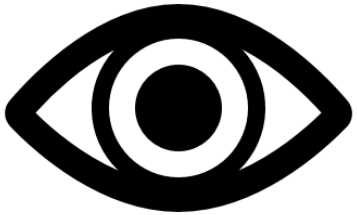


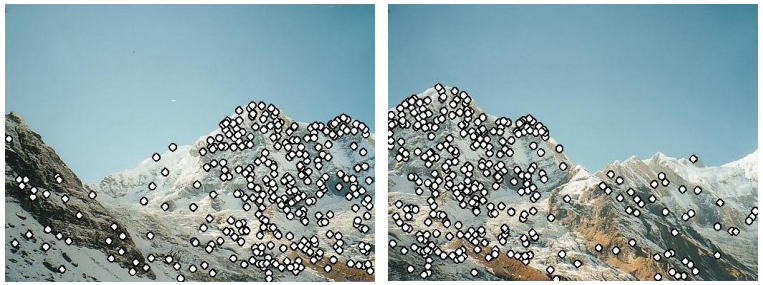
Advanced Image Processing

Local Image Features, Descriptors and Matching

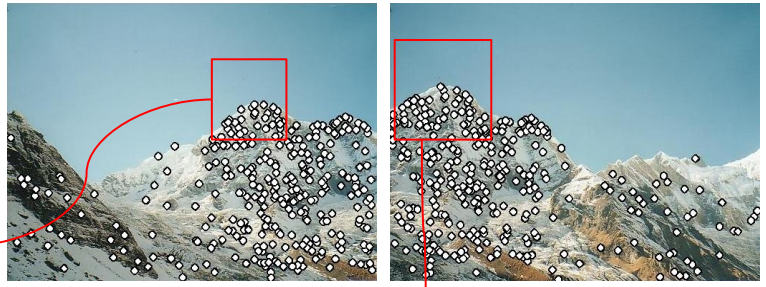
SYDE 671

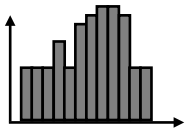
Local features: main components

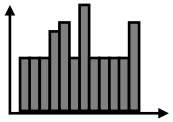
1) Detection:
Find a set of distinctive key points.



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

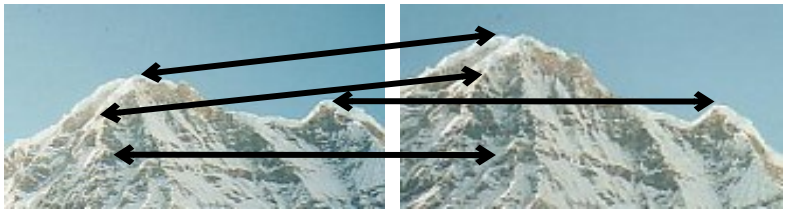


x_1  $\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$

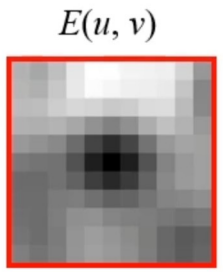
x_2  $\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

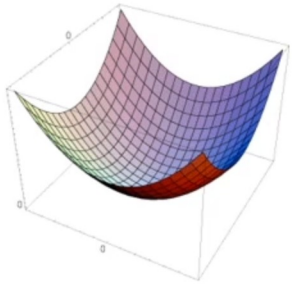
3) Matching:
Compute distance between feature vectors to find correspondence.

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

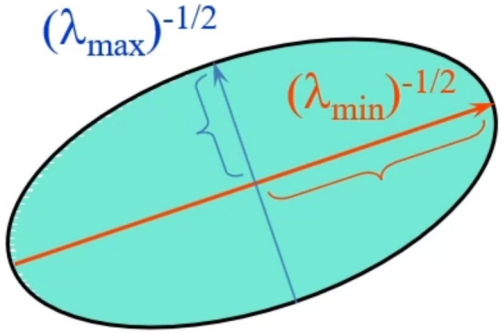


Review: Harris corner detector





- 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**.



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

Trace / determinant and eigenvalues

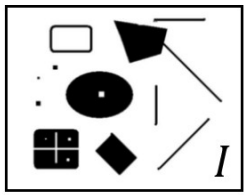
- Given n x n matrix A with eigenvalues $\lambda_{1...n}$

$$\operatorname{tr}(A) = \sum_{i=1}^{n} A_{ii} = \sum_{i=1}^{n} \lambda_i = \lambda_1 + \lambda_2 + \cdots + \lambda_n.$$

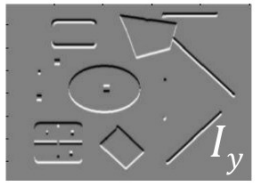
$$\det(A) = \prod_{i=1}^{n} \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_n.$$

- Cornerness Measure =
  $R = \lambda_1\lambda_2 - \alpha(\lambda_1+\lambda_2)^2 = \det(M) - \alpha\,\mathrm{trace}(M)^2$

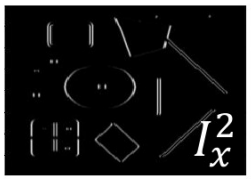
Harris Corner Detector [Harris88]

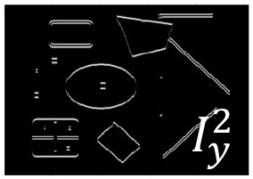


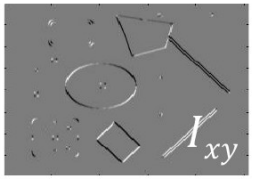
0. Input image
   We want to compute M at each pixel.



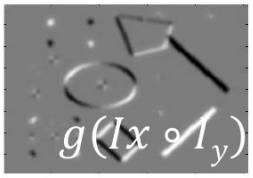
1. Compute image derivatives (optionally, blur first).







2. Compute M components
   as squares of derivatives.



3. Gaussian filter *g()* with width σ

4. Compute cornerness

$$C = \det(M) - \alpha \operatorname{trace}(M)^2$$
$$= g(I_x^2) \circ g(I_y^2) - g(I_x \circ I_y)^2 - \alpha\left[g(I_x^2) + g(I_y^2)\right]^2$$

5. Threshold on C to pick high cornerness

6. Non-maxima suppression to pick peaks.

Invariance and covariance

Are locations *invariant* to photometric transformations and *covariant* to geometric transformations?

- **Invariance:** image is transformed and corner locations do not change
- **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

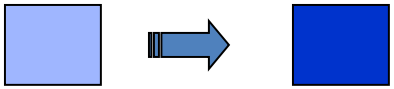
好像是Invariant就是不管图片怎么变，变之前检测到的feature在哪个位置，变之后就还在那个位置。

Covariant就是图片变了，但是变之前找到的feature变之后还能找到，虽然位置可能会不一样。

HOW INVARIANT ARE HARRIS CORNERS?

Affine intensity change

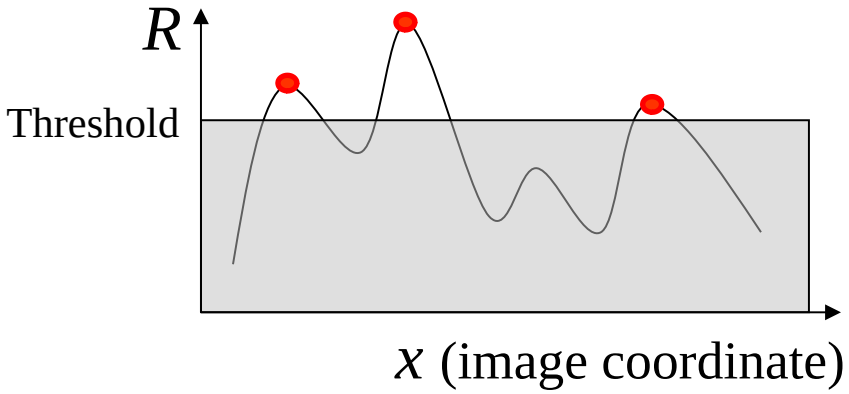
Linear

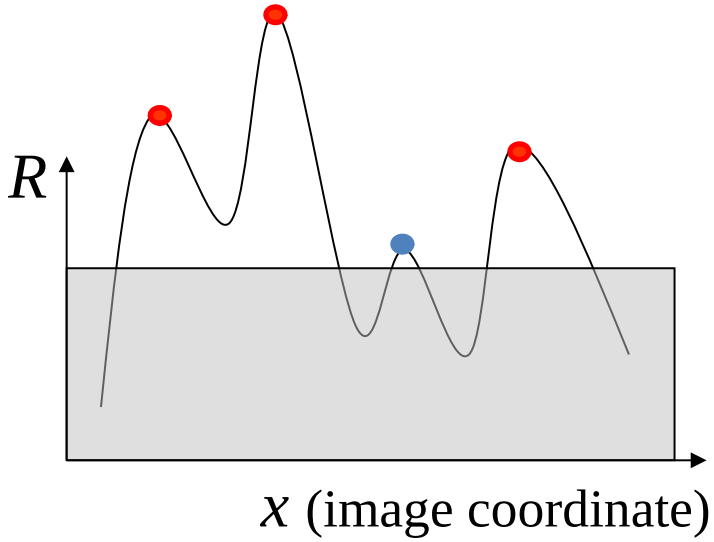


$$I \rightarrow a\,I + b$$

- Only derivatives are used so:
  invariance to intensity shift $I \rightarrow I + b$

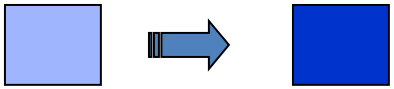
- Intensity scaling: $I \rightarrow a\,I$





? to affine intensity change

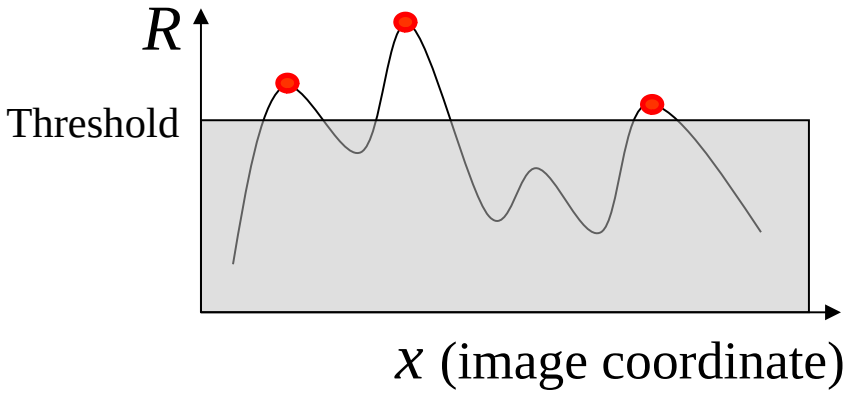
Affine intensity change

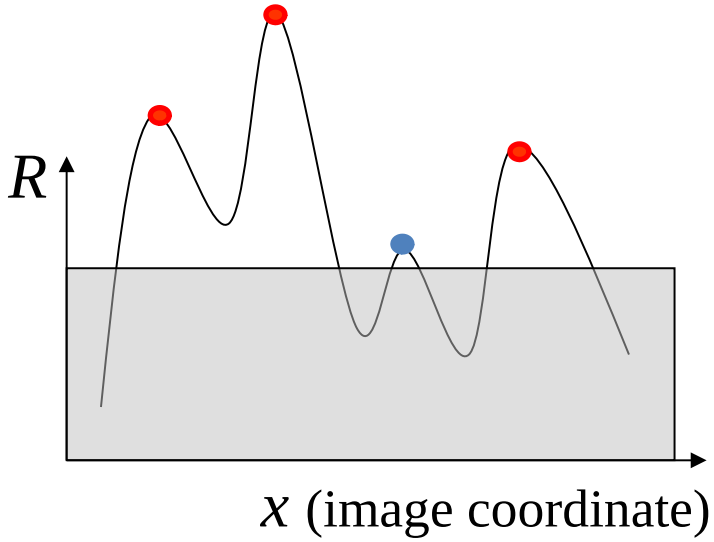


$$I \rightarrow a\,I + b$$

- Only derivatives are used so:
  invariance to intensity shift $I \rightarrow I + b$

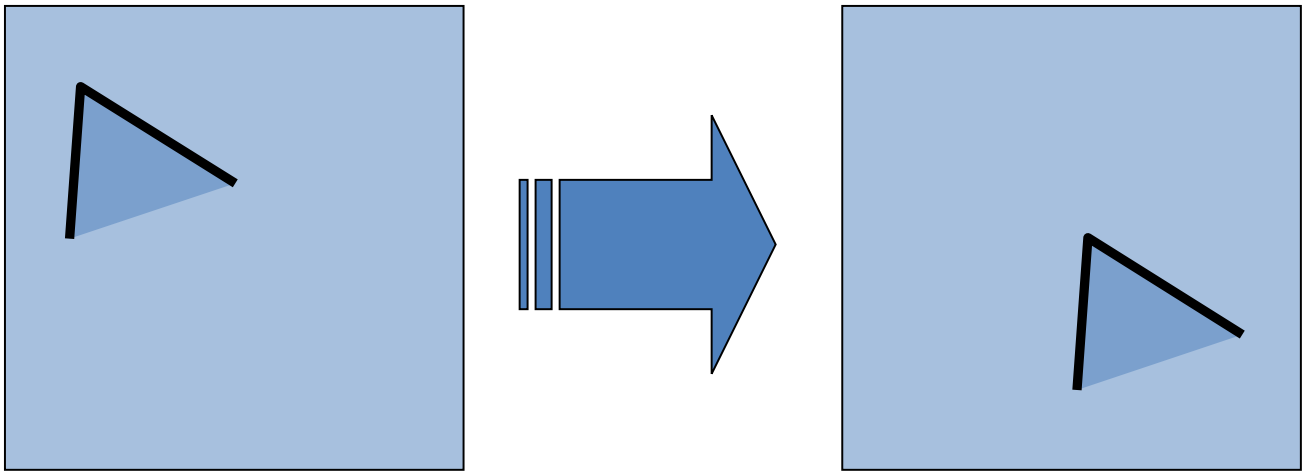
- Intensity scaling: $I \rightarrow a\,I$





Partially invariant to affine intensity change

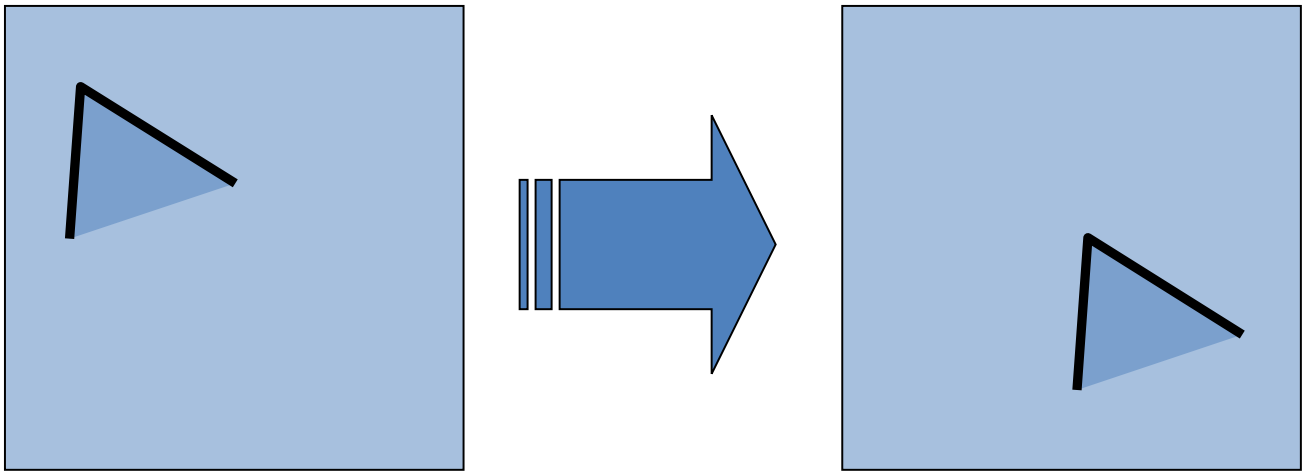
Image translation



- Derivatives and window function are shift-invariant.

Corner location is ? w.r.t. translation

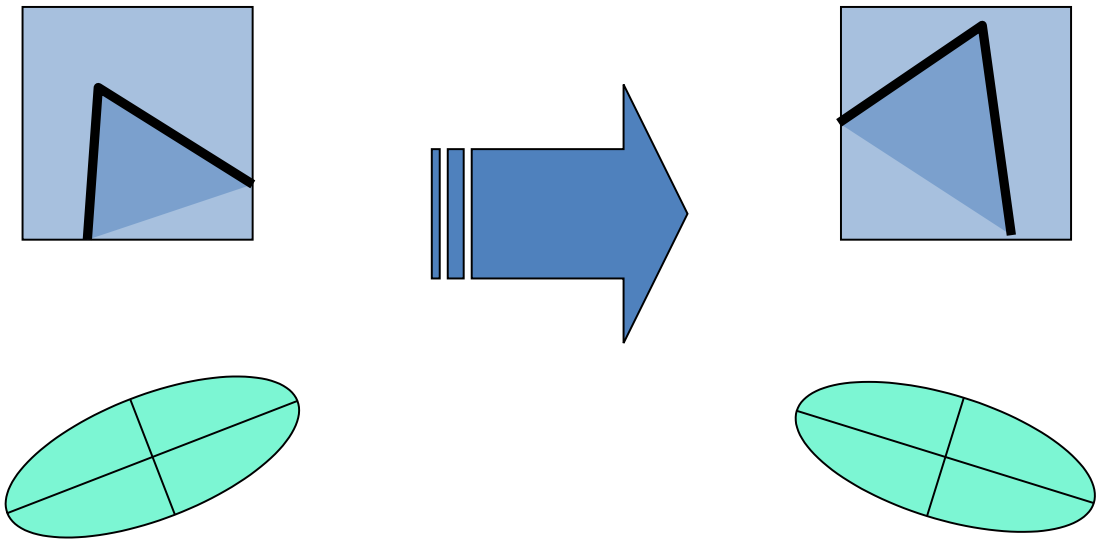
Image translation



- Derivatives and window function are shift-invariant.

Corner location is covariant w.r.t. translation

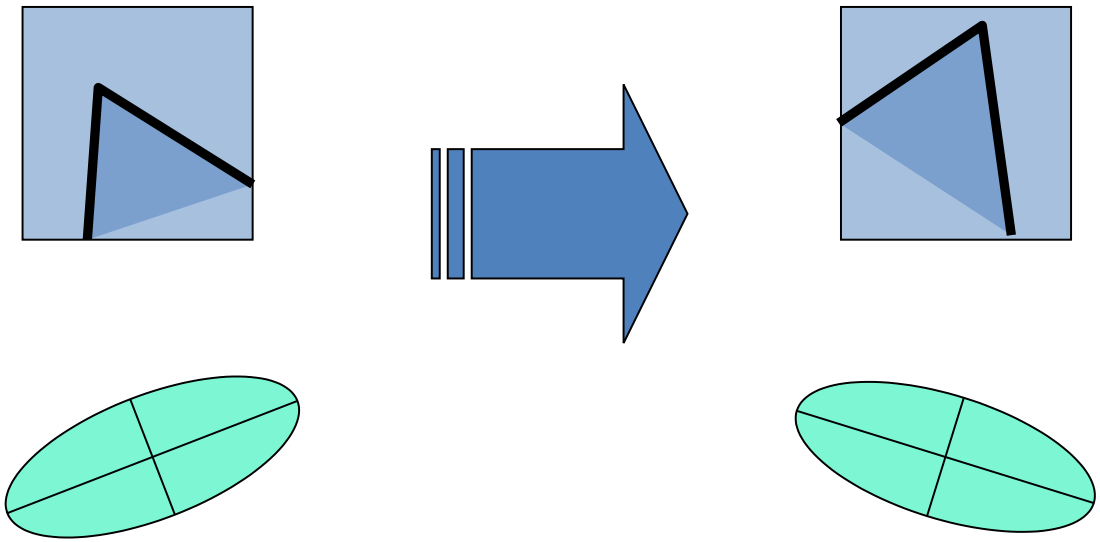
Image rotation



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

Corner location is ? w.r.t. rotation

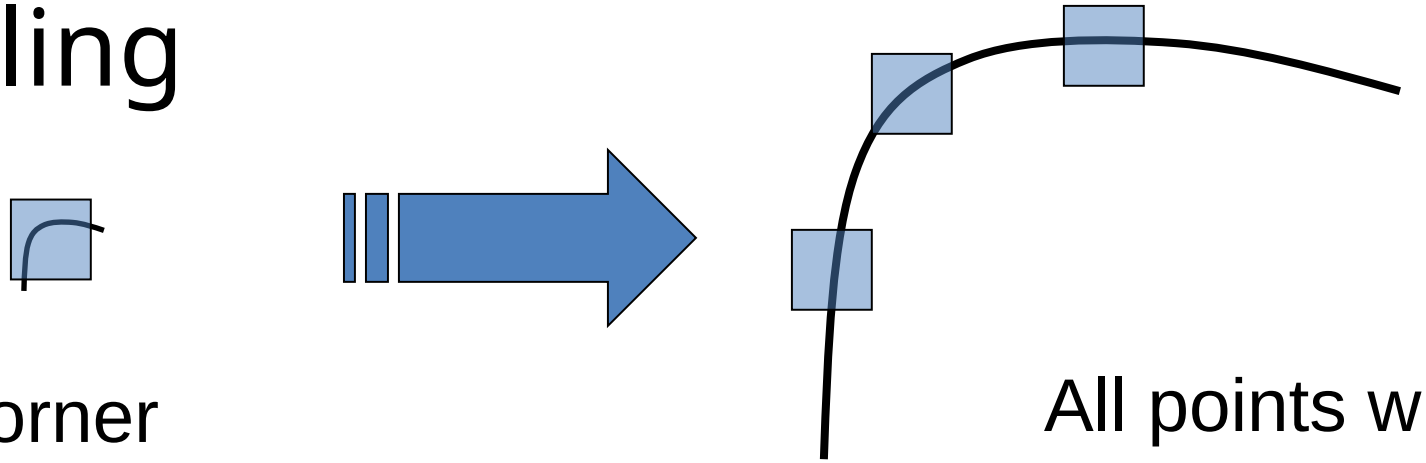
Image rotation



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

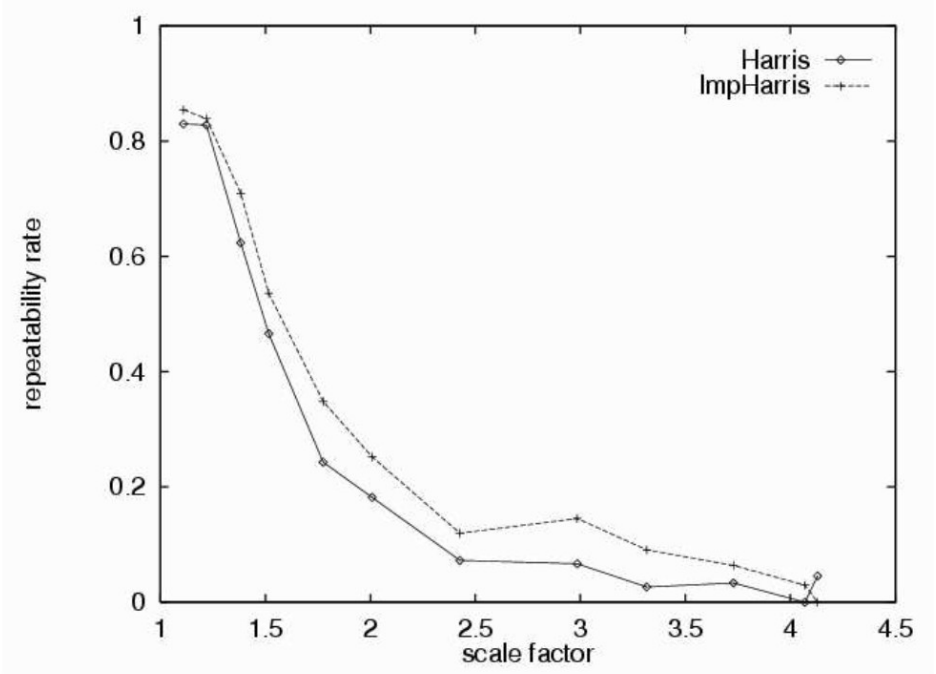
Corner location is covariant w.r.t. rotation

Scaling



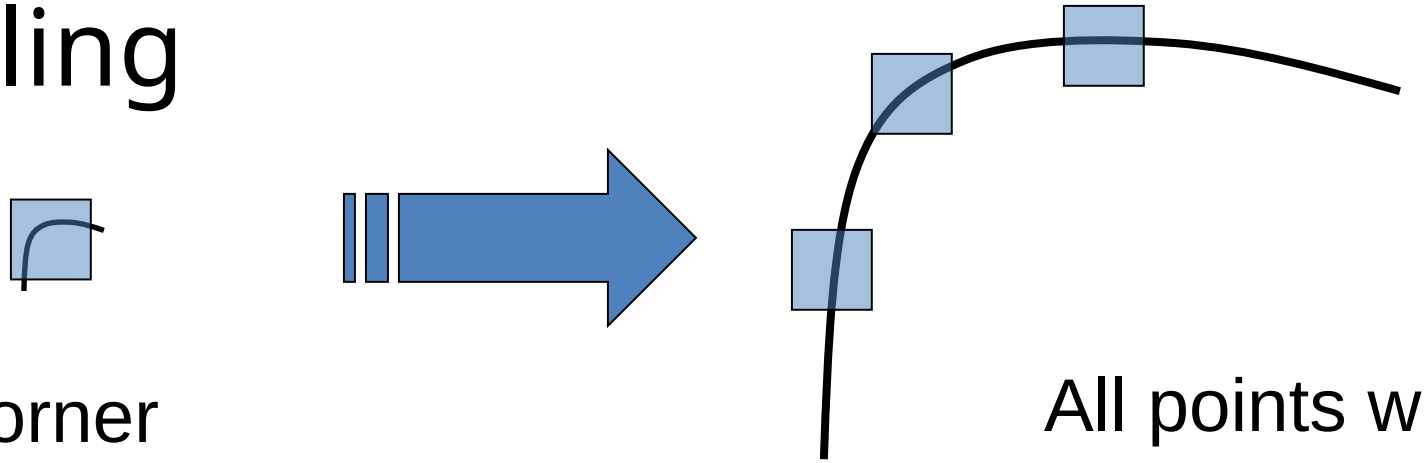
Corner

All points will be incorrectly classified as edges



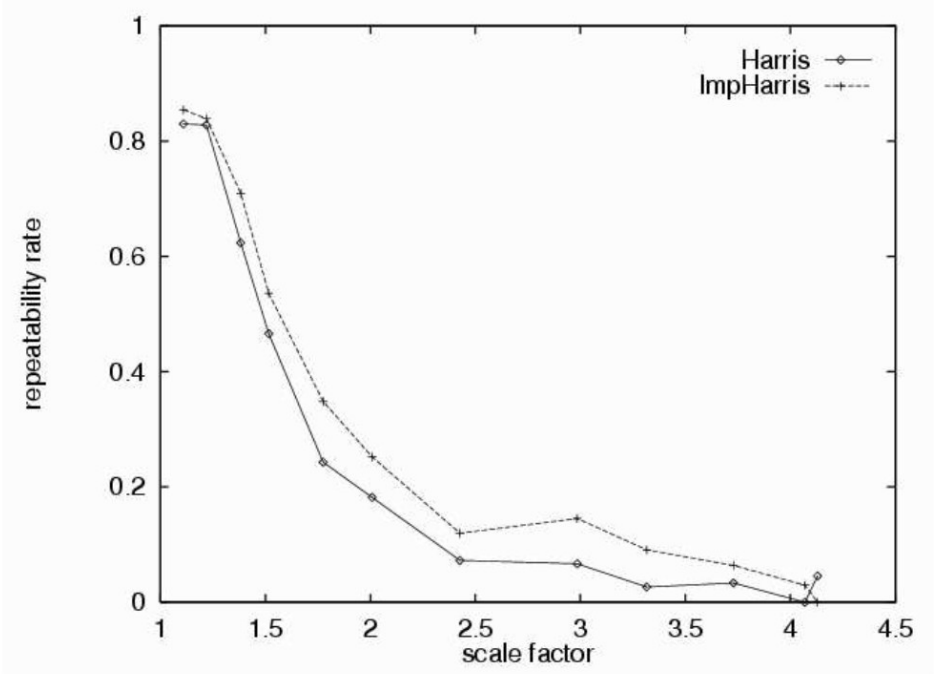
Corner location is ? to scaling

Scaling



Corner

All points will be incorrectly classified as edges



Corner location is not covariant nor invariant to scaling!

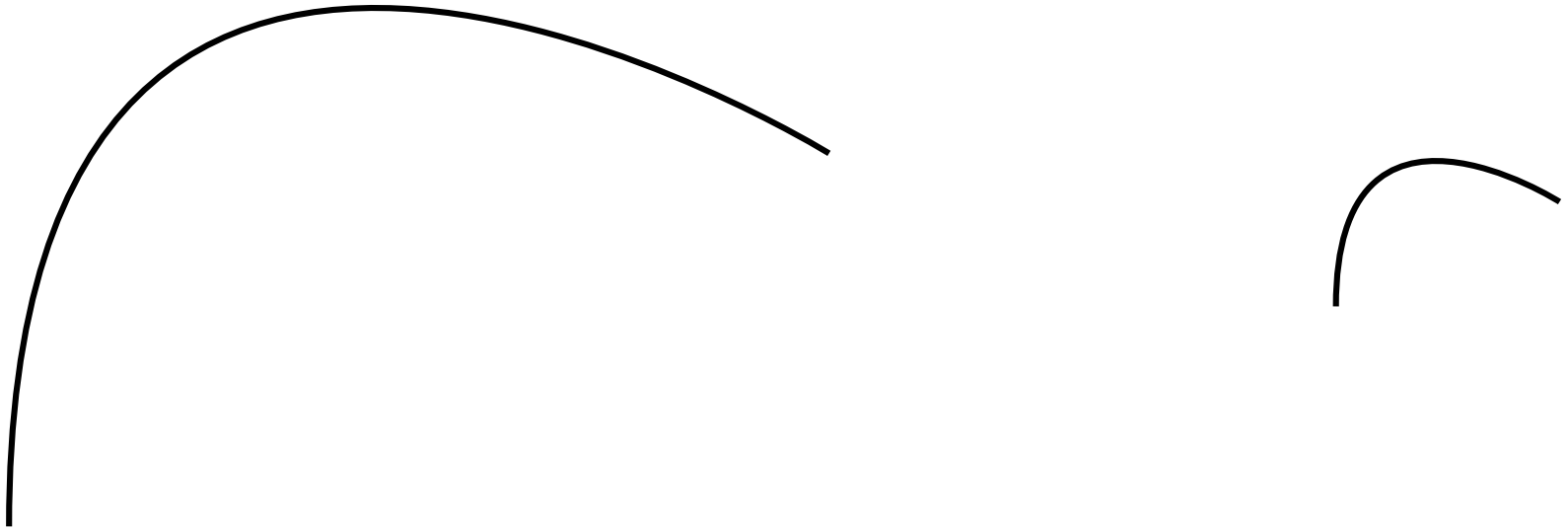
WHAT IS THE ‘SCALE’ OF A FEATURE POINT?

Does Scale Matter?

- When detecting corners, the `scale` of the window you use can change the corners you detect.

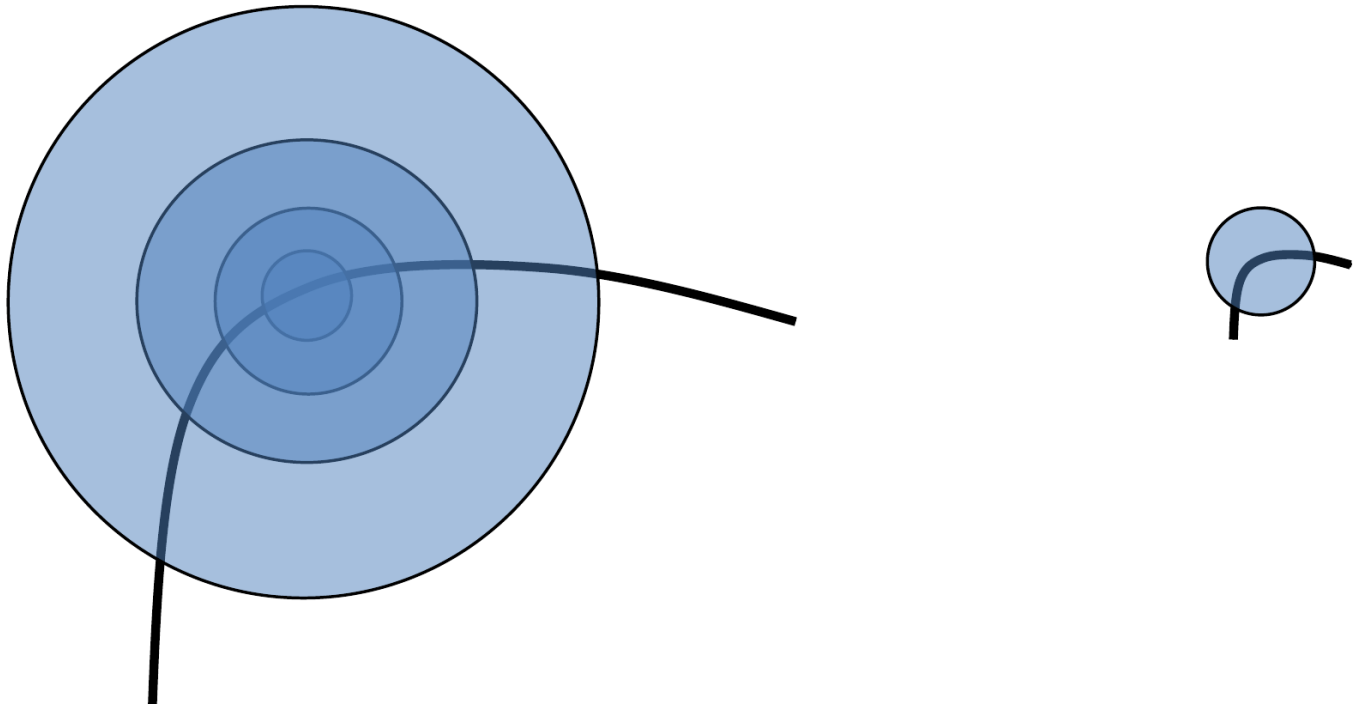
Scale Invariant Detection - Mindset

How can I link these?



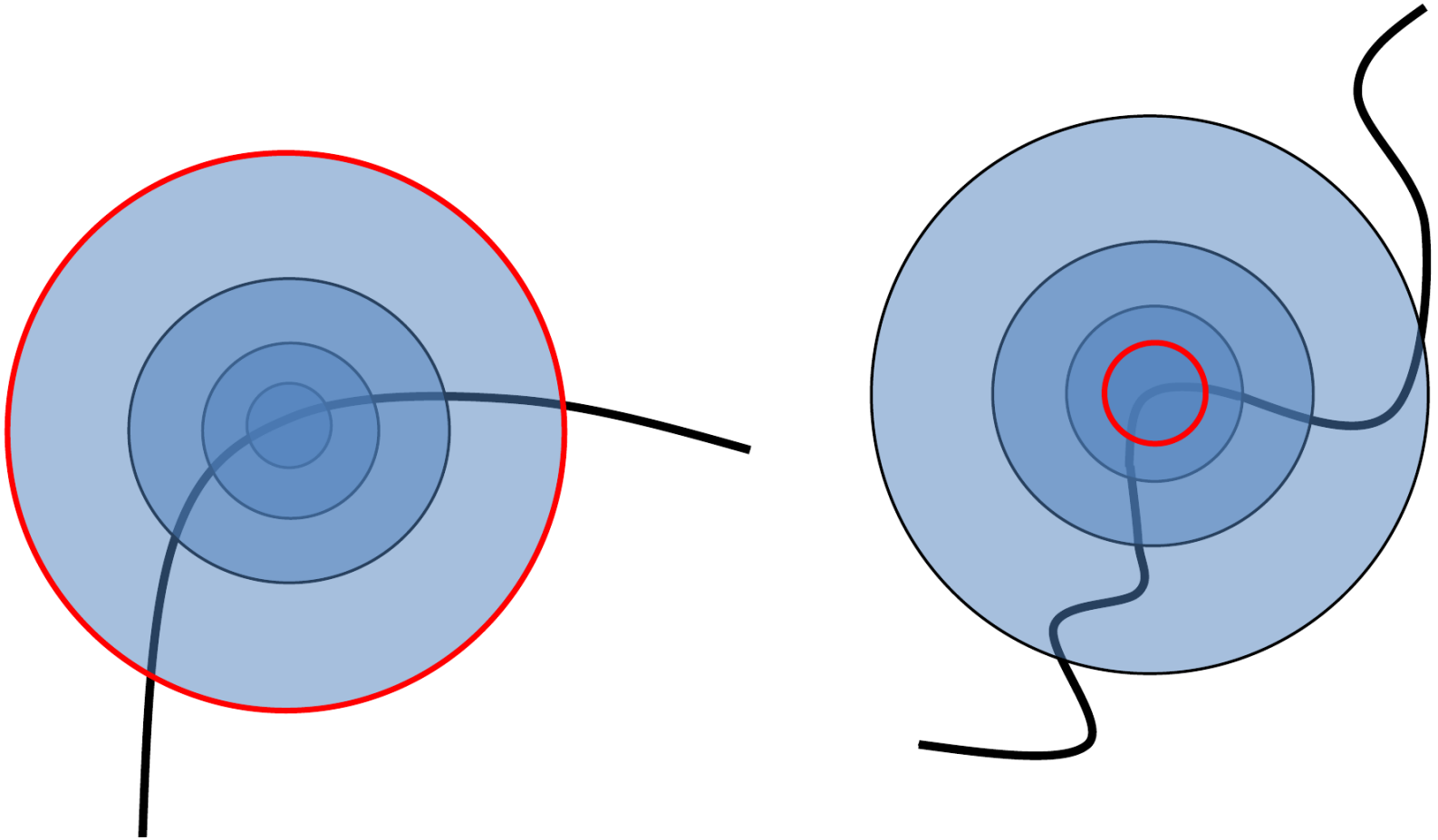
Scale Invariant Detection - Mindset

- Consider regions (e.g. circles) of different sizes around a point
- Find regions of corresponding sizes that will look the same in both images?



Scale Invariant Detection - Mindset

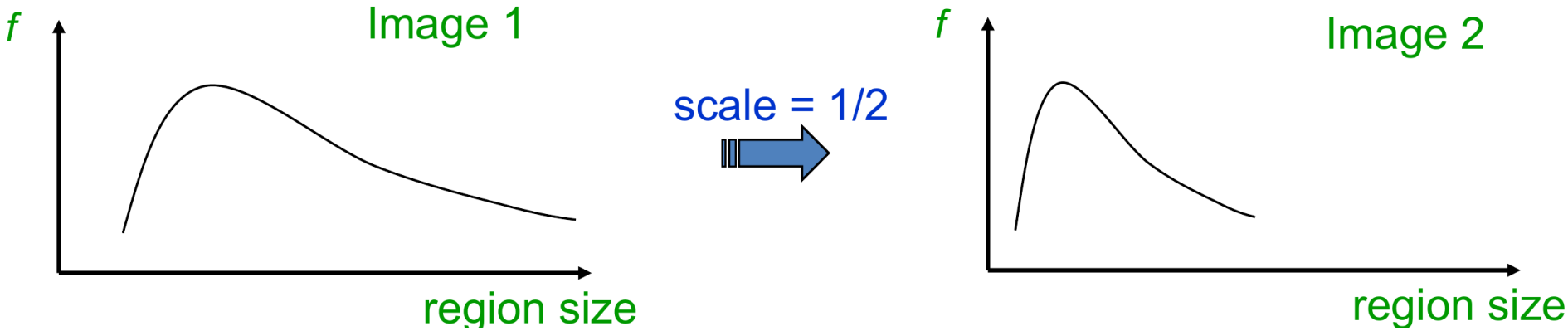
- The problem: how do we choose corresponding circles ***independently*** in each image?



Scale Invariant Detection

- Solution:
  - Design a function on the region (circle), which is "scale invariant" (the same for corresponding regions, even if they are at different scales)

    Example: average intensity. For corresponding regions (even of different sizes) it will be the same.
  - For a point in one image, we can consider it as a function of region size (circle radius)



Automatic Scale Selection

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

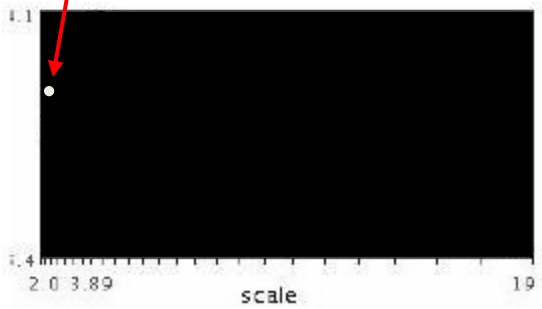
How to find patch sizes at which *f* response is equal?

What is a good *f* ?

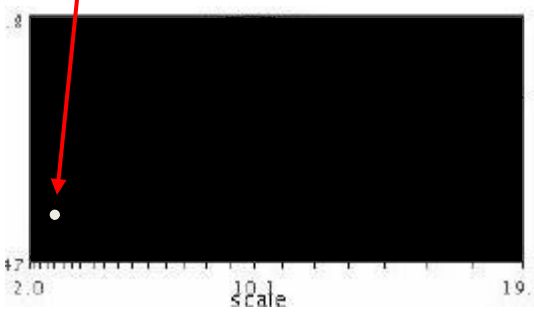
Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Response of some function *f*



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

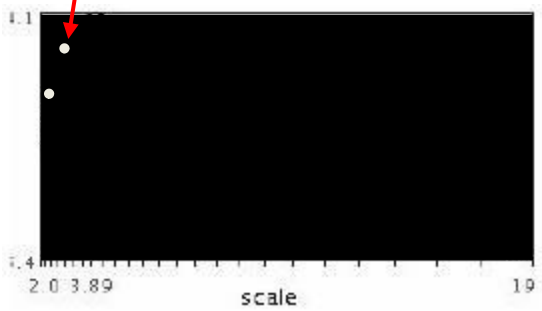


$f(I_{i_1 \ldots i_m}(x',\sigma'))$

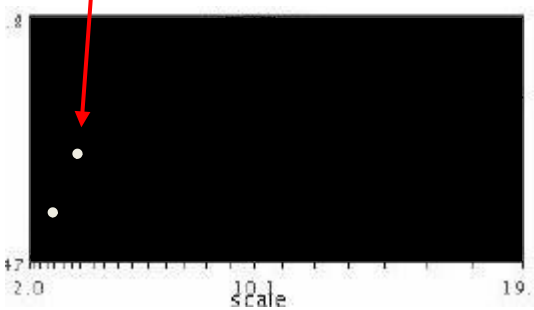
Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Response of some function *f*



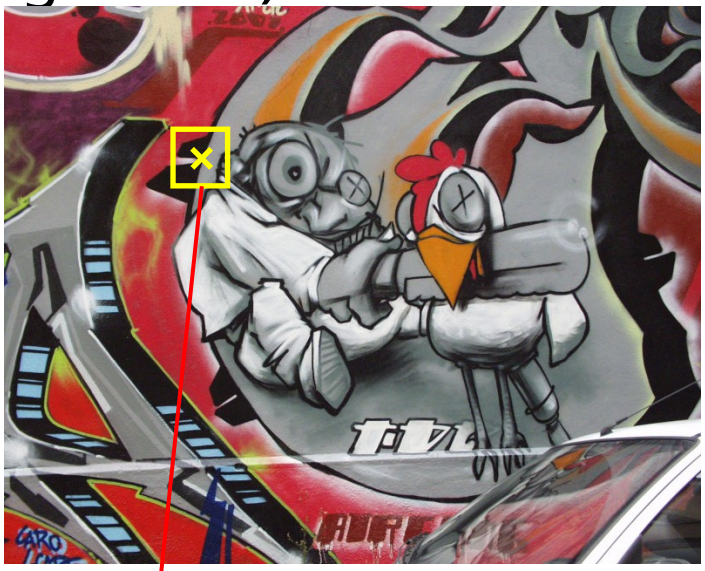
$f(I_{i_1 \ldots i_m}(x,\sigma))$



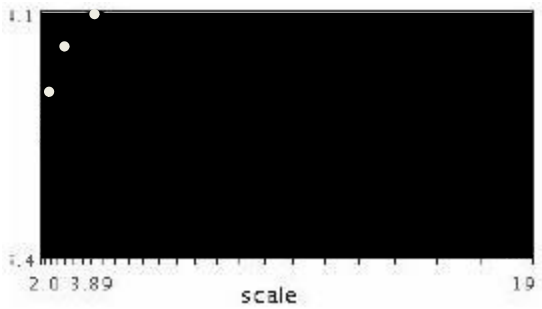
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Automatic Scale Selection

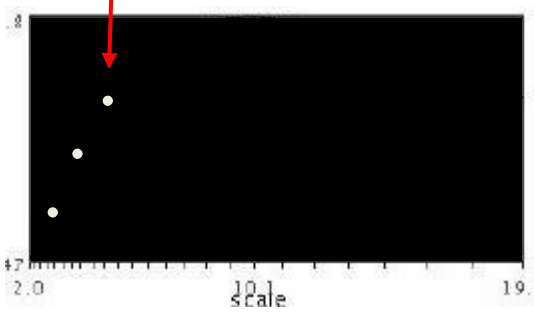
- Function responses for increasing scale (scale signature)



Response of some function *f*



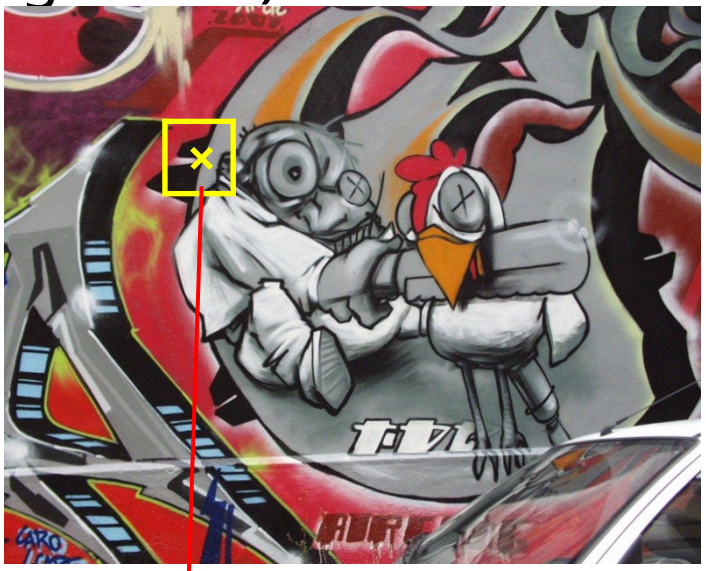
$f(I_{i_1 \ldots i_m}(x,\sigma))$



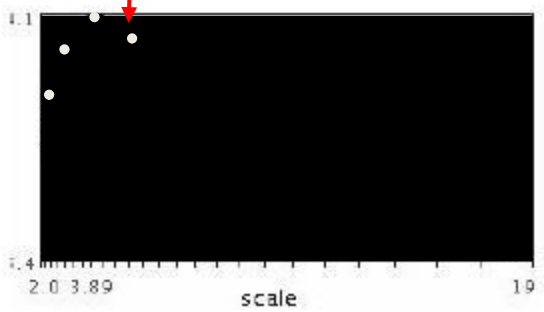
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Automatic Scale Selection

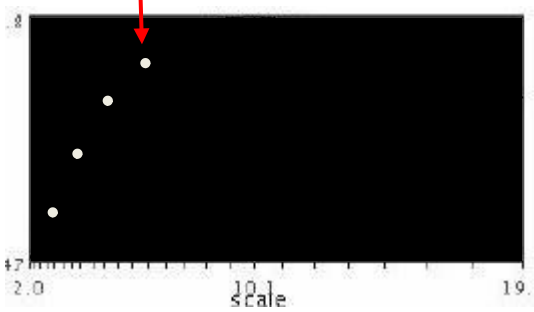
- Function responses for increasing scale (scale signature)



Response of some function *f*



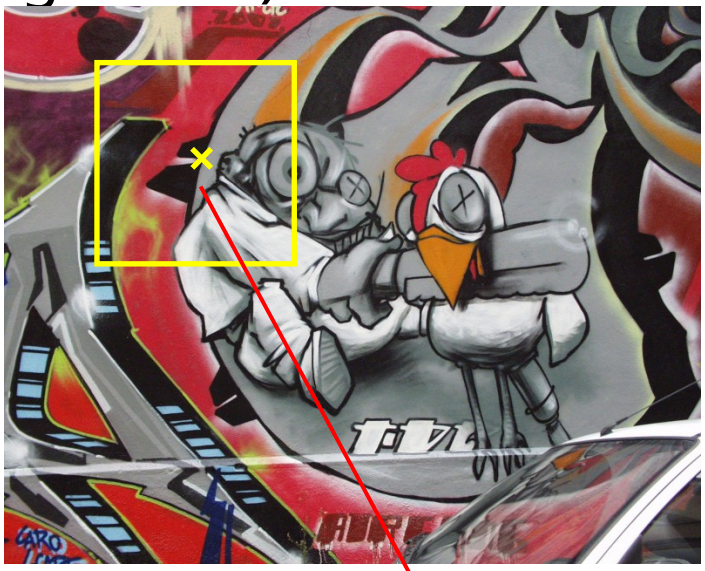
$f(I_{i_1 \ldots i_m}(x,\sigma))$

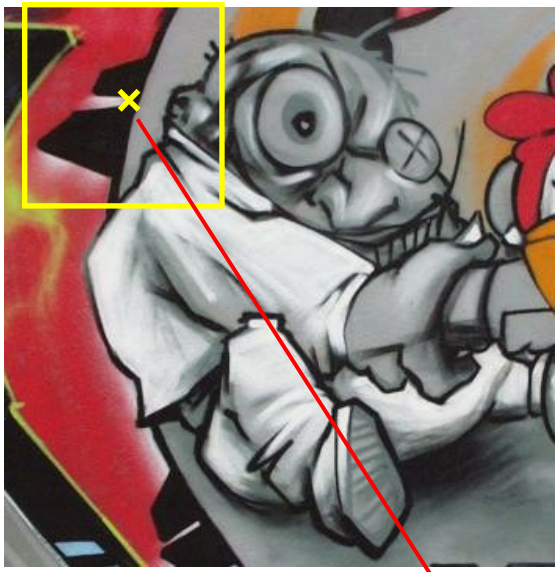


$f(I_{i_1 \ldots i_m}(x',\sigma'))$

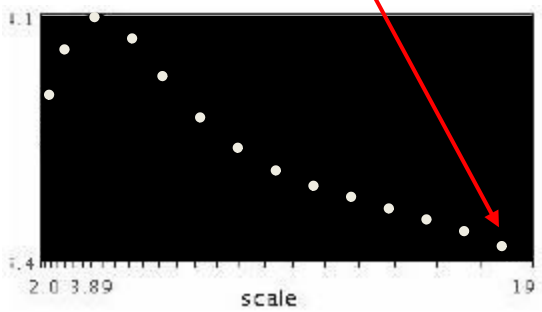
Automatic Scale Selection

- Function responses for increasing scale (scale signature)

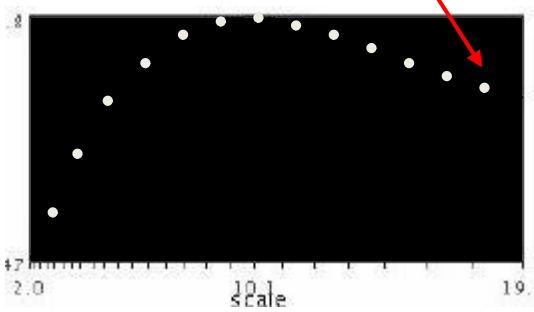




Response of some function *f*



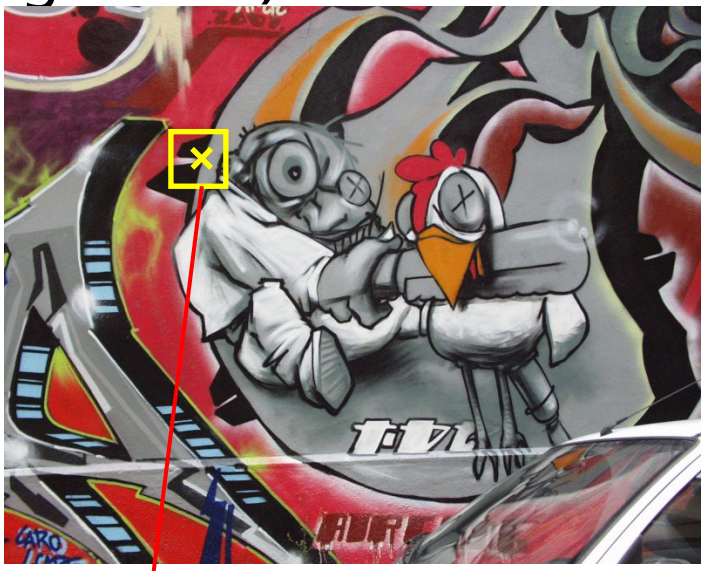
$f(I_{i_1 \ldots i_m}(x,\sigma))$



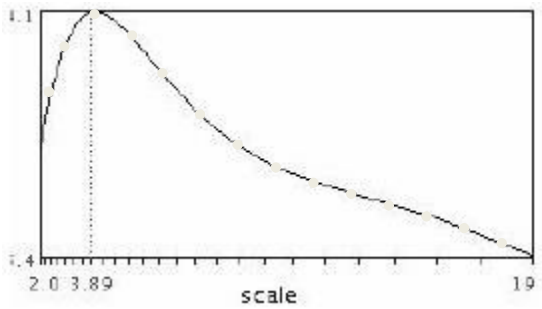
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Automatic Scale Selection

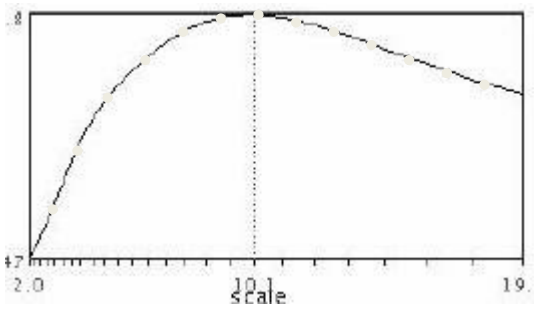
- Function responses for increasing scale (scale signature)



Response of some function *f*



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



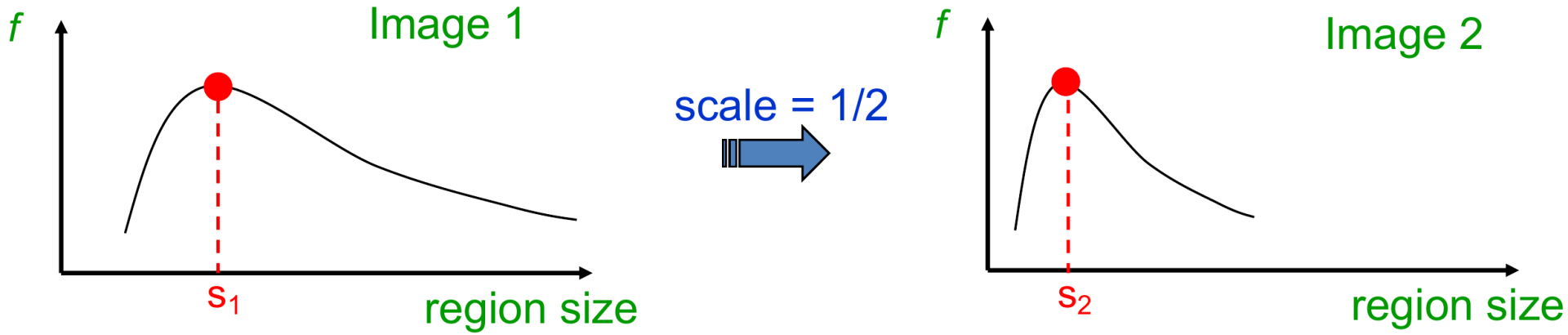
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Scale Invariant Detection

- Common approach:
  Take a local maximum of this function

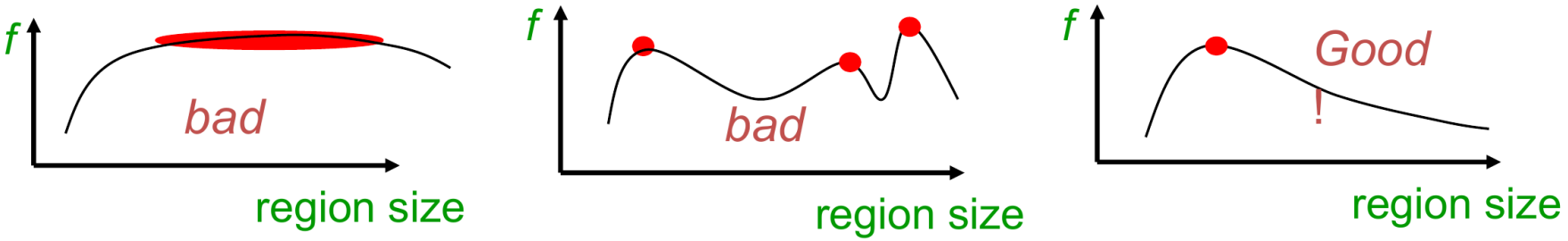
- Observation: region size, for which the maximum is achieved, should be *co-variant* with image scale.

Important: this scale invariant region size is found in each image independently!



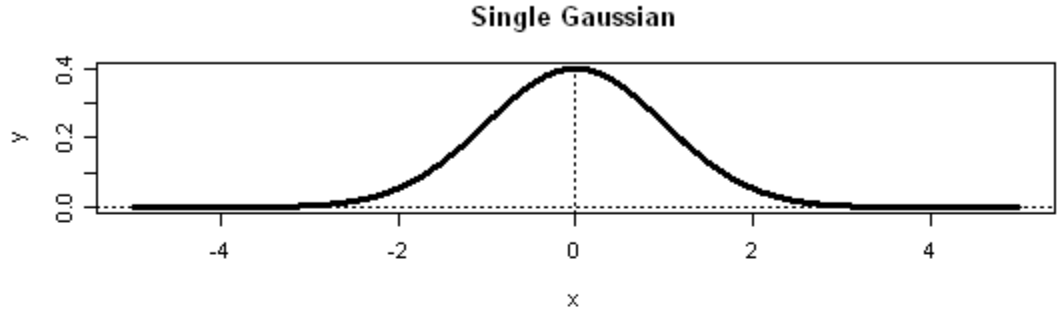
Scale Invariant Detection

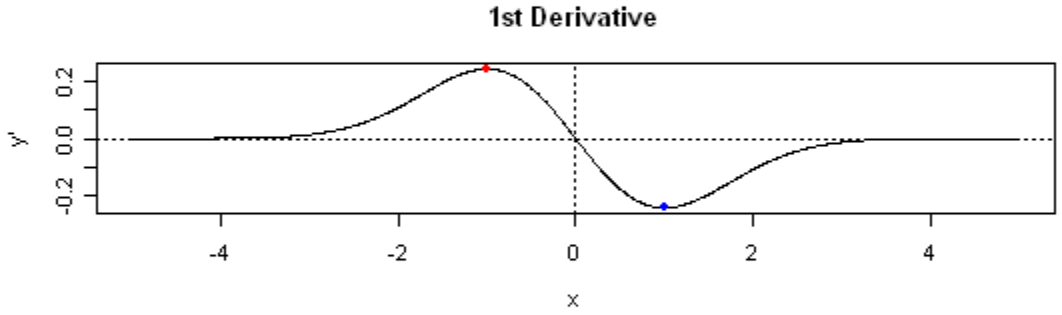
- A "good" function for scale detection:
  has one stable sharp peak

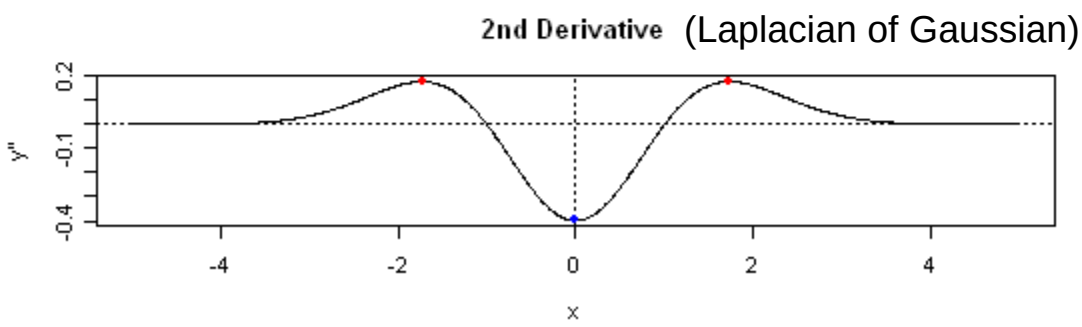


- For usual images: a good function would be one which responds to contrast (sharp local intensity change)

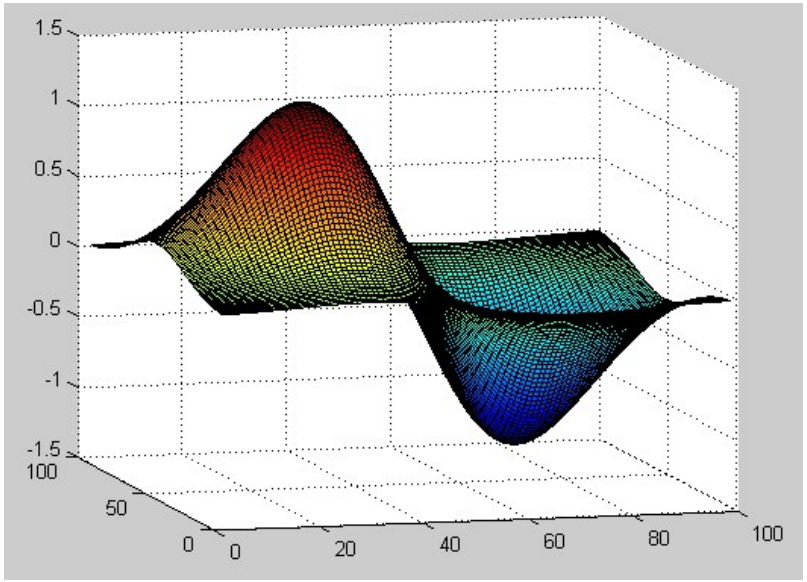
What Is A Useful Signature Function *f* ?







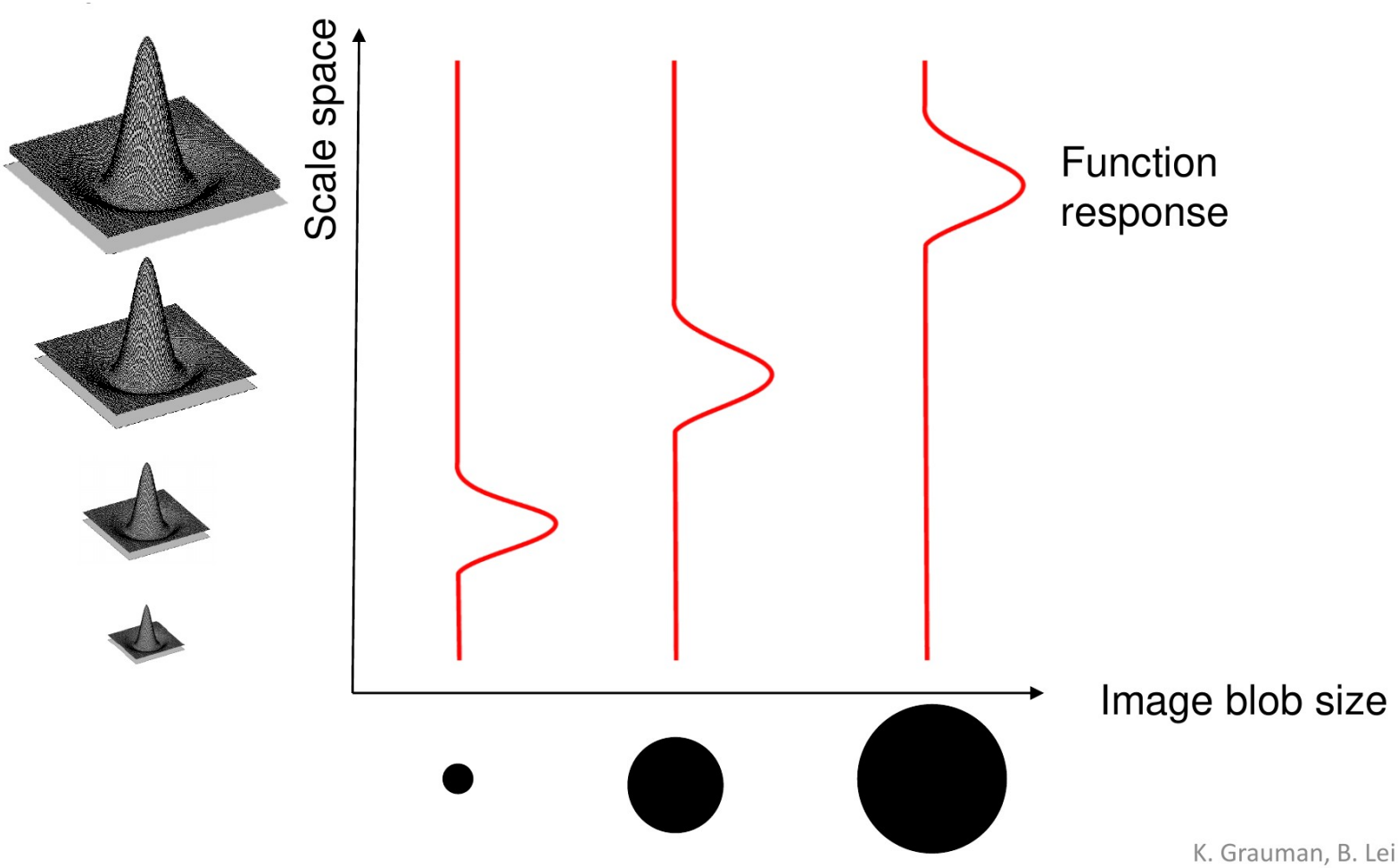
1st Derivative of Gaussian



What Is A Useful Signature Function *f* ?

"Blob" detector is common for corners

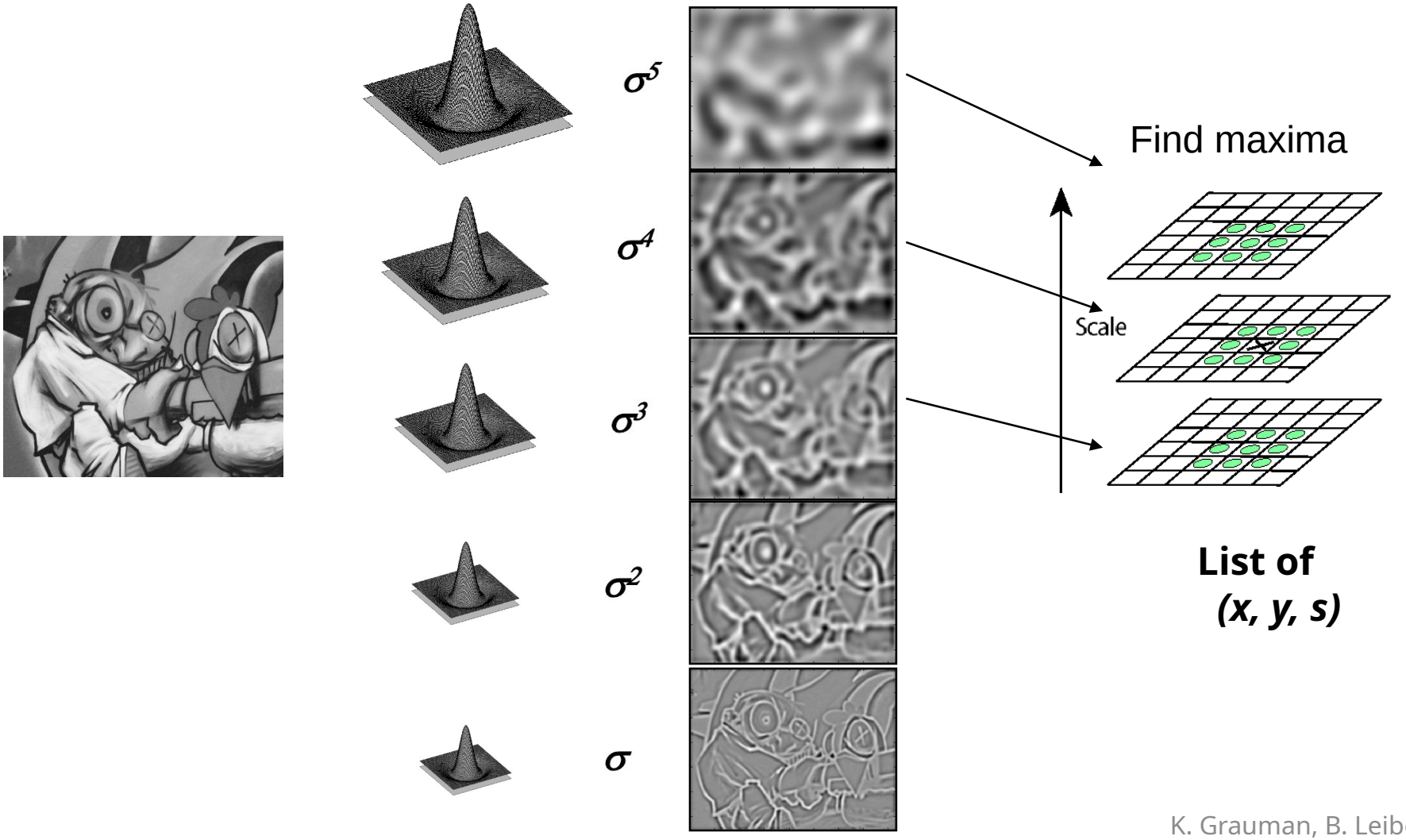
– Laplacian (2nd derivative) of Gaussian (LoG)



K. Grauman, B. Leibe

Find local maxima in position-scale space

LoG (Laplacian of Gaussian = second derivative of the gaussian kernel)



Find local maxima in position-scale space

We want to cover **as much as possible** the scale space **as fast possible**!

Tricks: the LoG can be approximated by DoG (Difference of Gaussian):

- Functions for determining scale $f = \text{Kernel} * \text{Image}$

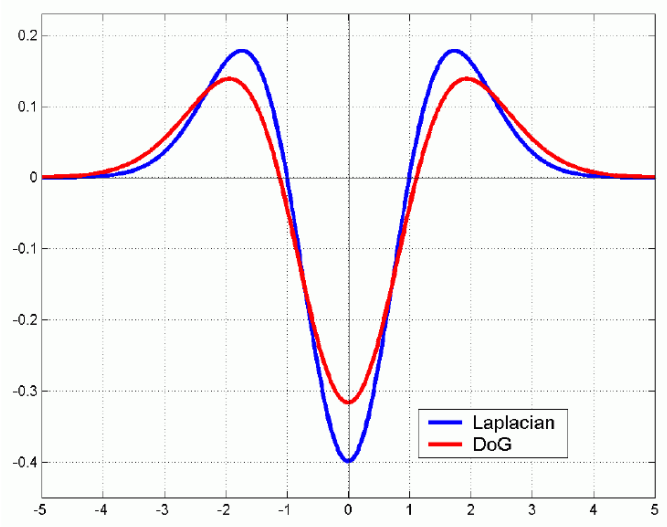
Kernels:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



where Gaussian

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

Note: both kernels are invariant to *scale* and *rotation*

Find local maxima in position-scale space

We want to cover **as much as possible** the scale space **as fast possible**!

Tricks: the LoG can be approximated by DoG (Difference of Gaussian):

$$\frac{\partial G}{\partial \sigma} = \sigma \Delta^2 G$$

Heat Diffusion Equation

$$\sigma \Delta^2 G = \frac{\partial G}{\partial \sigma} = \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

LoG

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \Delta^2 G$$

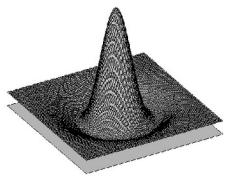
DoG

$$\text{Typical values}: \sigma = 1.6;\ k = \sqrt{2}$$

Alternative kernel

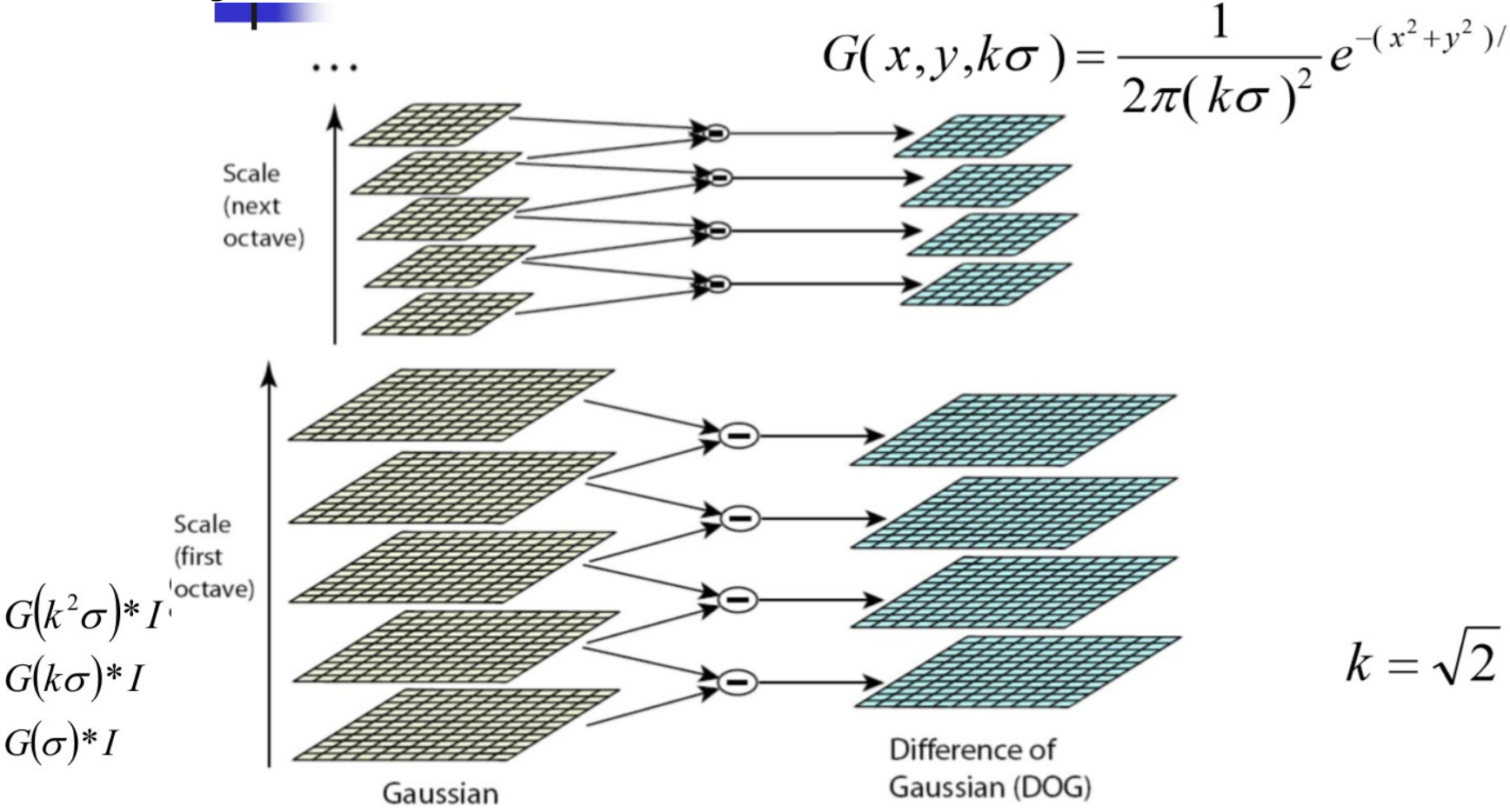
Approximate LoG with Difference-of-Gaussian (DoG).

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



SIFT Pyramid Scheme:

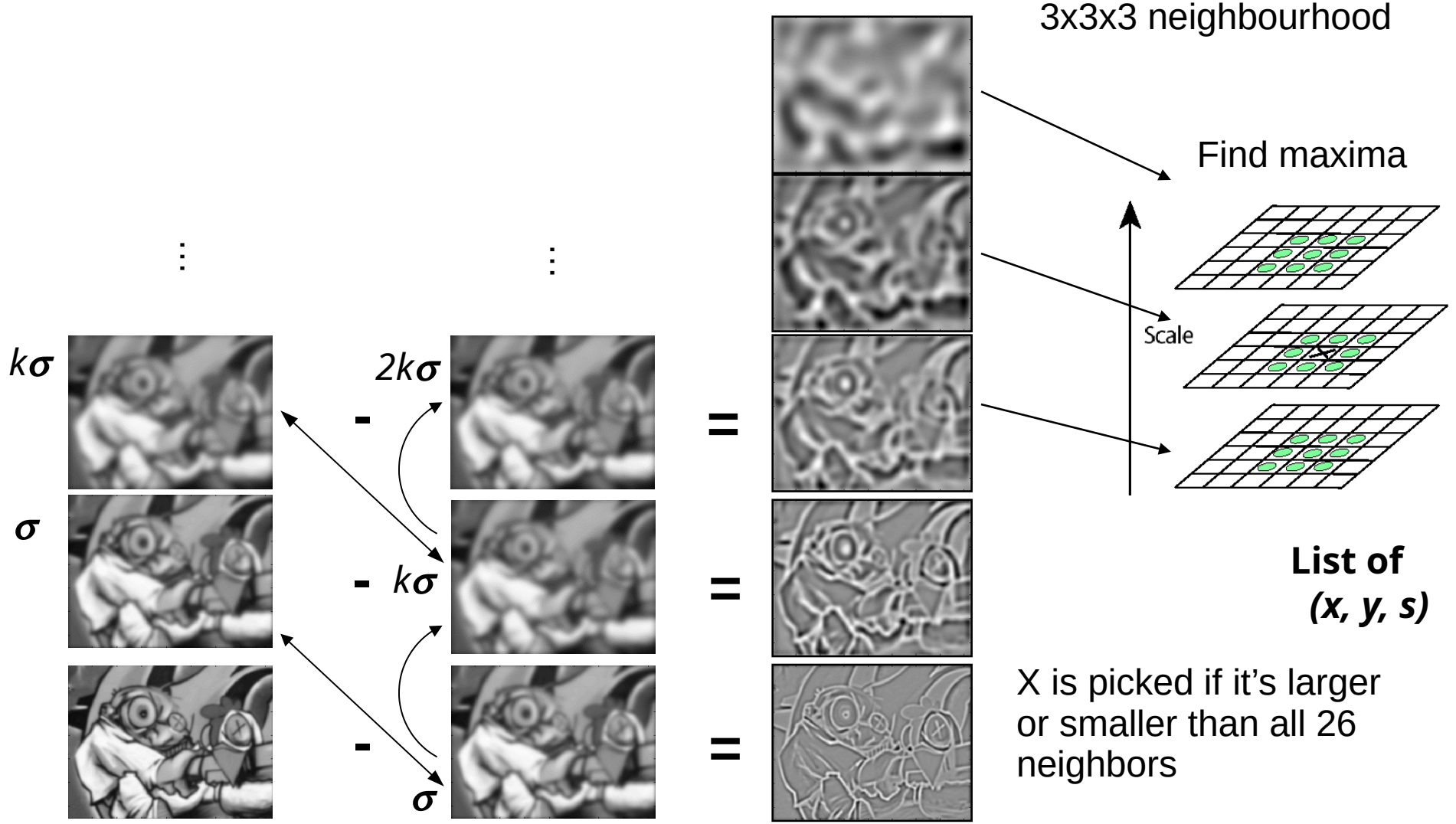
$$G(x,y,k\sigma) = \frac{1}{2\pi(k\sigma)^2} e^{-(x^2+y^2)/2k^2\sigma^2}$$



$$k = \sqrt{2}$$

- All images in a particular octave have the same resolution.
- The Images in each octave are convolved with progressively increasing σ.

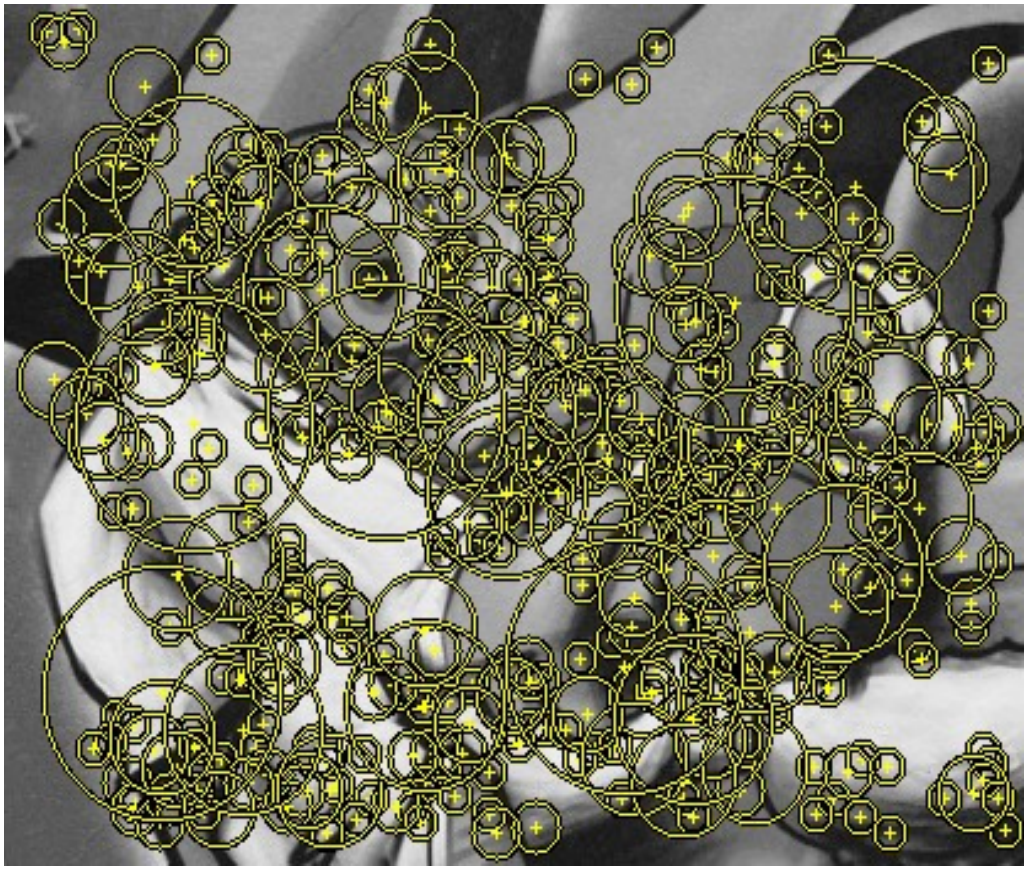
Find local maxima in position-scale space of DoG



K. Grauman, B. Leibe

Results: Difference-of-Gaussian

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



Are we Done?

Many issues persist:

- Noise: You can end up with interest points on seemingly flat-textureless surfaces.
- Edges: You can end up with interest points along edges. (Solved like we solved it with the Harris corner detector. M matrix!)
- Localization accuracy: We discretized the scale space to generate the Images, need to extrapolate to find a more accurate location
- Rotation Invariance?

Comparison of Keypoint Detectors

Table 7.1 Overview of feature detectors.

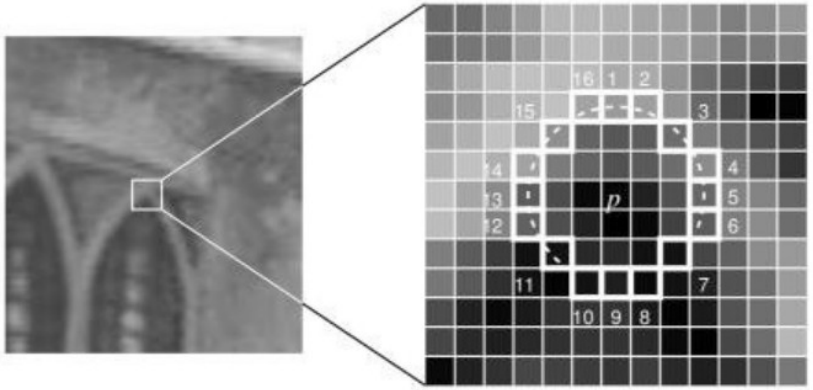
Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Localization Repeatability	Localization accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

Tuytelaars Mikolajczyk 2008

HUNDREDS MORE!

Traditional machine learning approach:

- Features from Accelerated Segment Test: Fast – AGAST – OAST … (based on decision trees)



- FREAK: Fast Retina Keypoint, etc.

Deep learning approach: LIFT (Learned Invariant Feature Transform), LF-Net, etc.

Many, many, many more, there are **dozens** of survey papers on features!!

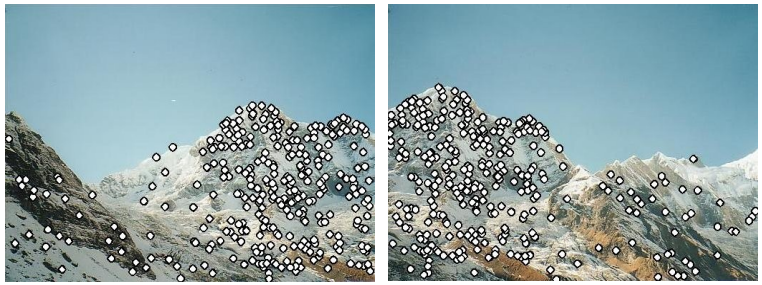
Local Image Descriptors

Read Szeliski 4.1

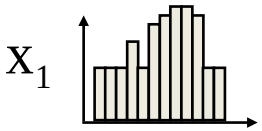
Acknowledgment: Many slides from James Hays, Derek Hoiem and Grauman & Leibe 2008 AAAI Tutorial

Local features: main components

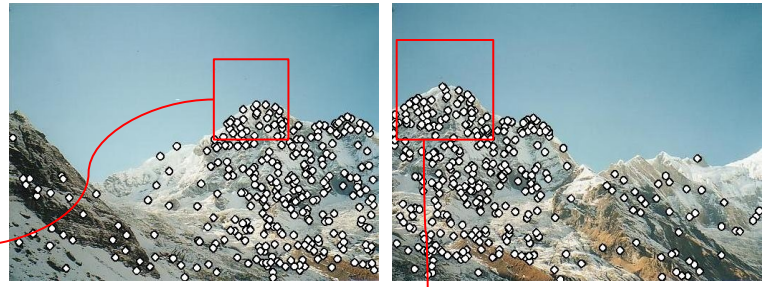
1) Detection:
Find a set of distinctive key points.

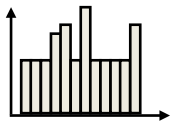


2) Description:
Extract feature descriptor around each interest point as vector.



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





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

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

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

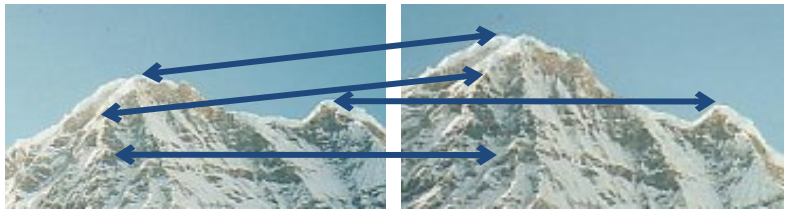
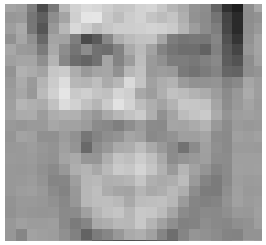


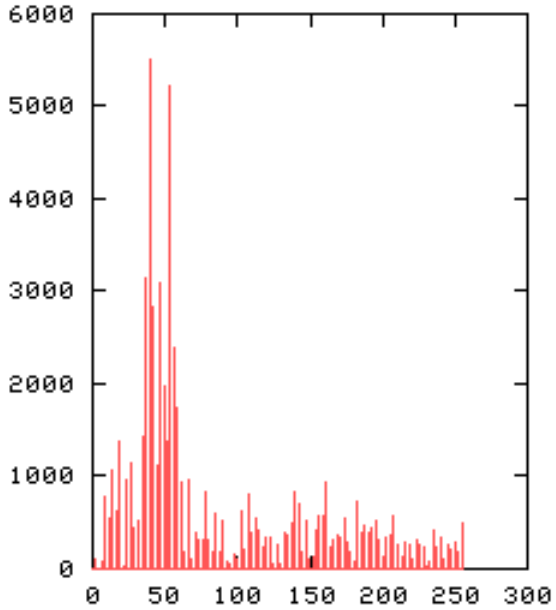
Image representations

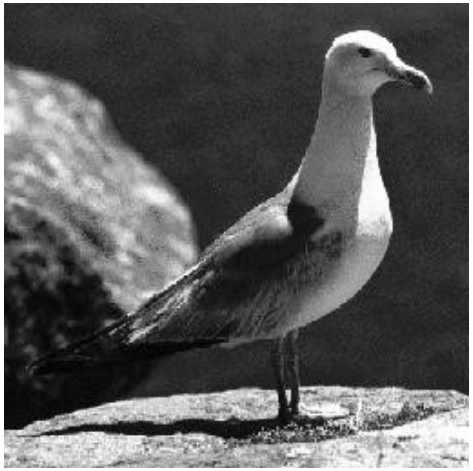
- Templates
  - Intensity, gradients, etc.



- Histograms
  - Color, texture, SIFT descriptors, etc.

Image Representations: Histograms





Global histogram to represent distribution of features

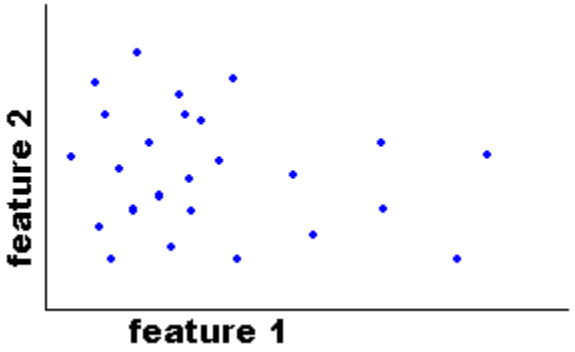
- Color, texture, depth, …

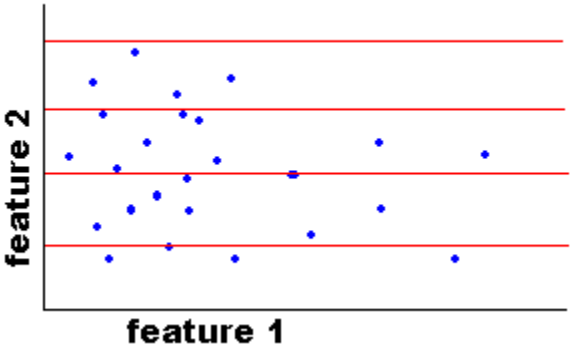
Local histogram per detected point

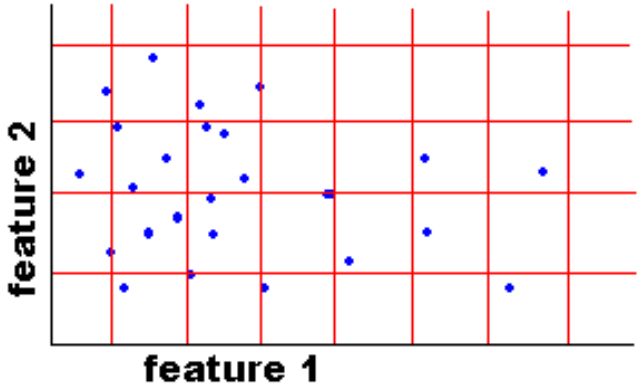
Images from Dave Kauchak

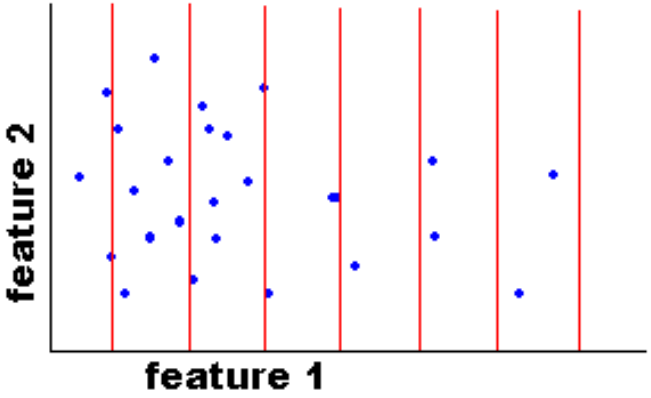
Image Representations: Histograms

Histogram: Probability or count of data in each bin









- Joint histogram
  - Requires lots of data
  - Loss of resolution to avoid empty bins

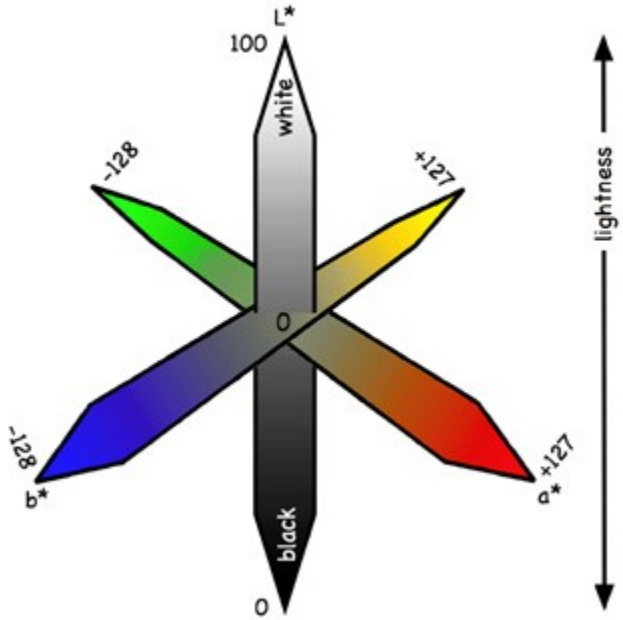
Marginal histogram

- Requires independent features
- More data/bin than joint histogram

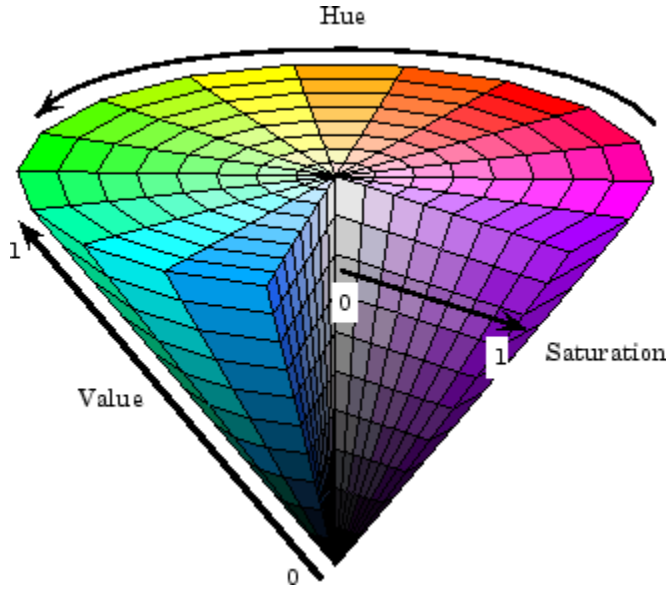
Images from Dave

For what things might we compute histograms?

- Color



L*a*b* color space

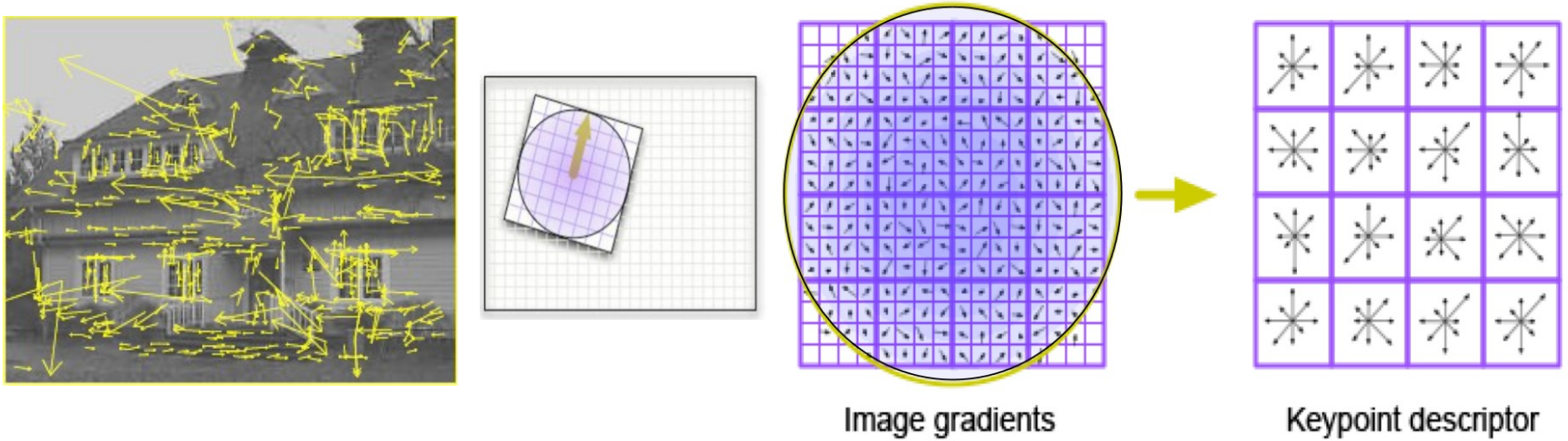


HSV color space

- Model local appearance

For what things might we compute histograms?

- Texture
- Local histograms of oriented gradients
- SIFT: Scale Invariant Feature Transform
  - Extremely popular (58k citations)



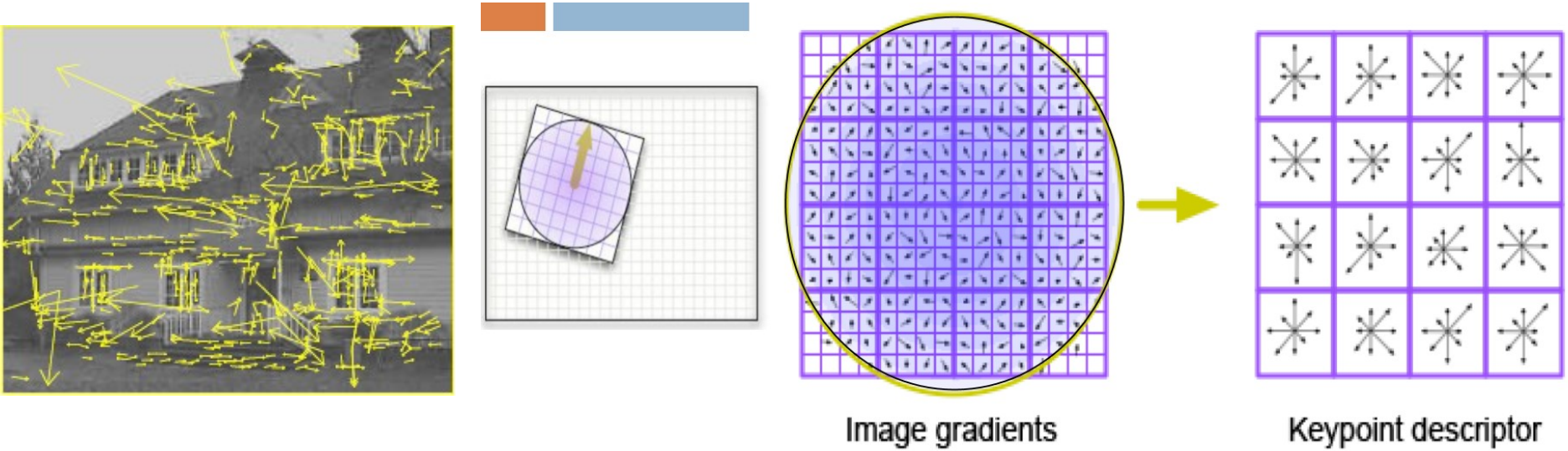
SIFT – Lowe IJCV 2004

SIFT descriptor formation

- A SIFT feature is given around a ‘point of interest’ by a vector <p,s,r,f>
  - p: pixel location
  - s: scale
  - r: Rotation
  - f: descriptor

View Point Dependent

(Hopefully) Independent



James Hays

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).

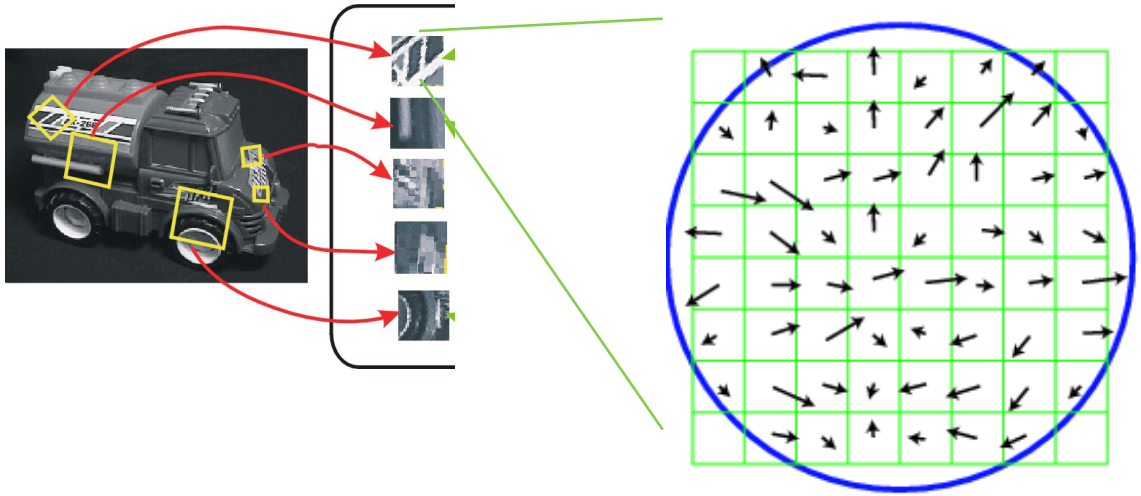
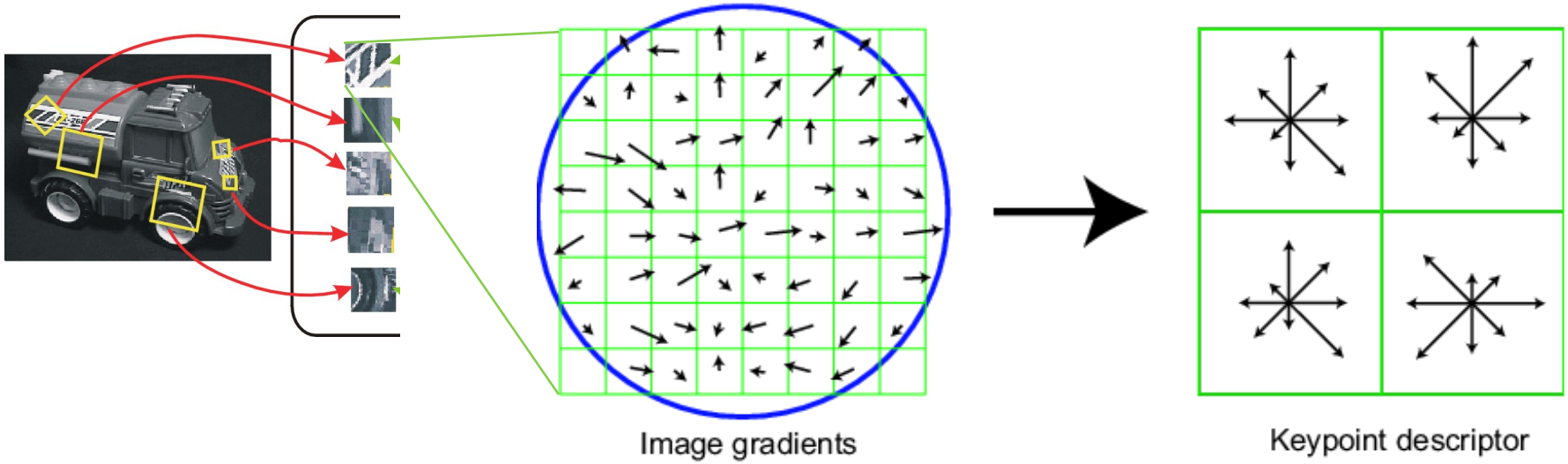


Image gradients

Actually 16x16, only showing 8x8

SIFT vector formation

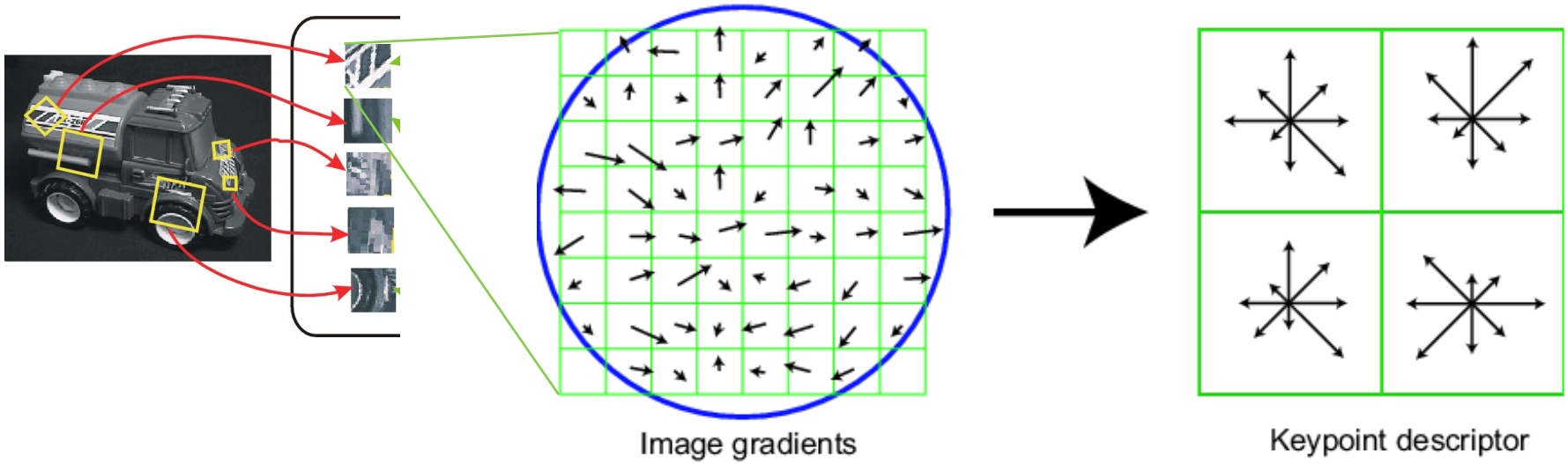
- Create a 4x4 array of gradient orientation histograms weighted by gradient magnitude.
- Bin into 8 orientations (45 degree bins) x 4x4 array = 128 dimensions.



Showing only 2x2 here but is 4x4

Reduce effect of illumination

- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - After normalization, clamp gradients > 0.2
  - Renormalize

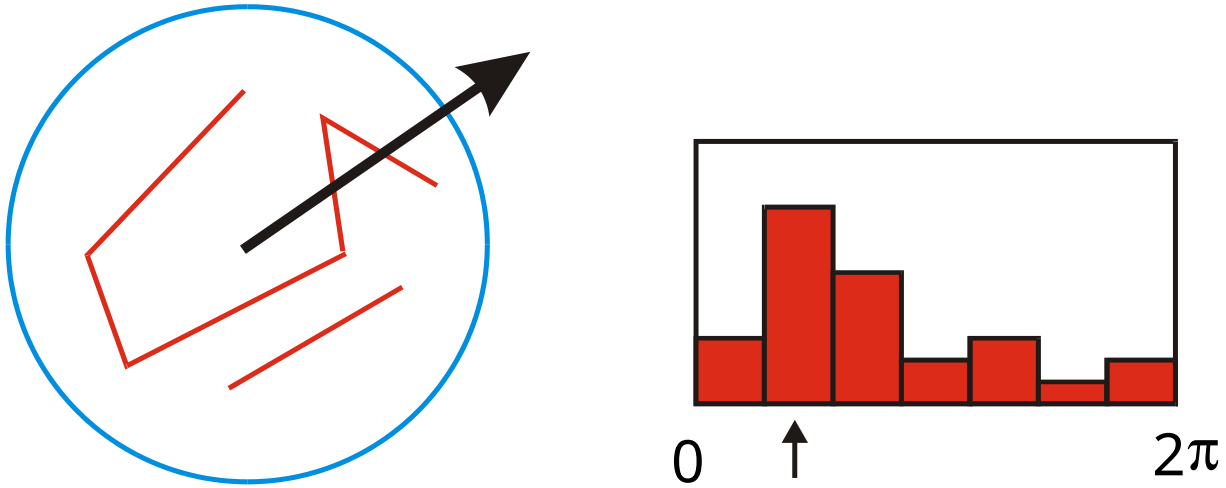


SIFT preprocessing

- Find Difference of Gaussian scale-space extrema as feature point locations
- Post-processing
  - Subpixel position interpolation
  - Discard low-contrast points
  - Eliminate points along edges
- Orientation estimation per feature point

SIFT Orientation estimation

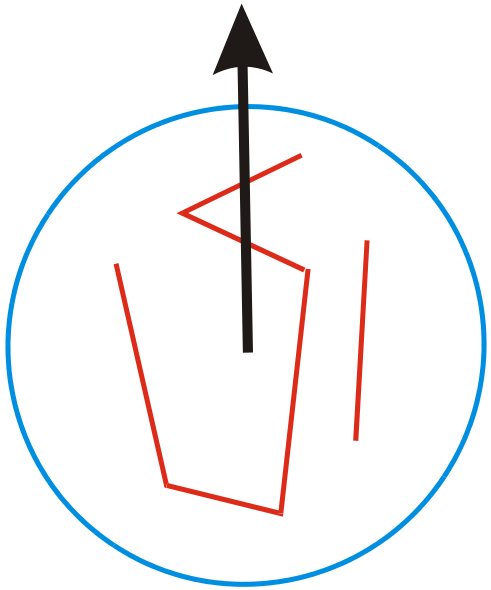
- Compute gradient orientation histogram
- Select dominant orientation θ

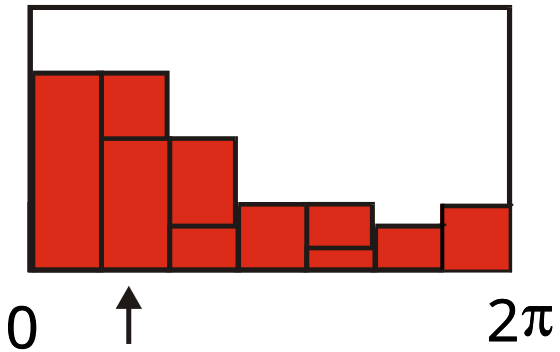


[Lowe, SIFT, 1999]

SIFT Orientation Normalization

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

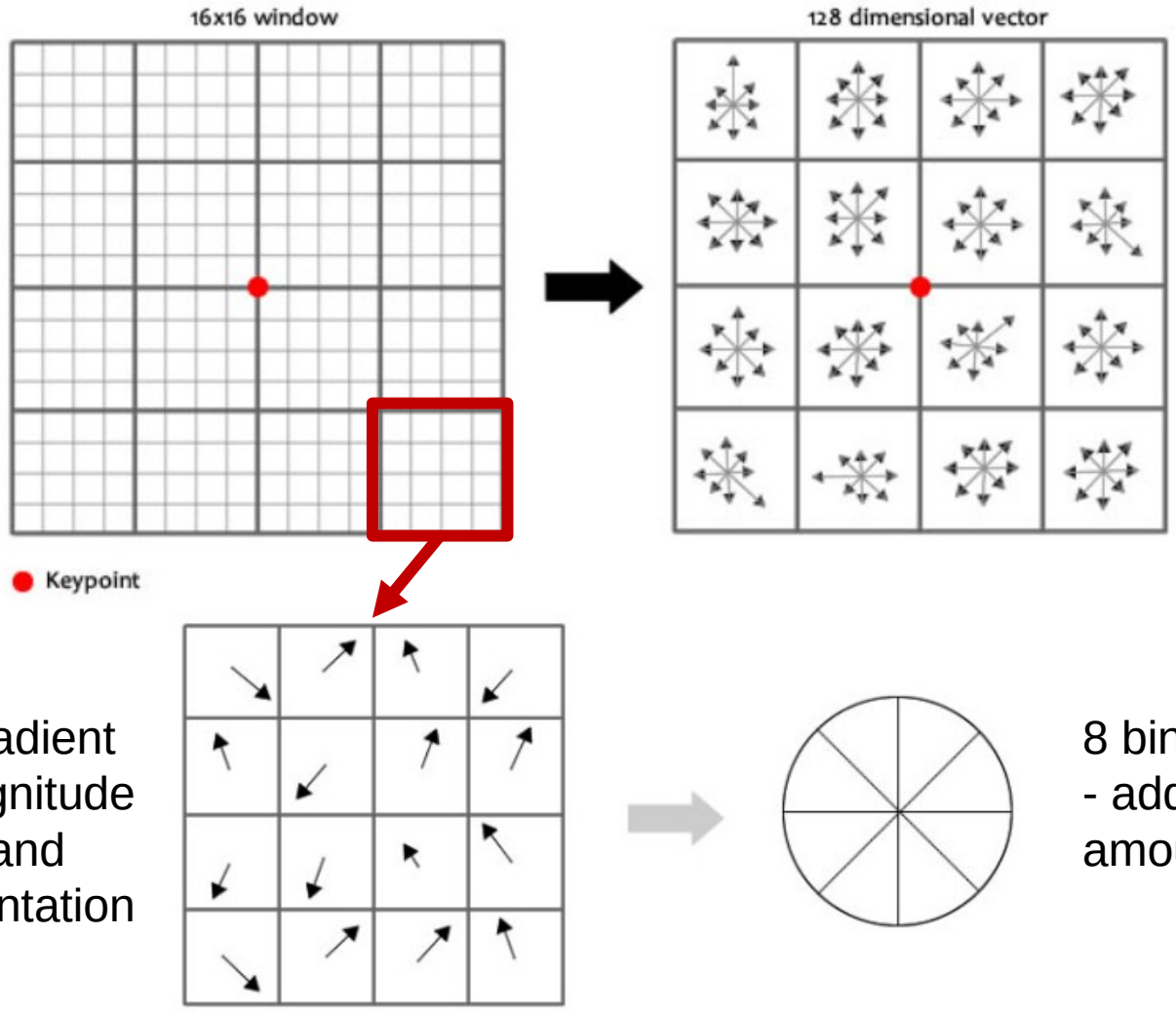




[Lowe, SIFT, 1999]

SIFT Descriptor Extraction

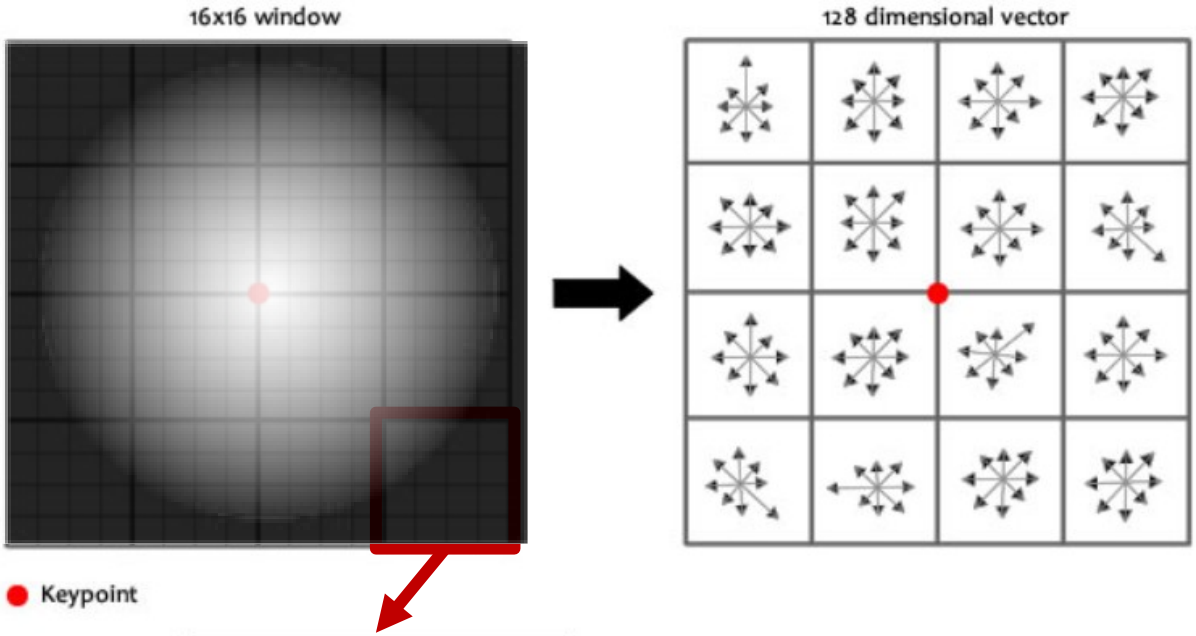
- Given a keypoint with scale and orientation



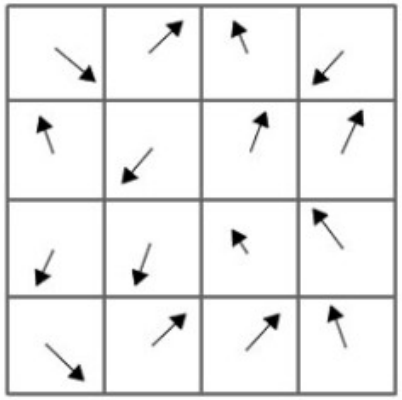
SIFT Descriptor Extraction

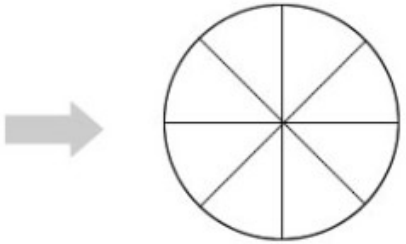
Weight 16x16 grid by Gaussian to add location robustness and reduce effect of outer reg

half window width



Gradient magnitude and orientation

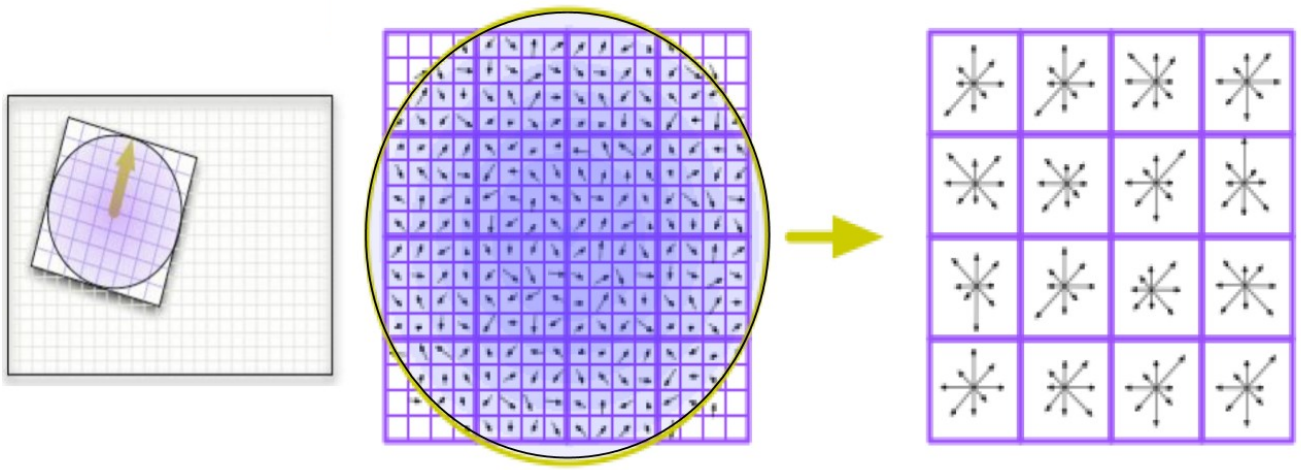




8 bin ‘histogram’ - add magnitude amounts!

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.



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

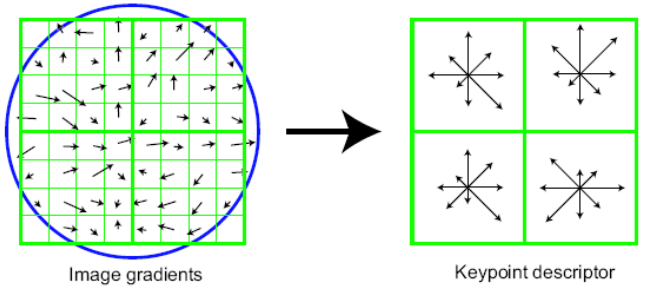
Specular highlights move between image pairs!

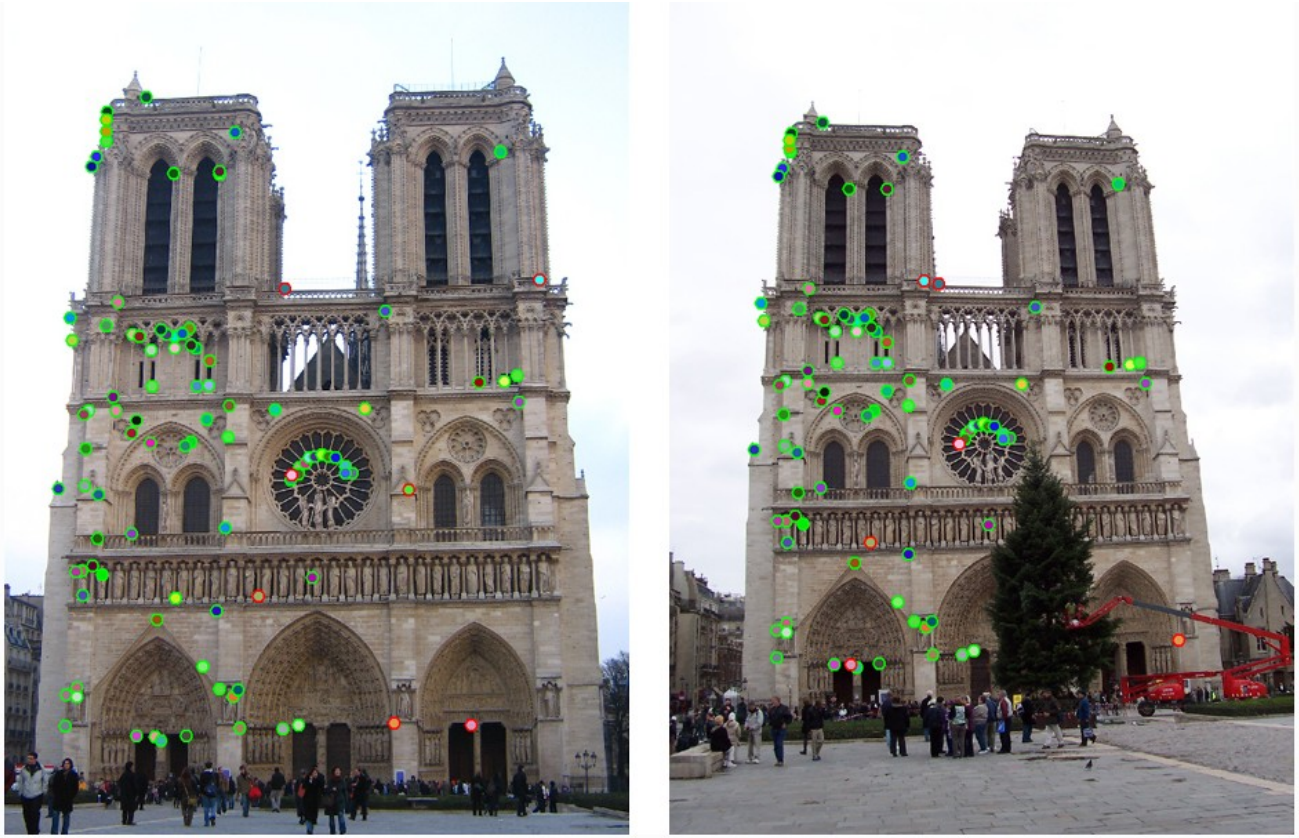
Efficient Implementation

- Filter using oriented kernels based on directions of histogram bins.
- Called ‘steerable filters’:
  - SURF

Review: Local Descriptors

- Most descriptors 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





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

SIFT-like descriptor in Project 2

- SIFT is hand designed based on intuition
- You implement your own SIFT-like descriptor
  - Ignore scale/orientation to start.

- Parameters: stick with defaults + minor tweaks
- Feel free to look at papers / resources for inspiration

A reference that might help you:
http://aishack.in/tutorials/sift-scale-invariant-feature-transform-features/

- Note an error in here: Gaussian weight should be applied to the larger 16x16 block and not the smaller 4x4 blocks. See Lowe's original paper, Sec. 6.1 par 2.

Lowe's original paper: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

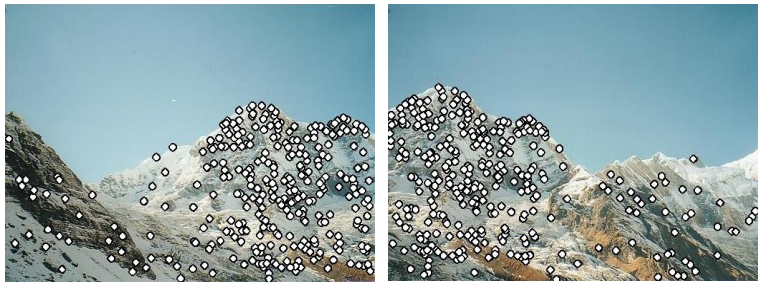
Feature Matching

Read Szeliski 4.1

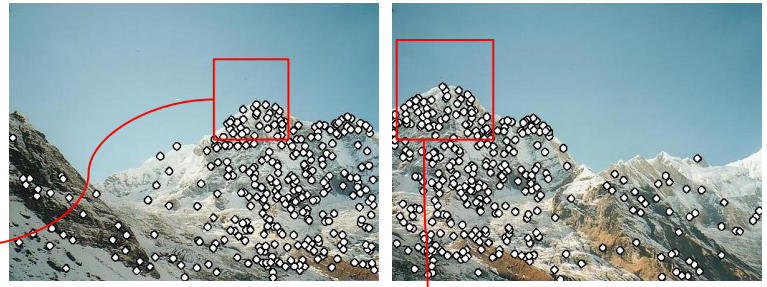
Many slides from James Hays, Derek Hoiem, and Grauman&Leibe 2008 AAAI Tutorial

Local features: main components

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



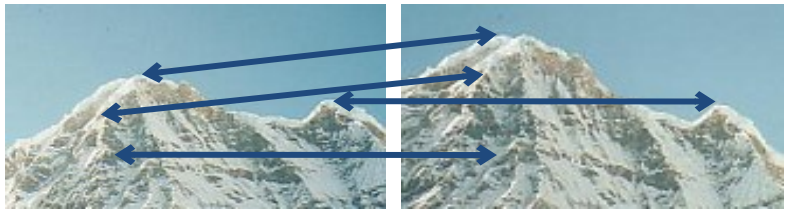
2) Description:
   Extract feature descriptor around each interest point as vector.



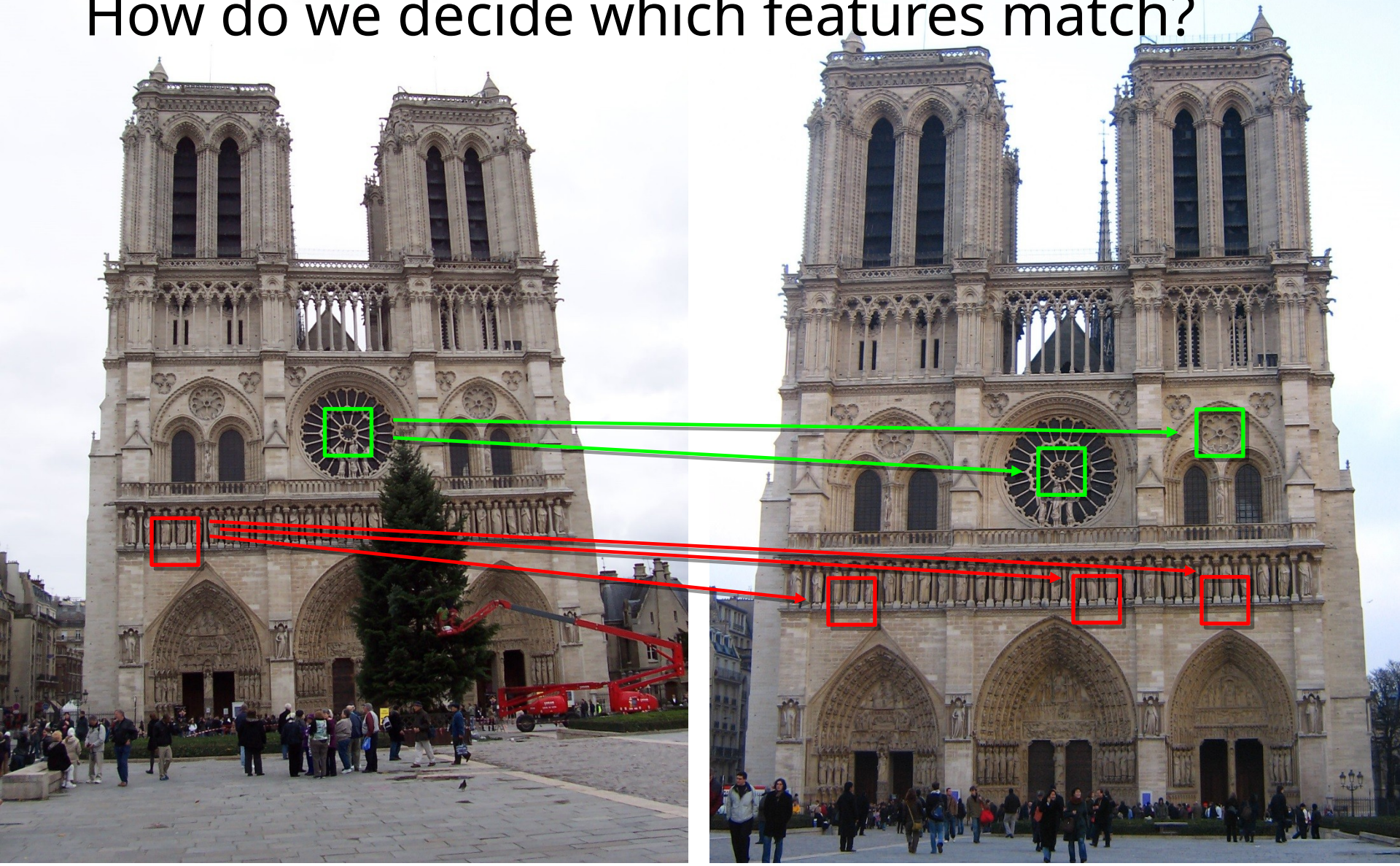
$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$

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

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

Euclidean distance vs. Cosine Similarity

- Euclidean distance:

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

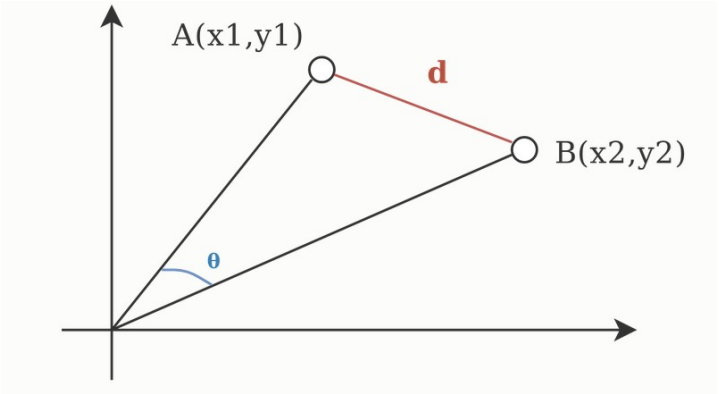
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

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

- Cosine similarity:

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

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



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:
  - Should 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.

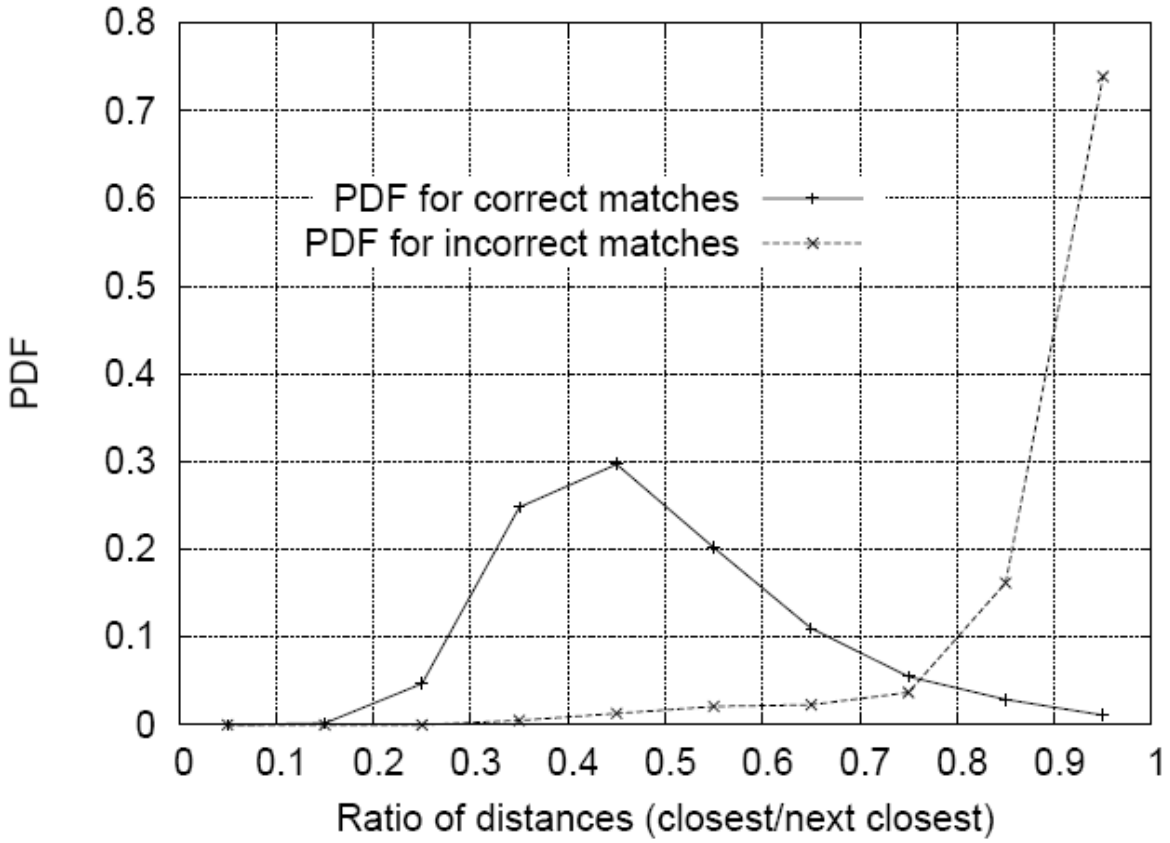
- If NN1 ≈ NN2, ratio will be ≈ 1 -> matches too close.
- As NN1 << NN2, ratio tends to 0.

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

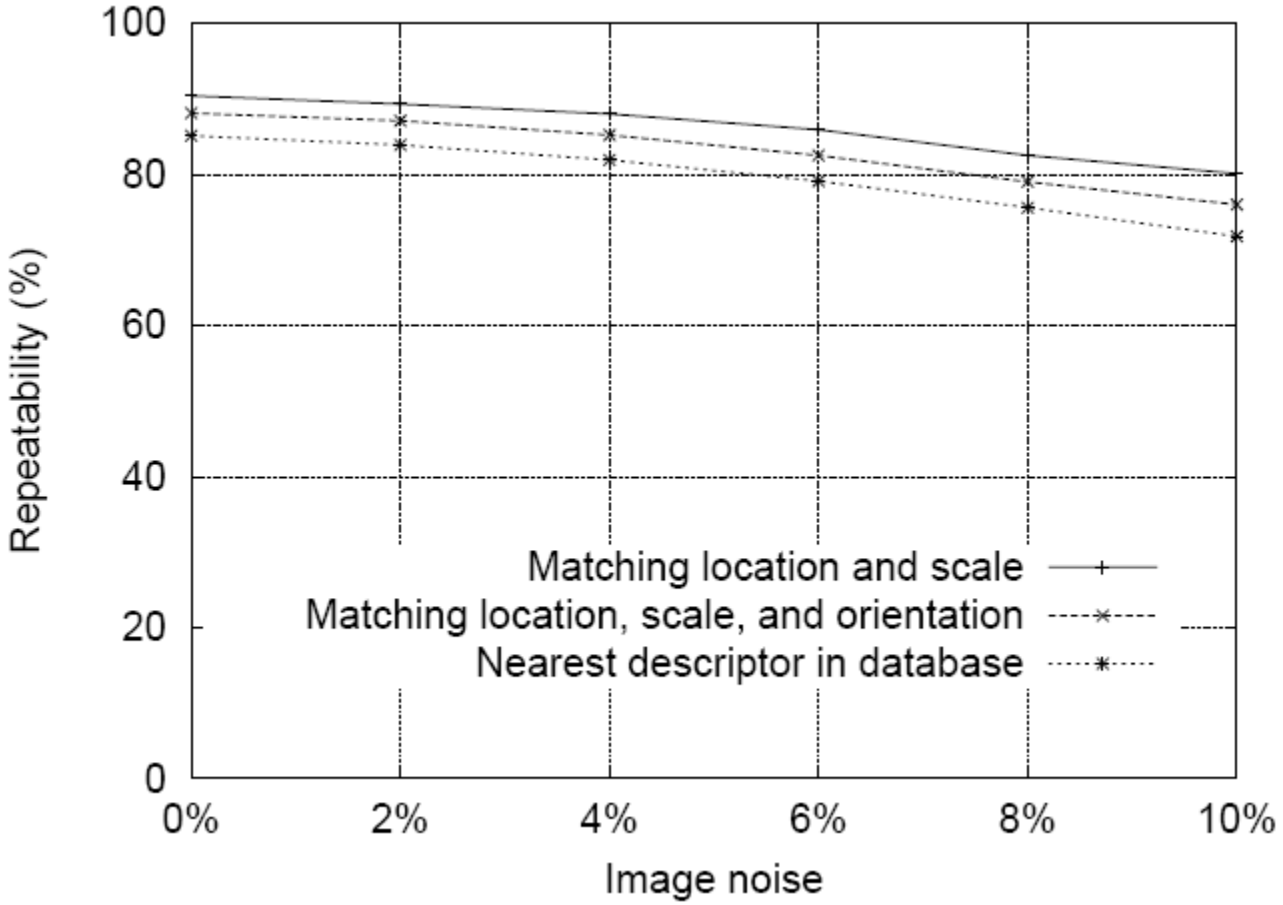


Ratio threshold depends on your application's view on the trade-off between the number of false positives and true positives!

Lowe IJCV 2004

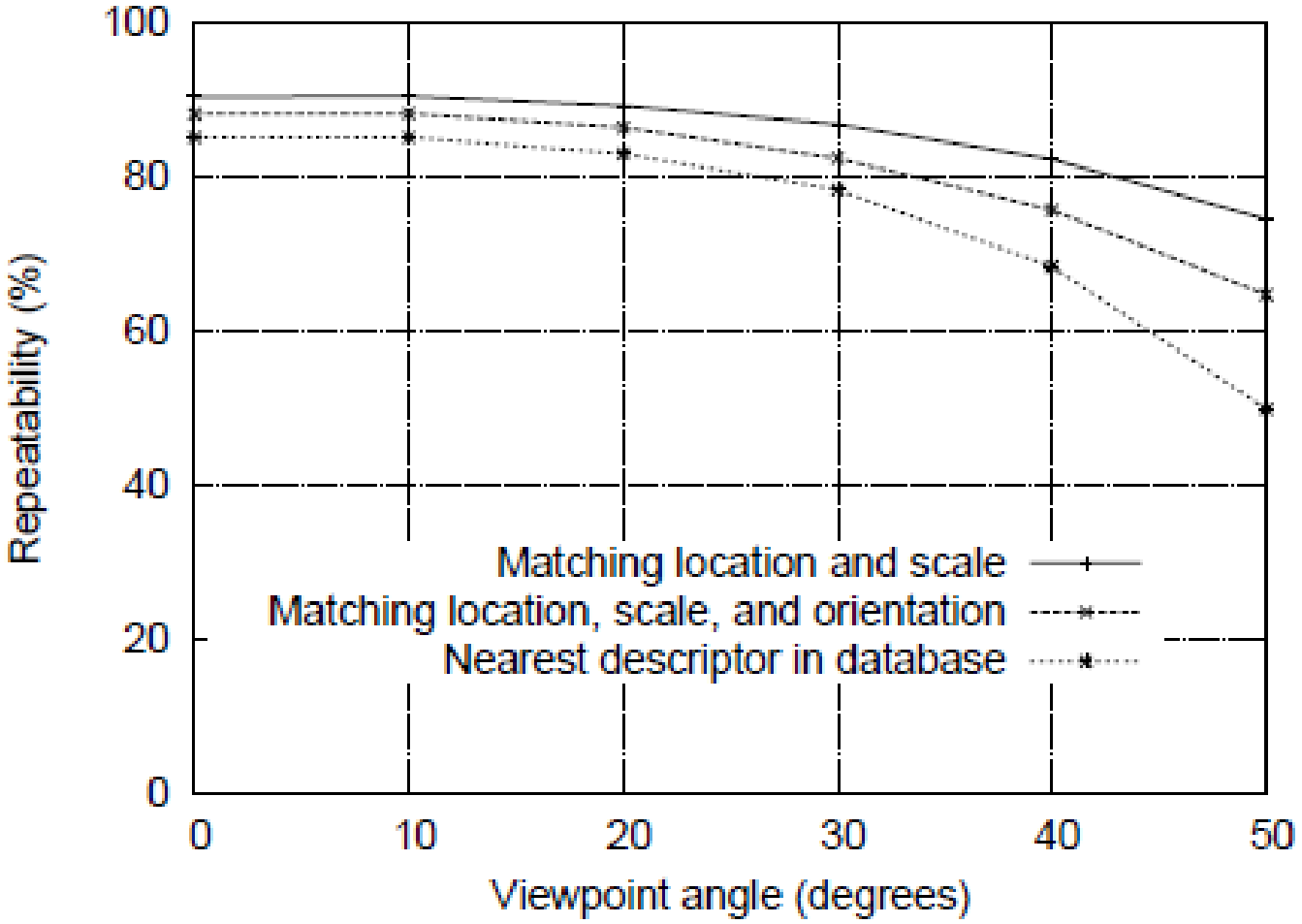
HOW GOOD IS SIFT?

SIFT Repeatability



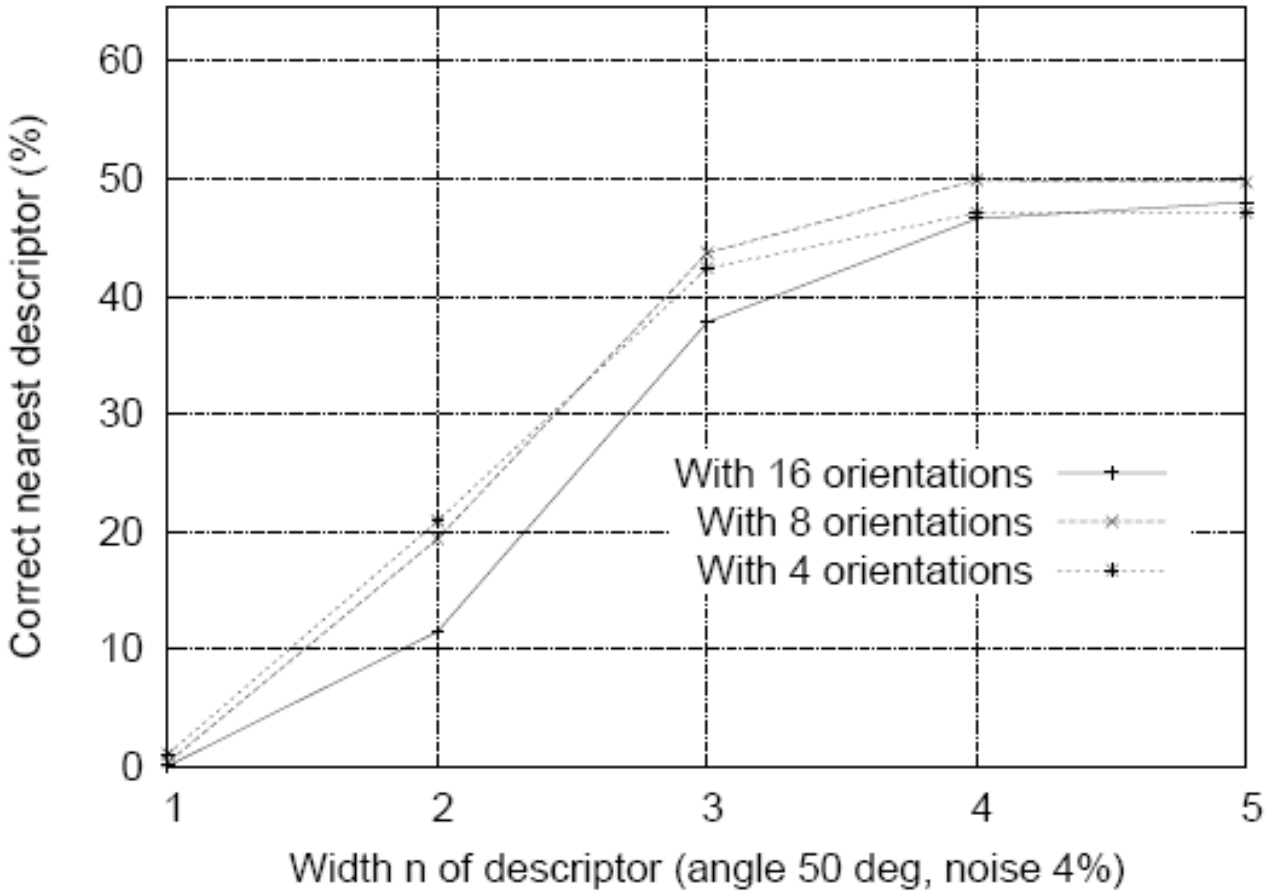
Lowe IJCV 2004

SIFT Repeatability



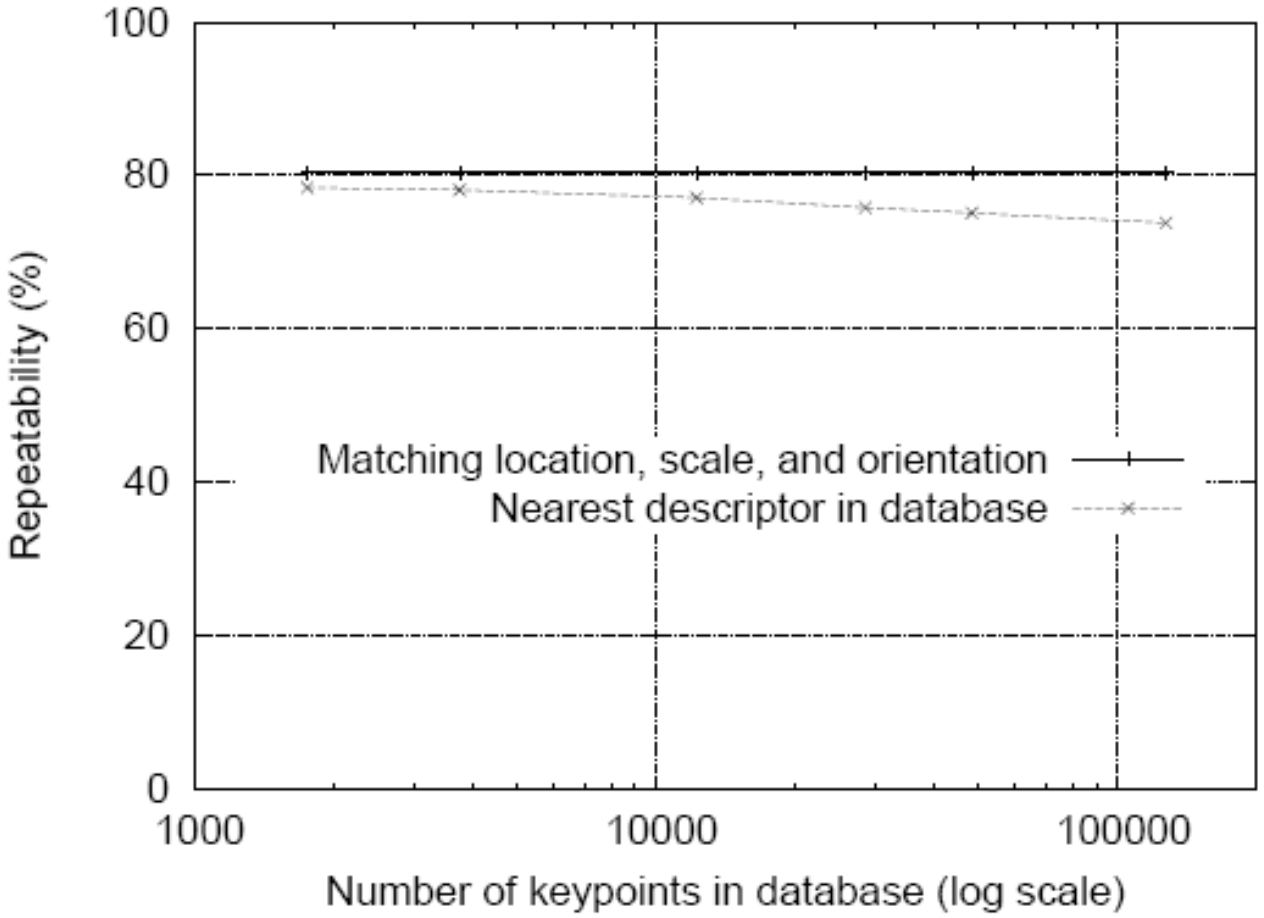
Lowe IJCV 2004

SIFT Repeatability



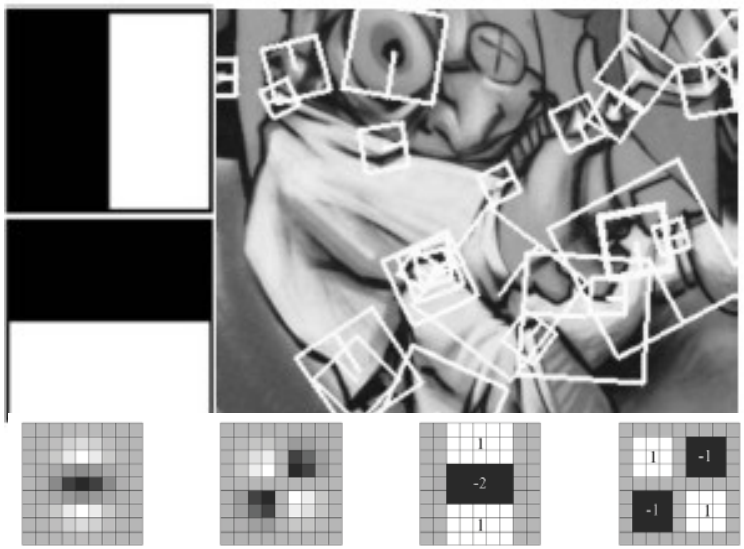
Lowe IJCV 2004

SIFT Repeatability



Lowe IJCV 2004

Local Descriptors: SURF



Fast approximation of SIFT idea

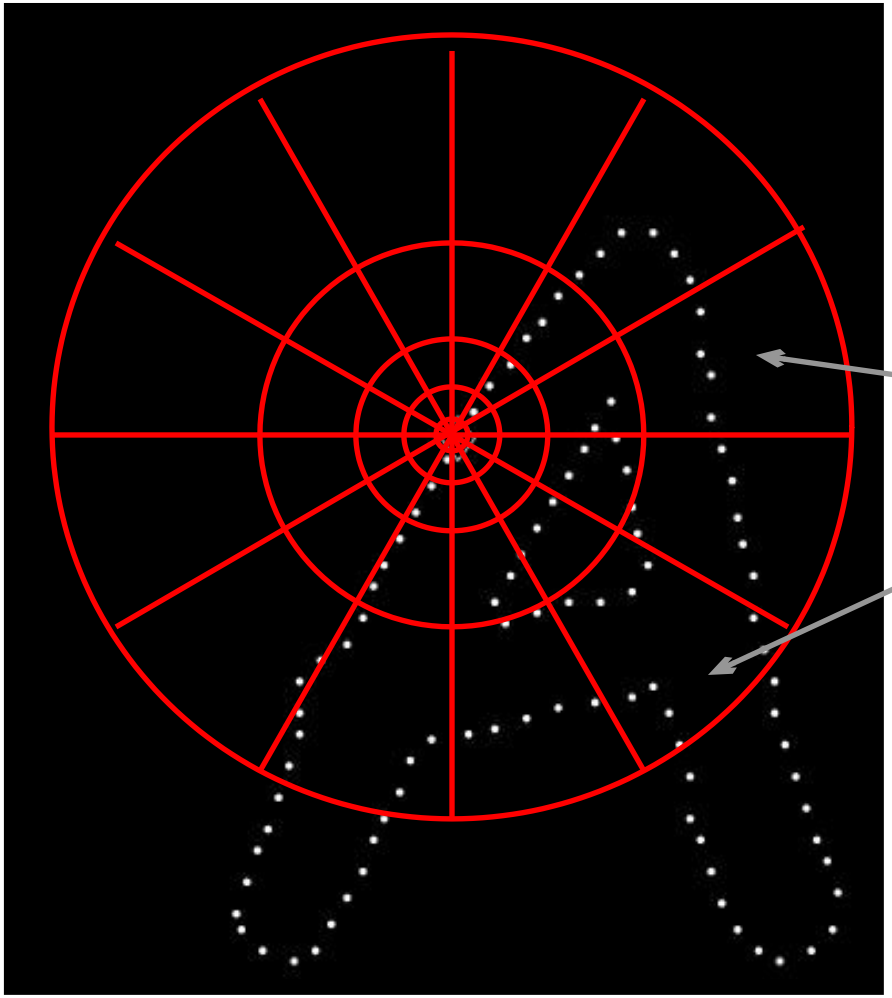
- **Efficient computation by 2D box filters & integral images**
  - **6 times faster than SIFT**
- **Equivalent quality for object identification**

GPU implementation available

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

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

Local Descriptors: Shape Context



Count the number of points inside each bin, e.g.:

Count = 4

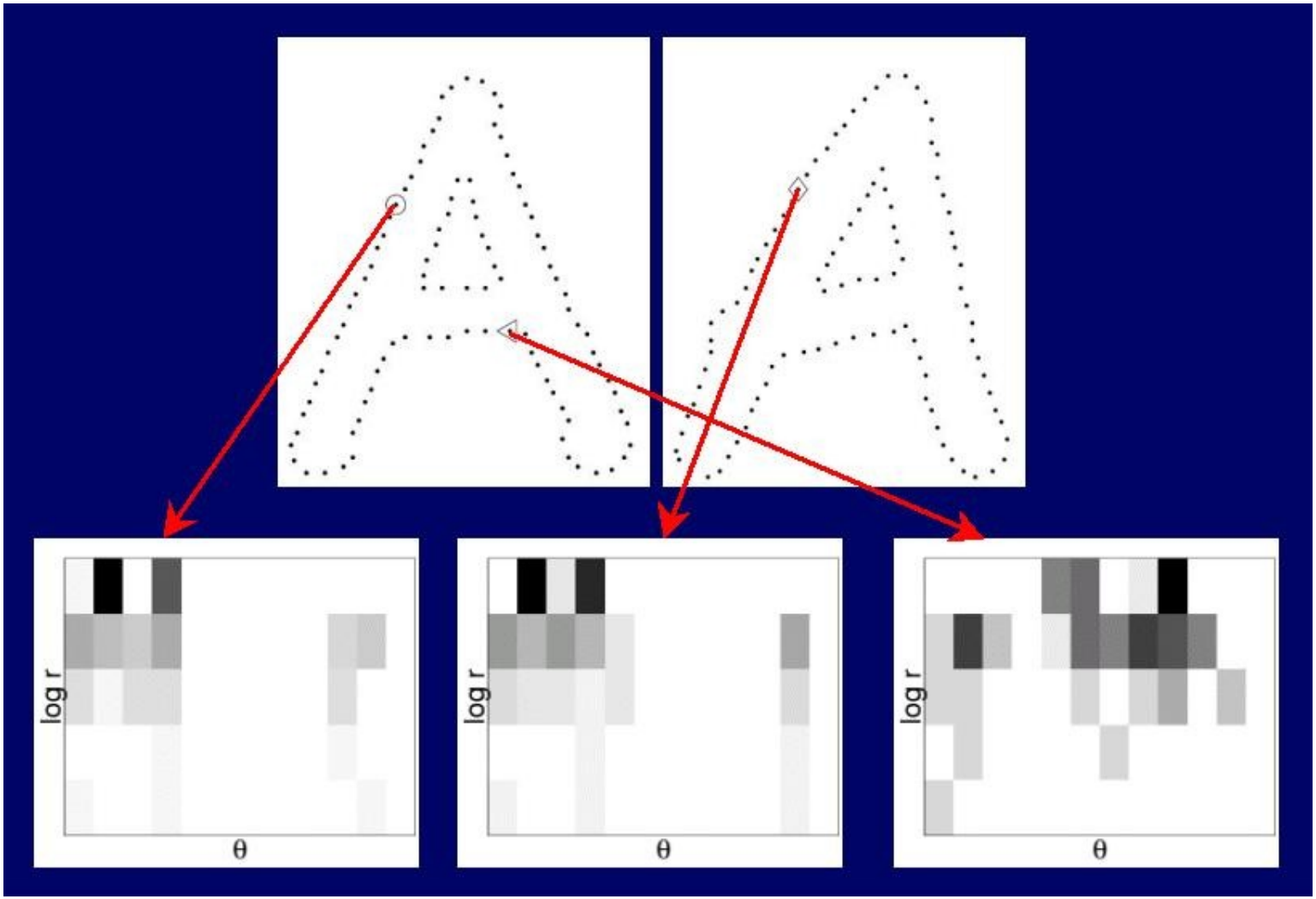
⋮

Count = 10

Log-polar binning:
More precision for nearby points, more flexibility for farther points.

Belongie & Malik, ICCV 2001

Shape Context Descriptor

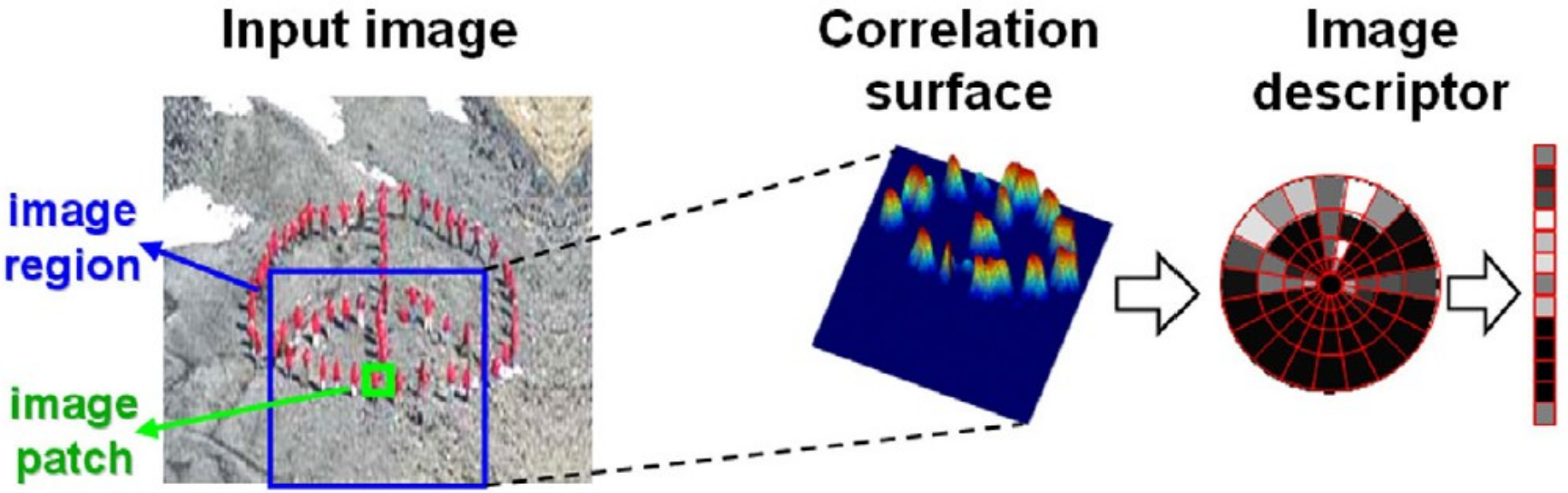


Self-similarity Descriptor

Figure 1. *These images of the same object (a heart) do NOT share common image properties (colors, textures, edges), but DO share a similar geometric layout of local internal self-similarities.*

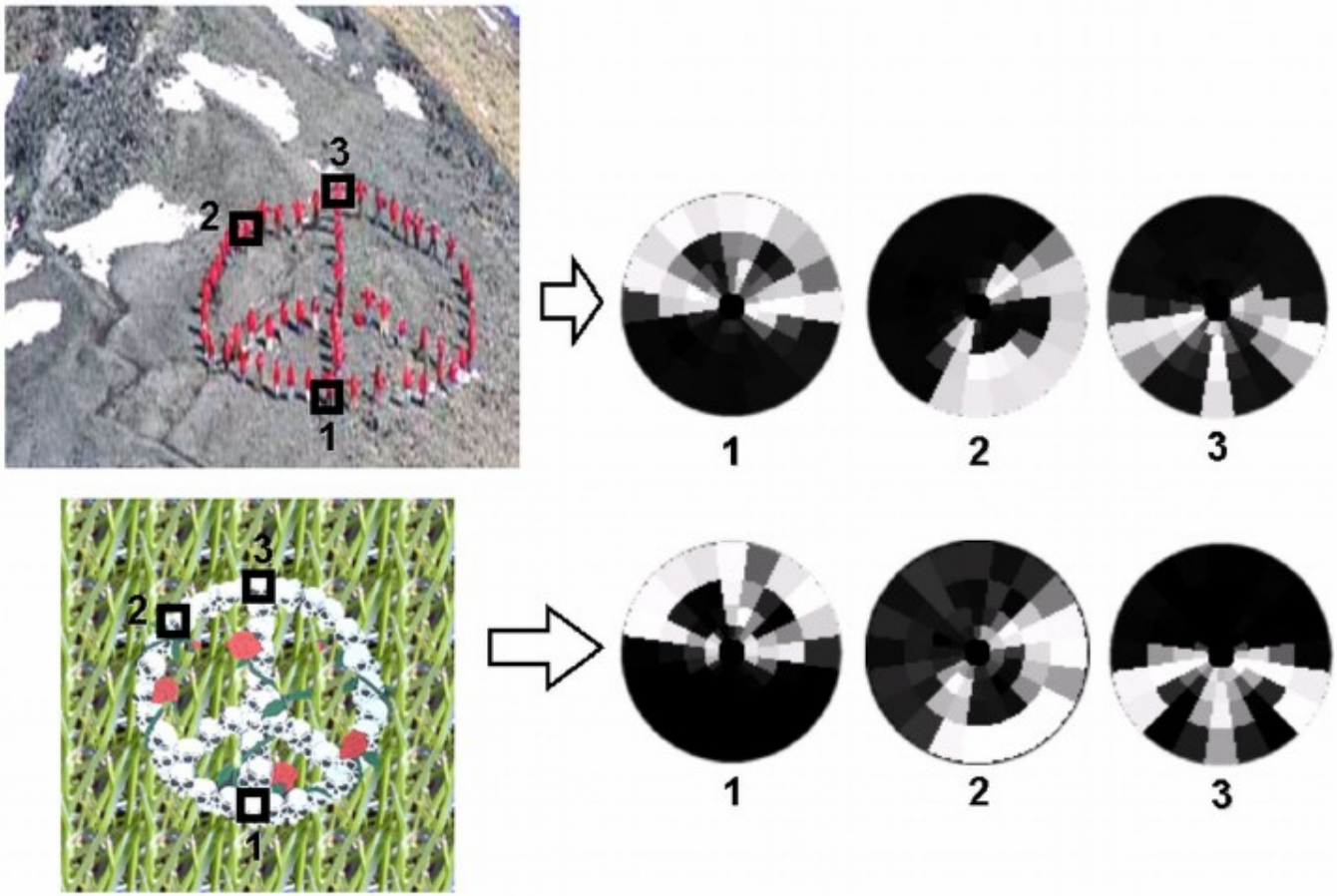
Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

Self-similarity Descriptor



Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

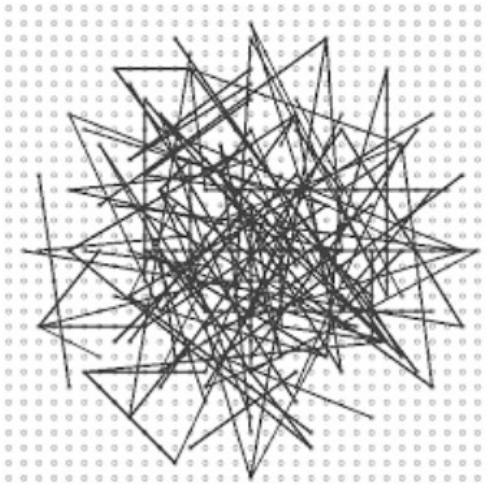
Self-similarity Descriptor



Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

Binary Descriptors

Binary Robust Independent Elementary Features Descriptors (BRIEF), LATCH, LBP, etc.



Oriented Brief (ORB): Oriented FAST feature detector + Brief Descriptor

Mix and Match! You can technically use any feature detector with any descriptor you want! Heck you can even use multiple descriptors at once!

Right features are application specific

- Shape: scene-scale, object-scale, detail-scale
  - 2D form, shading, shadows, texture, linear perspective
- Material properties: albedo, feel, hardness, …
  - Color, texture
- Motion
  - Optical flow, tracked points
- Distance
  - Stereo, position, occlusion, scene shape
  - If known object: size, other objects

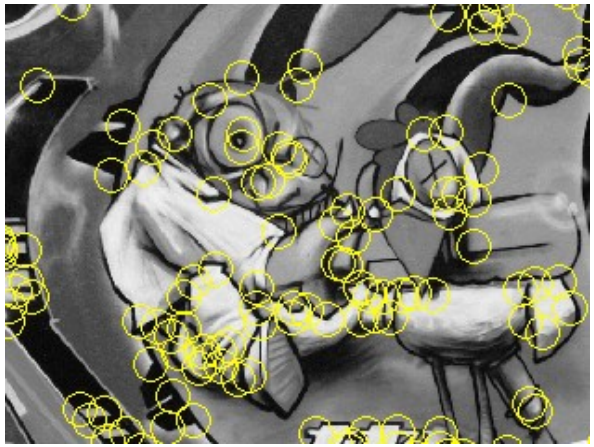
Available at a web site near you…

- 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

  - OpenCV has implemented a large library of various features and descriptors

Review: Interest points

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



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