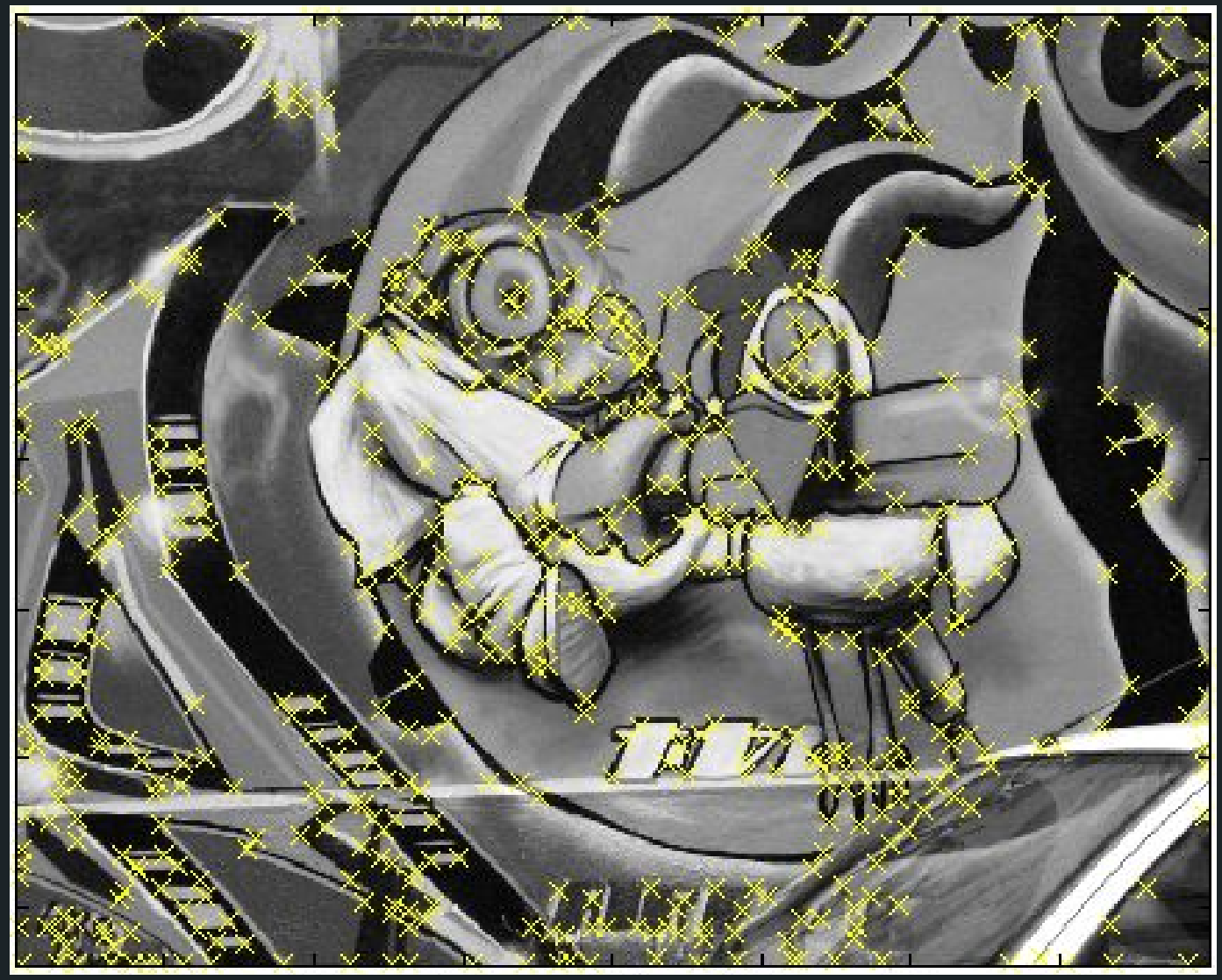
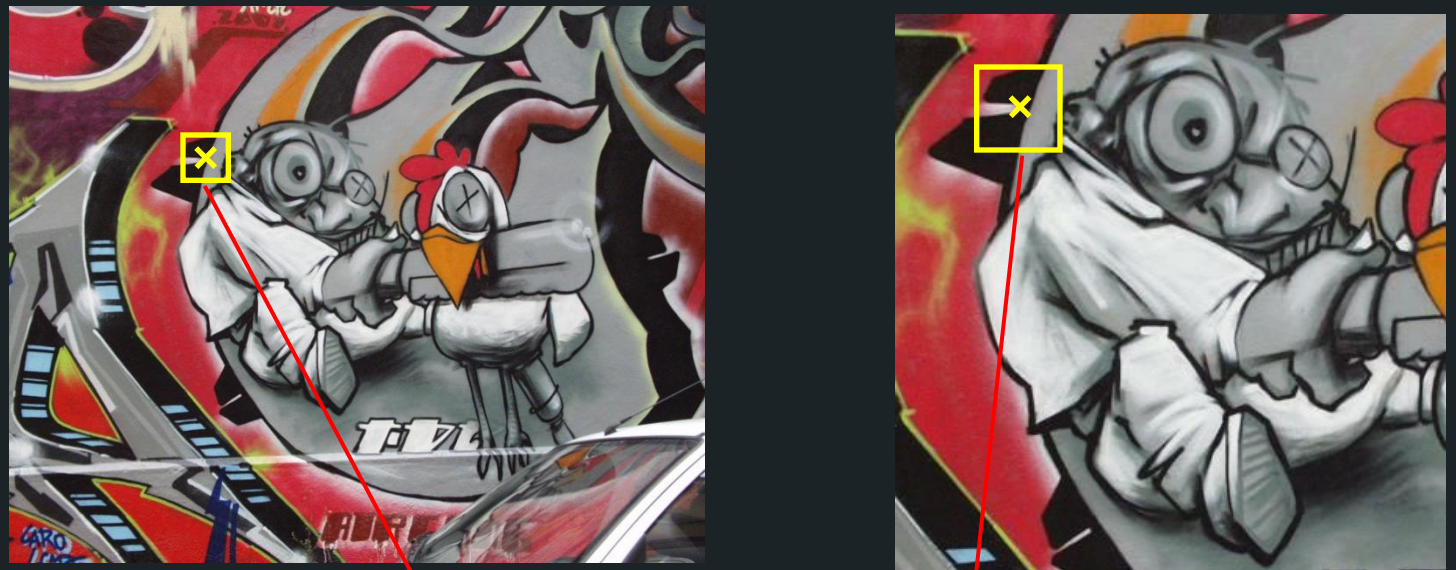
LOCAL IMAGE FEATURES

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



Automatic Scale Selection

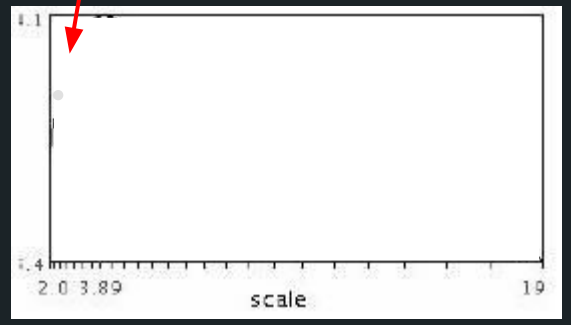


$$f(I_{i_1\ldots i_m}(x,\sigma)) \;=\; f(I_{i_1\ldots i_m}(x',\sigma'))$$

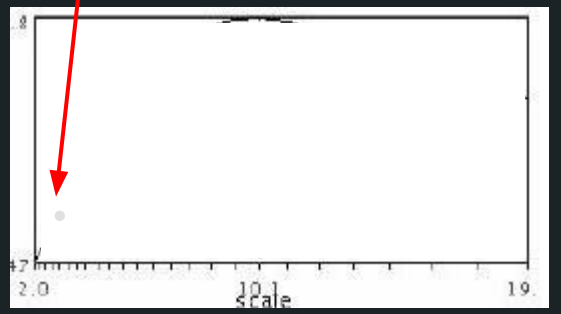
How to find corresponding patch sizes?

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



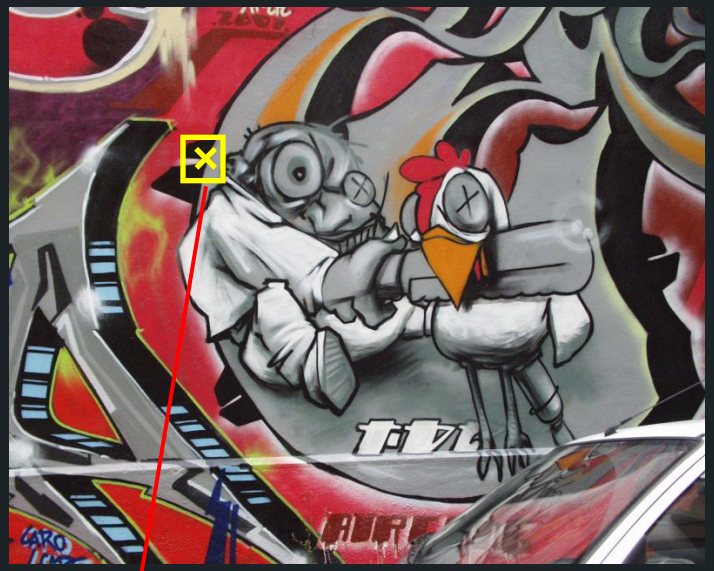
$f(I_{i_1} \ldots_{i_m}(x,\sigma))$

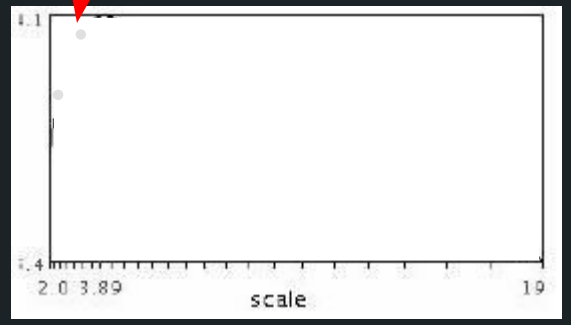


$f(I_{i_1} \ldots_{i_m}(x',\sigma))$

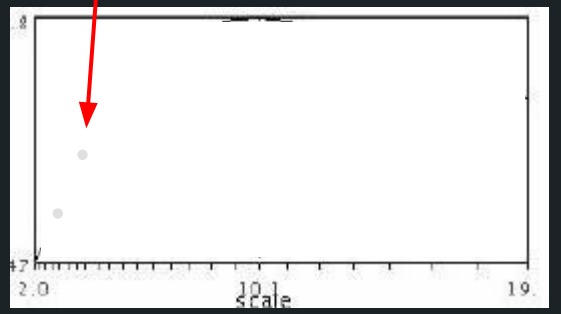
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





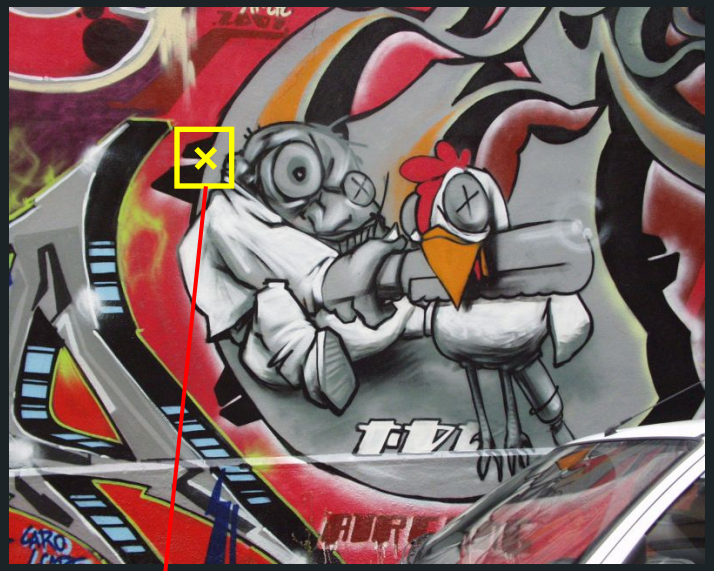
$f(I_{i_1 \square\, i_m}(x,\sigma))$

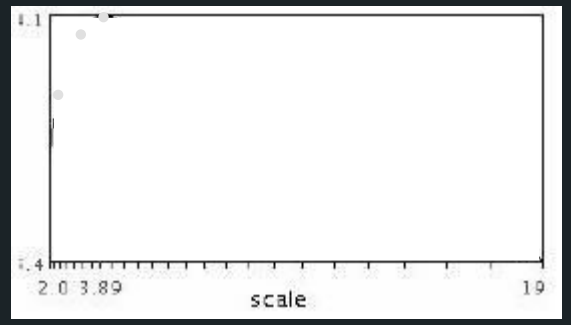


$f(I_{i_1 \square\, i_m}(x',\sigma))$

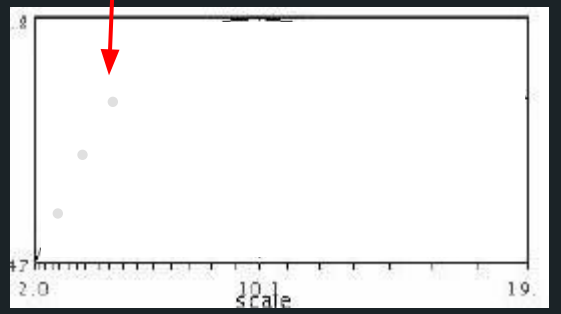
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





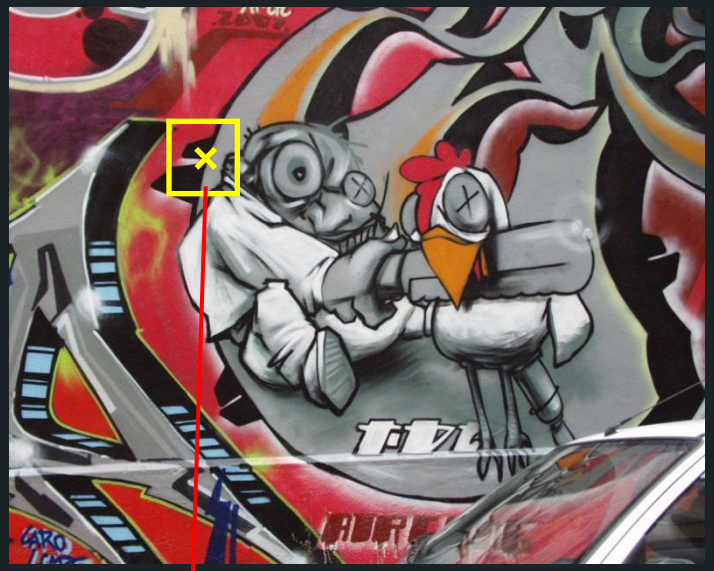
$f(I_{i_1} \ldots_{i_m}(x,\sigma))$

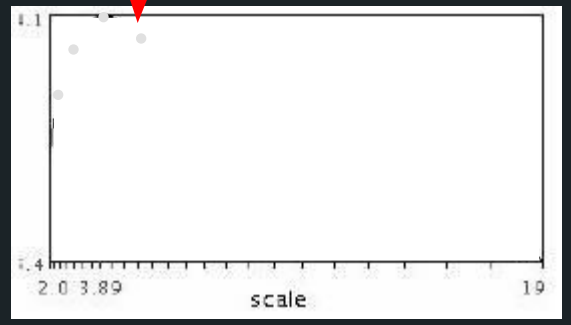


$f(I_{i_1} \ldots_{i_m}(x',\sigma))$

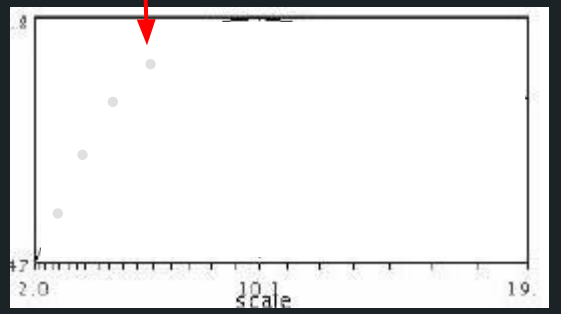
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





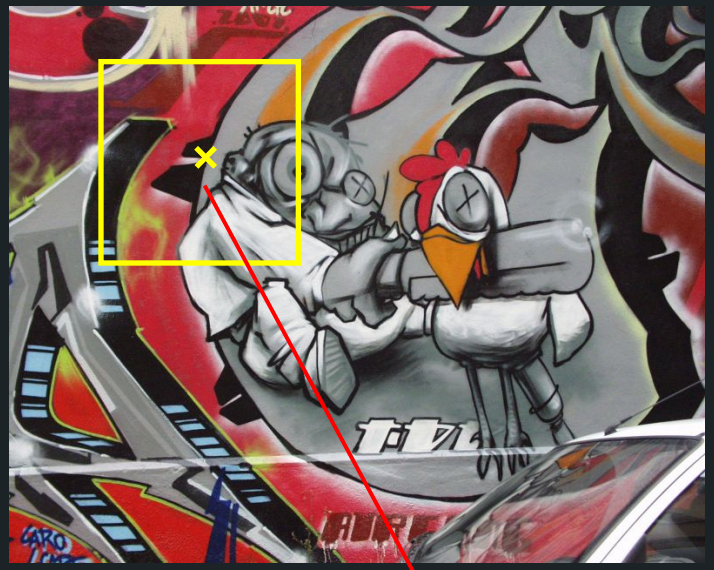
$f(I_{i_1 \ldots i_m}(x,\sigma))$

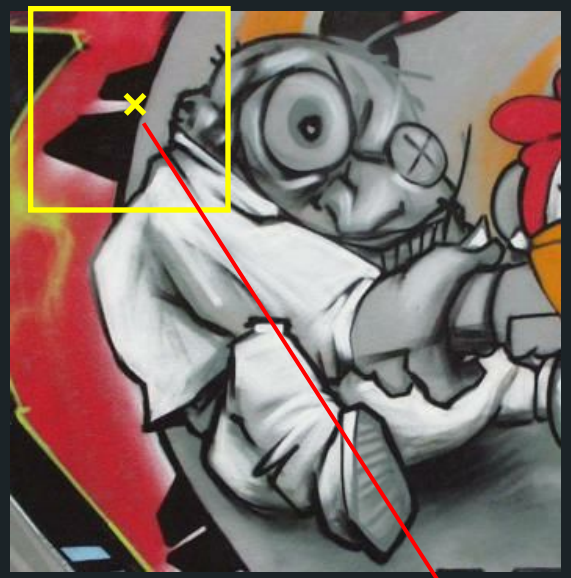


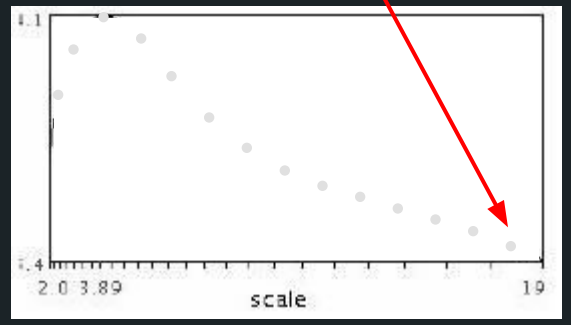
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

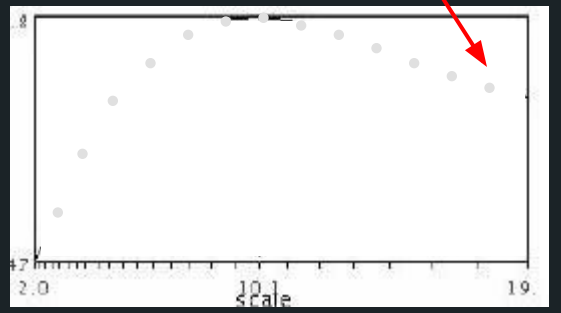
- Function responses for increasing scale (scale signature)







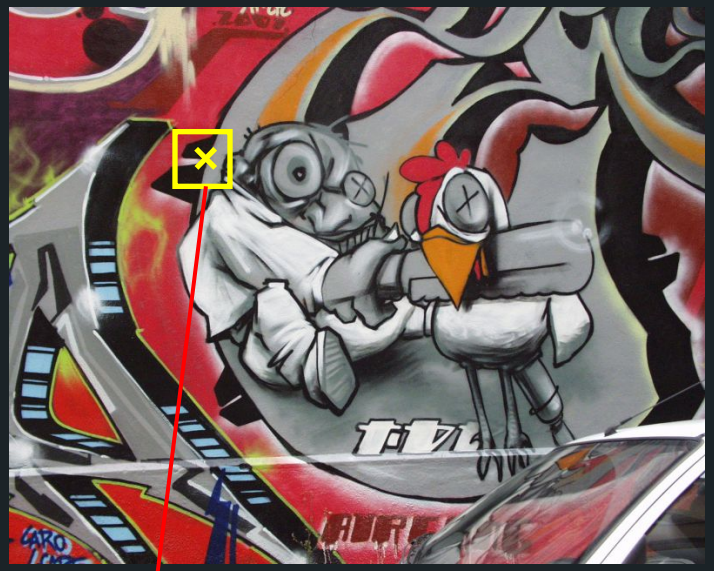
$f(I_{i_1} \square_{i_m}(x,\sigma))$

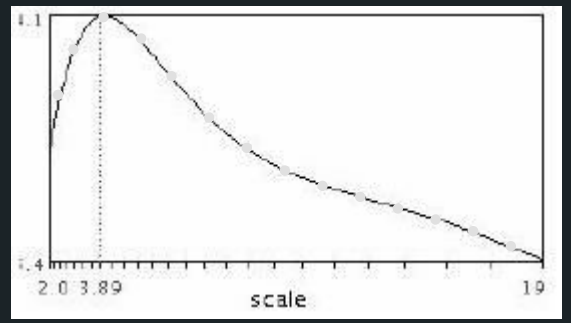


$f(I_{i_1} \square_{i_m}(x',\sigma))$

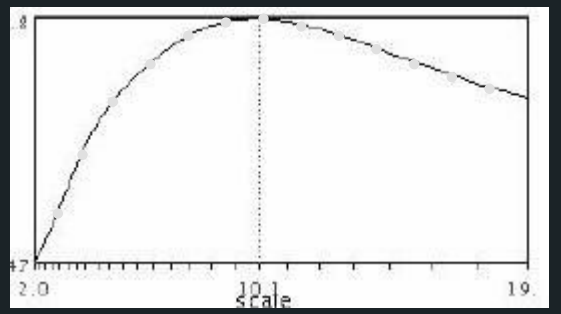
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





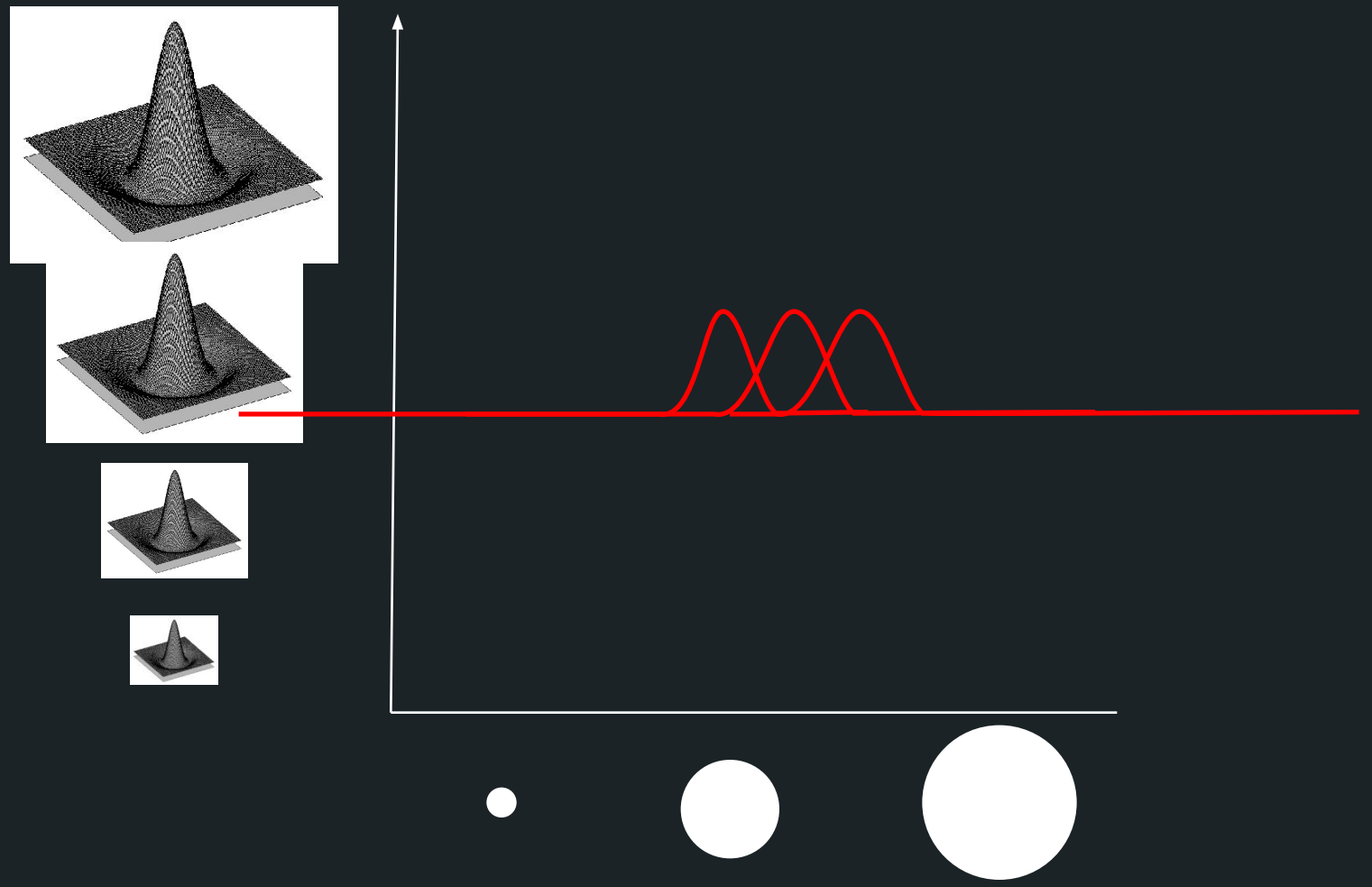
$f(I_{i_1 \square\, i_m}(x,\sigma))$



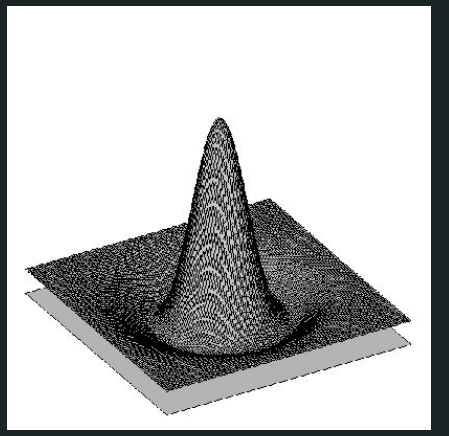
$f(I_{i_1 \square\, i_m}(x',\sigma'))$

What Is A Useful Signature Function?

- Difference-of-Gaussian = “blob” detector



Difference-of-Gaussian (DoG)



K. Grauman, B. Leibe

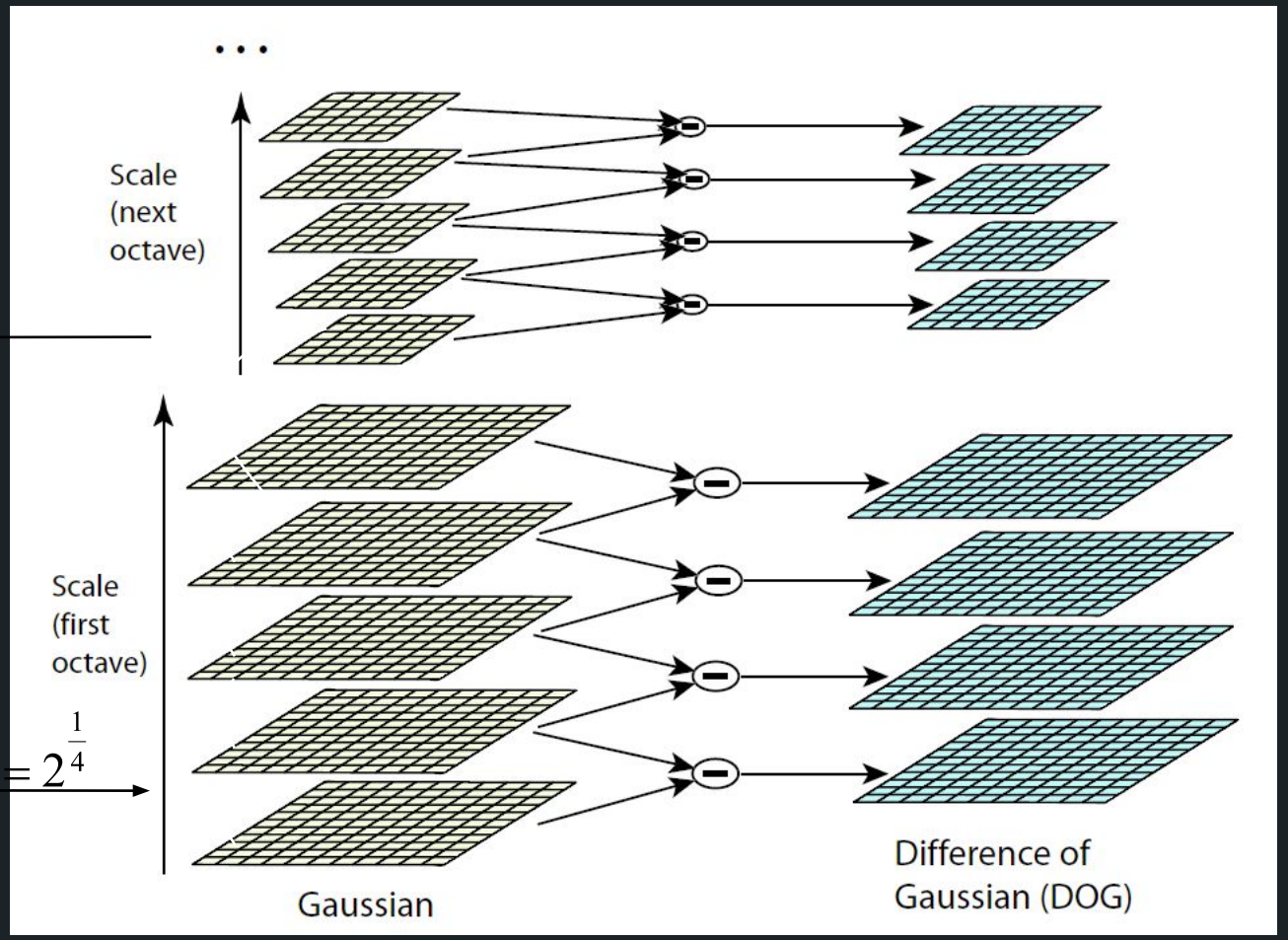
DoG – Efficient Computation

- Computation in Gaussian scale pyramid

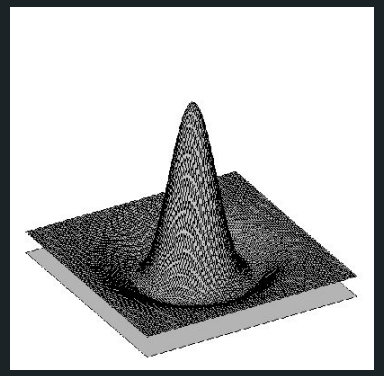
Sampling with step $\sigma^4 = 2$

Original image

$\sigma = 2^{\frac{1}{4}}$



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



$$L_{xx}(\sigma) + L_{yy}(\sigma)$$

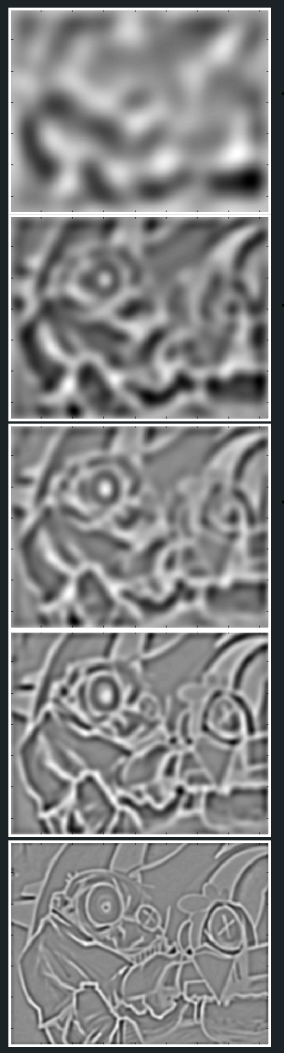
σ^5

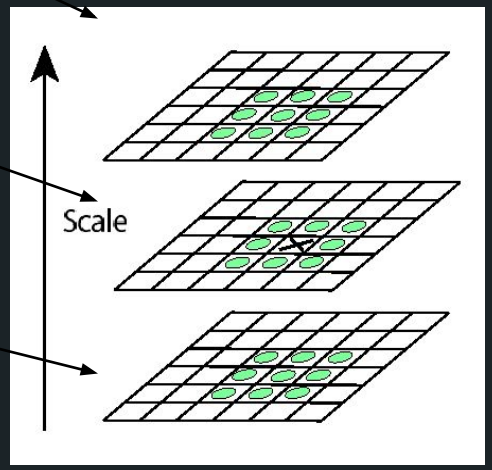
σ^4

σ^3

σ^2

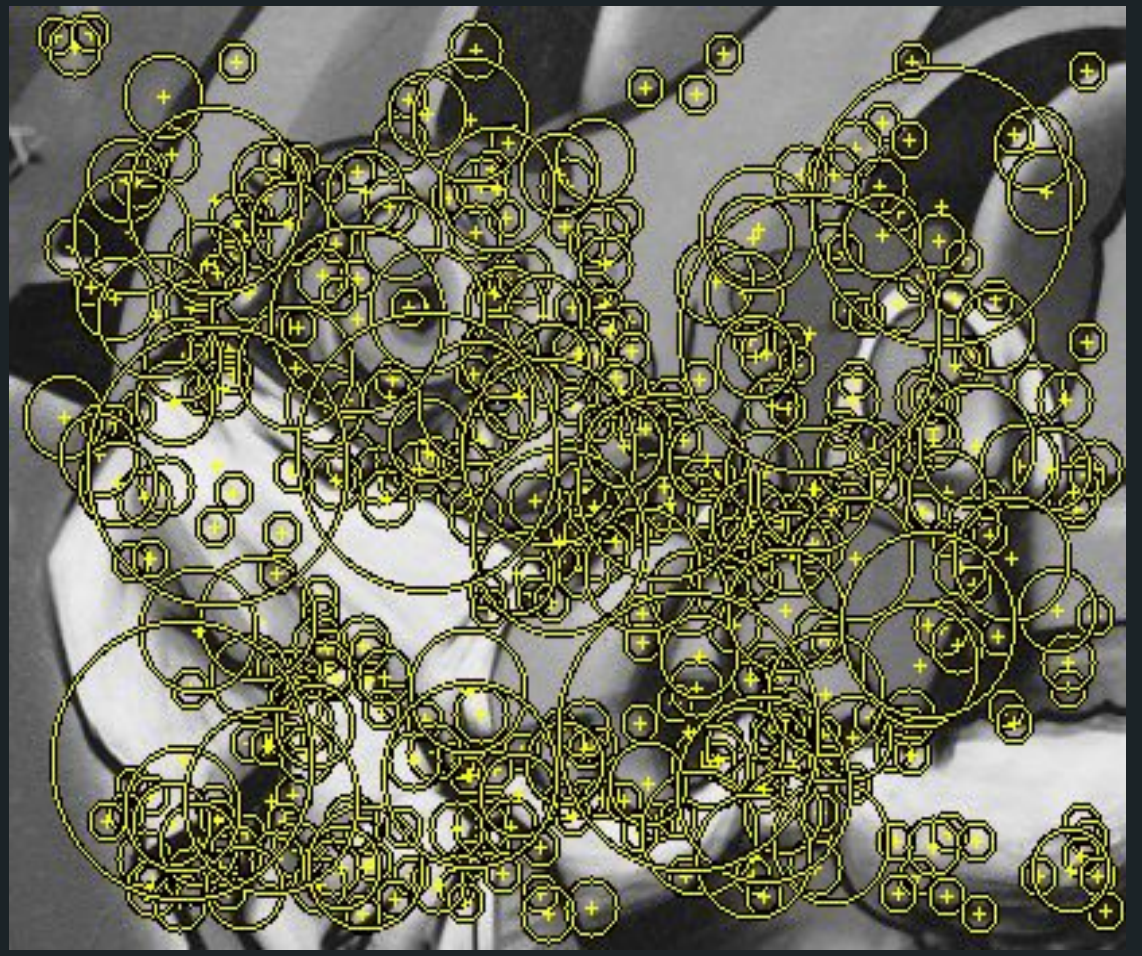
σ





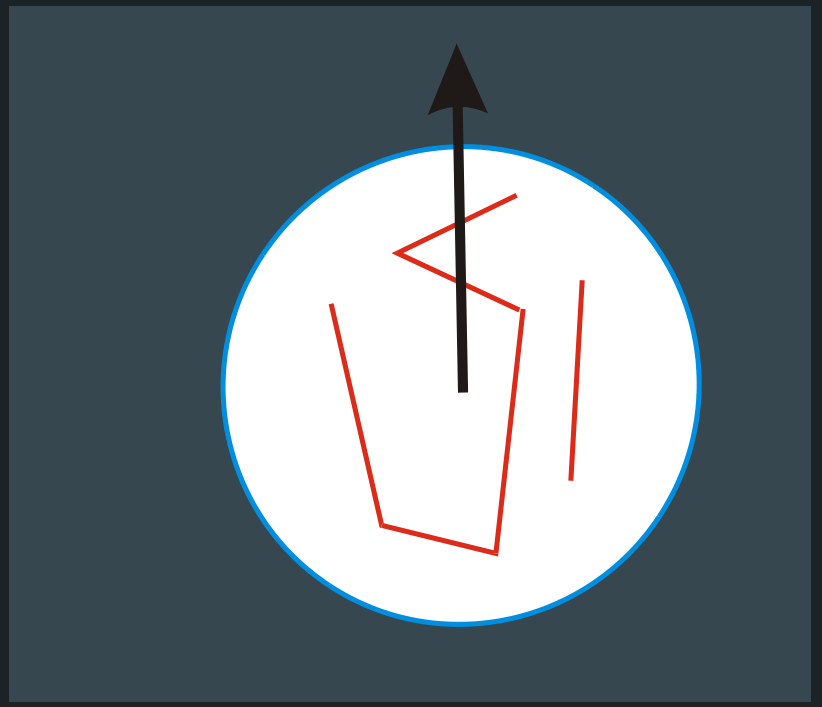
⇒ **List of**
(x, y, s)

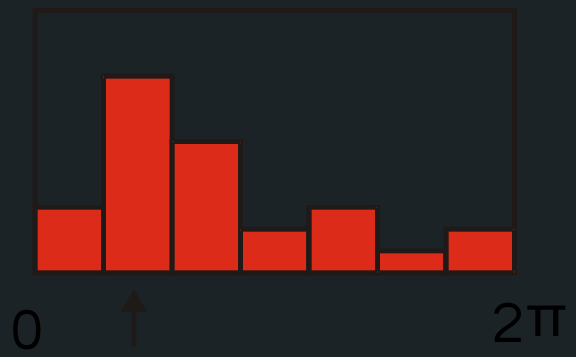
Results: Difference-of-Gaussian



Orientation Normalization

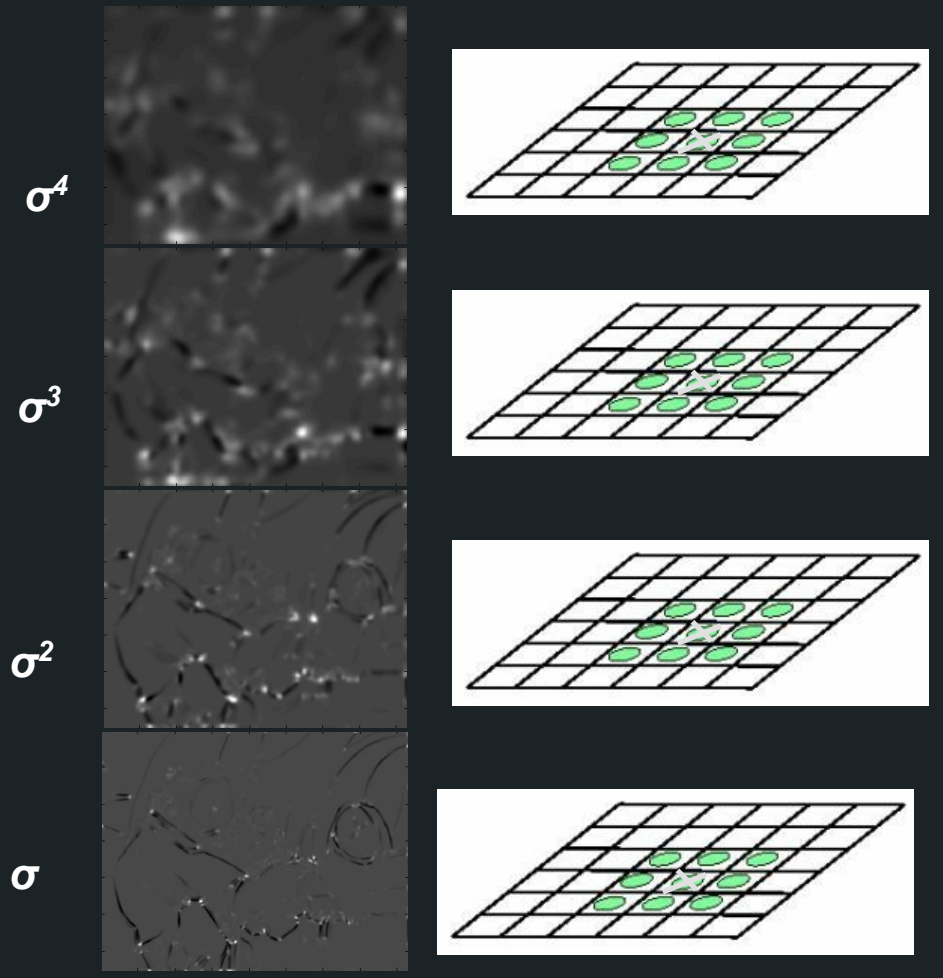
- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation





Harris-Laplace

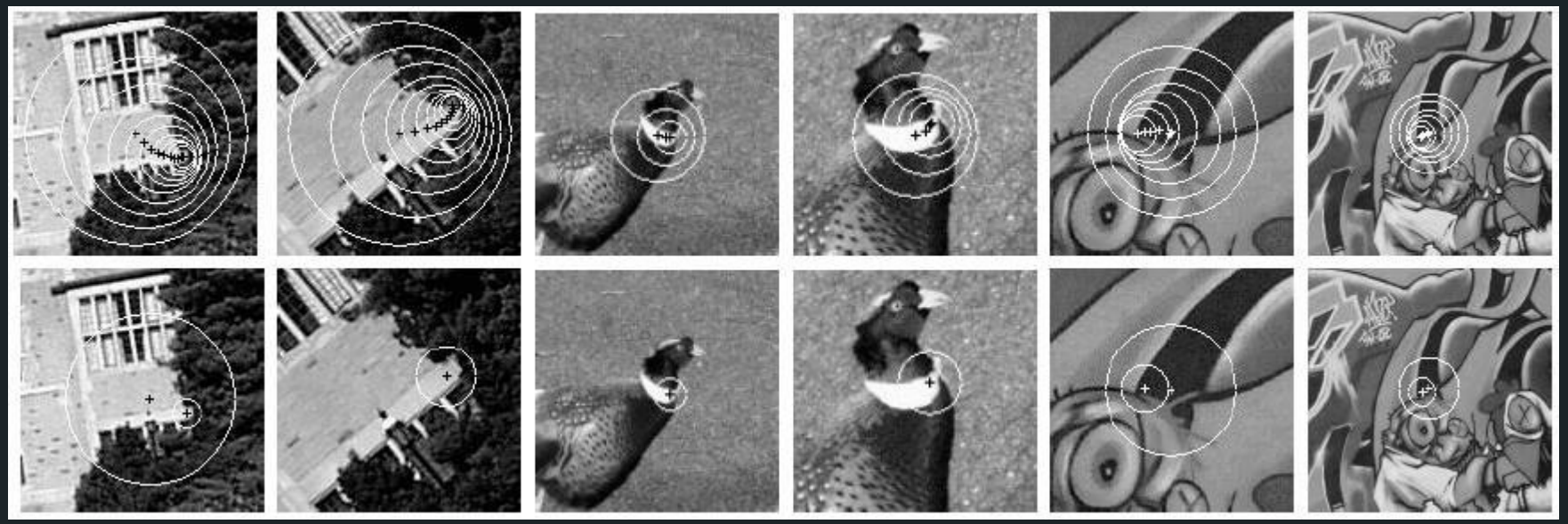
1. Initialization: Multiscale Harris corner detection



Harris-Laplace

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

Harris points



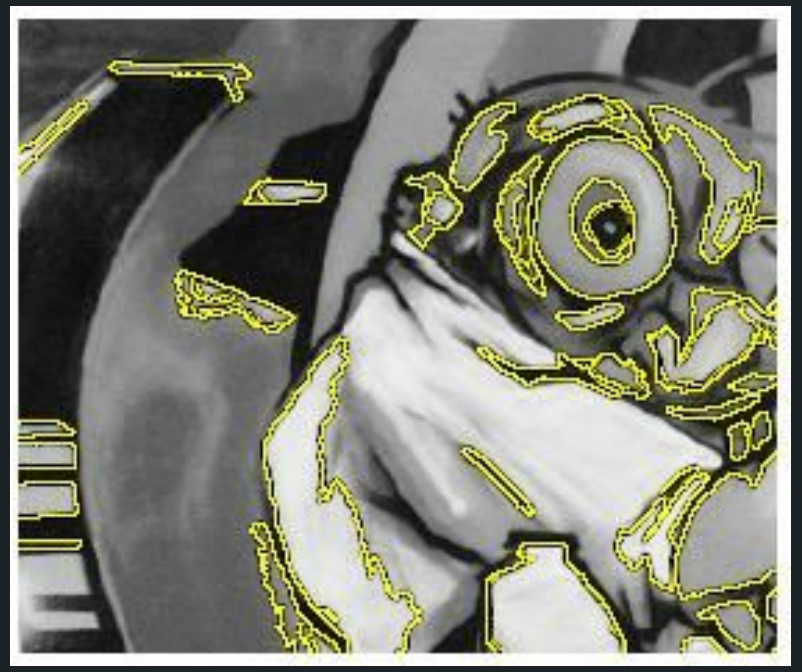
Harris-Laplace points

K. Grauman, B. Leibe

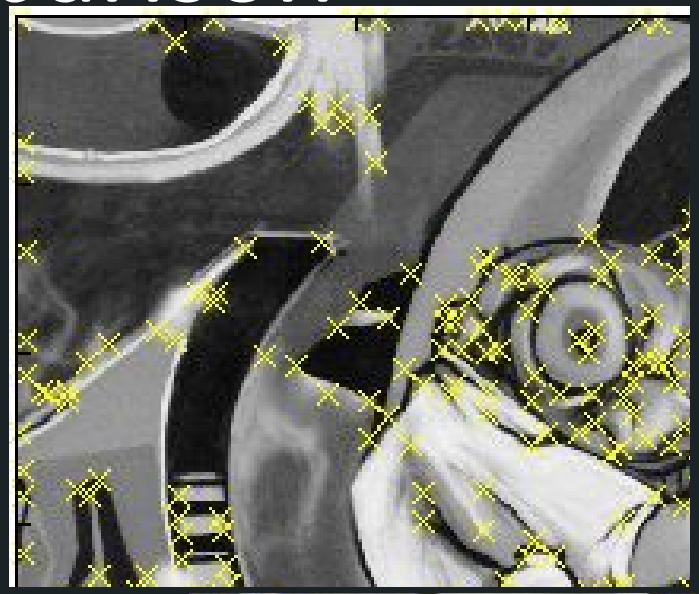
Maximally Stable Extremal Regions

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

Example Results: MSER



Comparison



Harris

LoG

Hessian

MSER

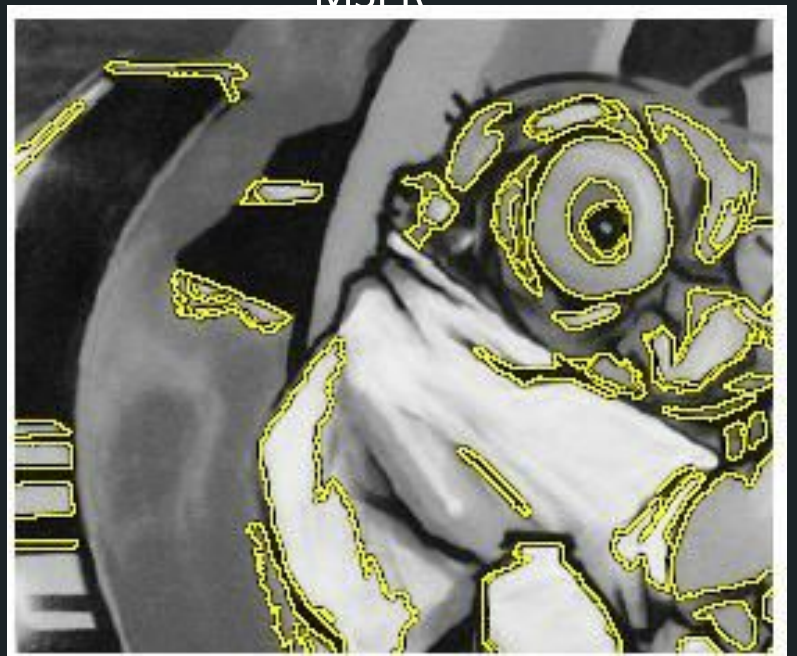
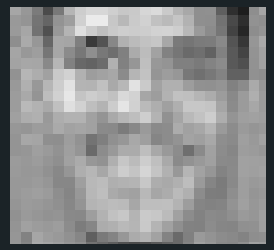


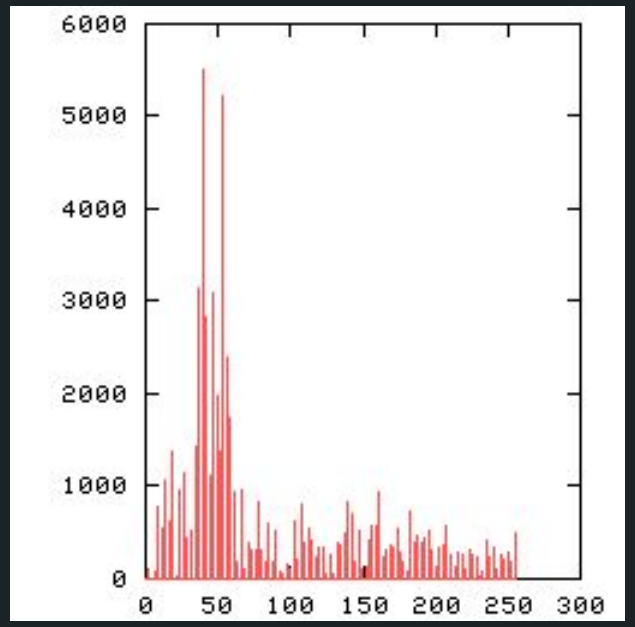
Image representations

- Templates
  - Intensity, gradients, etc.



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

Image Representations: Histograms



Global histogram

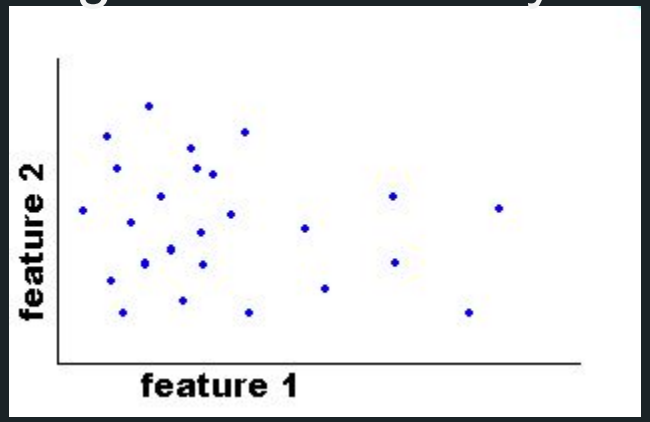
- Represent distribution of features
  - Color, texture, depth, …

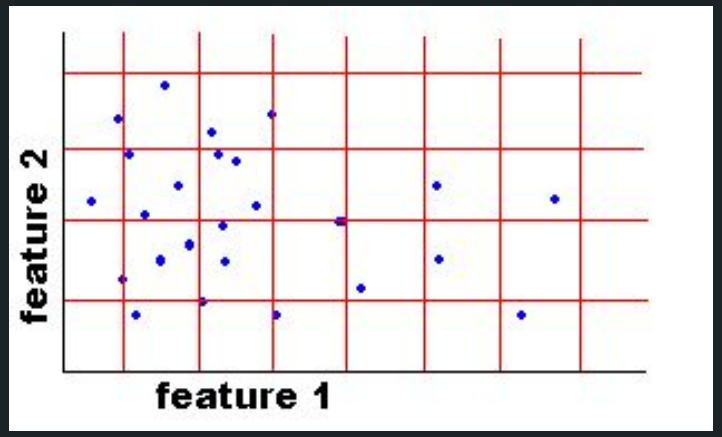
Space Shuttle Cargo Bay

Images from Dave Kauchak

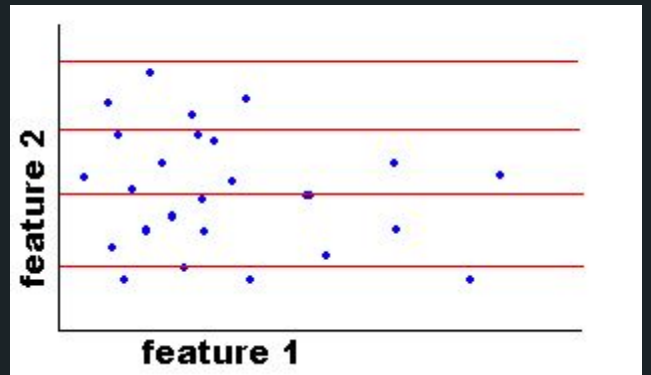
Image Representations: Histograms

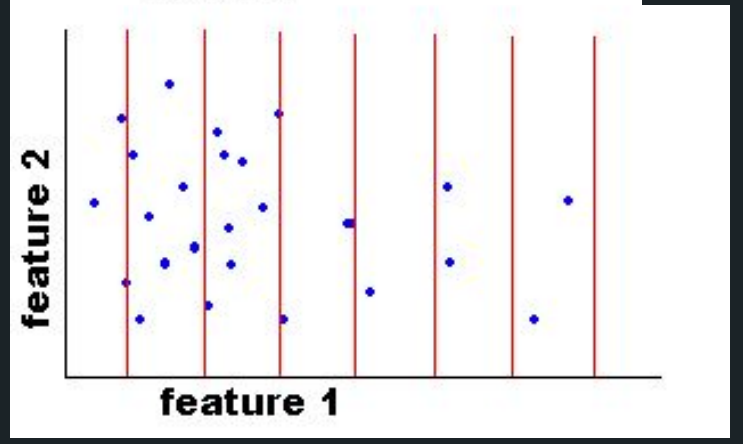
Histogram: Probability or count of data in each bin





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





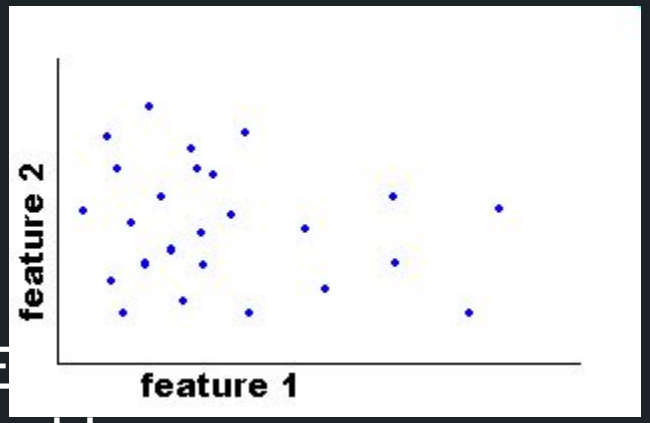
Marginal histogram

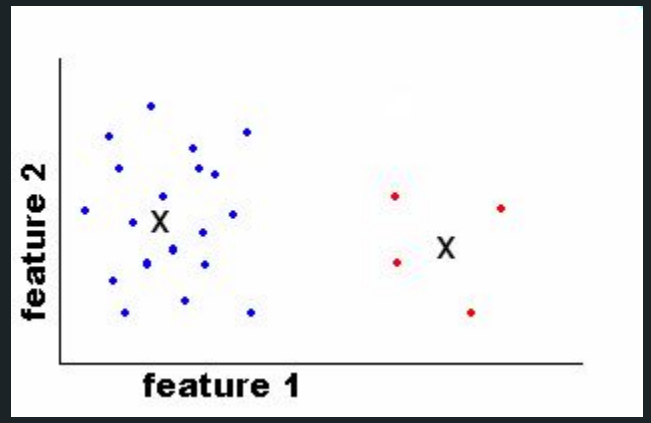
- Requires independent features
- More data/bin than joint histogram

Images from Dave Kauchak

Image Representations: Histograms

Clustering





EASE
Assembly

Use the same cluster centers for all images

Space Shuttle
Cargo Bay

Images from Dave Kauchak

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

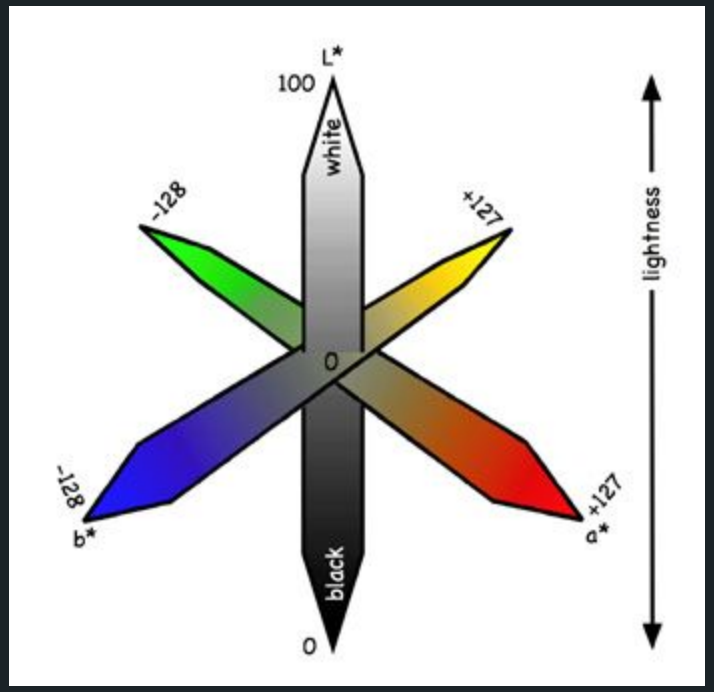
Few Bins
Need less data
Coarser representation

Many Bins
Need more data
Finer representation

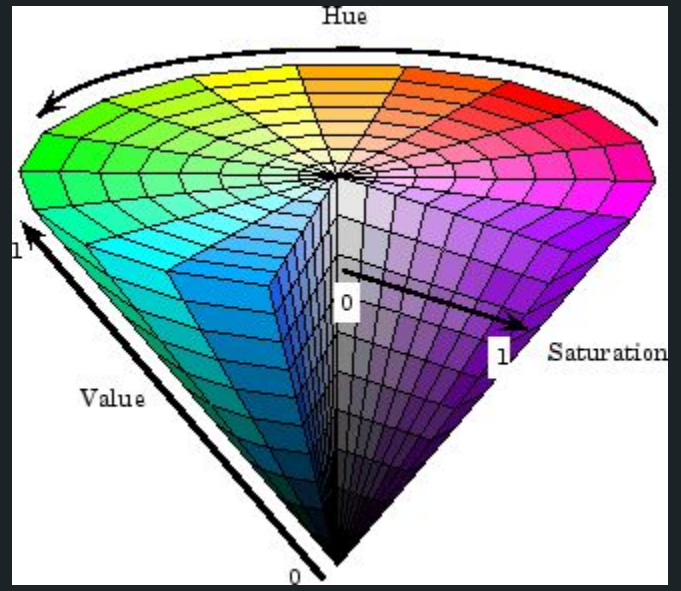
- Matching
  - Histogram intersection or Euclidean may be faster
  - Chi-squared often works better
  - Earth mover's distance is good for when nearby bins represent similar values

What kind of things do we compute histograms of?

- Color



L*a*b* color space

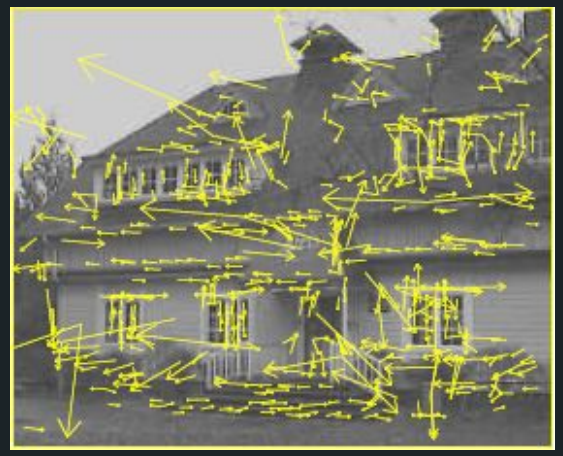


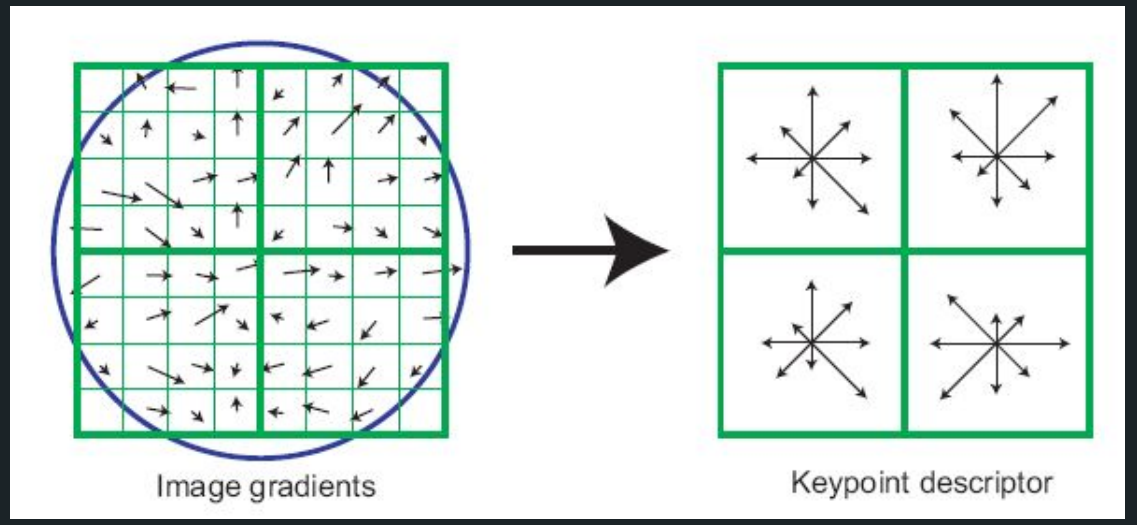
HSV color space

- Texture (filter banks or HOG over regions)

What kind of things do we compute histograms of?

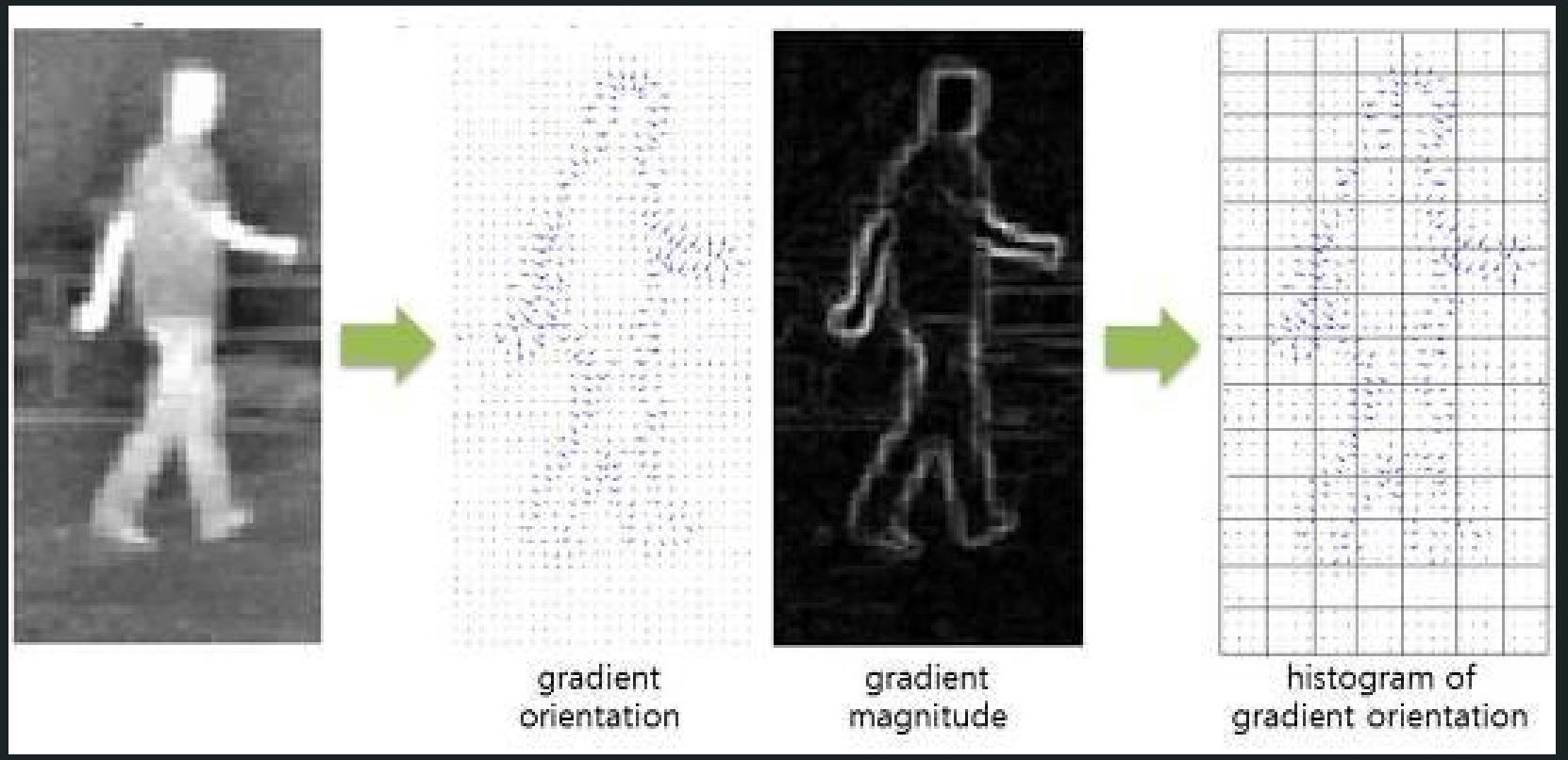
- Histograms of oriented gradients





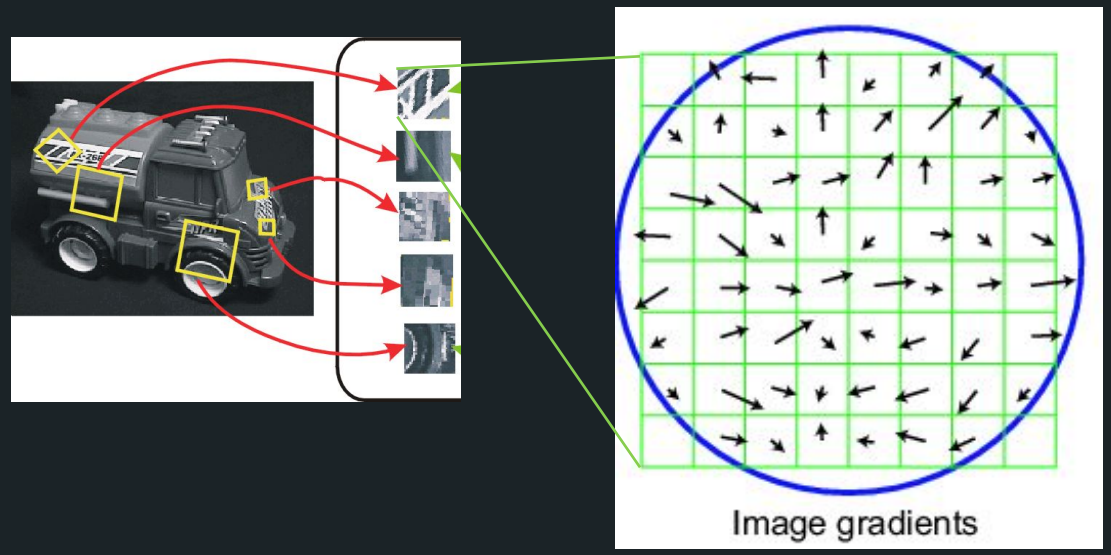
HoG(Histogram of Oriented Gradients)

Uses gradients(indirectly the edges and so the shape of objects)



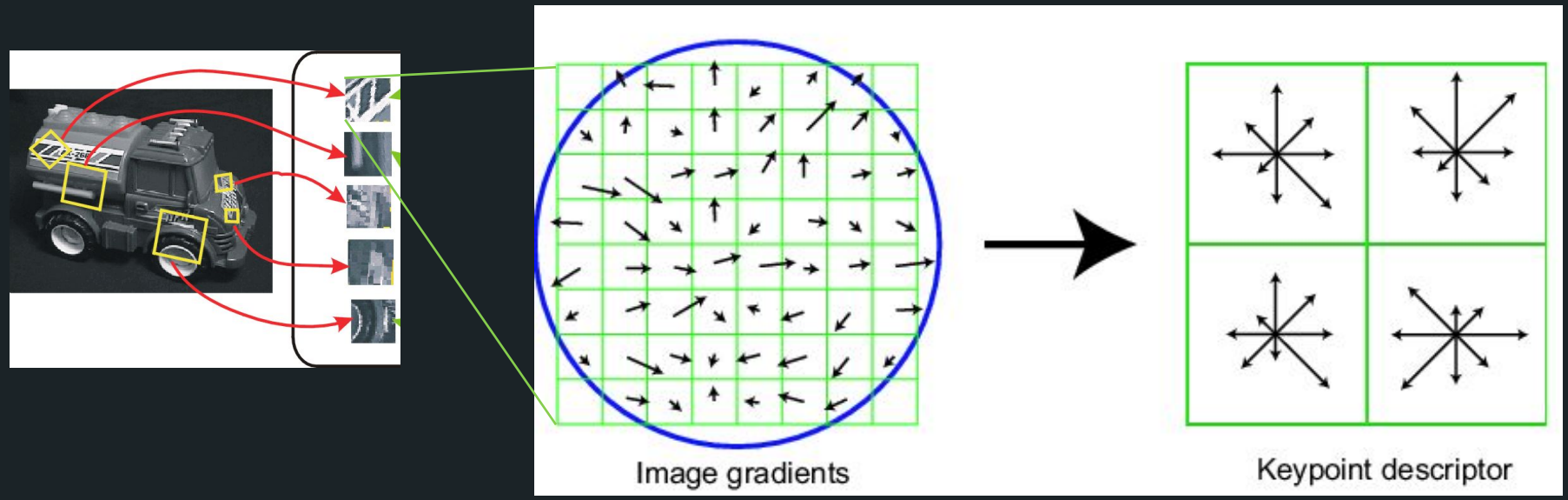
SIFT vector formation

- 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)



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.



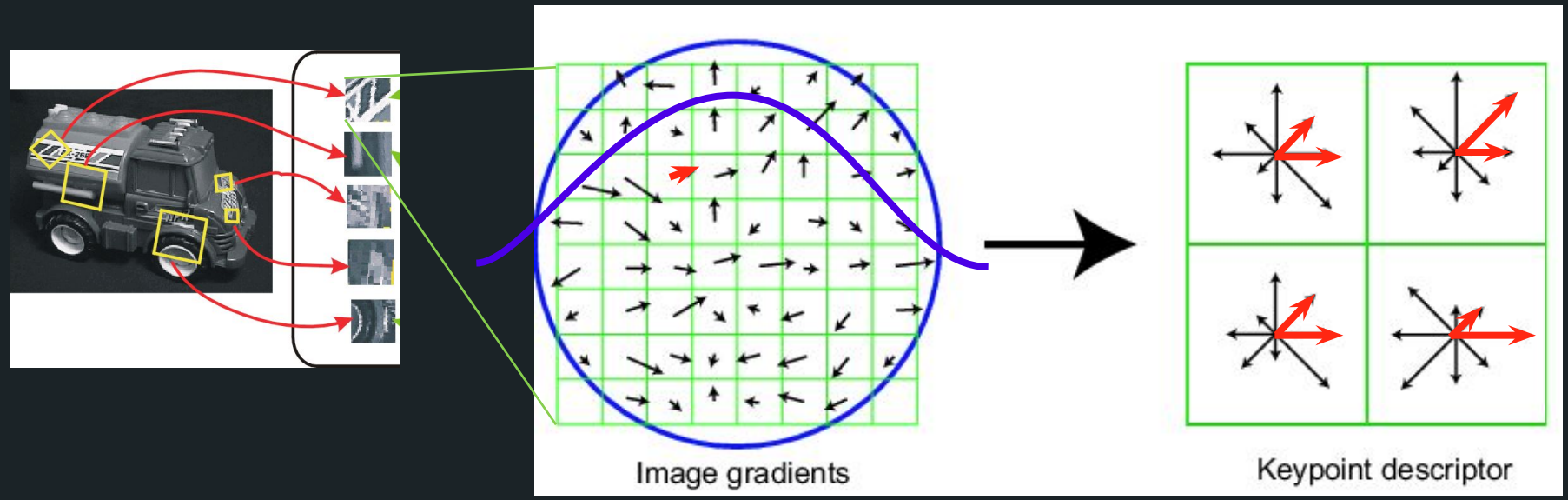
showing only 2x2 here but is 4x4

SIFT (Scale-invariant feature transform)

Invariant to image translation , scaling and rotation

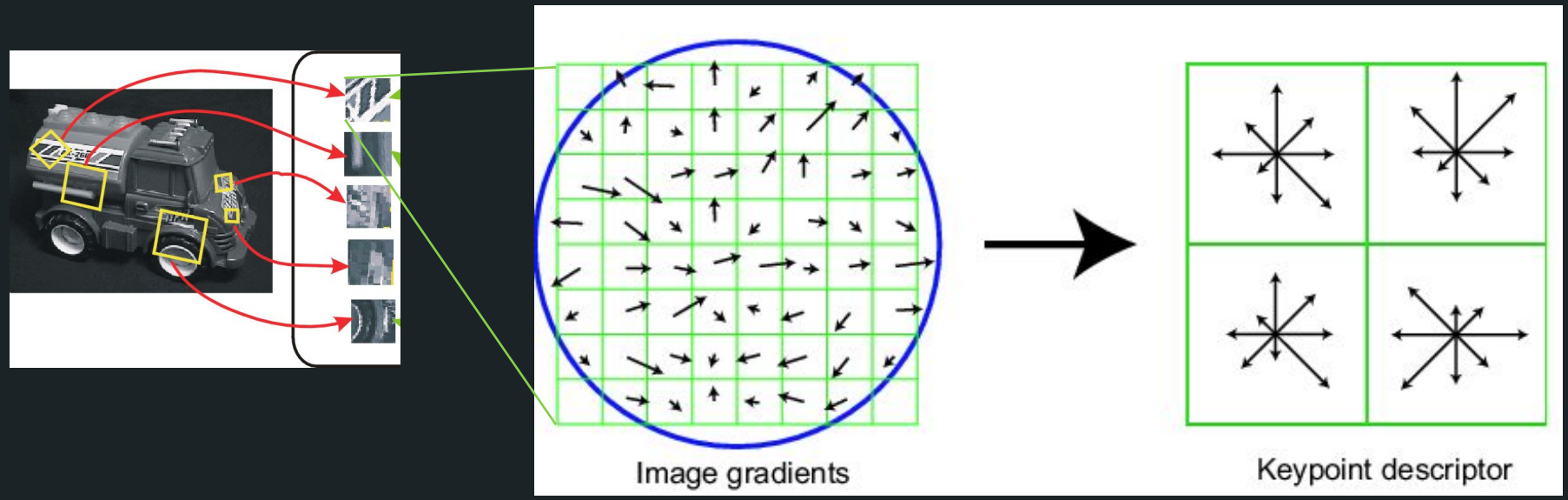
Ensure smoothness

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

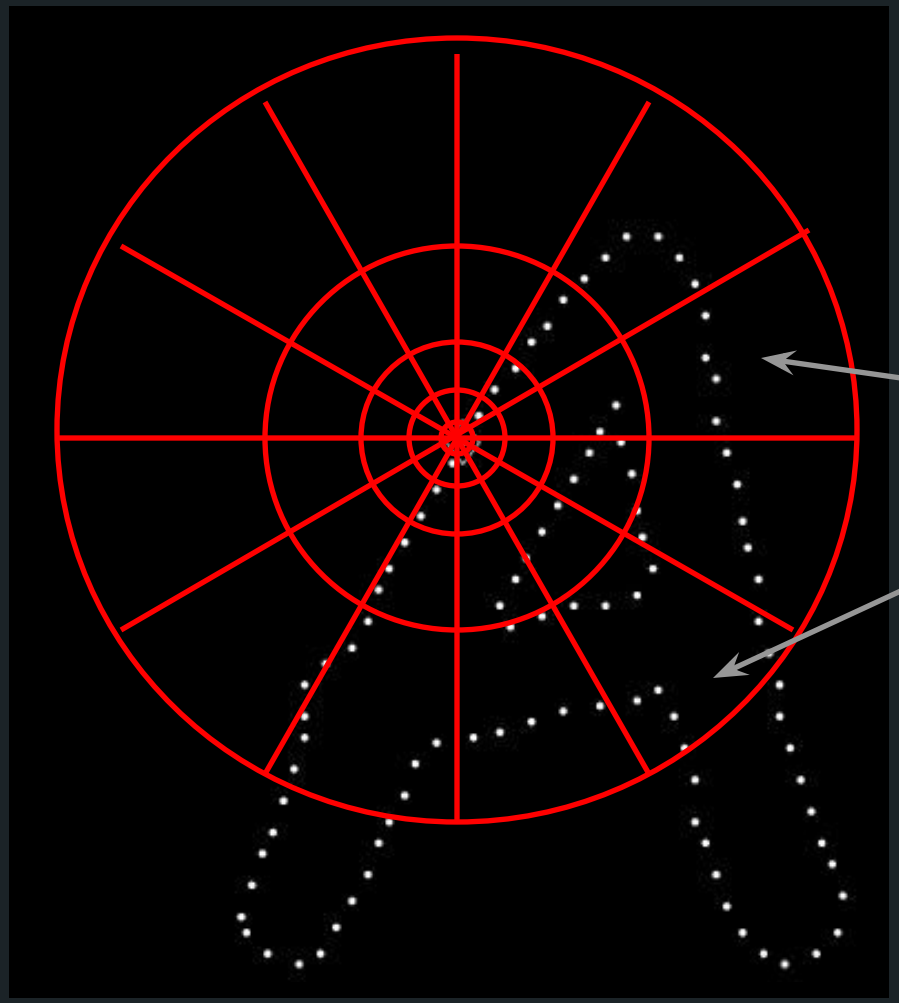


Reduce effect of illumination

- 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.:

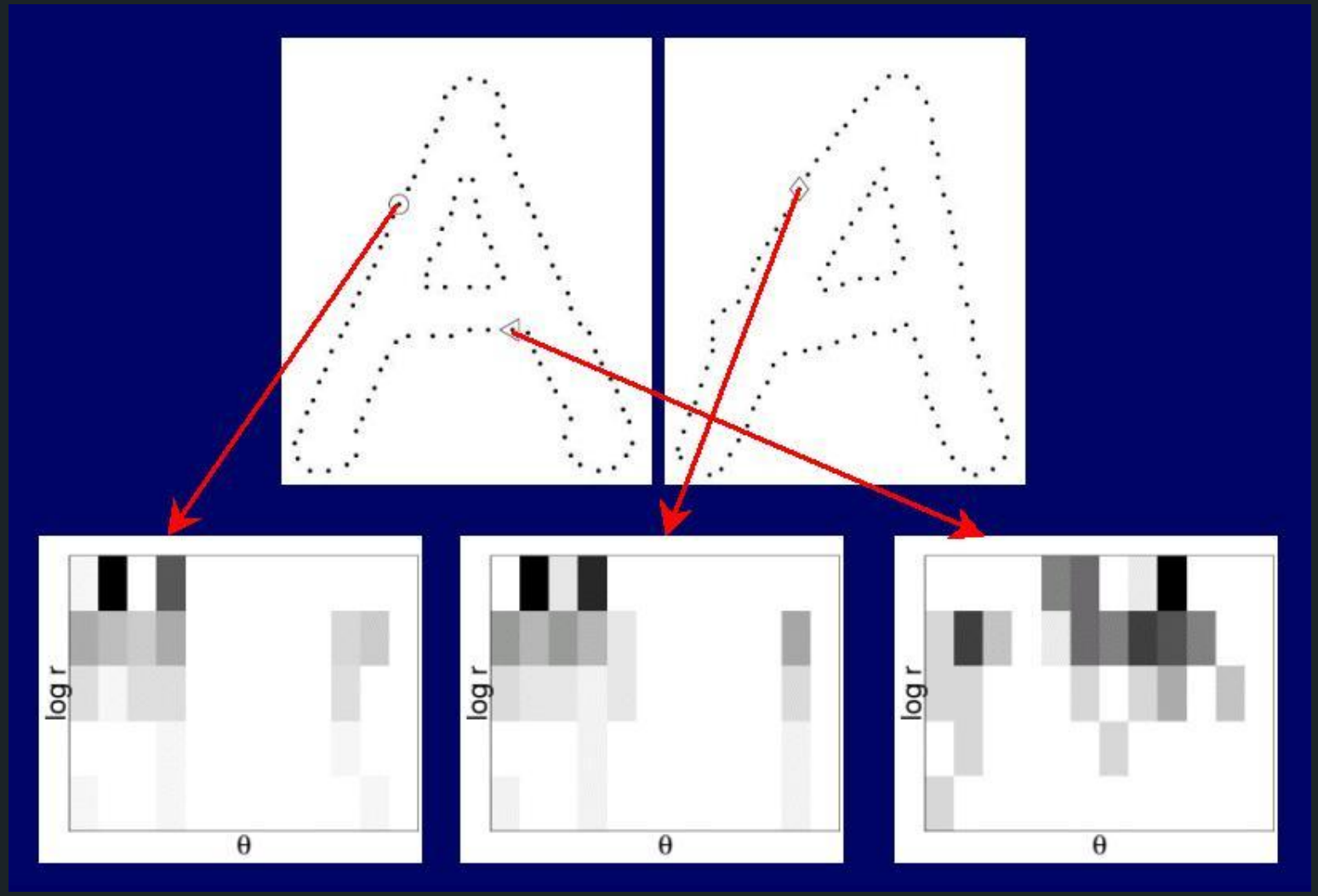
Count = 4

⋮

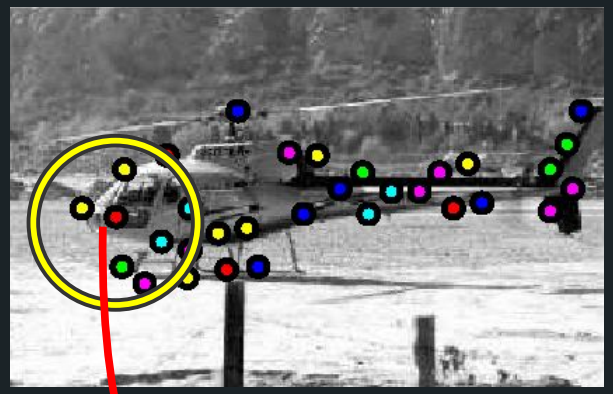
Count = 10

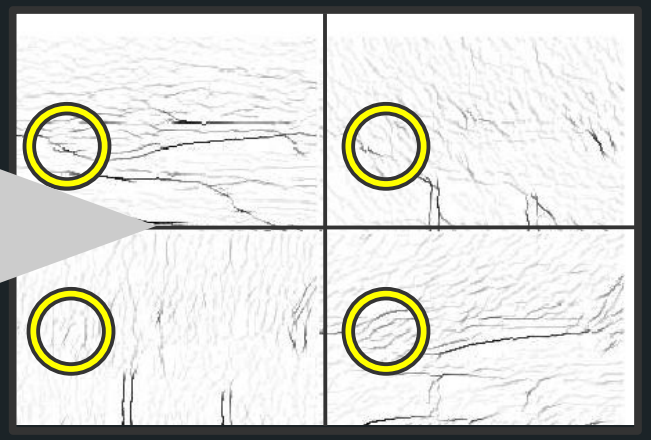
Log-polar binning: more precision for nearby points, more flexibility for farther points.

Shape Context Descriptor



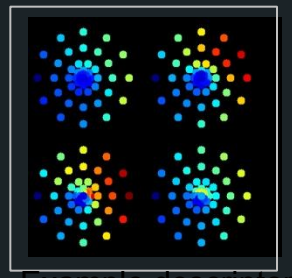
Local Descriptors: Geometric Blur



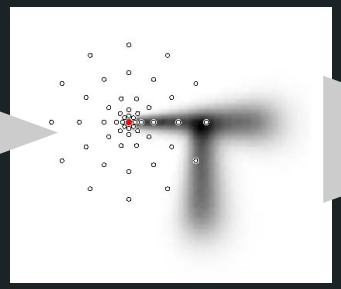


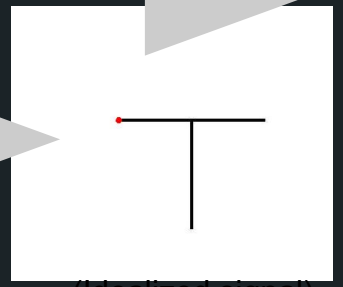
Compute edges at four orientations

Extract a patch in each channel



Example descriptor





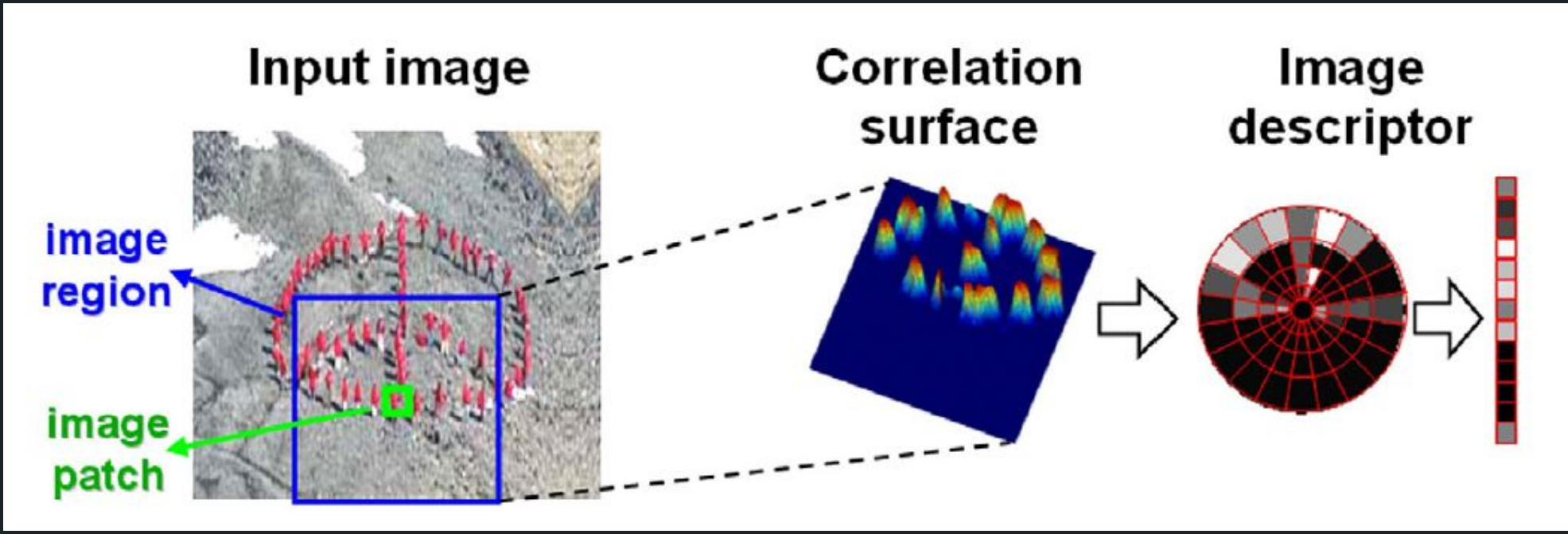
(Idealized signal)

Apply spatially varying blur and sub-sample

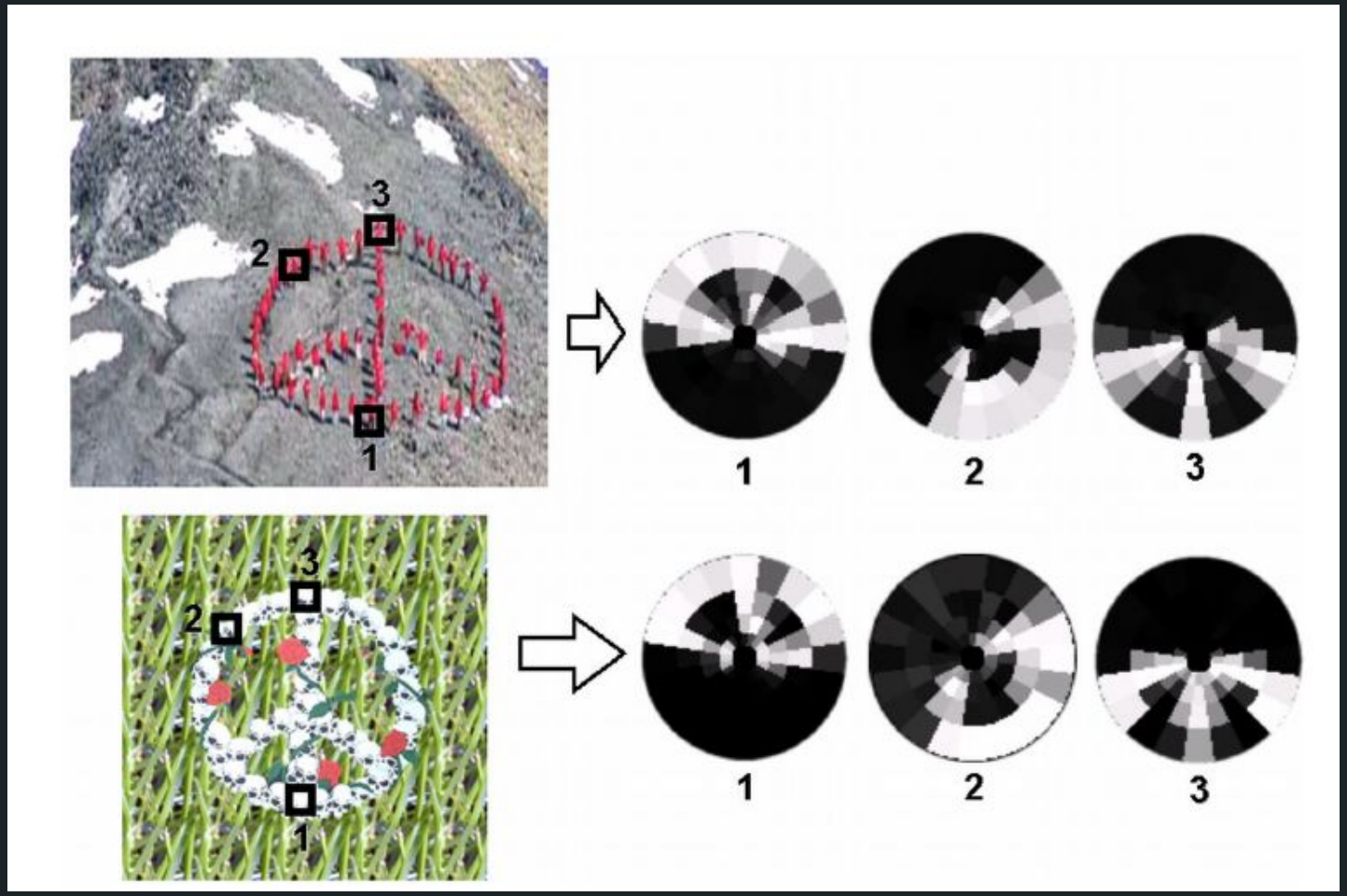
Self-similarity Descriptor

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.*

Self-similarity Descriptor



Self-similarity Descriptor



Right features depend on what you want to know

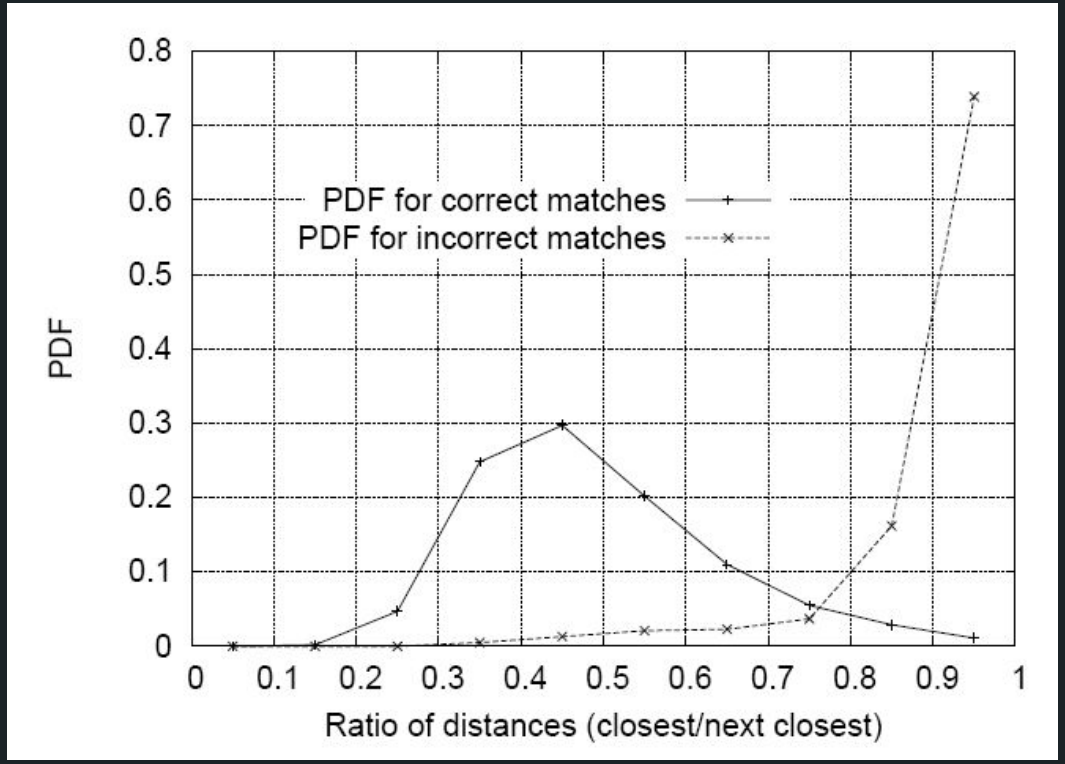
- Shape: scene-scale, object-scale, detail-scale
  - 2D form, shading, shadows, texture, linear perspective
- Material properties: albedo, feel, hardness, …
  - Color, texture
- Motion
  - Optical flow, tracked points
- Distance
  - Stereo, position, occlusion, scene shape
  - If known object: size,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

Local Descriptors

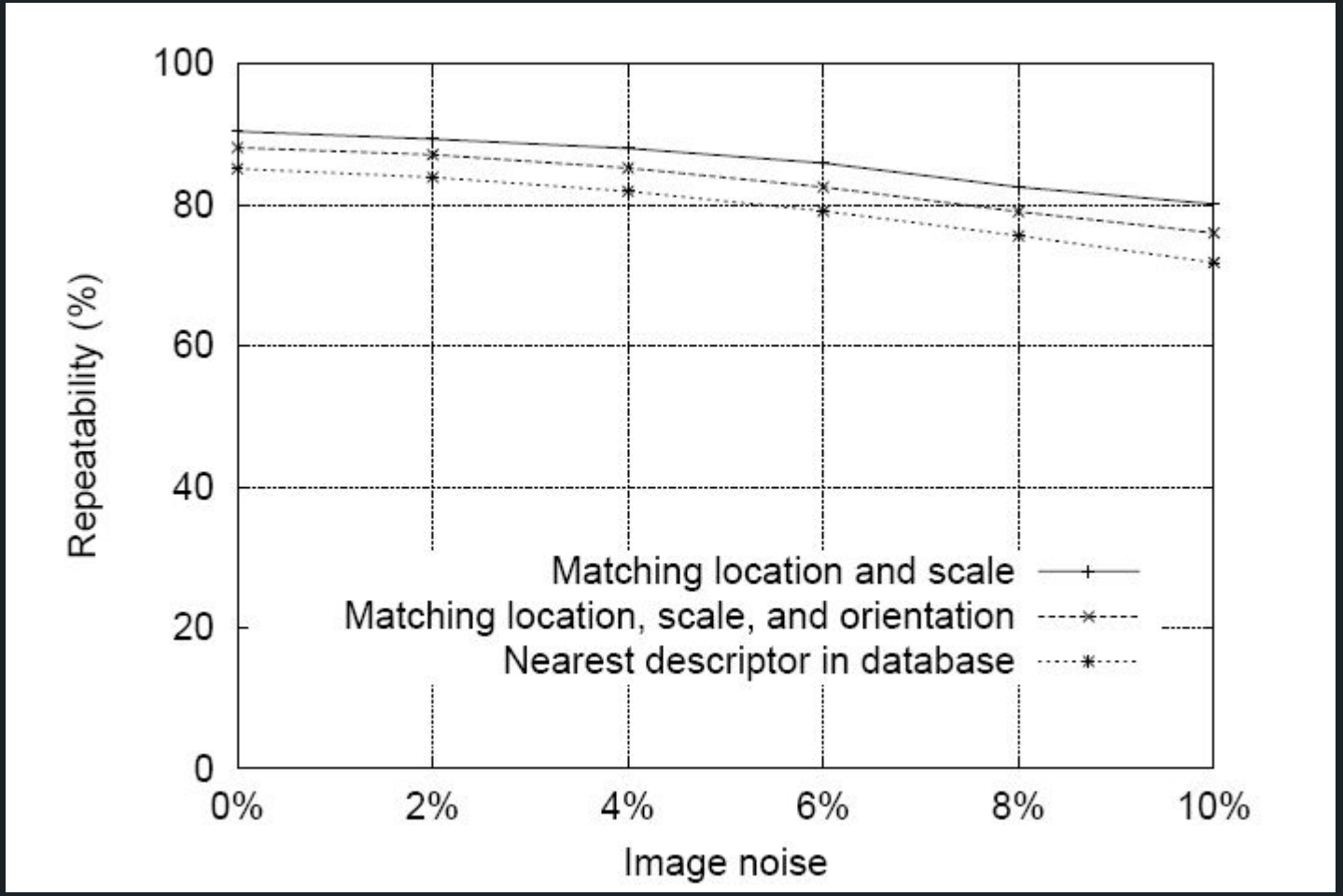
- Most features can be thought of as templates, histograms (counts), or combinations
- The ideal descriptor should be
  - Robust
  - Distinctive
  - Compact
  - Efficient
- Most available descriptors focus on edge/gradient information
  - Capture texture information
  - Color rarely used

Matching Local Features

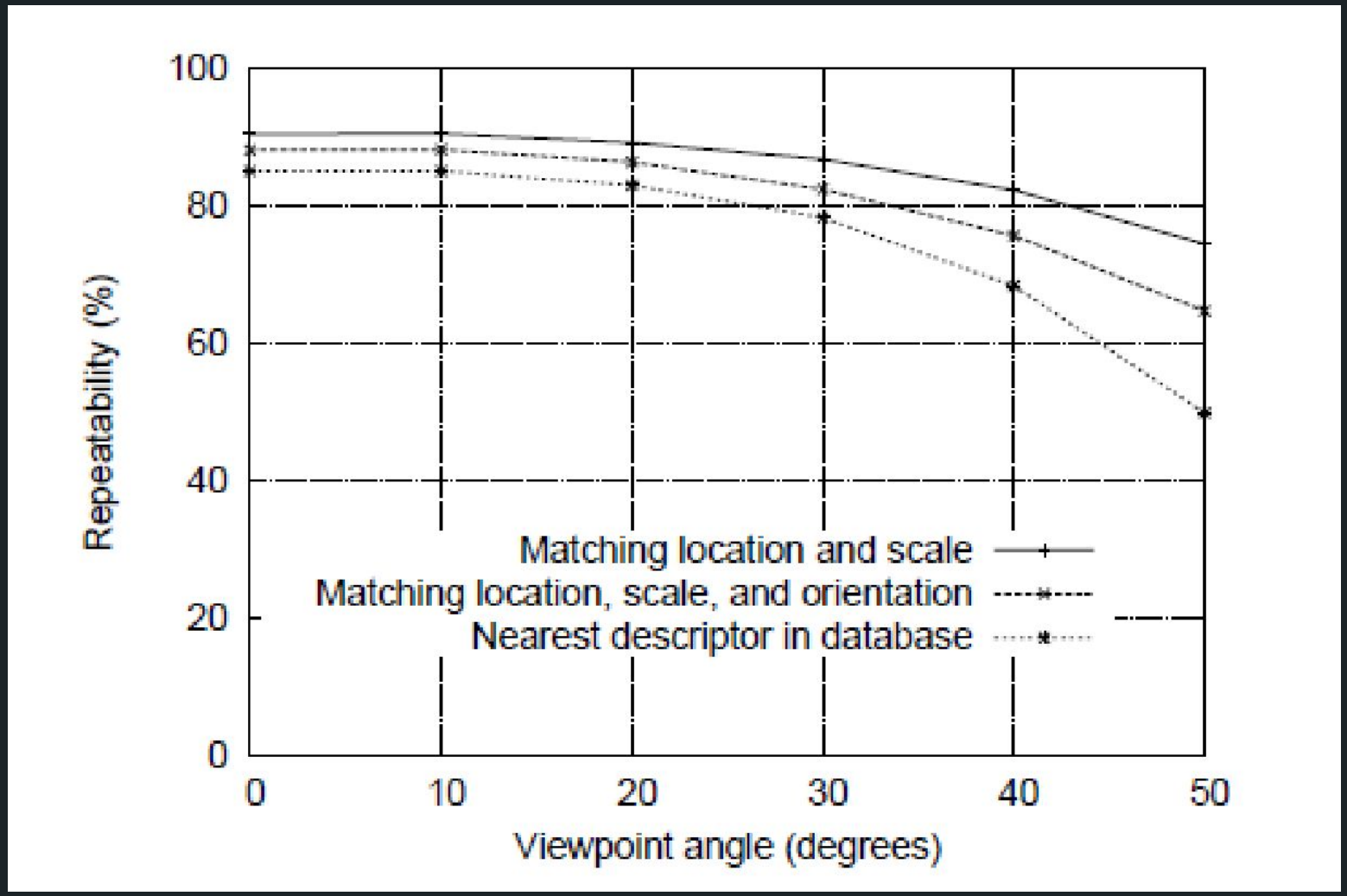
- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2nd nearest descriptor



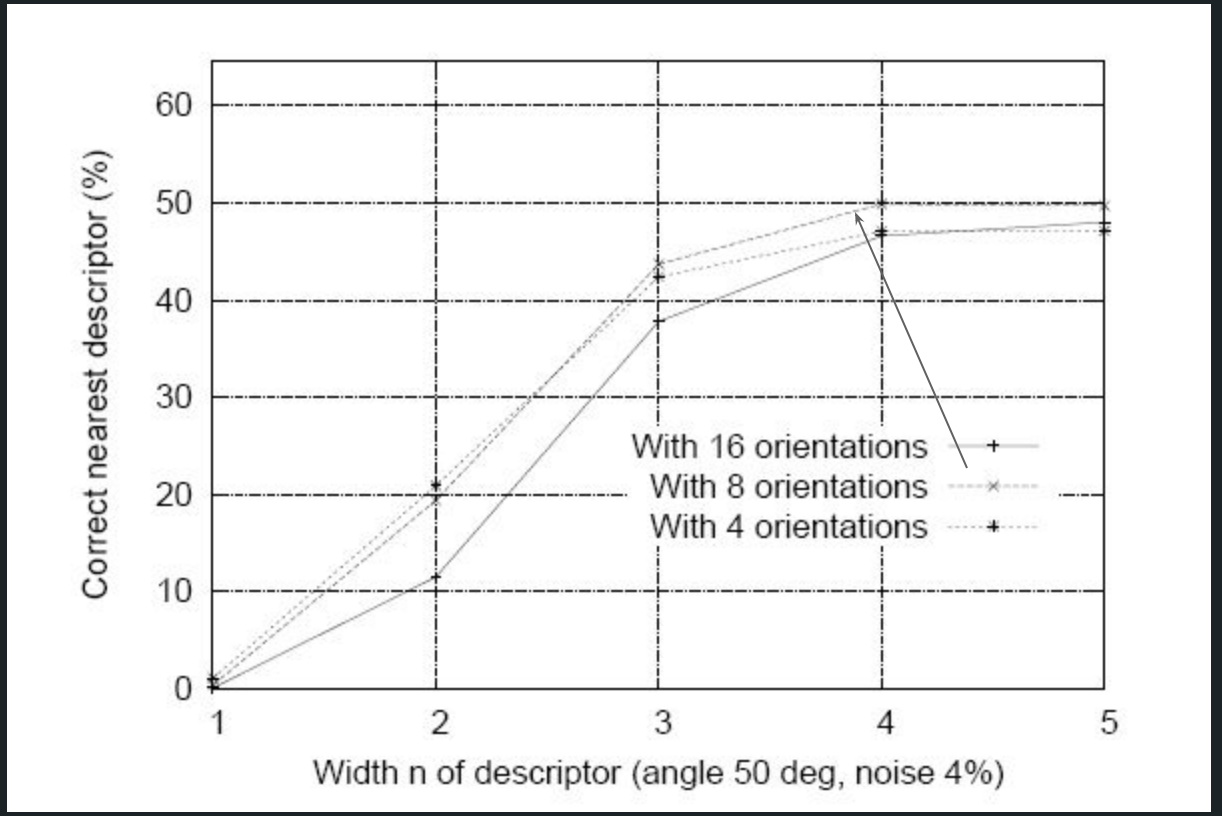
SIFT Repeatability



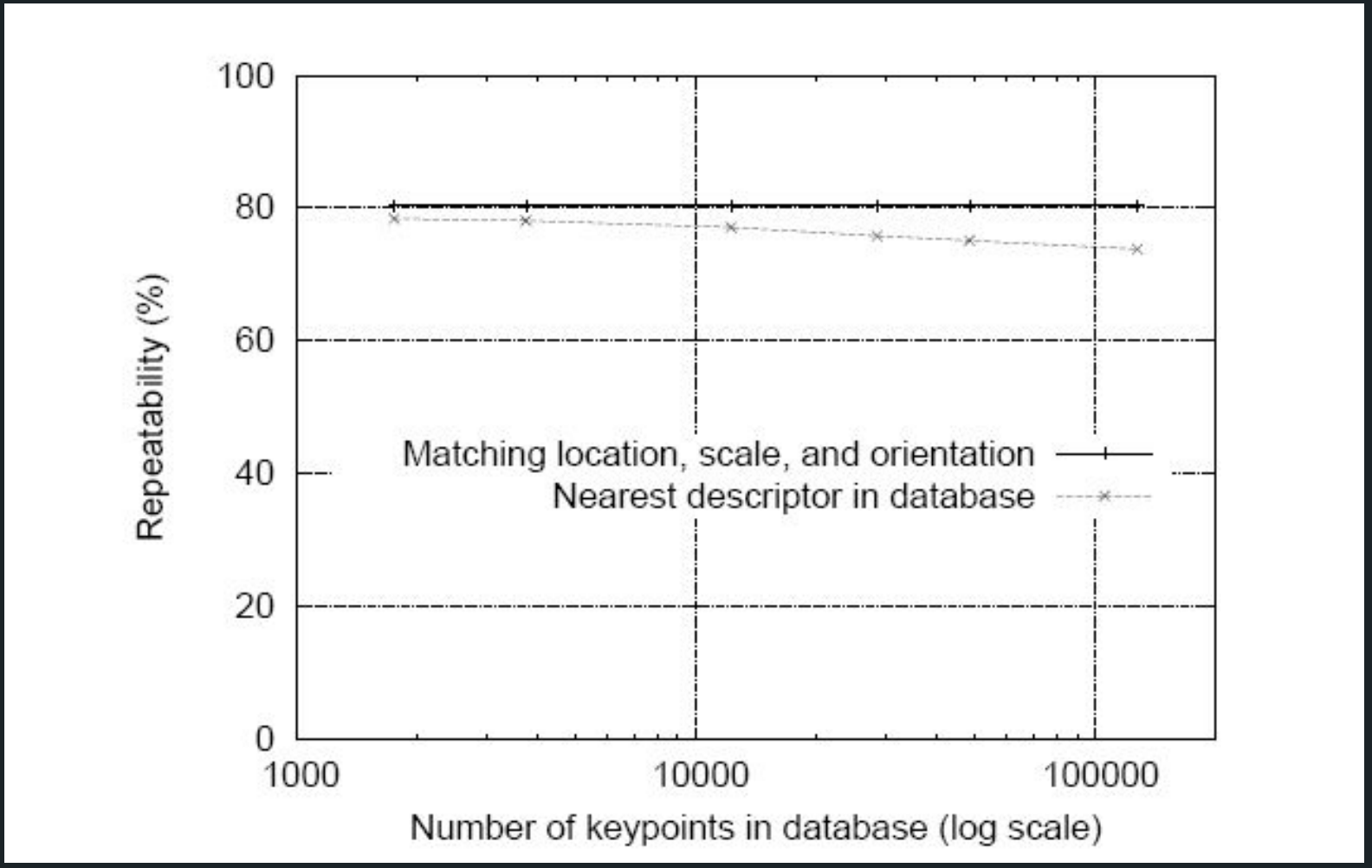
SIFT Repeatability



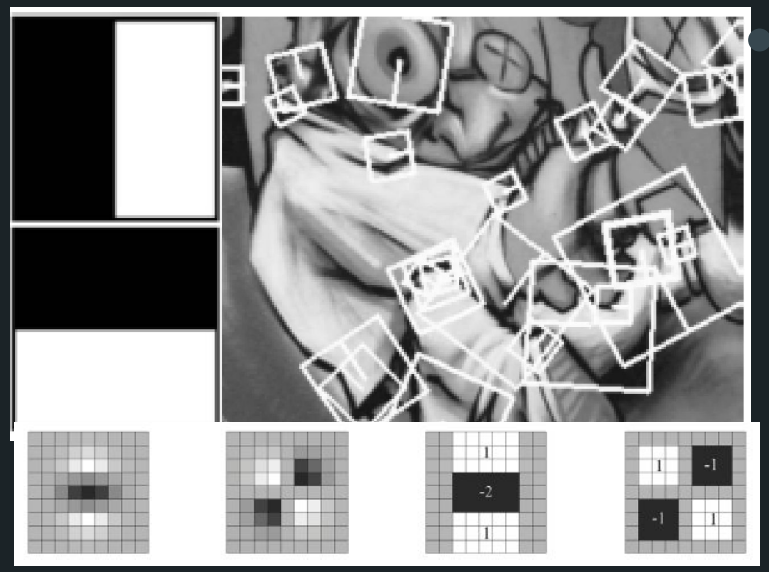
SIFT Repeatability



SIFT Repeatability

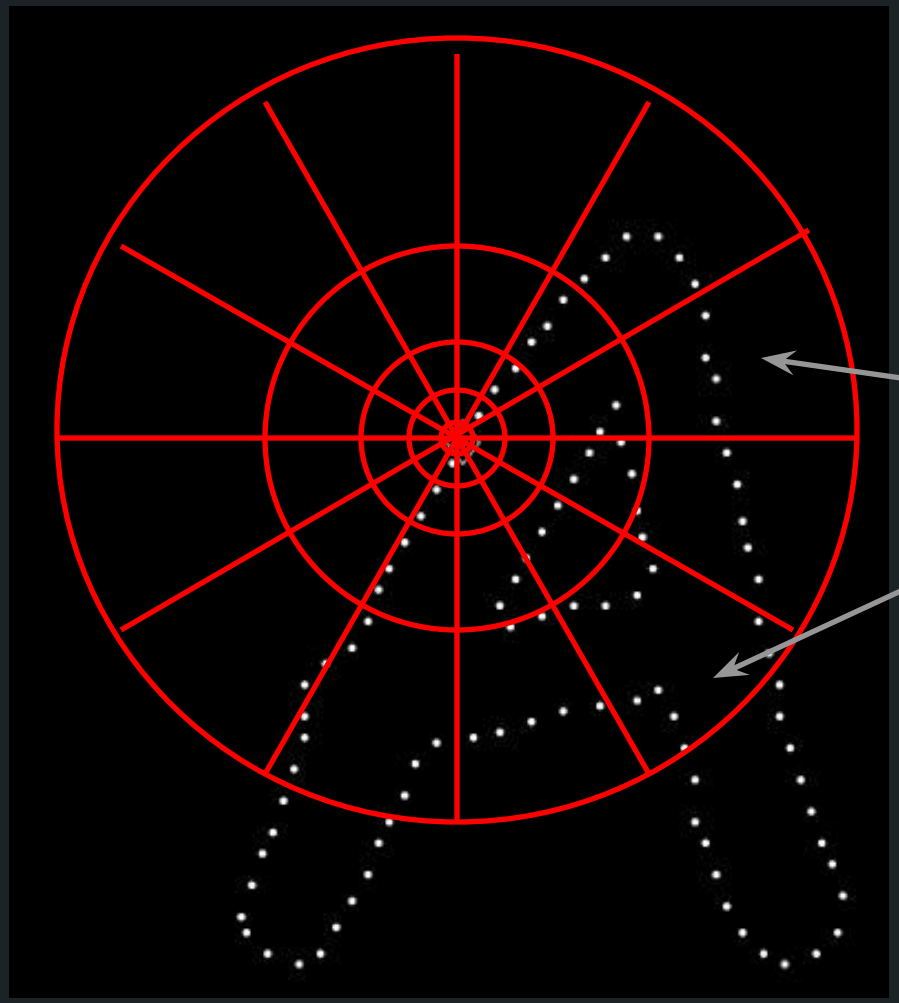


Local Descriptors: SURF



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

Local Descriptors: Shape Context



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

Count = 4

⋮

Count = 10

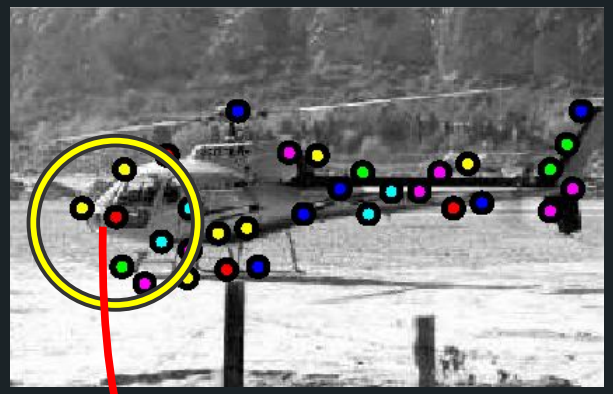
Log-polar binning: more precision for nearby points, more flexibility for farther points.

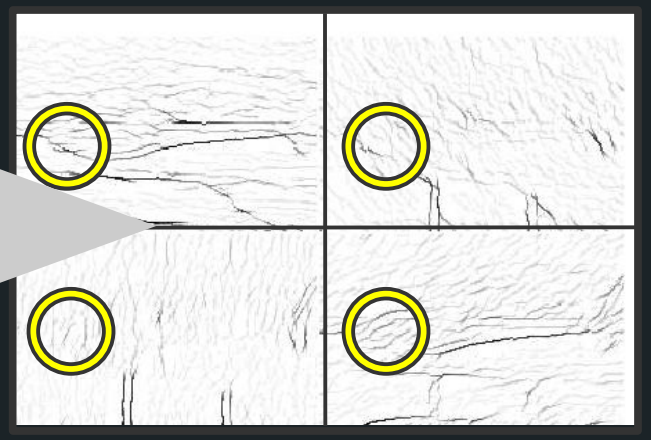
SURF(Speeded-Up Robust Features)

Similar to SIFT but much faster

More like a blob detector

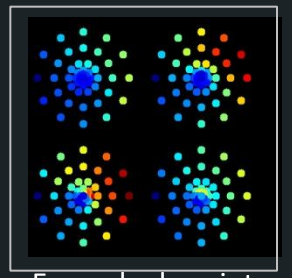
Local Descriptors: Geometric Blur



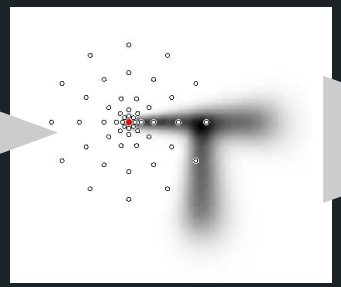


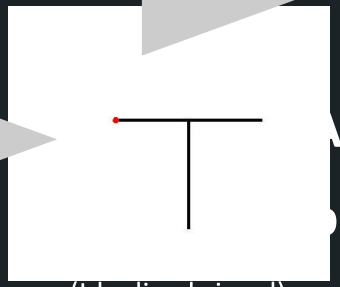
Compute edges at four orientations

Extract a patch in each channel



Example descriptor





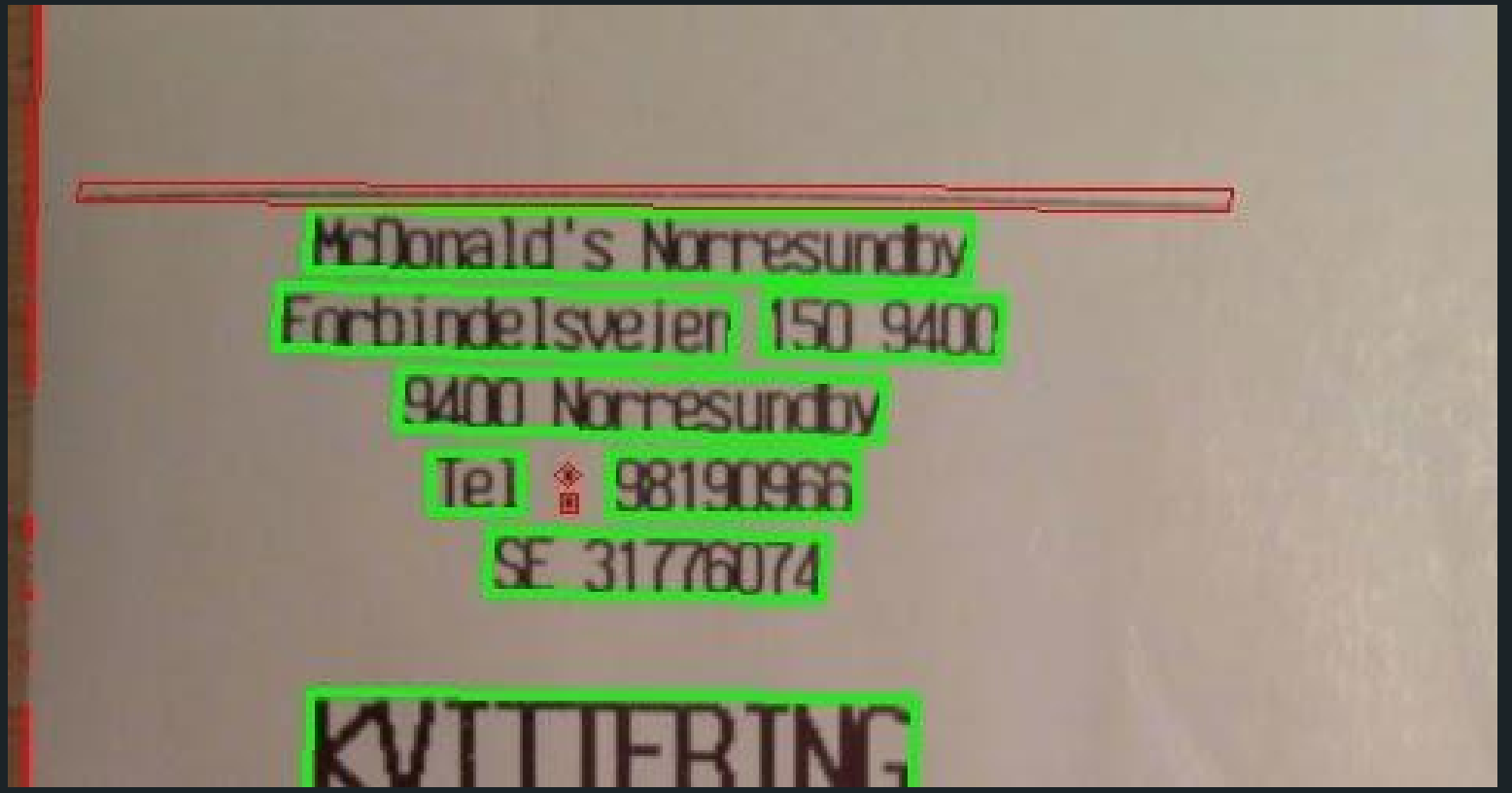
(Idealized signal)

Apply spatially varying blur and sub-sample

OTHER FEATURE DESCRIPTORS

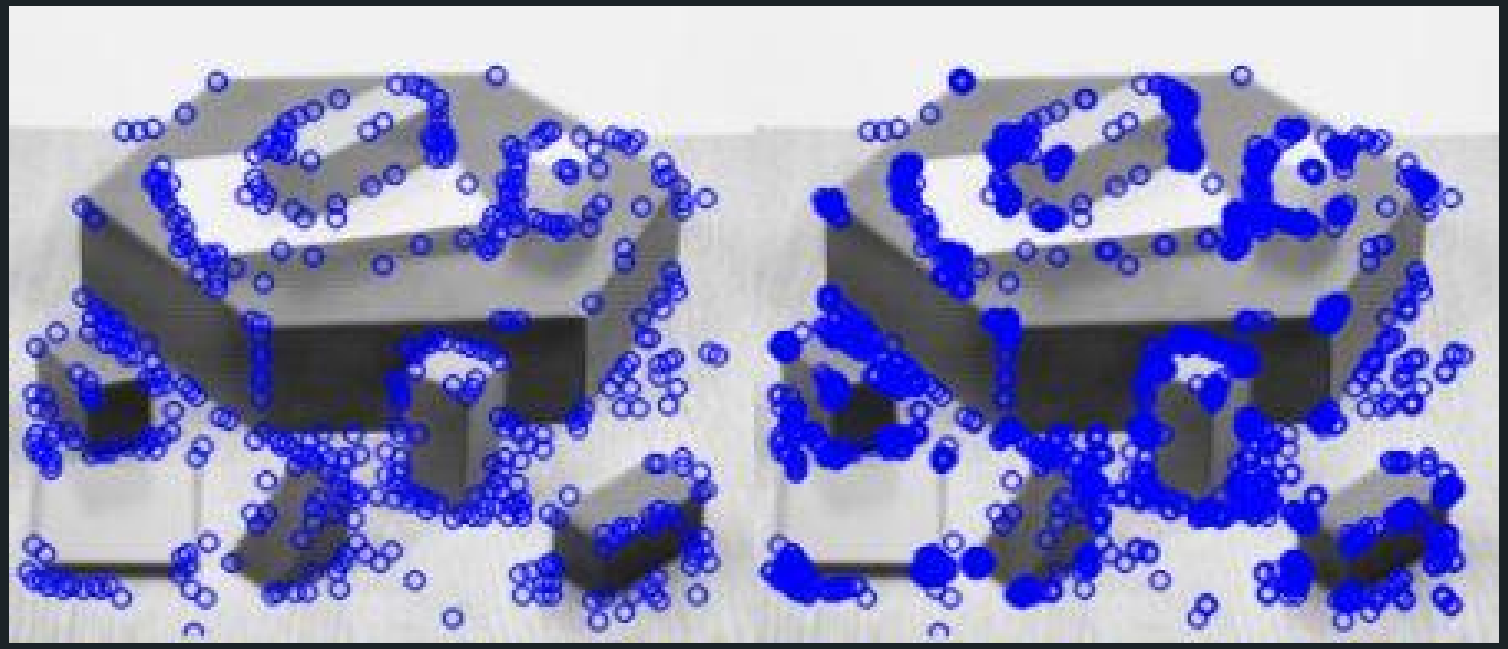
MSER (Maximally Stable Extremal Regions)

Blob Detection



FAST(Features from Accelerated Segment Test)

Machine learning as a corner detector



First image shows FAST with nonmaxSuppression and second one without nonmaxSuppression

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
  - 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			√	√	(√)	(√)	+	+	+	+

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