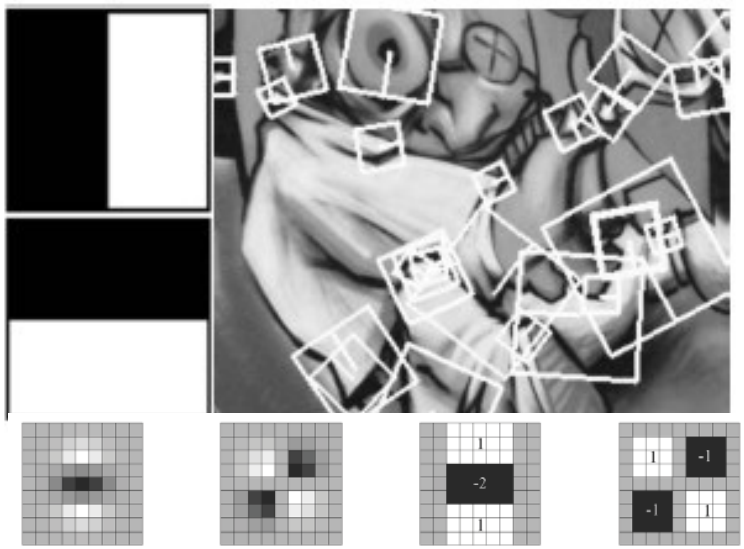
Lowe IJCV 2004



The top 100 most confident local feature matches from a baseline implementation of project 2. In this case, 93 were correct (highlighted in green) and 7 were incorrect (highlighted in red).

Project 2: Local Feature Matching

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]

Feature Matching

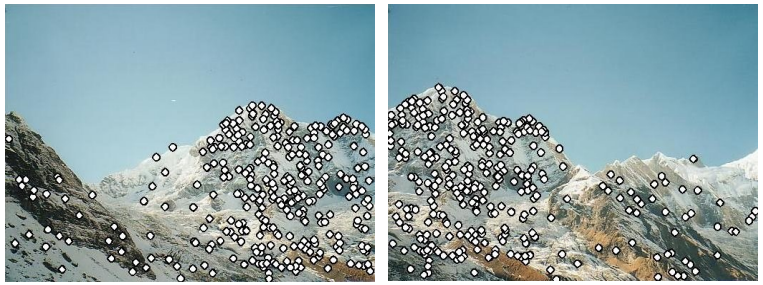
Read Szeliski 4.1

Many slides from James Hays, Derek Hoiem, and Grauman&Leibe 2008 AAAI Tutorial

Local features: main components

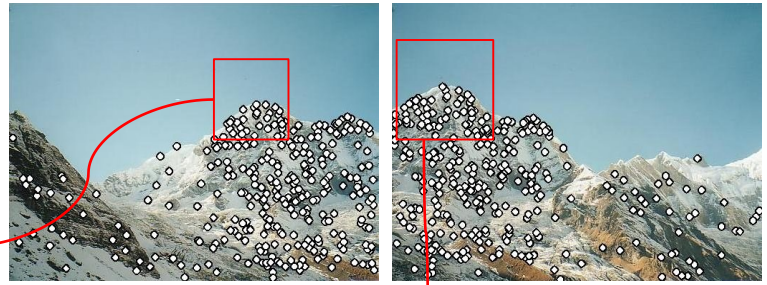
1) Detection:

Find a set of distinctive key points.



2) Description:

Extract feature descriptor around each interest point as vector.

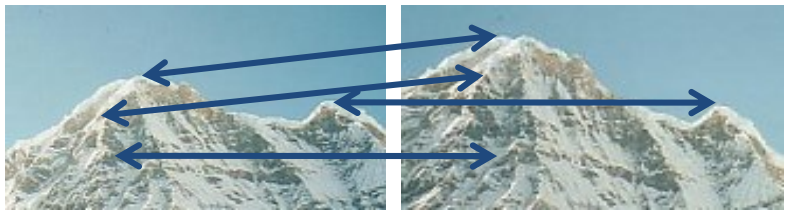


$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$

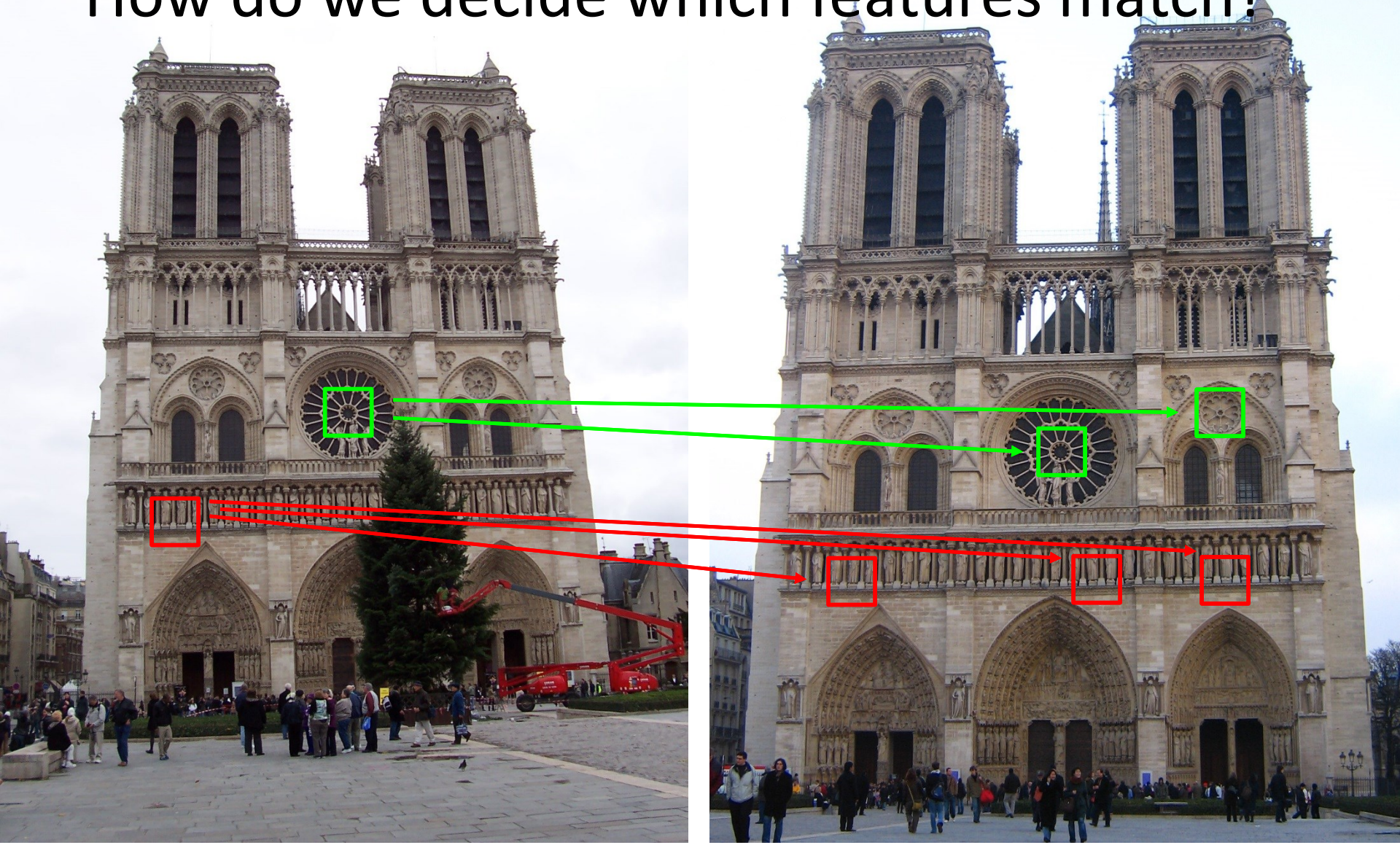
$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

3) Matching:

Compute distance between feature vectors to find correspondence.



How do we decide which features match?

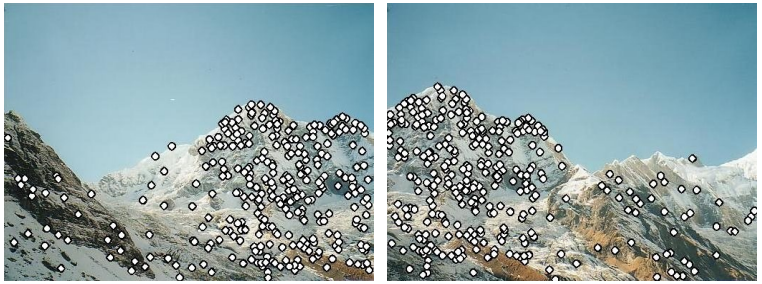


Distance: 0.34, 0.30, 0.40

Distance: 0.61, 1.22

Think-Pair-Share

- *Design a feature point matching scheme.*
- Two images, I_1 and I_2



- Two sets X_1 and X_2 of feature points
  - Each feature point $\mathbf{x_1}$ has a descriptor $\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$

- Distance, bijective/injective/surjective, noise, confidence, computational complexity, generality…

Euclidean distance vs. Cosine Similarity

- Euclidean distance:

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

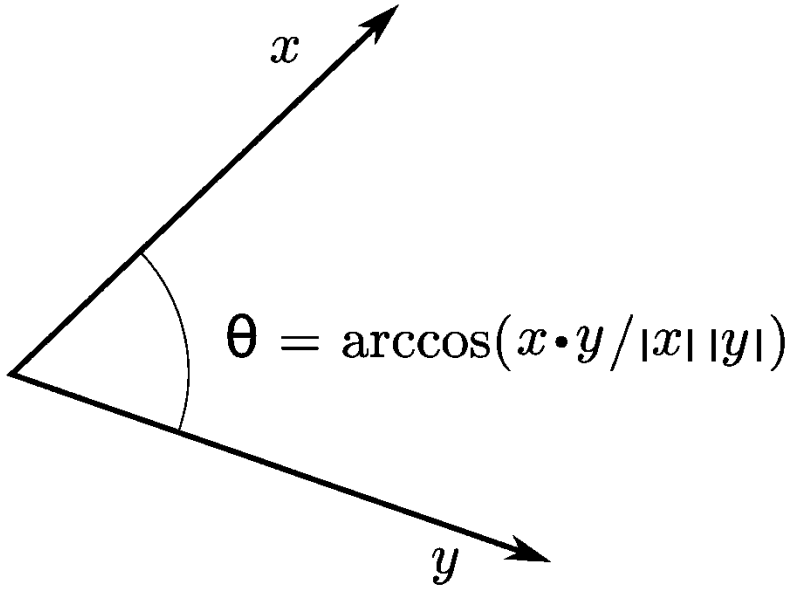
$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

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

- Cosine similarity:

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

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



Wikipedia

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.

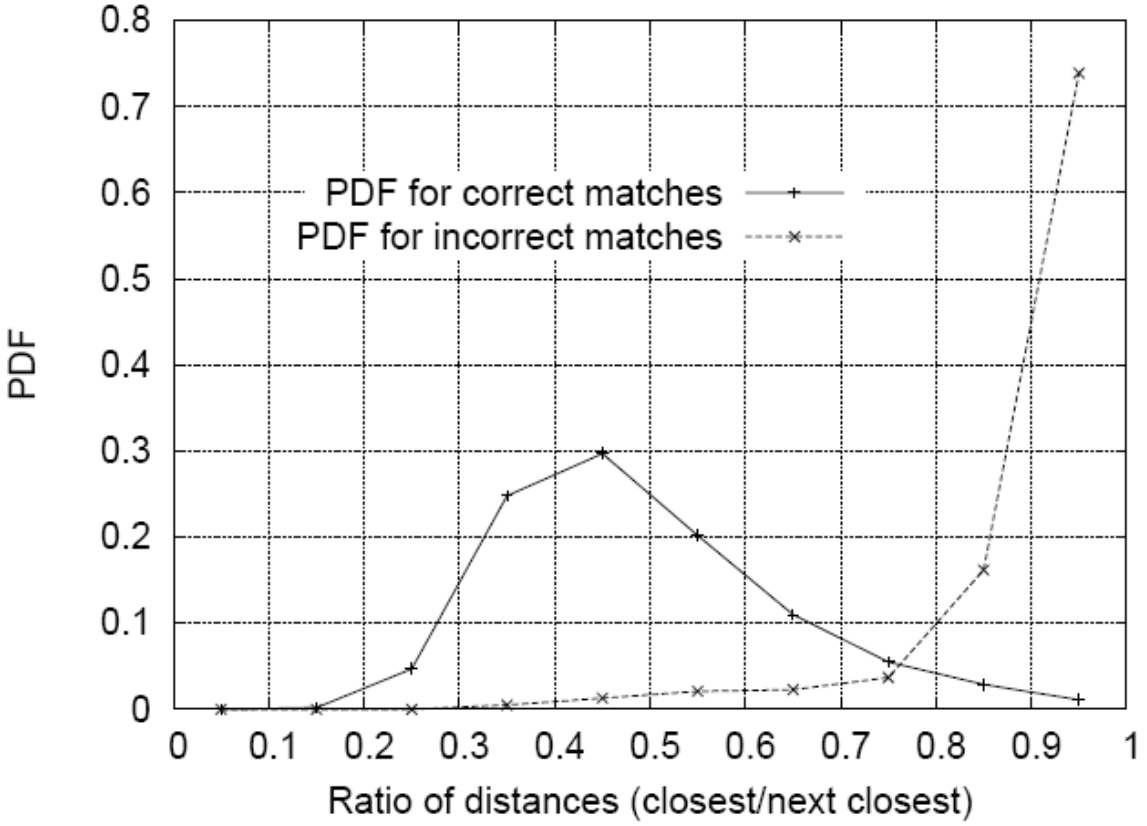
- If NN1 ≈ NN2, ratio $\frac{NN1}{NN2}$ will be ≈ 1 -> matches too close.
- As NN1 << NN2, ratio $\frac{NN1}{NN2}$ tends to 0.

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
- 40,000 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!

Lowe IJCV 2004

Efficient compute cost

- Naïve looping: Expensive

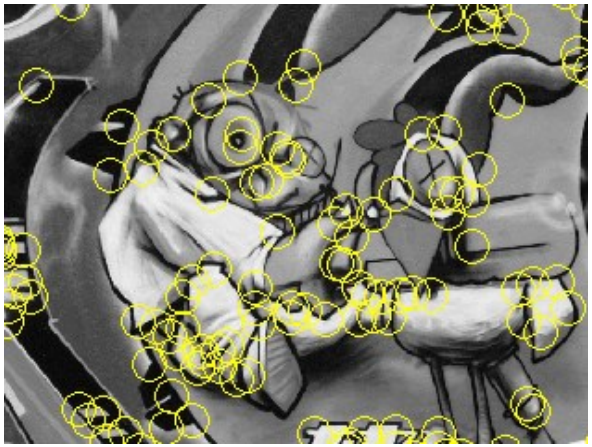
- Operate on matrices of descriptors
- E.g., for row vectors,

$$\texttt{features\_image1} * \texttt{features\_image2}^{T}$$

produces matrix of dot product results for all pairs of features

Review: Interest points

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



- Descriptors: robust and selective
  - Spatial histograms of orientation
  - SIFT