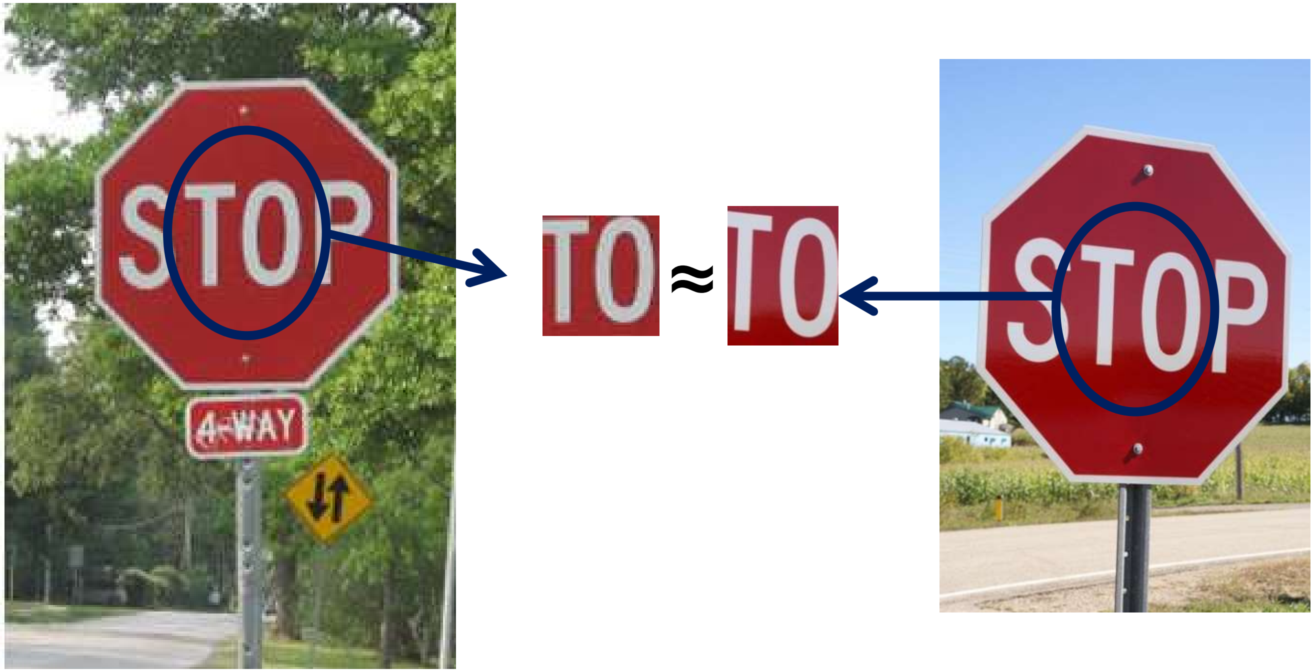
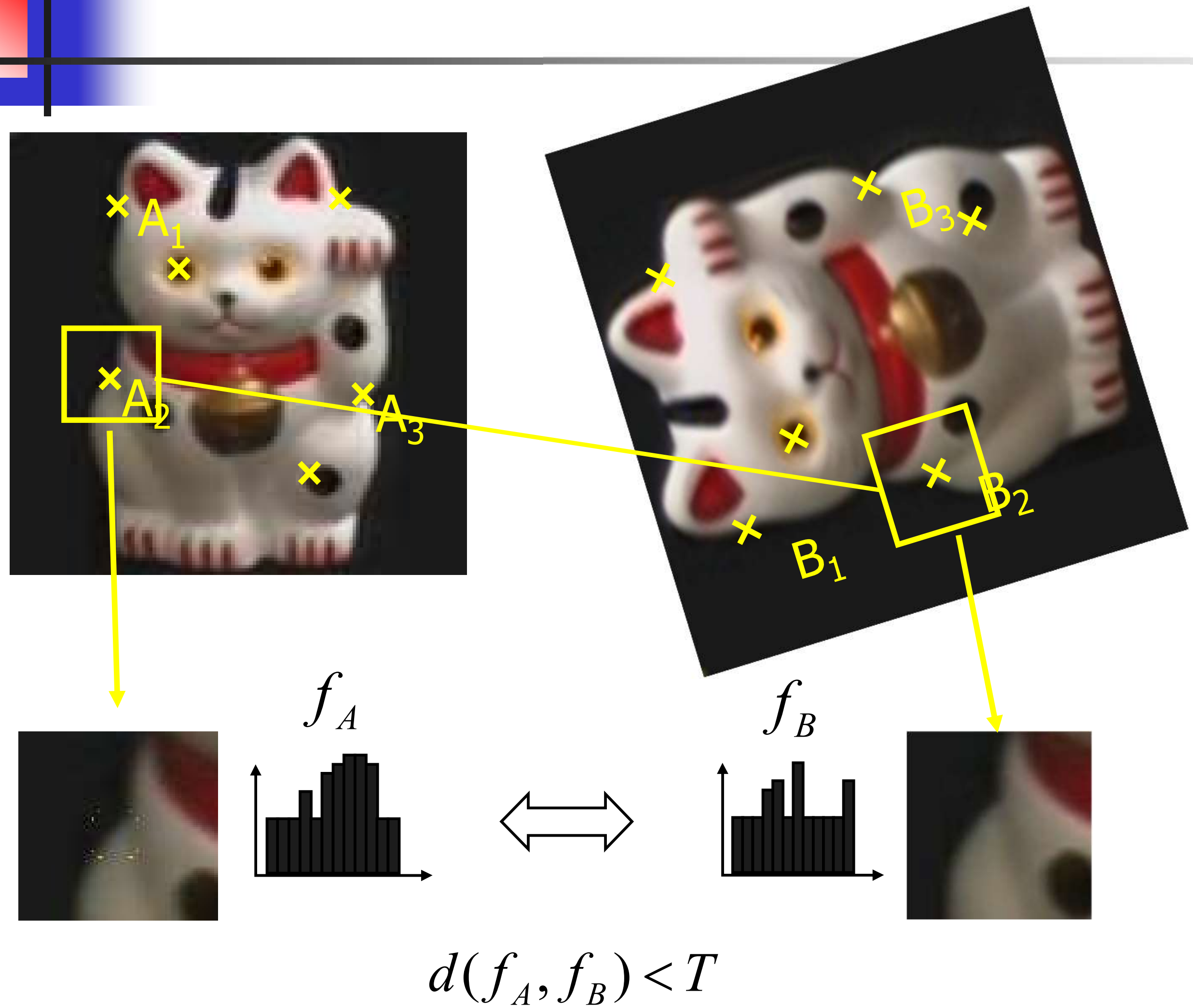
Lecture 7: Feature Matching

Correspondence and alignment

- Correspondence: matching points, patches, edges, or regions across images



Overview of Keypoint Matching

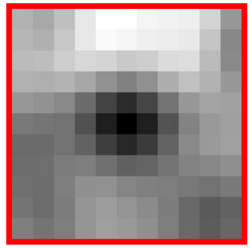


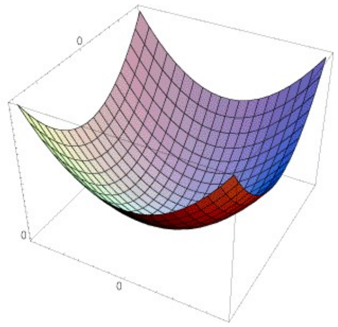
1. Find a set of distinctive key-points
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

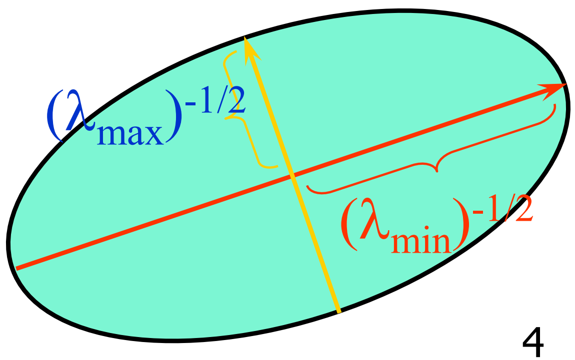
Review: Harris corner detector

- Approximate distinctiveness by local auto-correlation.
- Approximate local auto-correlation by second moment matrix
- Quantify distinctiveness (or cornerness) as function of the eigenvalues of the second moment matrix.
- But we don’t actually need to compute the eigenvalues by using the determinant and trace of the second moment matrix.

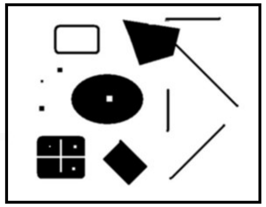
$E(u, v)$







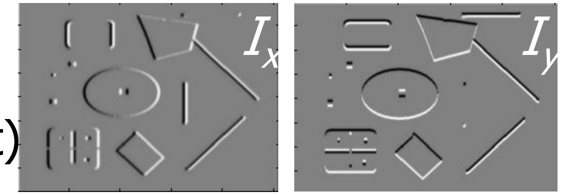
Harris Detector



- Second moment matrix

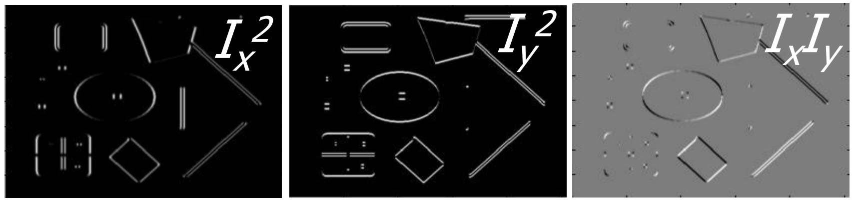
$$\mu(\sigma_I,\sigma_D)=g(\sigma_I)*\begin{bmatrix} I_x^2(\sigma_D) & I_xI_y(\sigma_D) \\ I_xI_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives (optionally, blur first)



$$\det M = \lambda_1\lambda_2$$
$$\operatorname{trace} M = \lambda_1+\lambda_2$$

2. Square of derivatives



3. Gaussian filter $g(\sigma_I)$

4. Cornerness function – both eigenvalues are strong

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

5. Non-maxima suppression

So far: can localize in x-y, but not scale

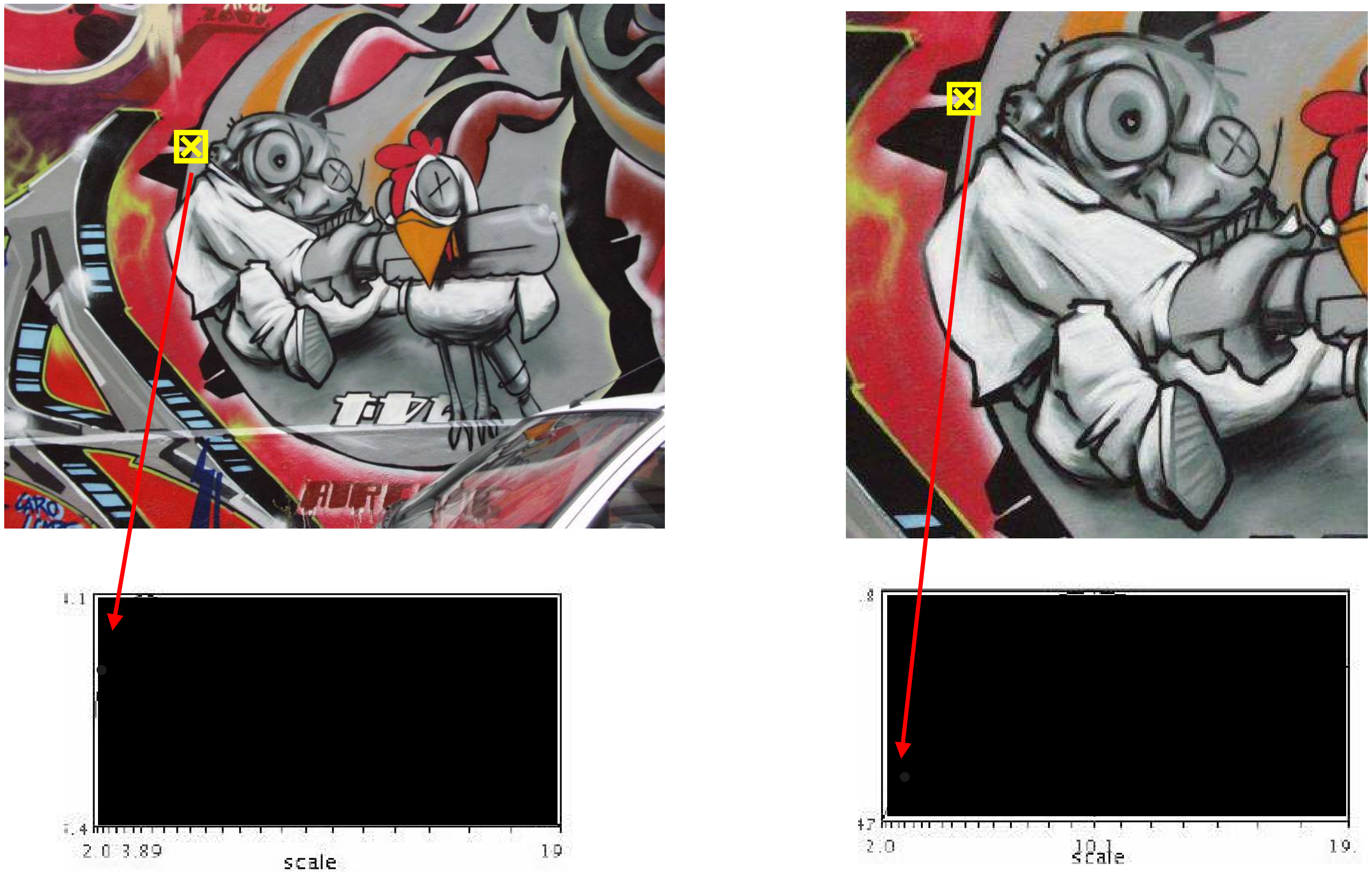
Automatic Scale Selection

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

How to find corresponding patch sizes?

Automatic Scale Selection

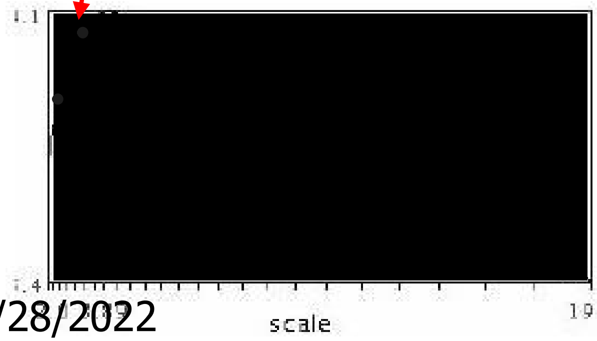
- Function responses for increasing scale (scale signature)



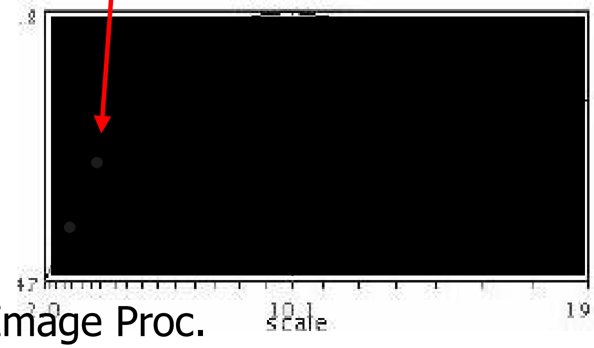
$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

Function responses for increasing scale (scale signature)



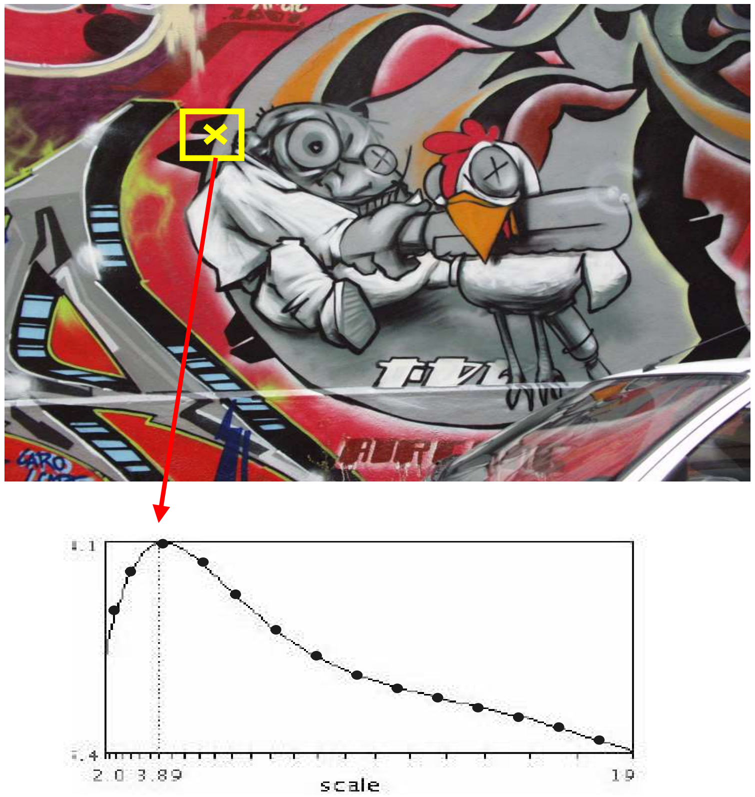
$f(I_{i_1 \ldots i_m}(x,\sigma))$



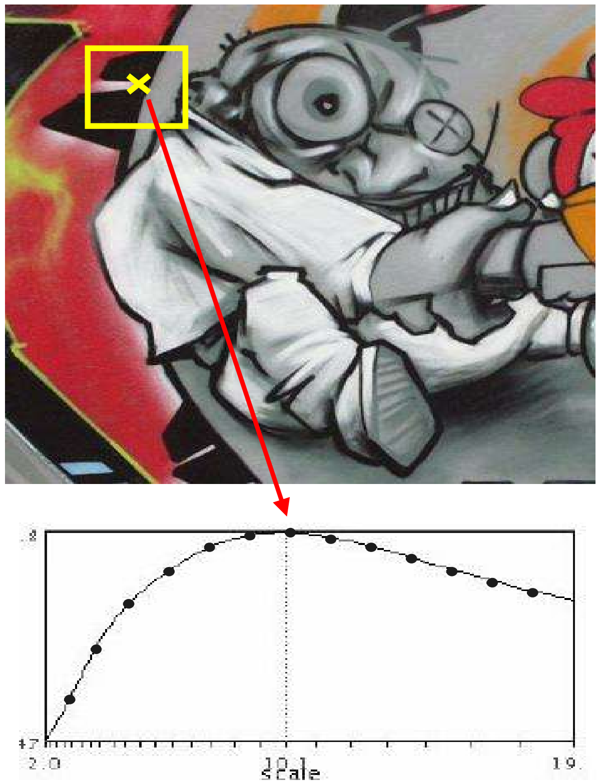
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



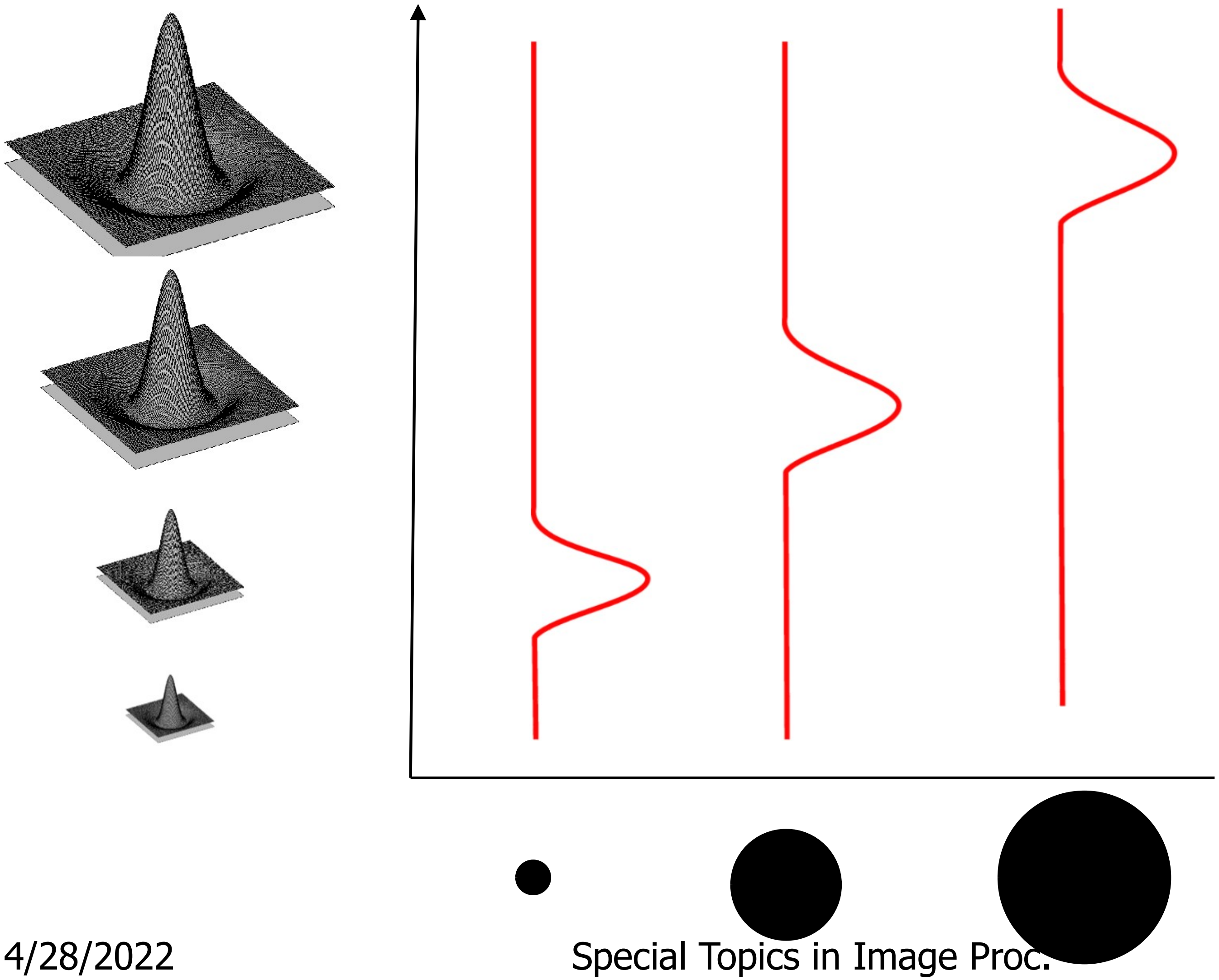
$f(I_{i_1 \ldots i_m}(x, \sigma))$



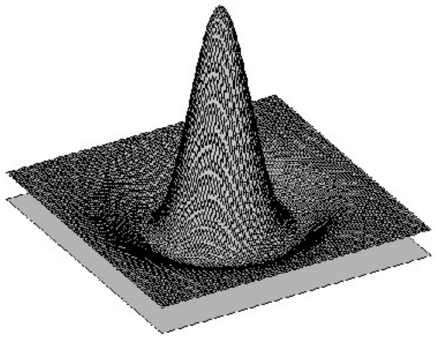
$f(I_{i_1 \ldots i_m}(x', \sigma'))$

What Is A Useful Signature Function?

- Difference-of-Gaussian = “blob” detector

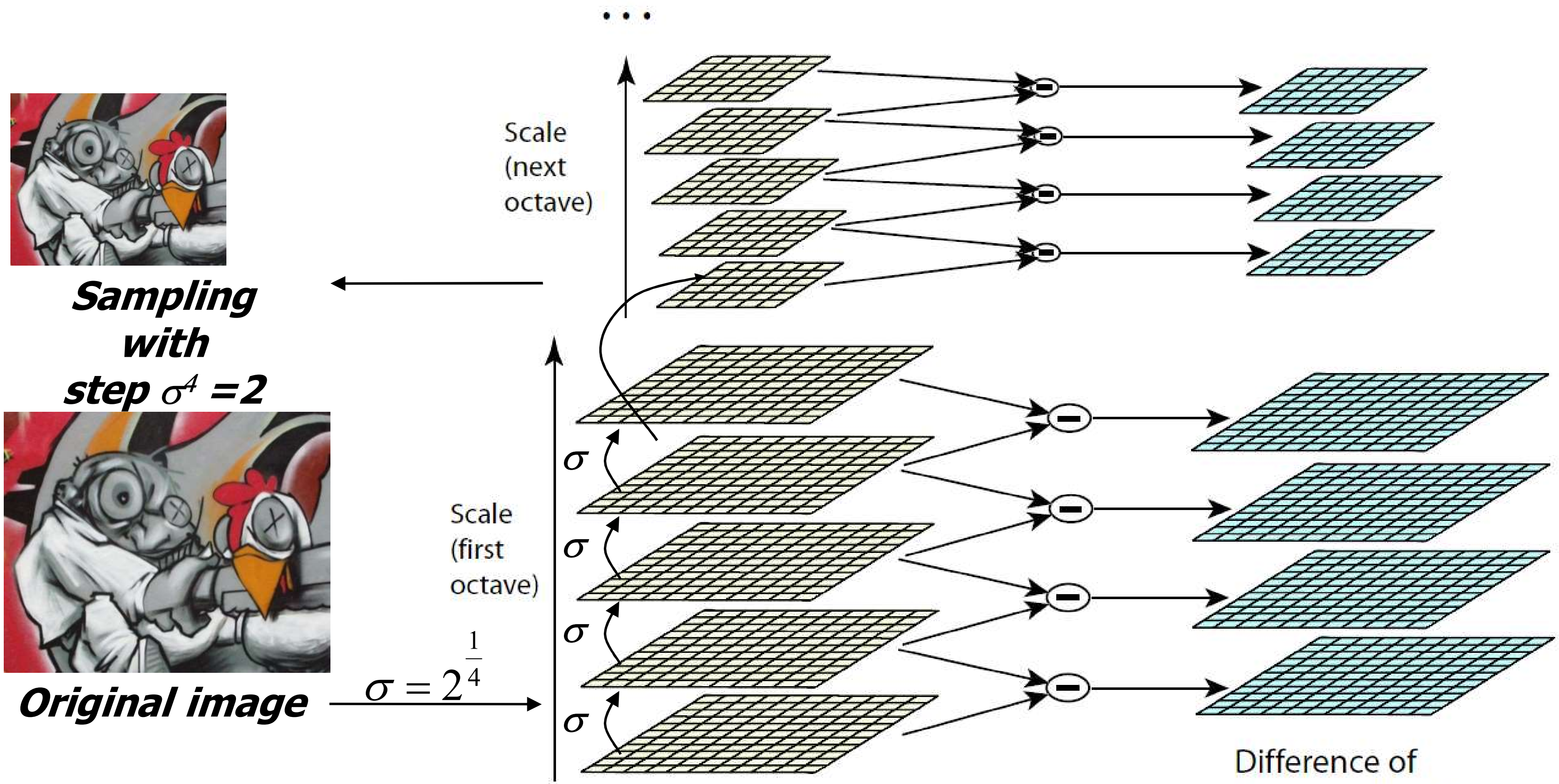


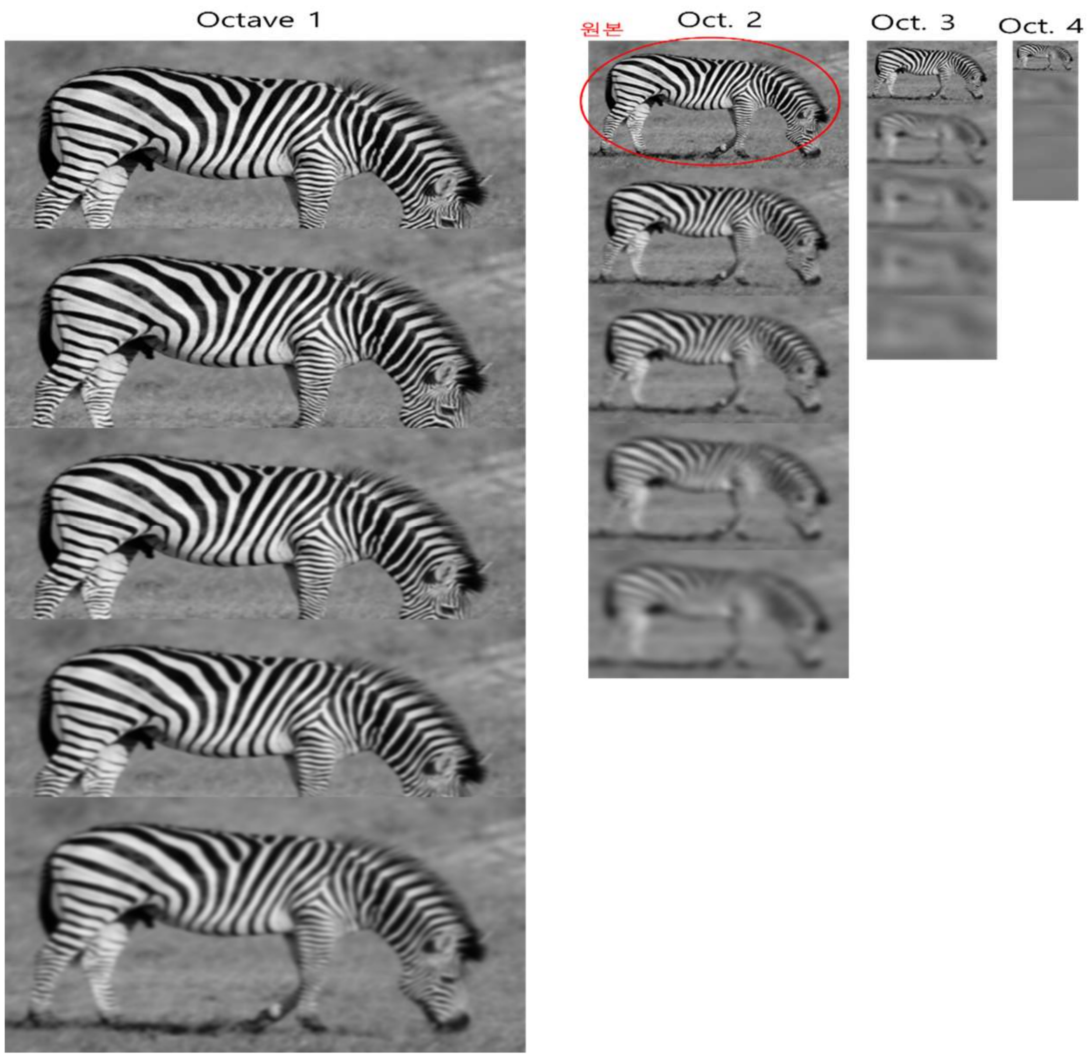
Difference-of-Gaussian (DoG)

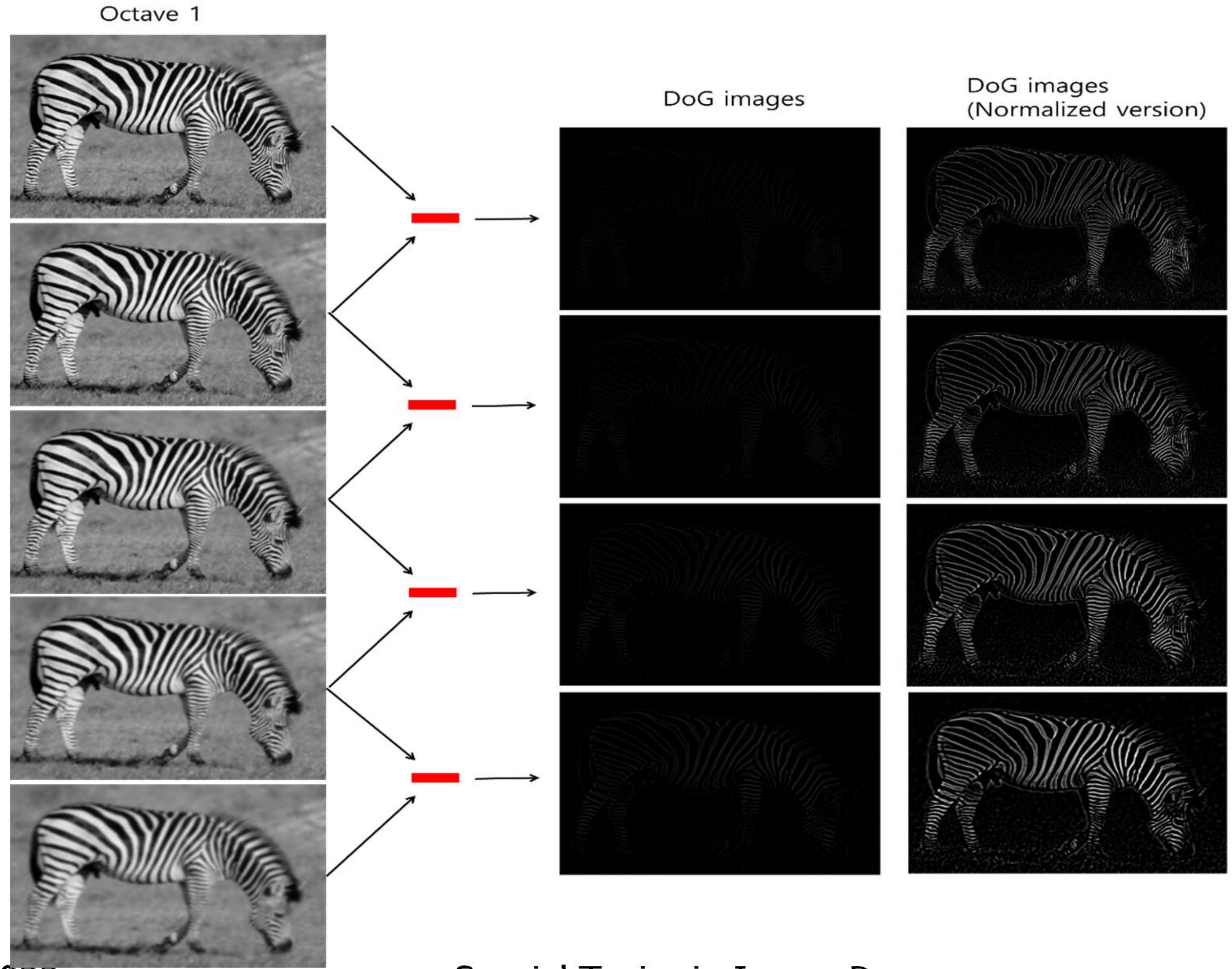


DoG – Efficient Computation

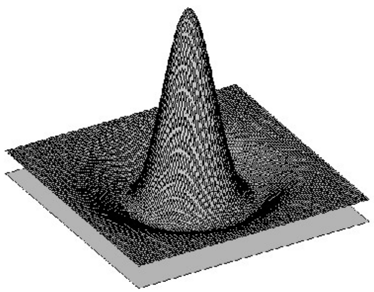
- Computation in Gaussian scale pyramid

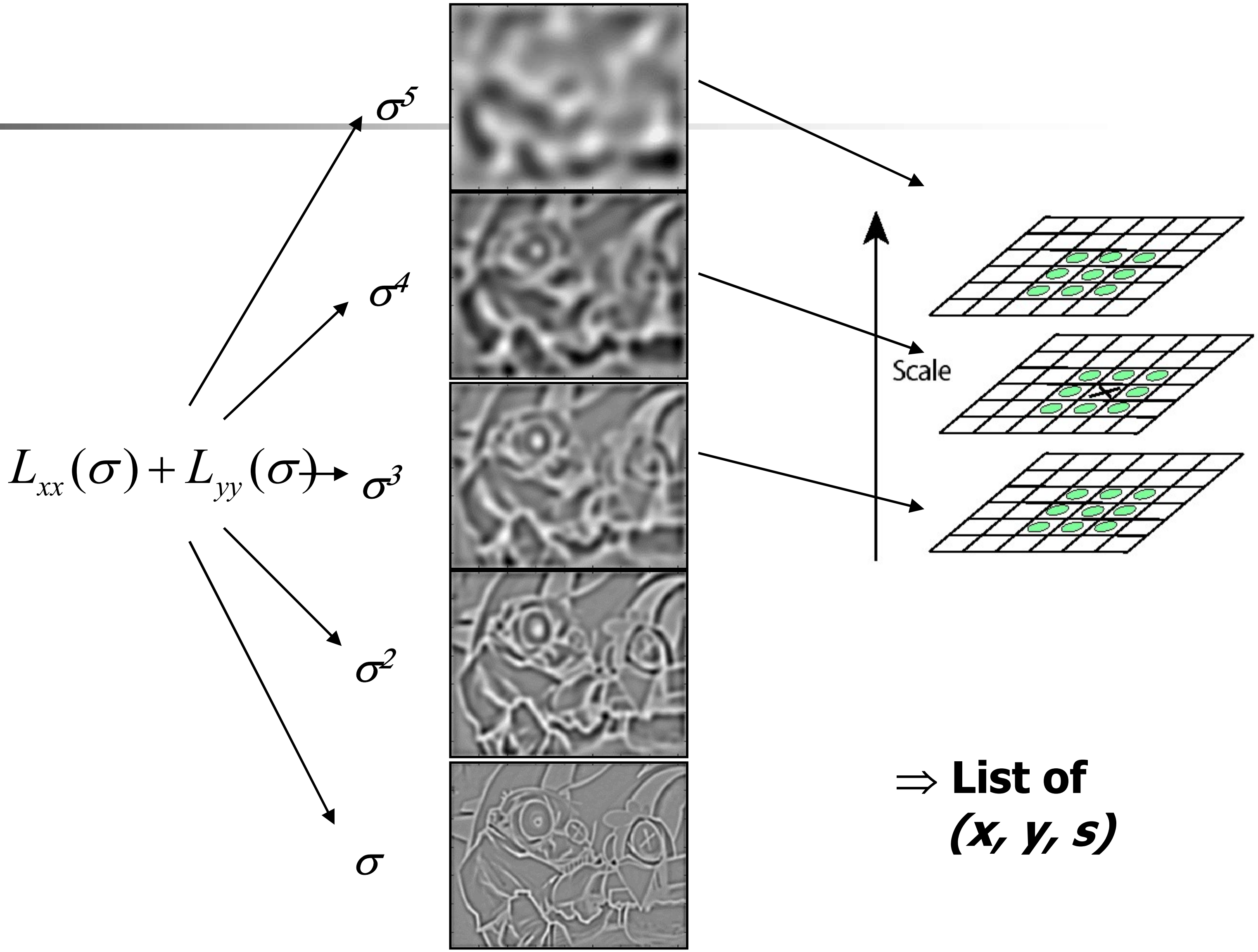



Octave 1
원본
Oct. 2
Oct. 3
Oct. 4


Octave 1
DoG images
DoG images
(Normalized version)

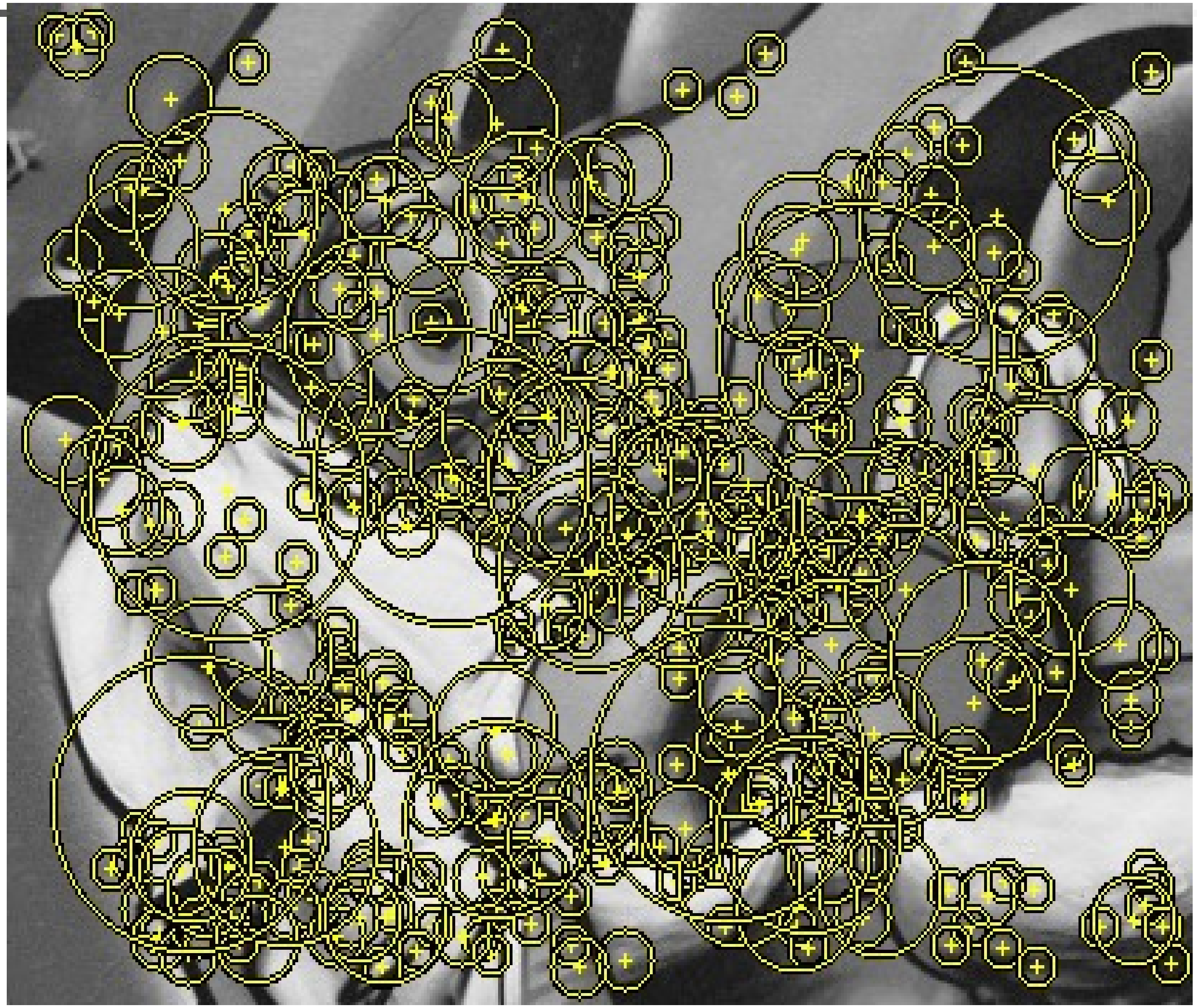
Find local maxima in position-scale space of Difference-of-Gaussian





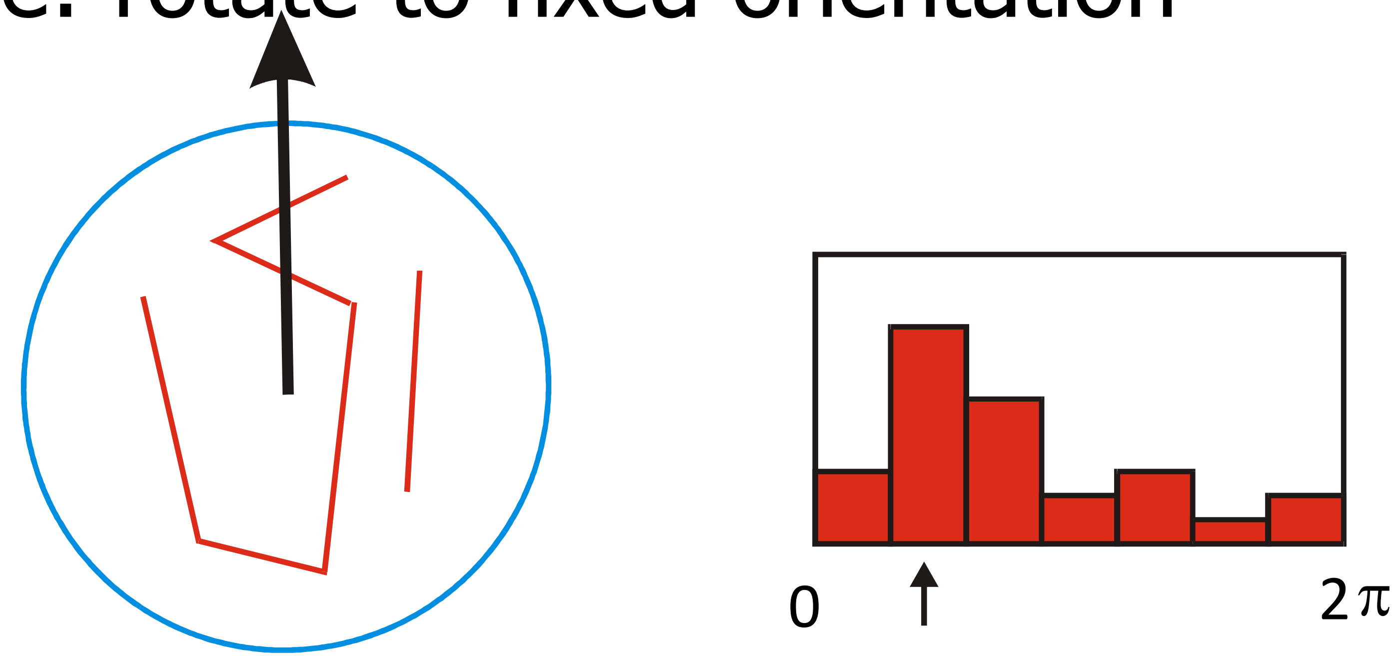
$\Rightarrow$ **List of *(x, y, s)***

Results: Difference-of-Gaussian



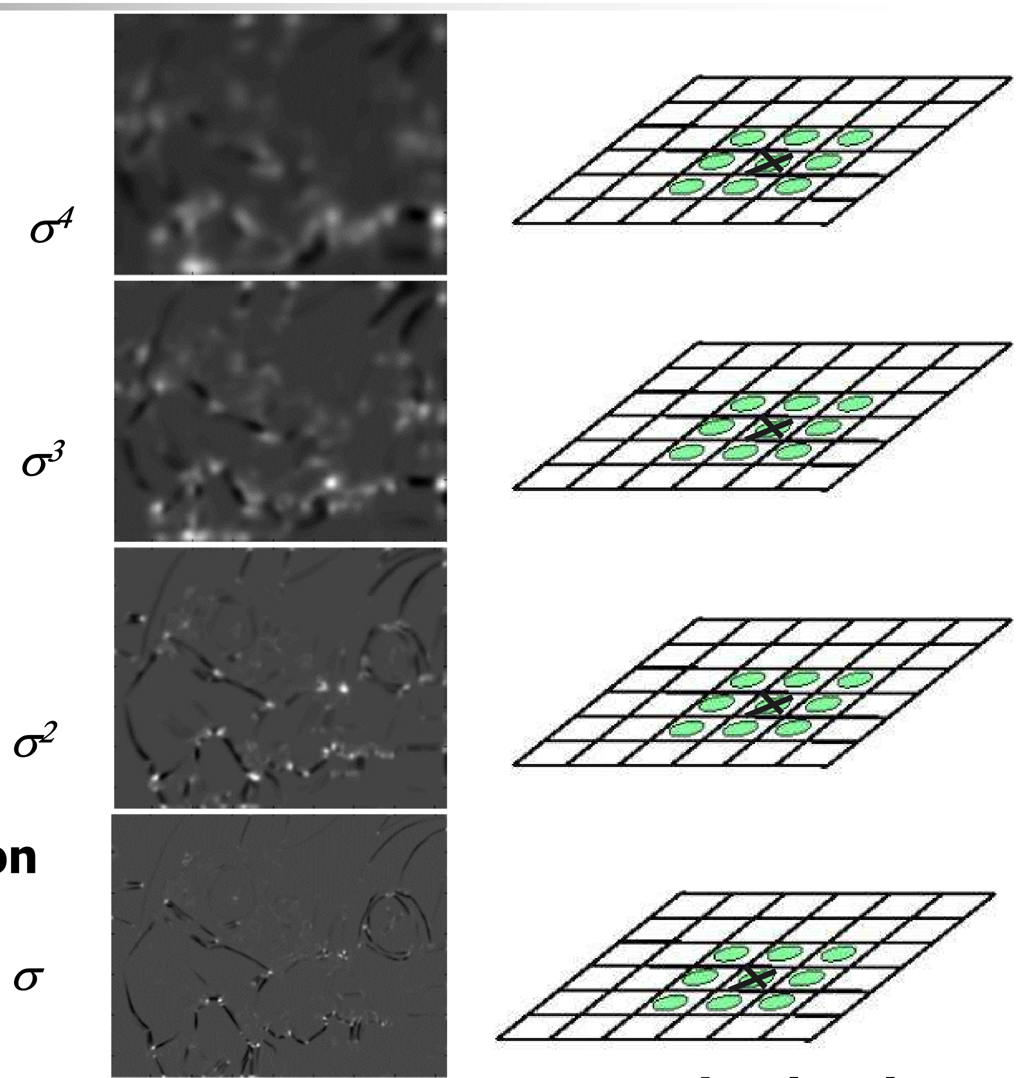
Orientation Normalization

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



Harris-Laplace

1. Initialization: Multiscale Harris corner detection



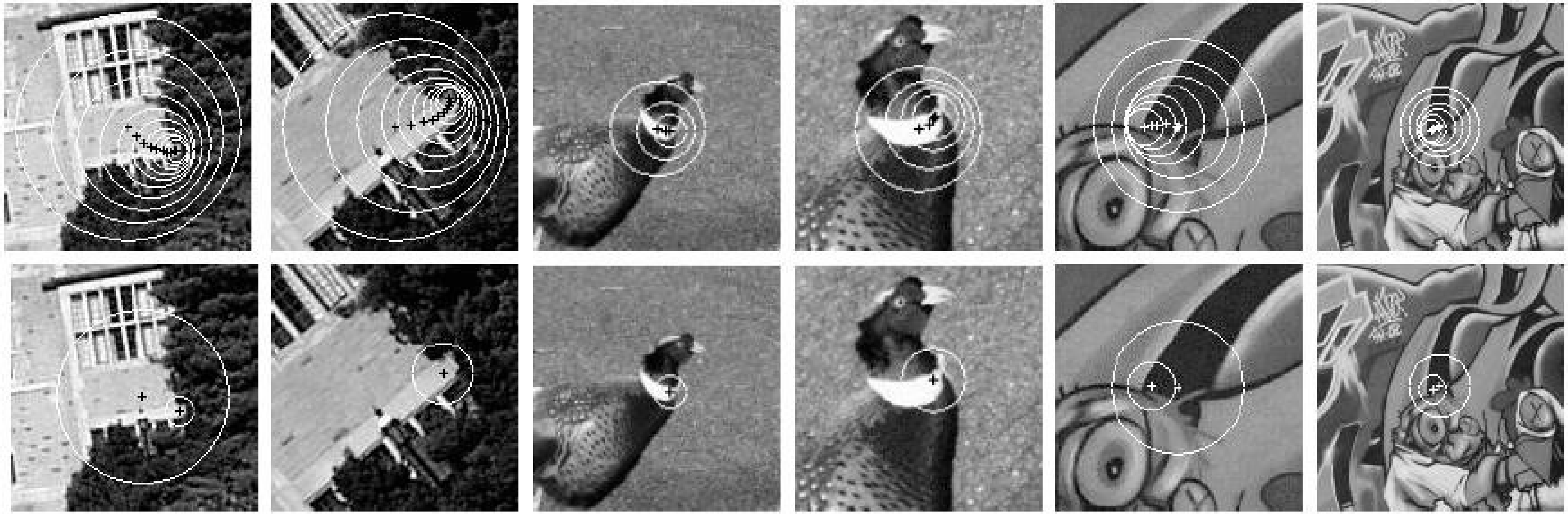
Computing Harris function

Detecting local maxima

Harris-Laplace

1. Initialization: Multiscale Harris corner detection
2. Scale selection based on Laplacian
   (same procedure with Hessian $\Rightarrow$ Hessian-Laplace)

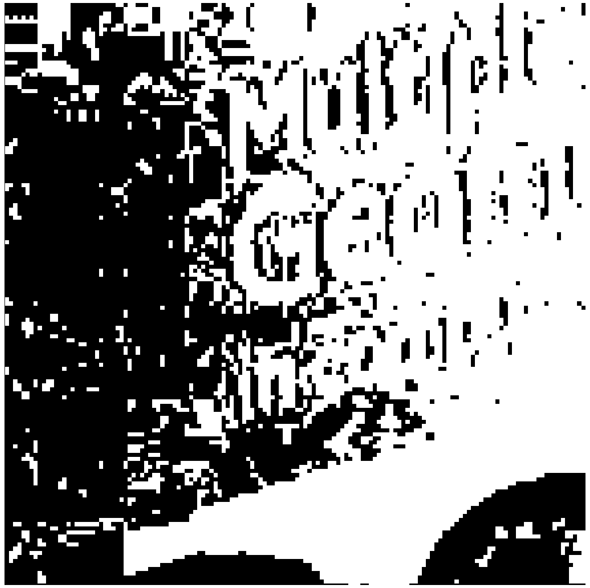
Harris points



Harris-Laplace points

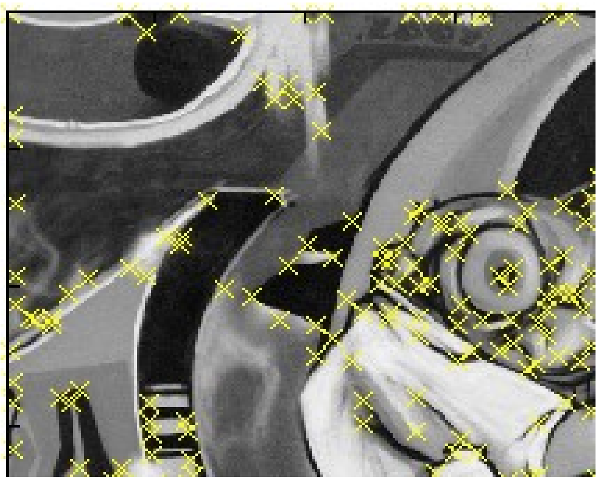
Maximally Stable Extremal Regions(MSER)

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



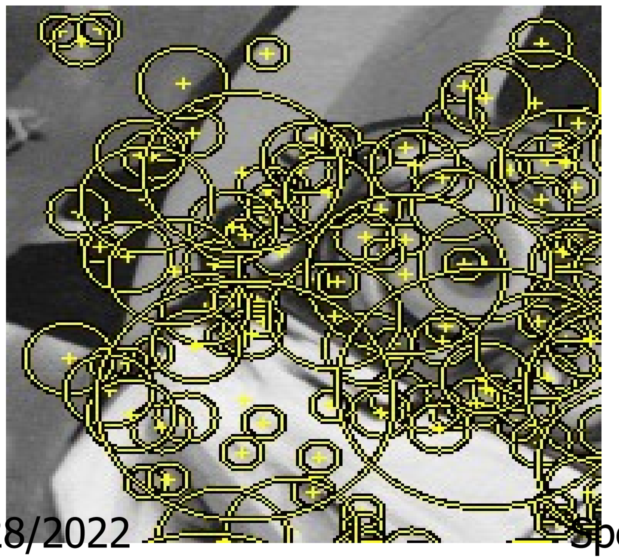
Example Results: MSER

Comparison



Harris

Hessian



LoG

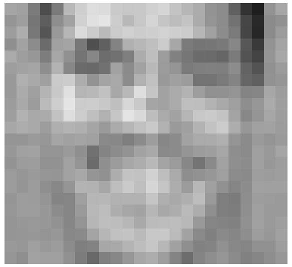
MSER

Available at a web site

- For most local feature detectors, executables are 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

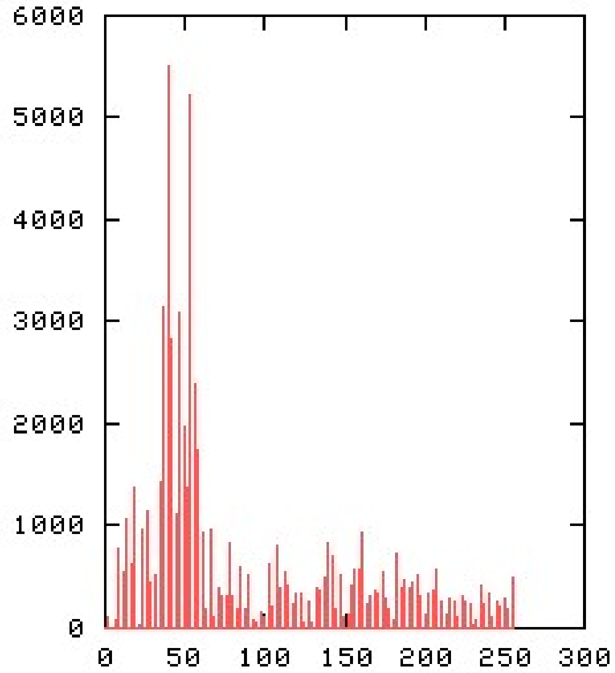
Image representations

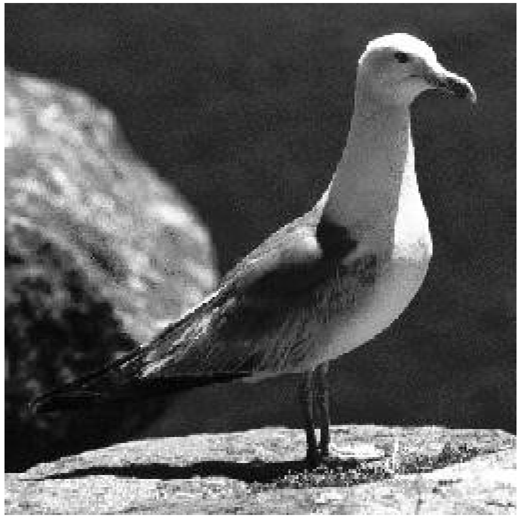
- Templates
  - Intensity, gradients, etc.



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

Image Representations: Histograms



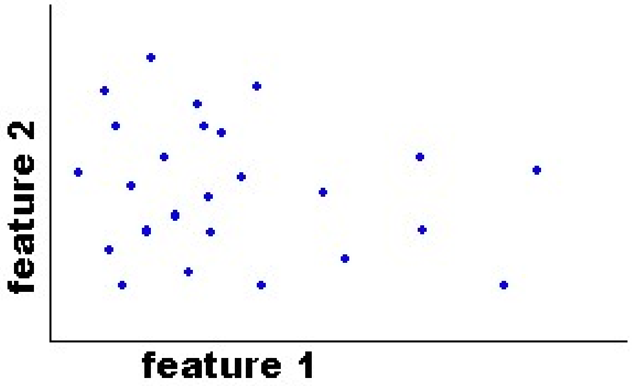


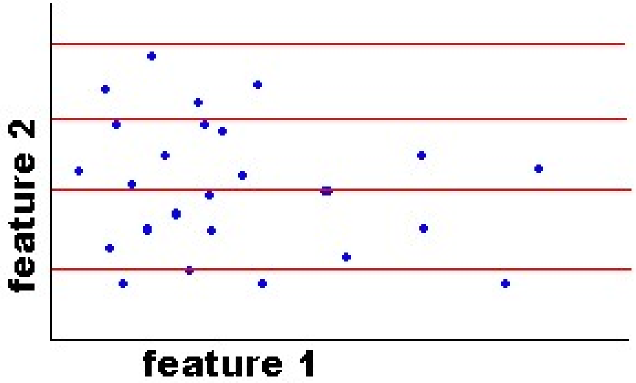
Global histogram

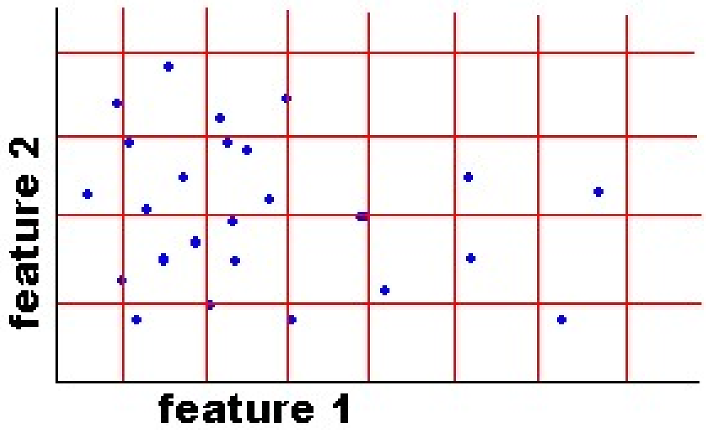
- Represent distribution of features
  - Color, texture, depth, …

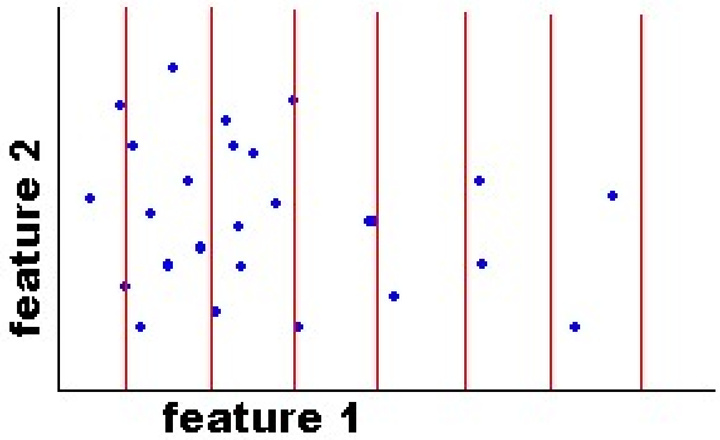
Image Representations: Histograms

Histogram: Probability or count of data in each bin









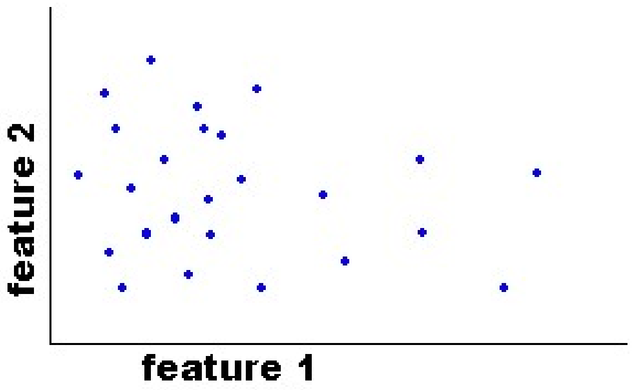
- Joint histogram
  - Requires lots of data
  - Loss of resolution to avoid empty bins

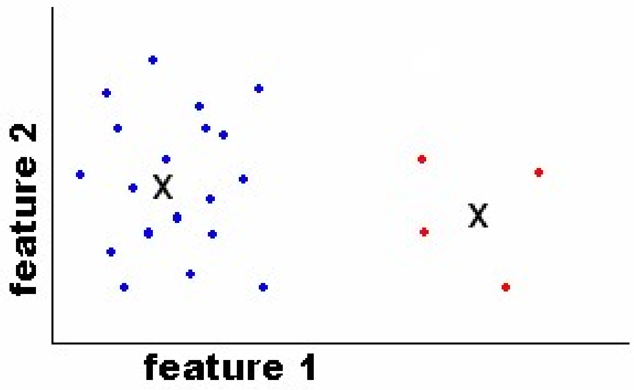
Marginal histogram

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

Image Representations: Histograms

Clustering





Use the same cluster centers for all images

Computing histogram distance

$$\text{histint}(h_i, h_j) = 1 - \sum_{m=1}^{K} \min\left(h_i(m), h_j(m)\right)$$

Histogram intersection (assuming normalized histograms)

$$\chi^2(h_i, h_j) = \frac{1}{2} \sum_{m=1}^{K} \frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)}$$

Chi-squared Histogram matching distance

Cars found by color histogram matching using chi-squared

Histograms: Implementation issues

- Quantization
  - Grids: fast but applicable only with few dimensions
  - Clustering: slower but can quantize data in higher dimensions

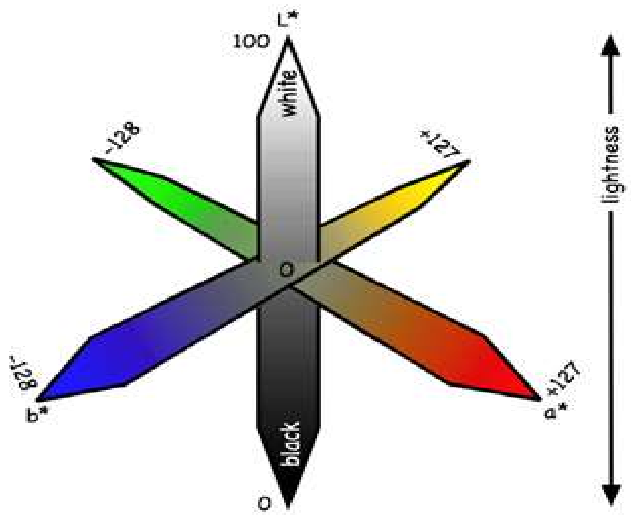
Few Bins
Need less data
Coarser representation

Many Bins
Need more data
Finer representation

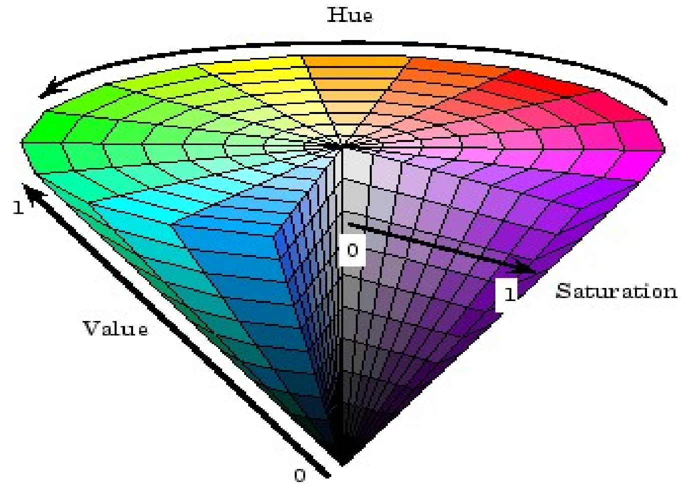
- Matching
  - Histogram intersection or Euclidean may be faster
  - Chi-squared often works better
  - Earth mover’s distance is good for when nearby bins represent similar values

What kind of things do we compute histograms of?

- Color



L*a*b* color space

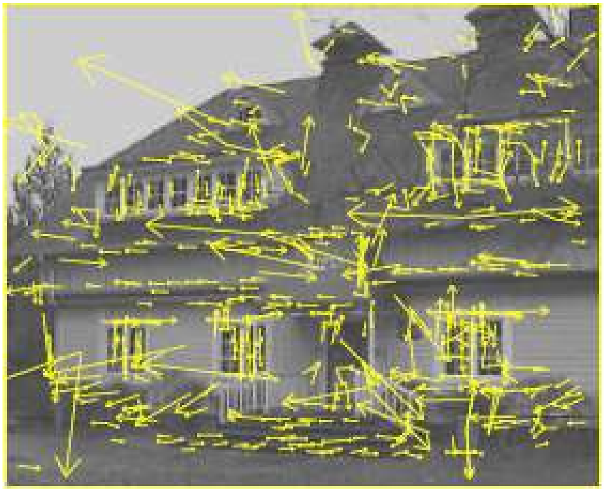


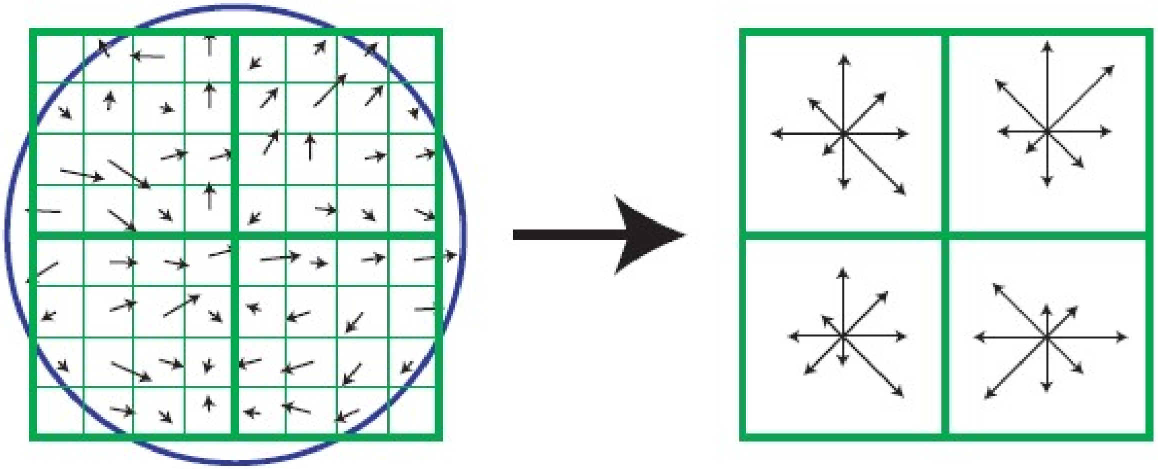
HSV color space

- Texture (filter banks or HOG over regions)

What kind of things do we compute histograms of?

- Histograms of oriented gradients

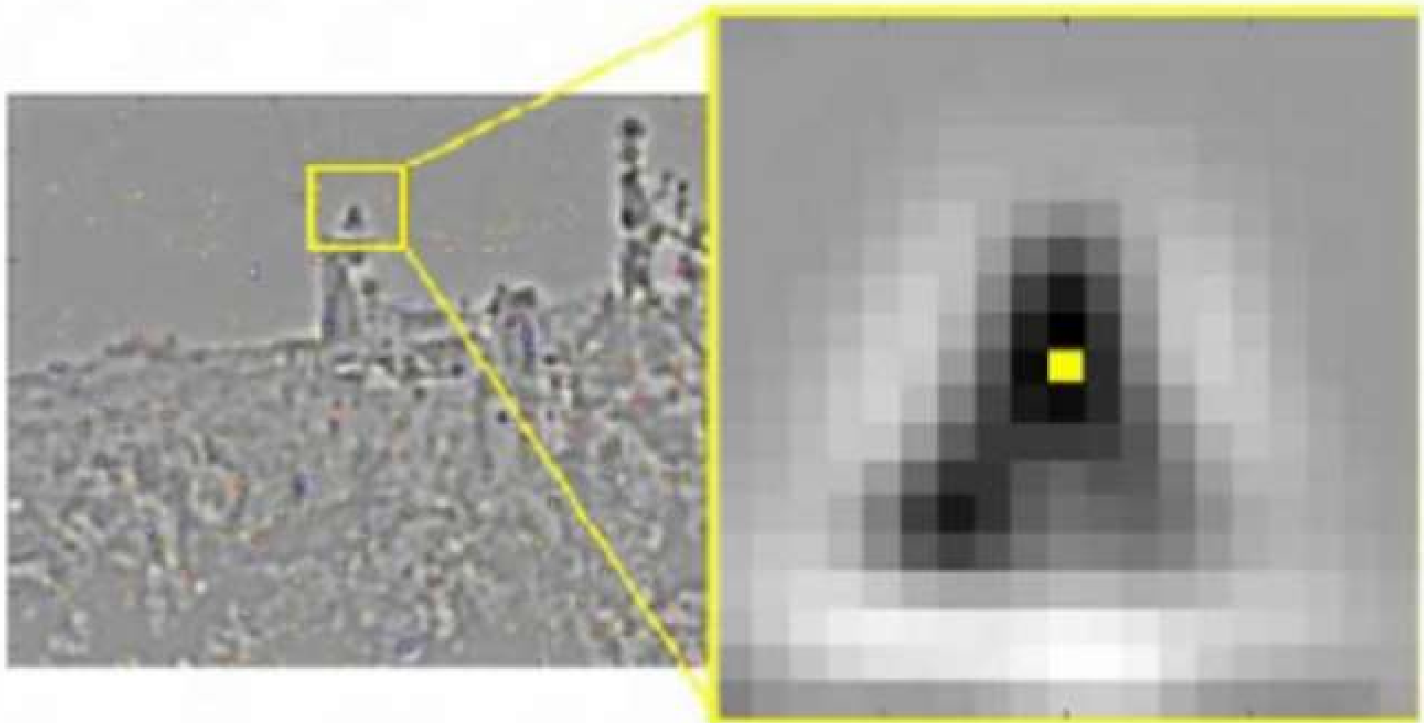




$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

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

How to generate Keypoint descriptor Based on SIFT(1)

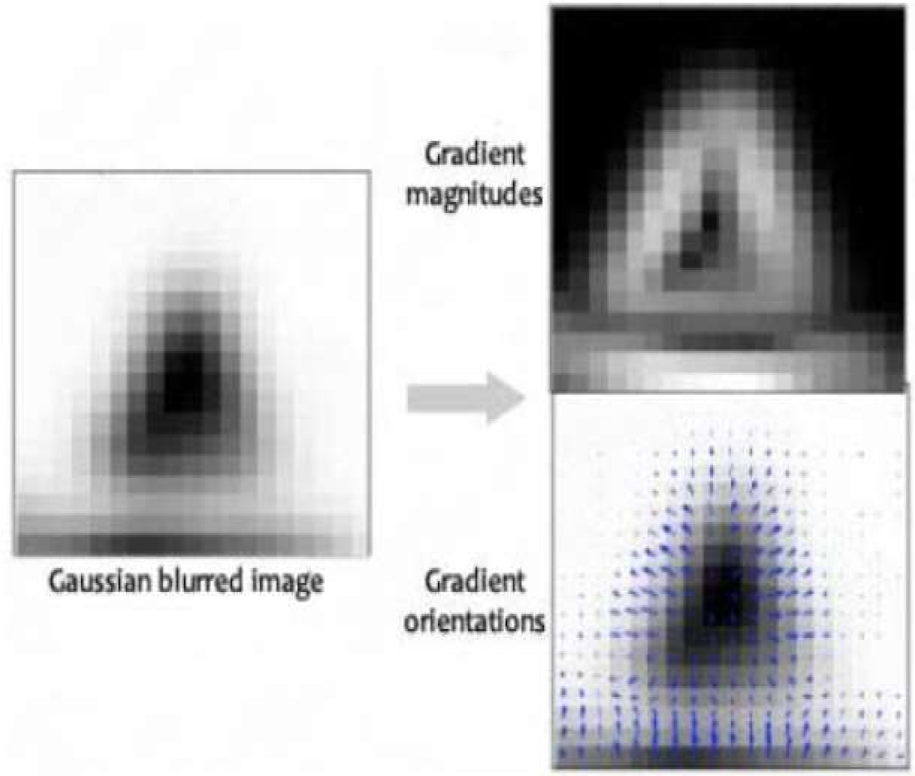


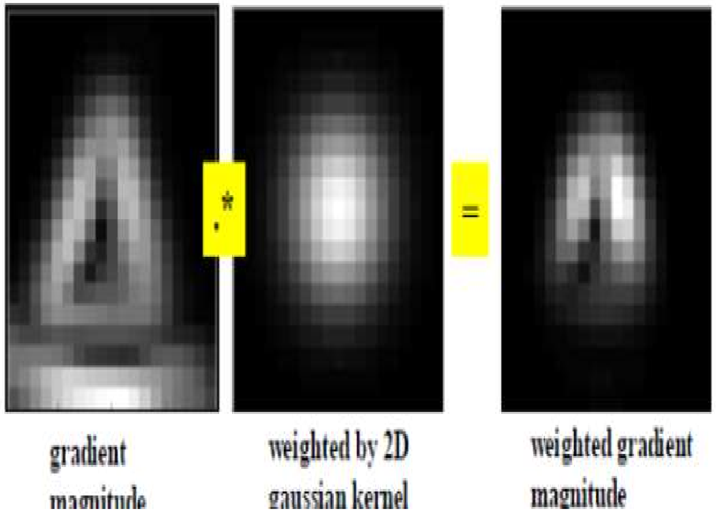
A keypoint

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

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

How to generate Keypoint descriptor Based on SIFT(2)





How to generate Keypoint descriptor based on SIFT(3)

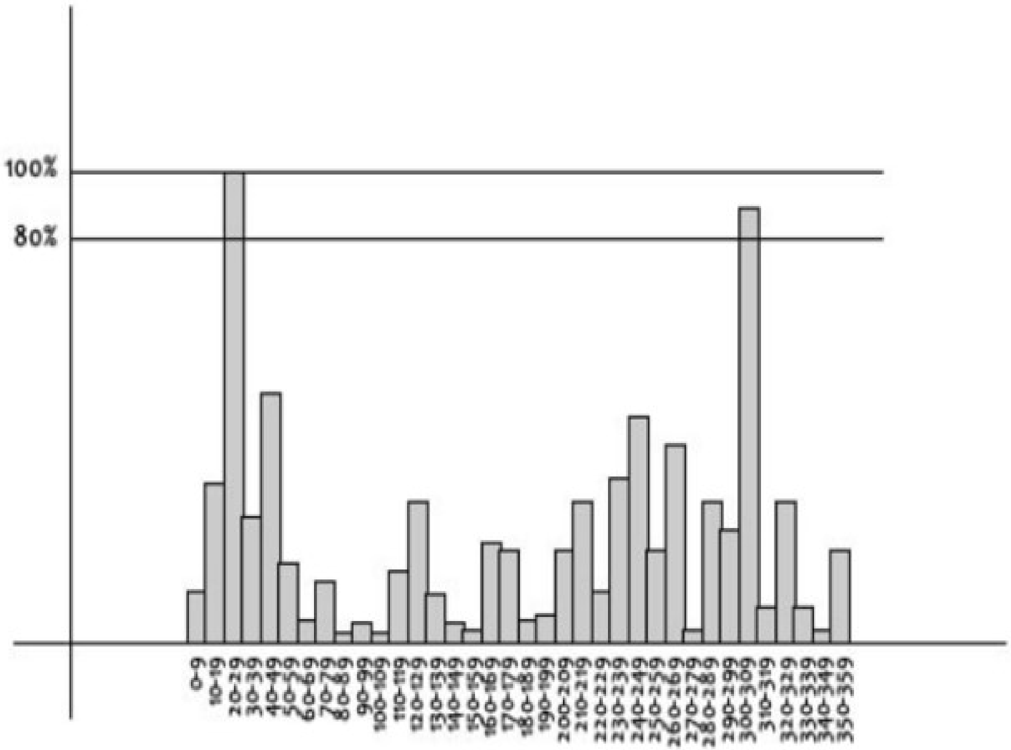


그림 15. keypoint 주변 그레디언트 방향 히스토그램 [1]

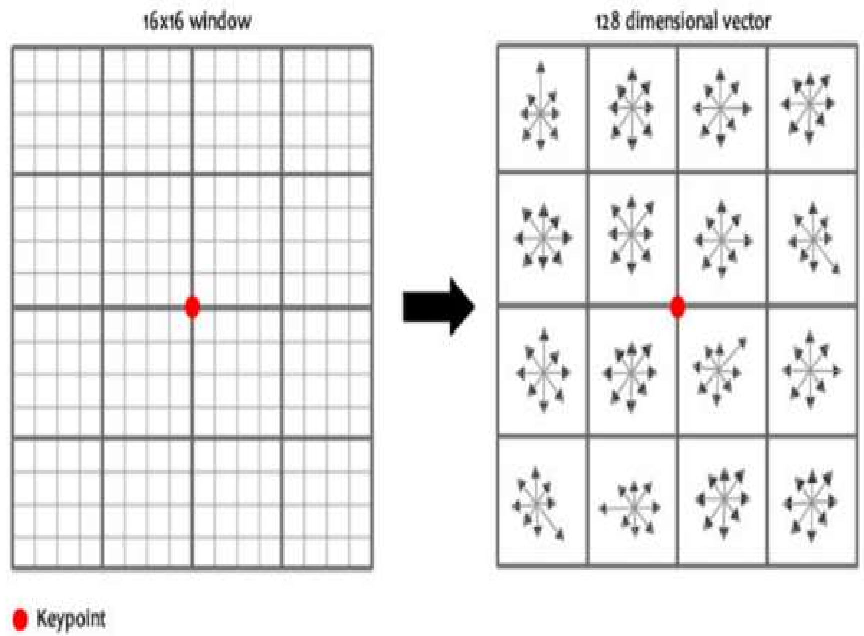


그림 16. 하나의 keypoint의 특징을 나타내는 128개의 숫자를 얻는 과정

SIFT vector formation

- Computed on rotated and scaled version of window according to computed orientation & scale
  - resample the window
- Based on gradients weighted by a Gaussian of variance half the window (for smooth falloff)

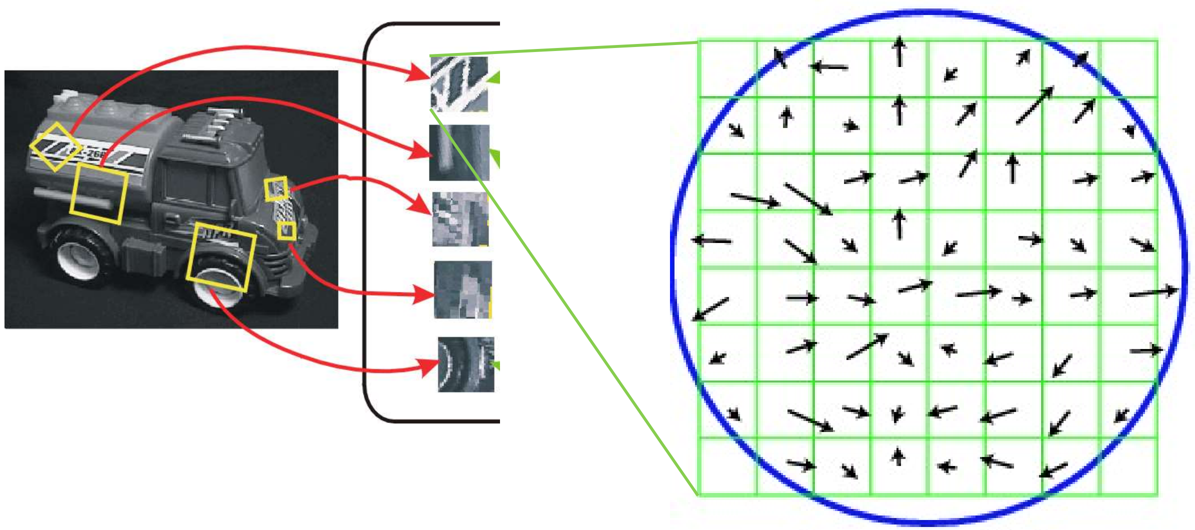
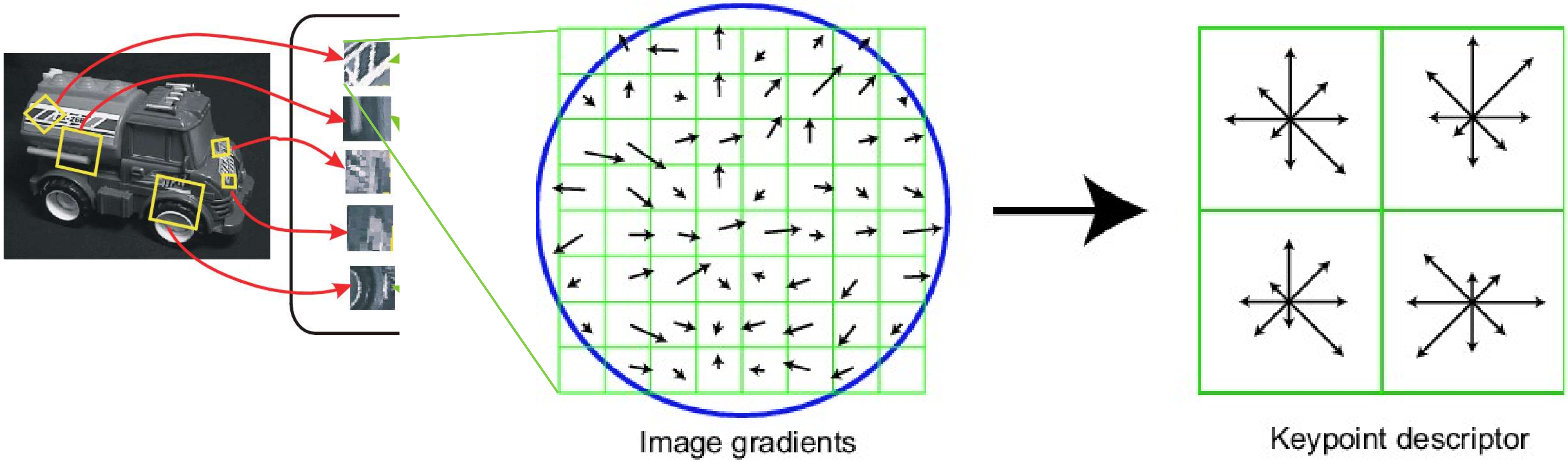


Image gradients

SIFT vector formation

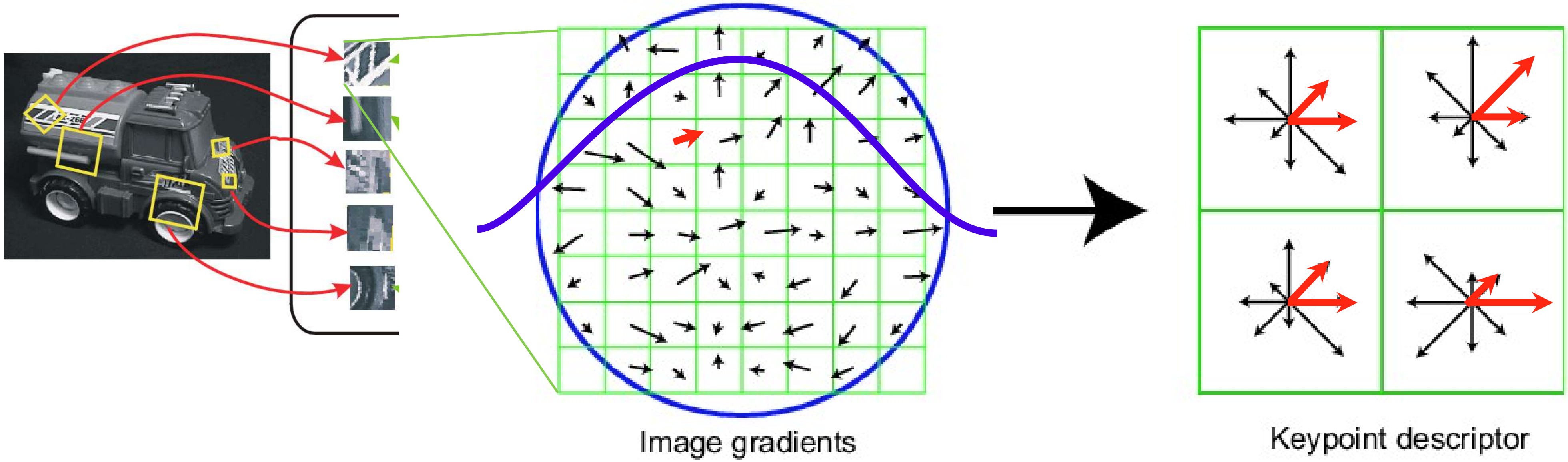
- 4x4 array of gradient orientation histogram weighted by magnitude
- 8 orientations x 4x4 array = 128 dimensions
- Motivation: some sensitivity to spatial layout, but not too much.



showing only 2x2 here but is 4x4

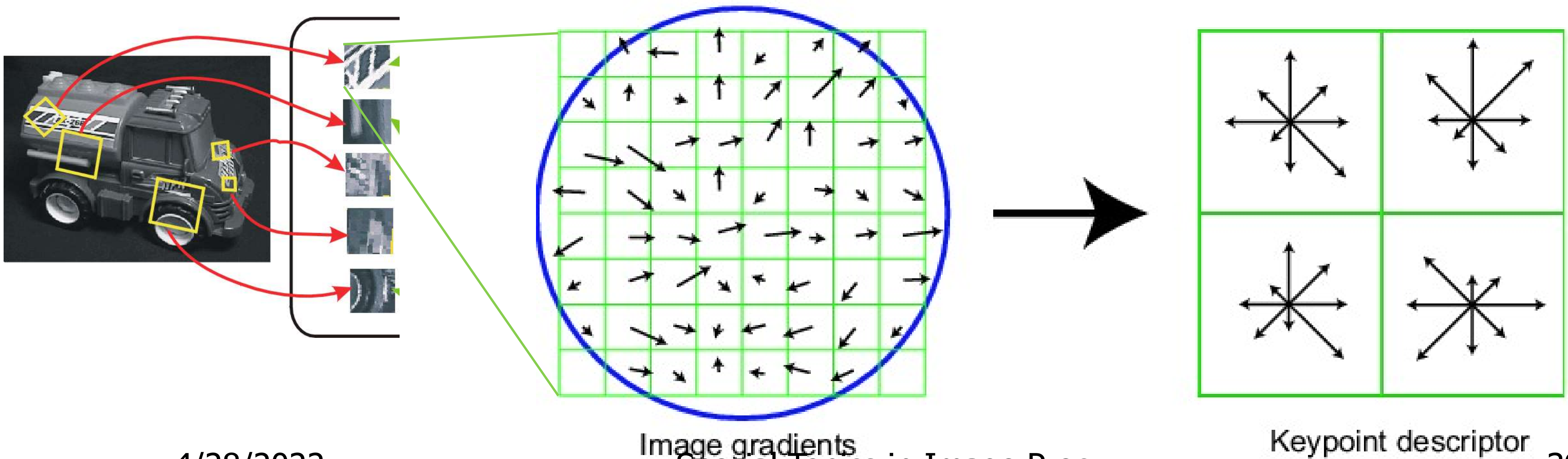
Ensure smoothness

- Gaussian weight
- Trilinear interpolation
  - a given gradient contributes to 8 bins:
    4 in space times 2 in orientation

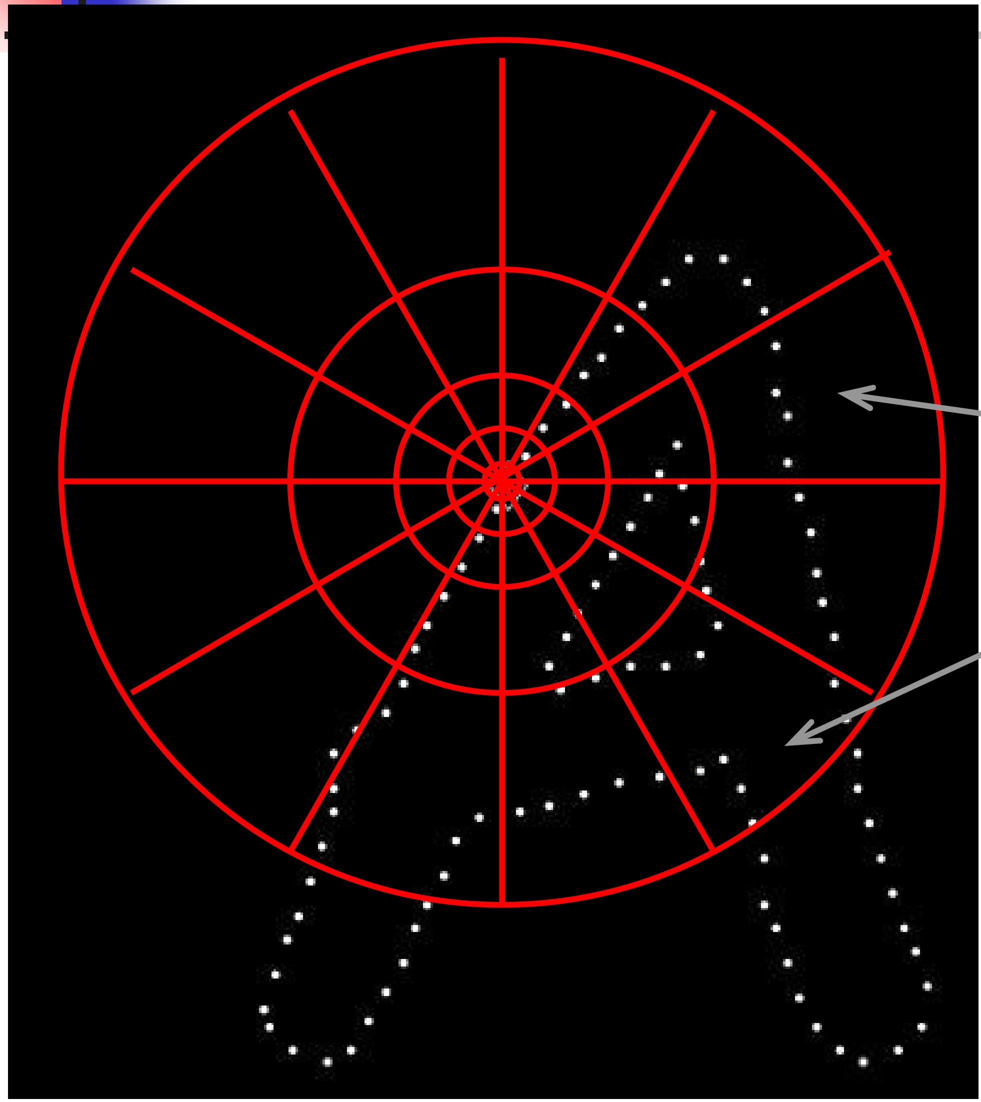


Reduce effect of illumination

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



Local Descriptors: Shape Context



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

Count = 4

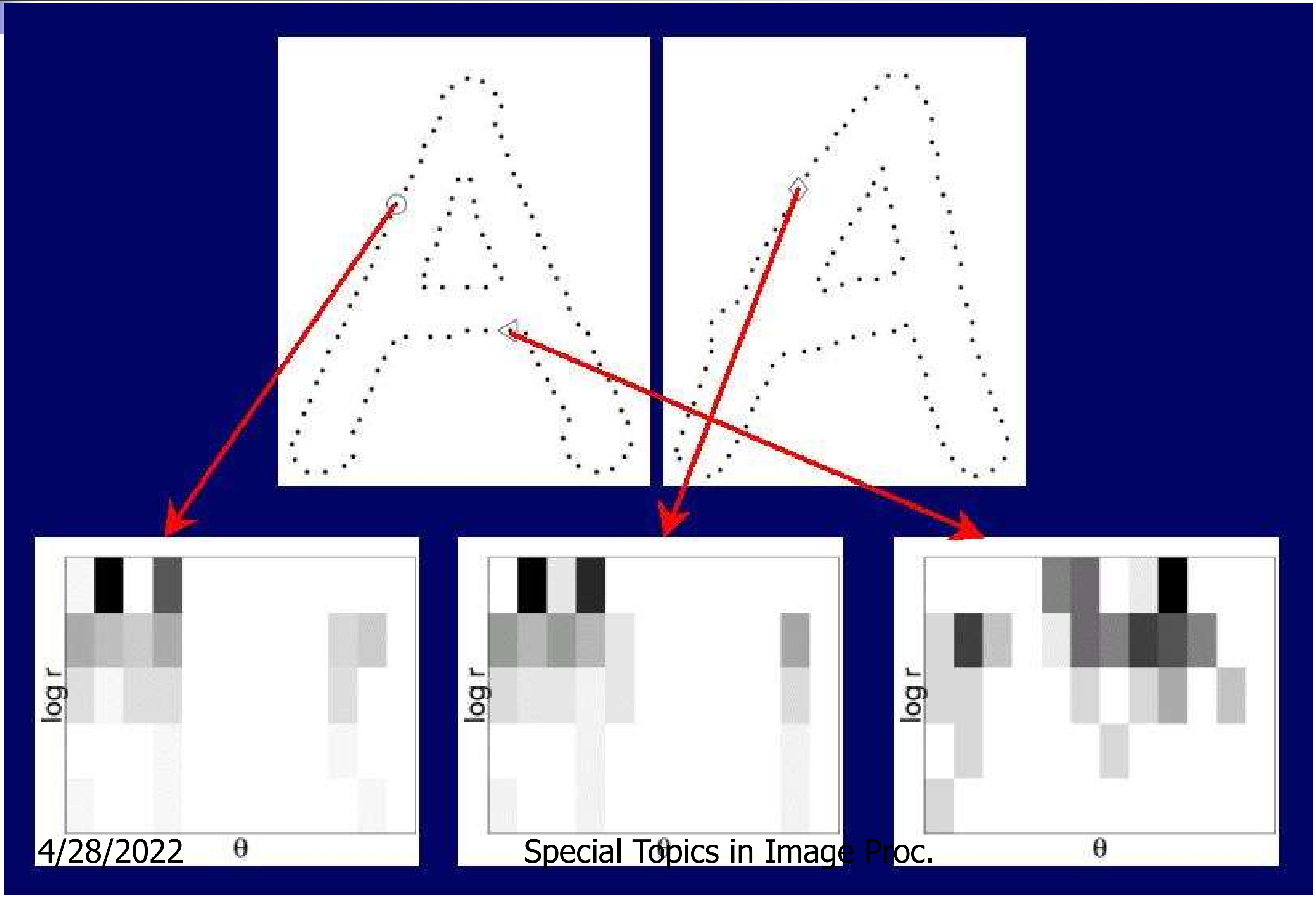
⋮

Count = 10

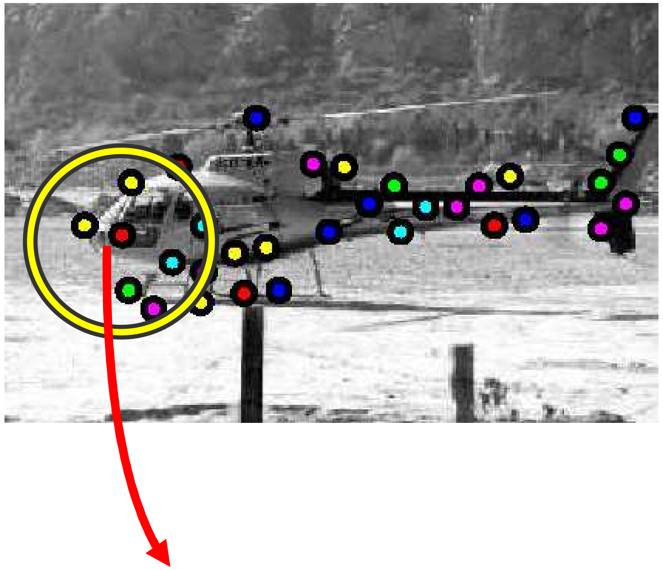
Log-polar binning: more precision for nearby points, more flexibility for farther points.

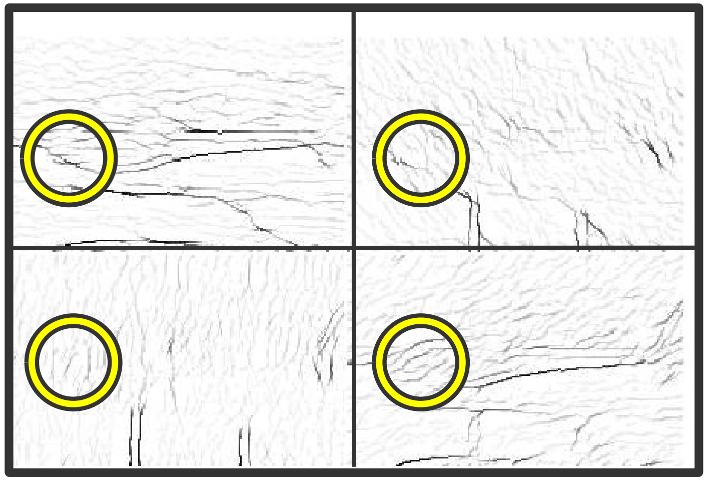
Belongie & Malik, ICCV 2001

Shape Context Descriptor



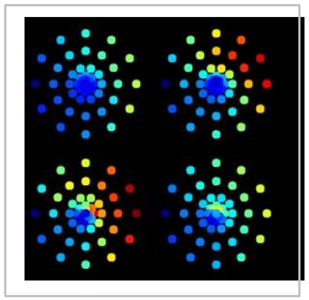
Local Descriptors: Geometric Blur



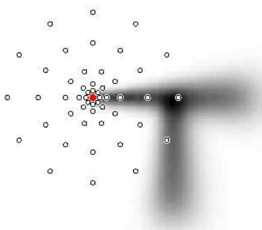


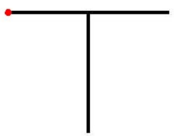
Compute edges at four orientations

Extract a patch in each channel



Example descriptor





Apply spatially varying blur and sub-sample

(Idealized signal)

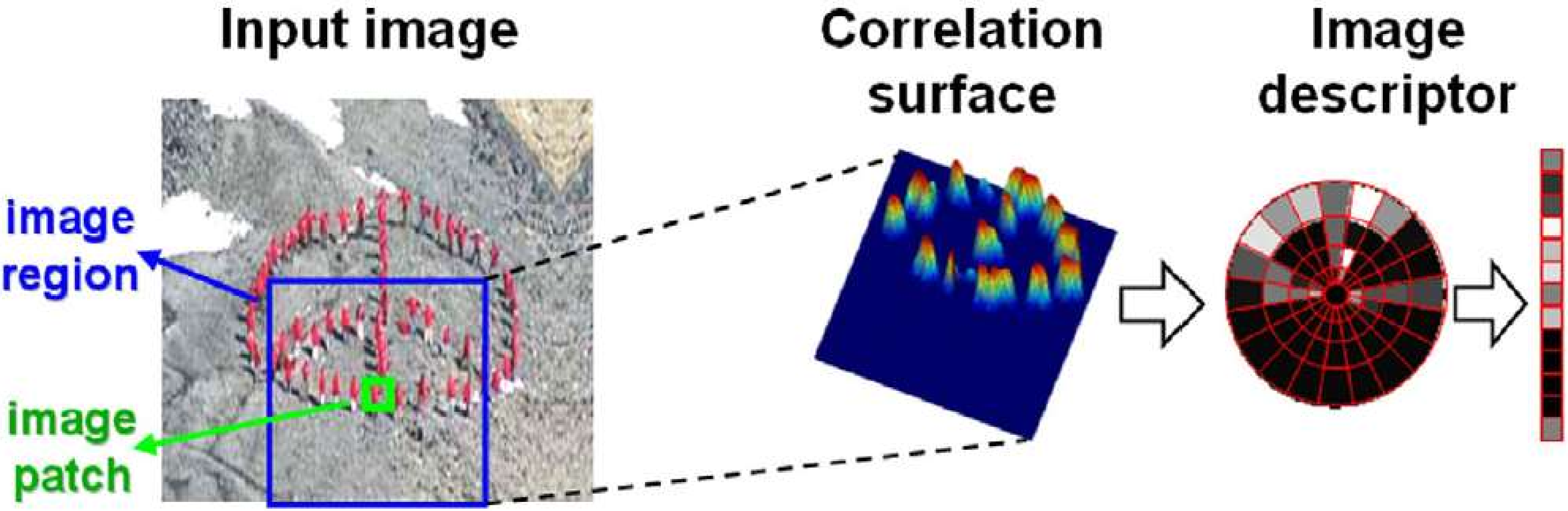
Berg & Malik, CVPR 2001

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.*

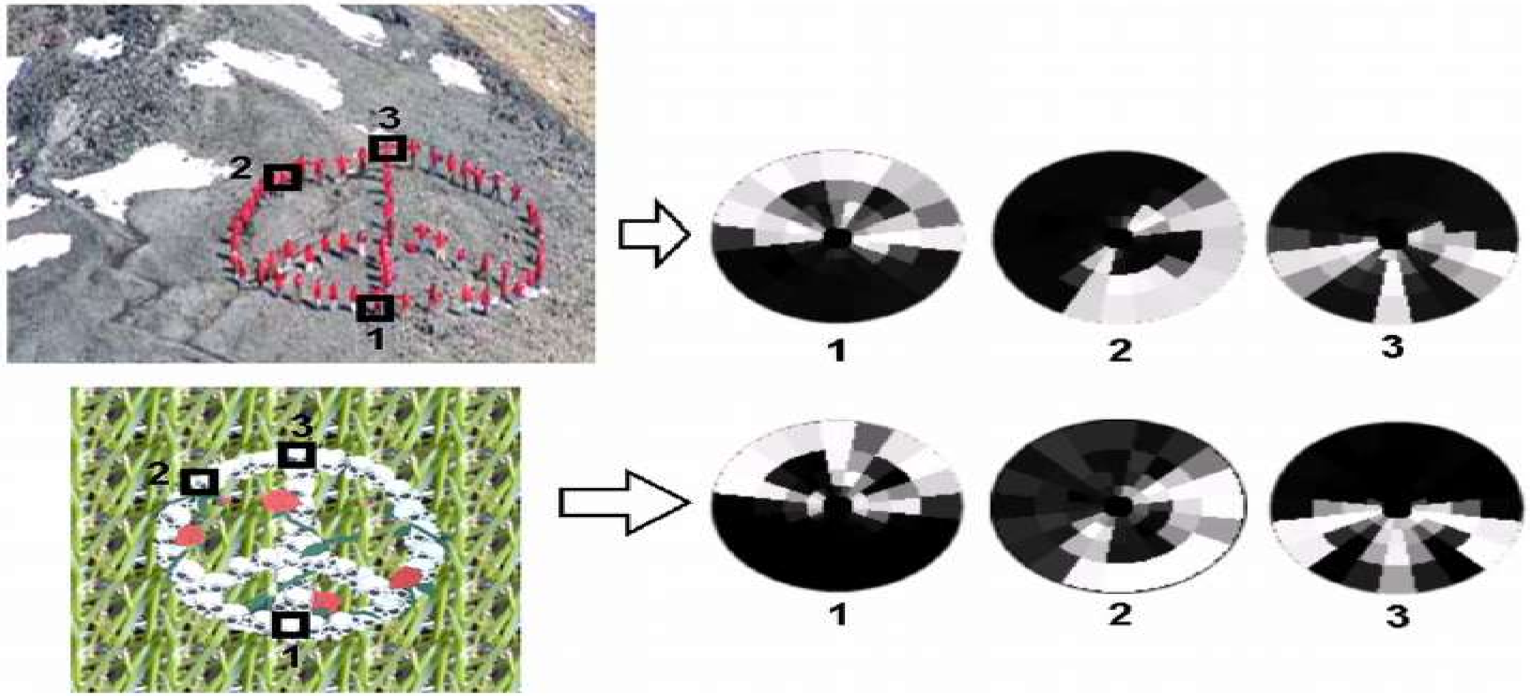
Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

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

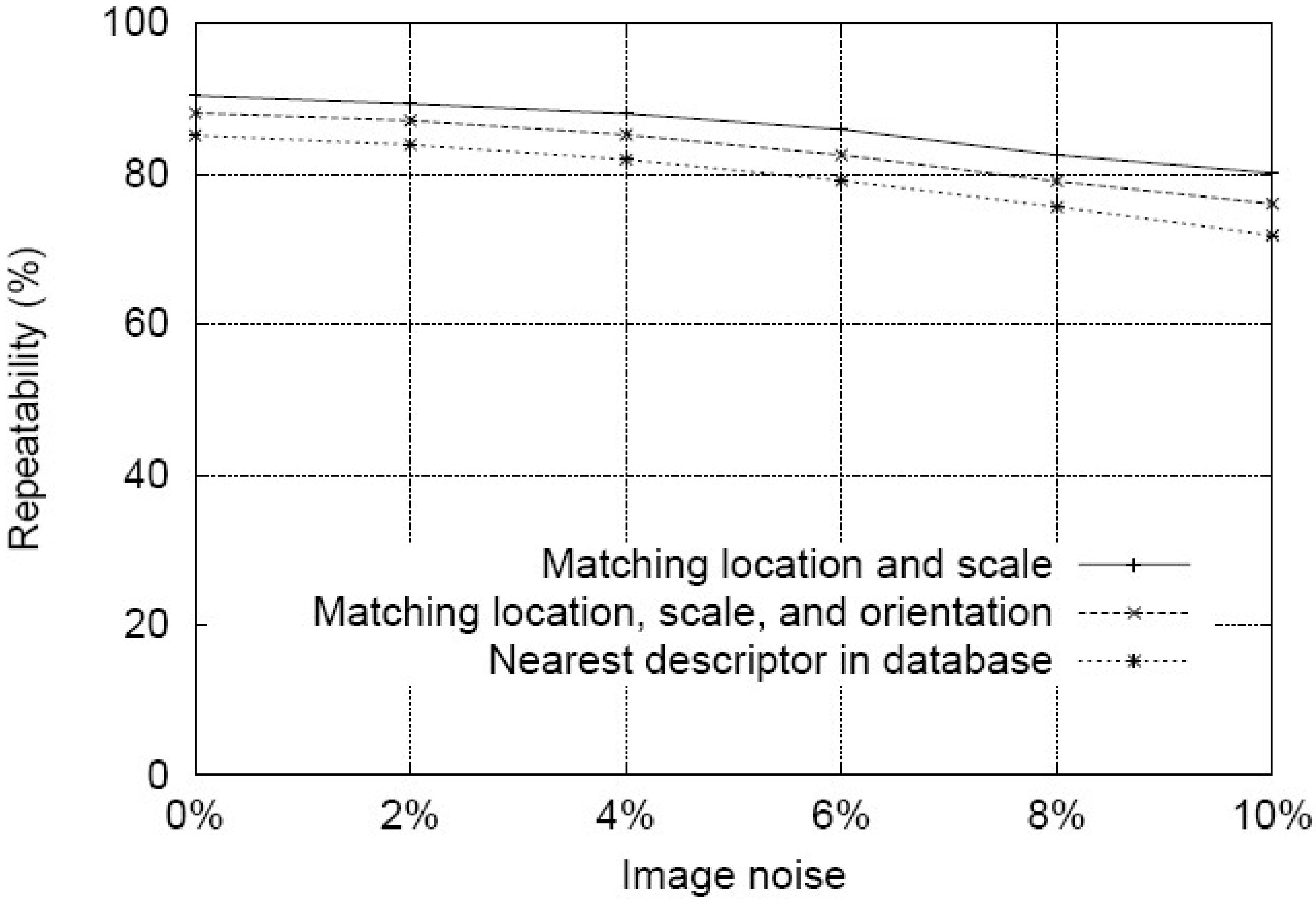
Right features depend on what you want to know

- 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

Local Descriptors

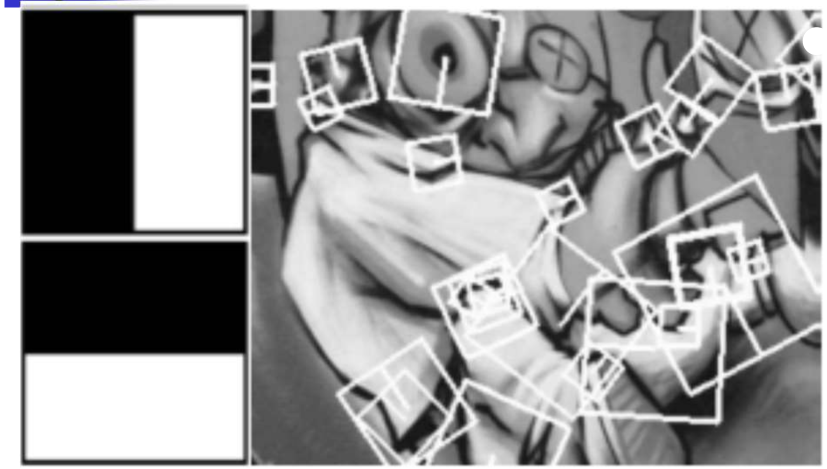
- Most features can be thought of as templates, histograms (counts), or combinations
- The ideal descriptor should be
  - Robust
  - Distinctive
  - Compact
  - Efficient
- Most available descriptors focus on edge/gradient information
  - Capture texture information
  - Color rarely used

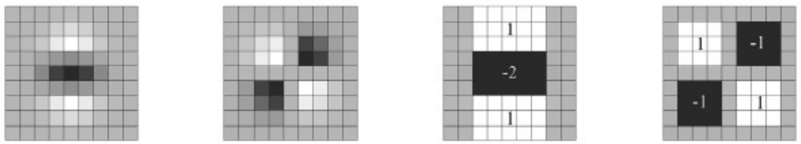
SIFT(Scale Invariant Feature Transform) Repeatability



Lowe IJCV 2004

Local Descriptors: SURF





- **Fast approximation of SIFT idea**
  - **Efficient computation by 2D box filters & integral images**
    $\Rightarrow$ 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]

Choosing a detector

- 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

- Why choose?
  - Get more points with more detectors

- There have been extensive evaluations/comparisons
  - [Mikolajczyk et al., IJCV'05, PAMI'05]
  - All detectors/descriptors shown here work well

Comparison of Keypoint Detectors

Table 7.1 Overview of feature detectors.

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	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			√	√	(√)	(√)	+	+	+	+

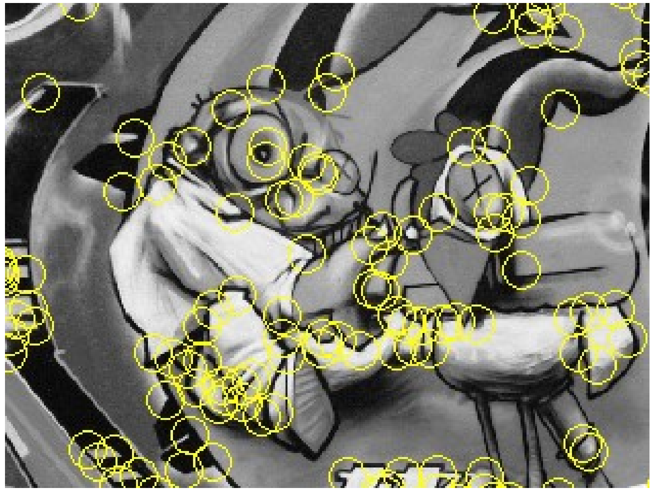
Tuytelaars Mikolajczyk 2008

Choosing a descriptor

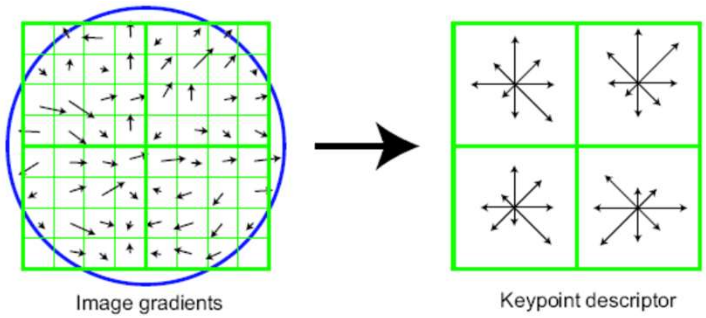
- Again, need not stick to one
- For object instance recognition or stitching, SIFT or variant is a good choice

Things to remember

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



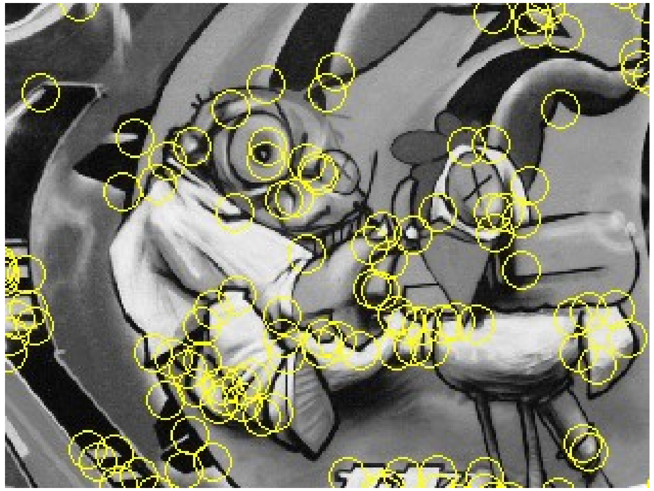
- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT



Feature Matching and Robust Fitting

Review: Interest points

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



Review: Choosing an interest point detector

- What do you want it for?
  - Precise localization in x-y: Harris
  - Good localization in scale: Difference of Gaussian
  - Flexible region shape: MSER

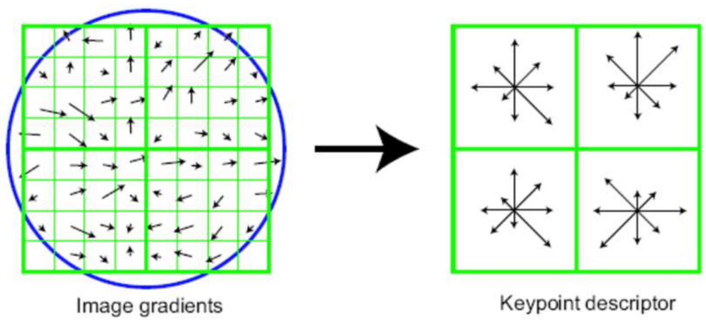
- Best choice often application dependent
  - Harris-/Hessian-Laplace/DoG work well for many natural categories
  - MSER works well for buildings and printed things

- Why choose?
  - Get more points with more detectors

- There have been extensive evaluations/comparisons
  - [Mikolajczyk et al., IJCV'05, PAMI'05]
  - All detectors/descriptors shown here work well

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



Feature Matching

- Szeliski 4.1.3
  - Simple feature-space methods
  - Evaluation methods
  - Acceleration methods
  - Geometric verification (Chapter 6)

Feature Matching

- Simple criteria: One feature matches to another if those features are nearest neighbors and their distance is below some threshold.
- Problems:
  - Threshold is difficult to set
  - Non-distinctive features could have lots of close matches, only one of which is correct

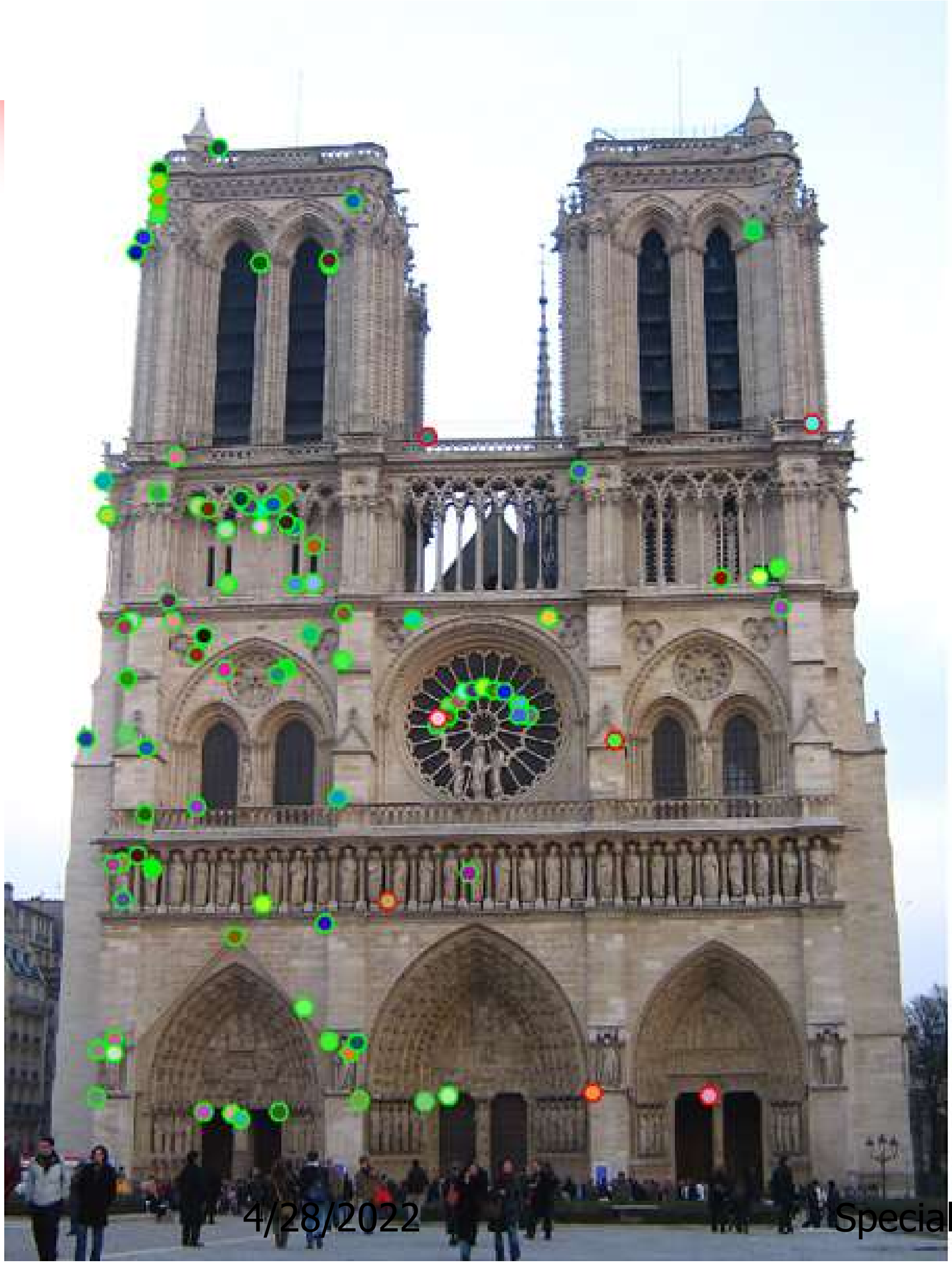
Comparison of Keypoint Detectors

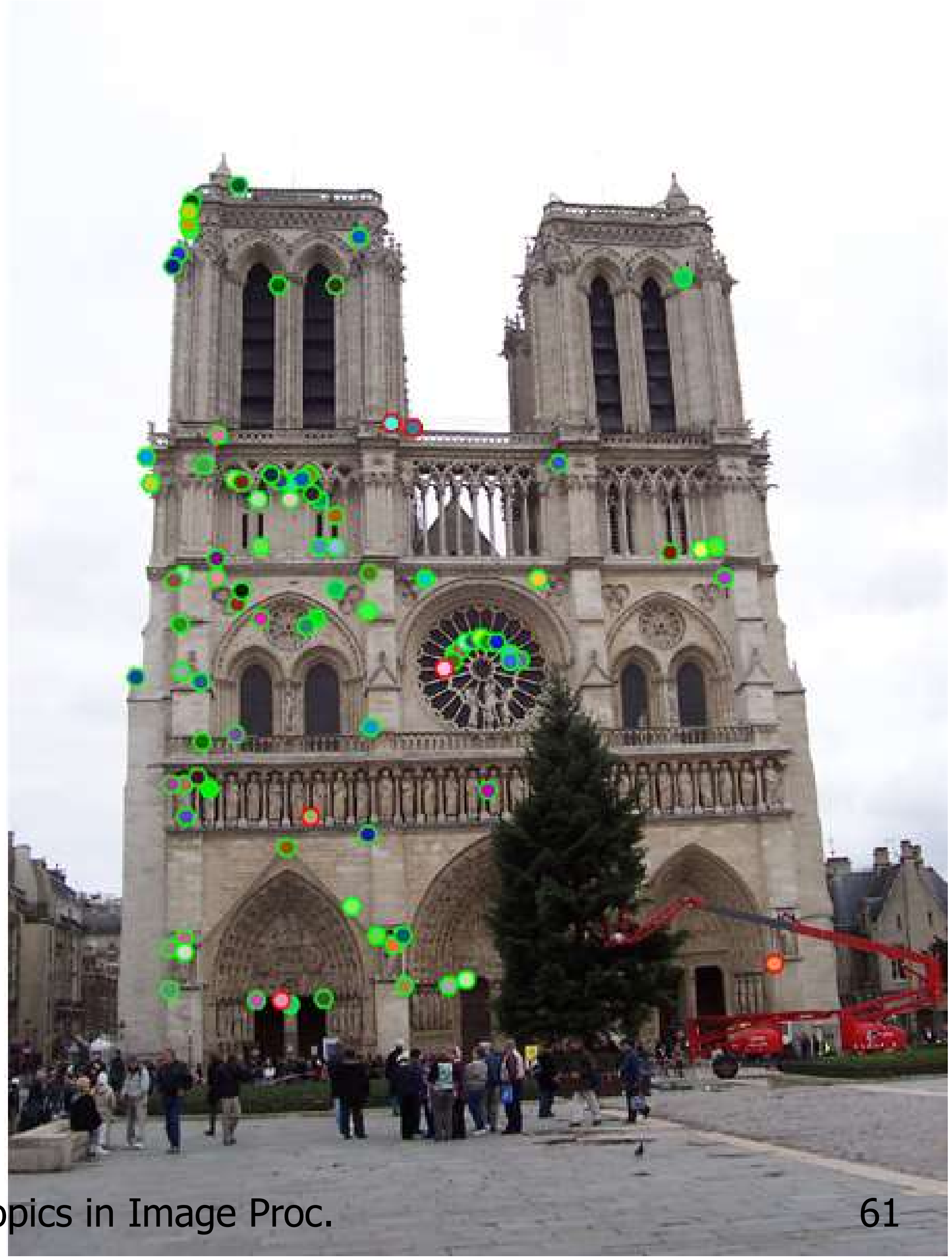
Table 7.1 Overview of feature detectors.

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	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

How do we decide which features match?





Fitting and Alignment

- Fitting: find the parameters of a model that best fit the data
- Alignment: find the parameters of the transformation that best align matched points

Fitting and Alignment

- Design challenges
  - Design a suitable **goodness of fit** measure
    - Similarity should reflect application goals
    - Encode robustness to outliers and noise
  - Design an **optimization** method
    - Avoid local optima
    - Find best parameters quickly

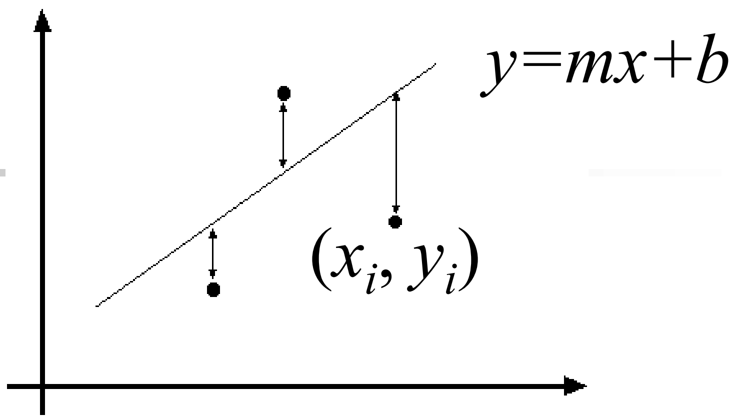
Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Iterative closest point (ICP)

- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

Least squares line fitting

$$E=\sum_{i=1}^{n}(y_i-mx_i-b)^2$$



- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m\,x_i + b$
- Find (m, b) to minimize

$$E=\sum_{i=1}^{n}\left(\begin{bmatrix}x_i & 1\end{bmatrix}\begin{bmatrix}m\\ b\end{bmatrix}-y_i\right)^2=\left\|\begin{bmatrix}x_1 & 1\\ \vdots & \vdots\\ x_n & 1\end{bmatrix}\begin{bmatrix}m\\ b\end{bmatrix}-\begin{bmatrix}y_1\\ \vdots\\ y_n\end{bmatrix}\right\|^2=\|\mathbf{Ap}-\mathbf{y}\|^2$$

$$=\mathbf{y}^T\mathbf{y}-2(\mathbf{Ap})^T\mathbf{y}+(\mathbf{Ap})^T(\mathbf{Ap})$$

$$\frac{dE}{dp}=2\mathbf{A}^T\mathbf{Ap}-2\mathbf{A}^T\mathbf{y}=0$$

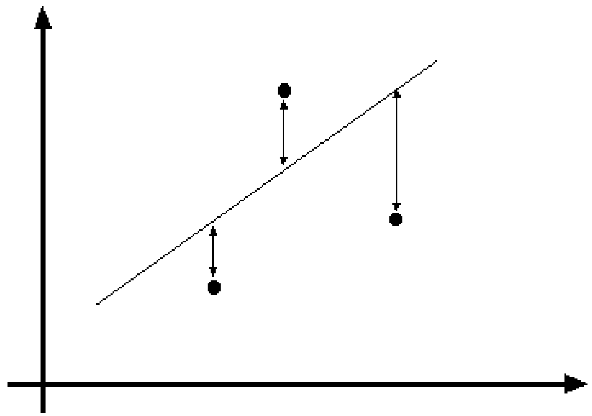
Matlab: `p = A \ y;`

$$\mathbf{A}^T\mathbf{Ap}=\mathbf{A}^T\mathbf{y}\Rightarrow\mathbf{p}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{y}$$

Modified from S. Lazebnik

Problem with "vertical" least squares

- Not rotation-invariant
- Fails completely for vertical lines



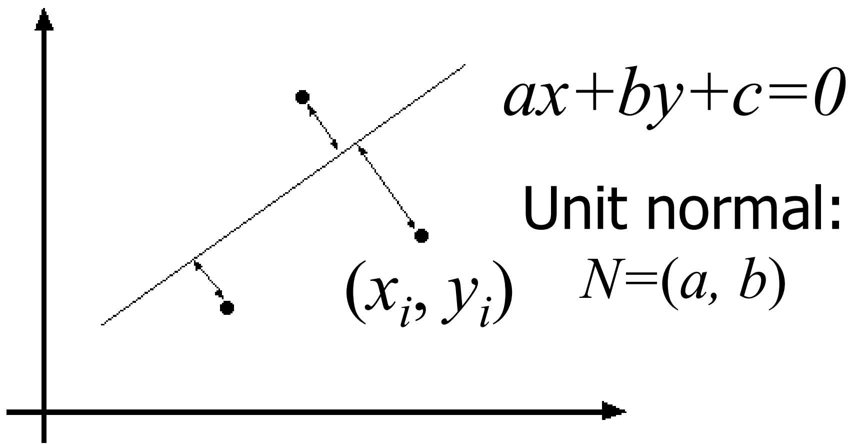
Total least squares

If $(a^2+b^2=1)$ then

Distance between point (x_i, y_i) is

$$|ax_i + by_i + c|$$

proof:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html



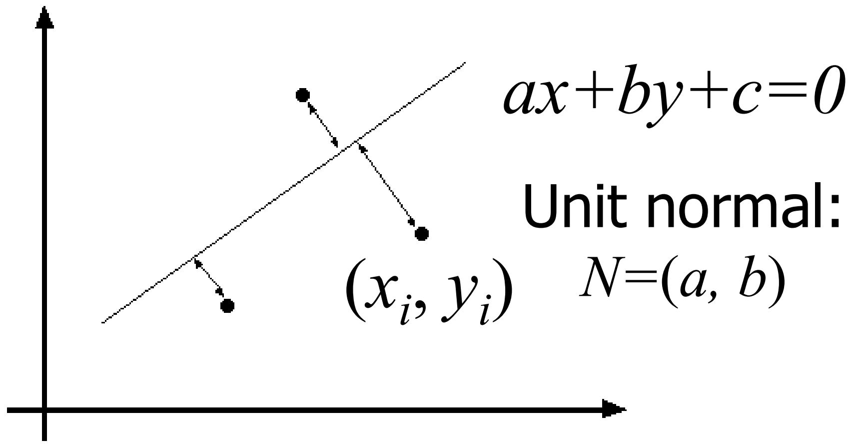
Total least squares

If ($a^2+b^2=1$) then

Distance between point (x_i, y_i) is

$$|ax_i + by_i + c|$$

Find (a, b, c) to minimize the sum of squared perpendicular distances

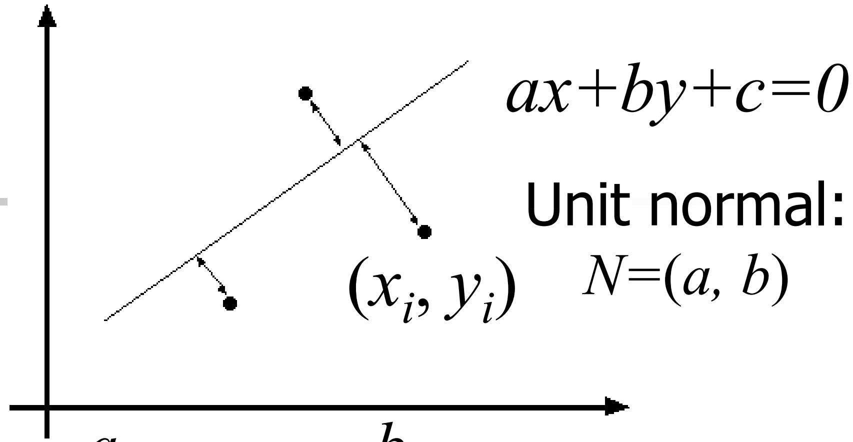


$$\boxed{E = \sum_{i=1}^{n} (ax_i + by_i + c)^2}$$

Total least squares

Find (a, b, c) to minimize the sum of squared perpendicular distances

$$\boxed{E = \sum_{i=1}^{n} (a x_i + b y_i + c)^2}$$



$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} 2(a x_i + b y_i + c) = 0 \qquad c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}$$

$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}$$

$$\text{minimize}\, \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p} \quad \text{s.t. } \mathbf{p}^T \mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize} \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}}{\mathbf{p}^T \mathbf{p}}$$

Solution is eigenvector corresponding to smallest eigenvalue of $A^T A$

See details on Raleigh Quotient: http://en.wikipedia.org/wiki/Rayleigh_quotient

Two Common Optimization Problems

Problem statement | **Solution**

$$\text{minimize } \|\mathbf{Ax}-\mathbf{b}\|^2$$

$$\text{least squares solution to } \mathbf{Ax}=\mathbf{b}$$

$$\mathbf{x}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

$$\mathbf{x}=\mathbf{A}\setminus\mathbf{b}$$ (matlab)

Problem statement | **Solution**

$$\text{minimize } \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} \quad \text{s.t. } \mathbf{x}^T\mathbf{x}=1$$

$$\text{minimize} \frac{\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}}{\mathbf{x}^T\mathbf{x}}$$

$$\text{non - trivial lsq solution to } \mathbf{Ax}=0$$

$$[\mathbf{v},\lambda]=\text{eig}(\mathbf{A}^T\mathbf{A})$$

$$\lambda_1<\lambda_{2..n}:\mathbf{x}=\mathbf{v}_1$$

Least squares (global) optimization

Good

- Clearly specified objective
- Optimization is easy

Bad

- May not be what you want to optimize
- Sensitive to outliers
  - Bad matches, extra points
- Doesn't allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.