BMB5113
COMPUTER VISION

DETECTORS

Detectors

- Edge detectors
- Corner detectors
- Line detectors
- Circle detectors

  …
- Specific object detectors?

Robust Feature Exraction

- Detectors provide us with some features
- Matching those features is important to solve many problems such as
  - object or scene recognition
  - solving for 3D structure from multiple images
  - stereo correspondence
  - motion tracking
- Robust feature detectors should be:
  - Translation invariant
  - Scale invariant
  - Rotation invariant
  - Illumination invariant

An Example

- In a video frame there can be many local features to track
  - if a point is picked on a large blank wall then it is not easy to find that same point in the next frames
  - if a point is unique then it is a pretty good chance of finding that point again
  - usually corners are good points to track because of their having strong derivatives in two orthogonal directions

Descriptors

- Tries to define the characteristics of what is actually found by the detector
  - studies in local regions
  - computes some features in the local regions
    - magnitudes, gradients, angles, moments, projections etc.
  - Generates some histograms for the computed features
- Purpose may differ depending on detector
  - Point descriptors
  - Shape descriptors
  - Boundary descriptors

Edge Detectors

- Some well-known simple edge detectors
  - Marr-Hildreth edge detector
  - Canny-edge detector

Python Functions

```
import scipy.ndimage as snd
sx_sobel = snd.filters.sobel(im, axis=1, mode='constant')
sy_prewitt = snd.filters.prewitt(im, axis=0, mode='constant')


s_log = snd.filters.gaussian_laplace(im, (5,5))
s_laplace = snd.filters.laplace(im, mode='constant')

# conda install scikit-image
from skimage import feature
edges2 = feature.canny(im, sigma=3)
```

Marr-Hildreth Edge Detection

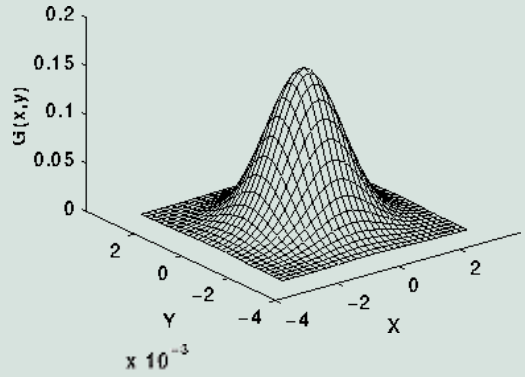
- Involves two phases
  1. Laplacian of Gaussian (LoG) filtering
  2. Finding zero-crossings

- Can detect direction-independent (isotropic) edge points.

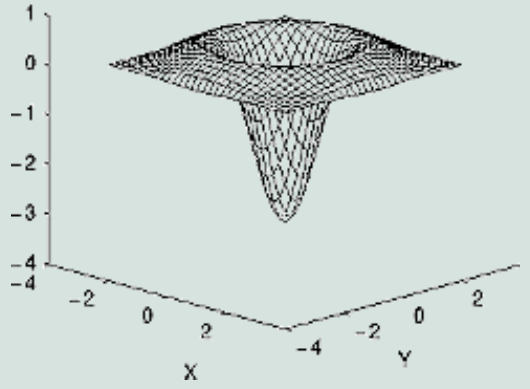
2-D LoG Filtering

- Second-order partial derivative of 2-D Gaussian function

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



$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Finding Zero-crossings

- Pixel locations where sign changes occur.
- Typical cases that are controlled:

$$T = constant(0.7) * \frac{\sum_{i=0,j=0}^{M-1,N-1} |I_{LoG}|}{M \times N}$$

Case I:

If $|a_{22}| + |a_{23}|$>T

If $|a_{22}| + |a_{21}|$>T

If $|a_{22}| + |a_{12}|$>T

If $|a_{22}| + |a_{32}|$>T

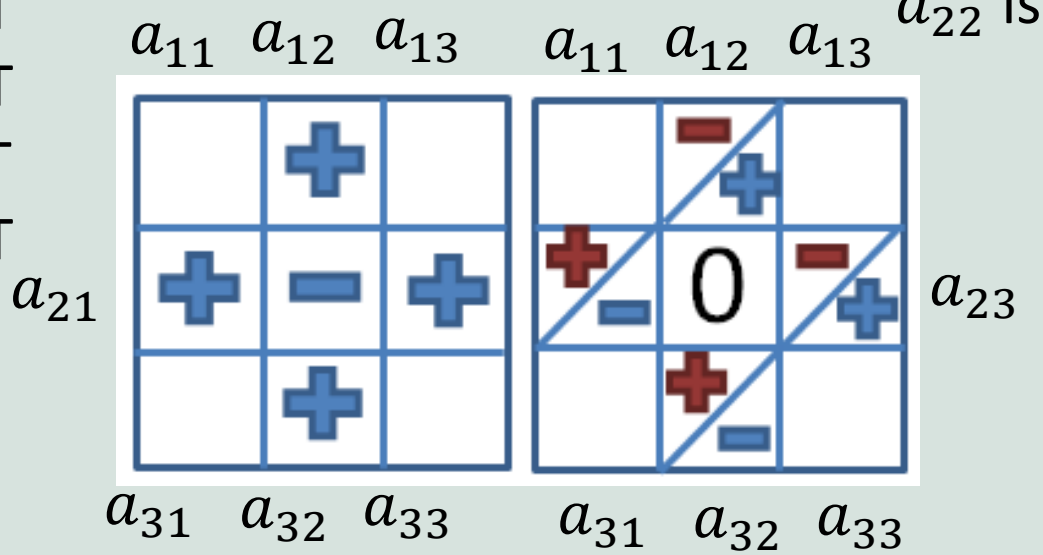
a_{22} is zero-cross

Case II:

If $|a_{12}| + |a_{32}|$>2*T

If $|a_{21}| + |a_{23}|$>2*T

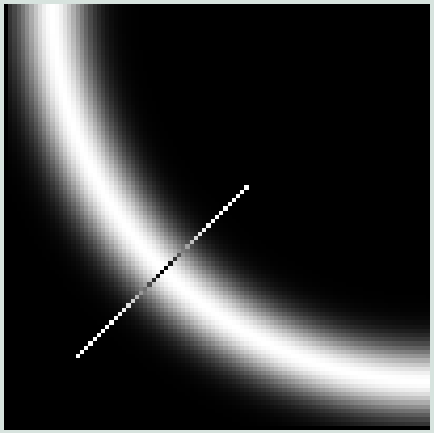
a_{22} is zero-cross

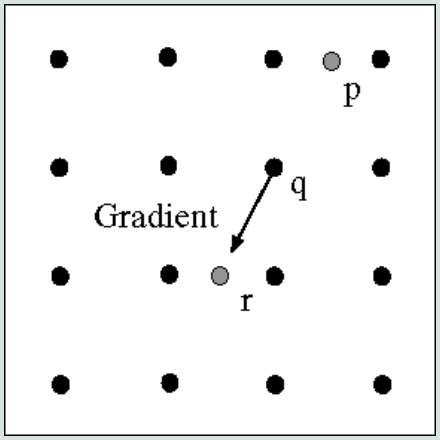


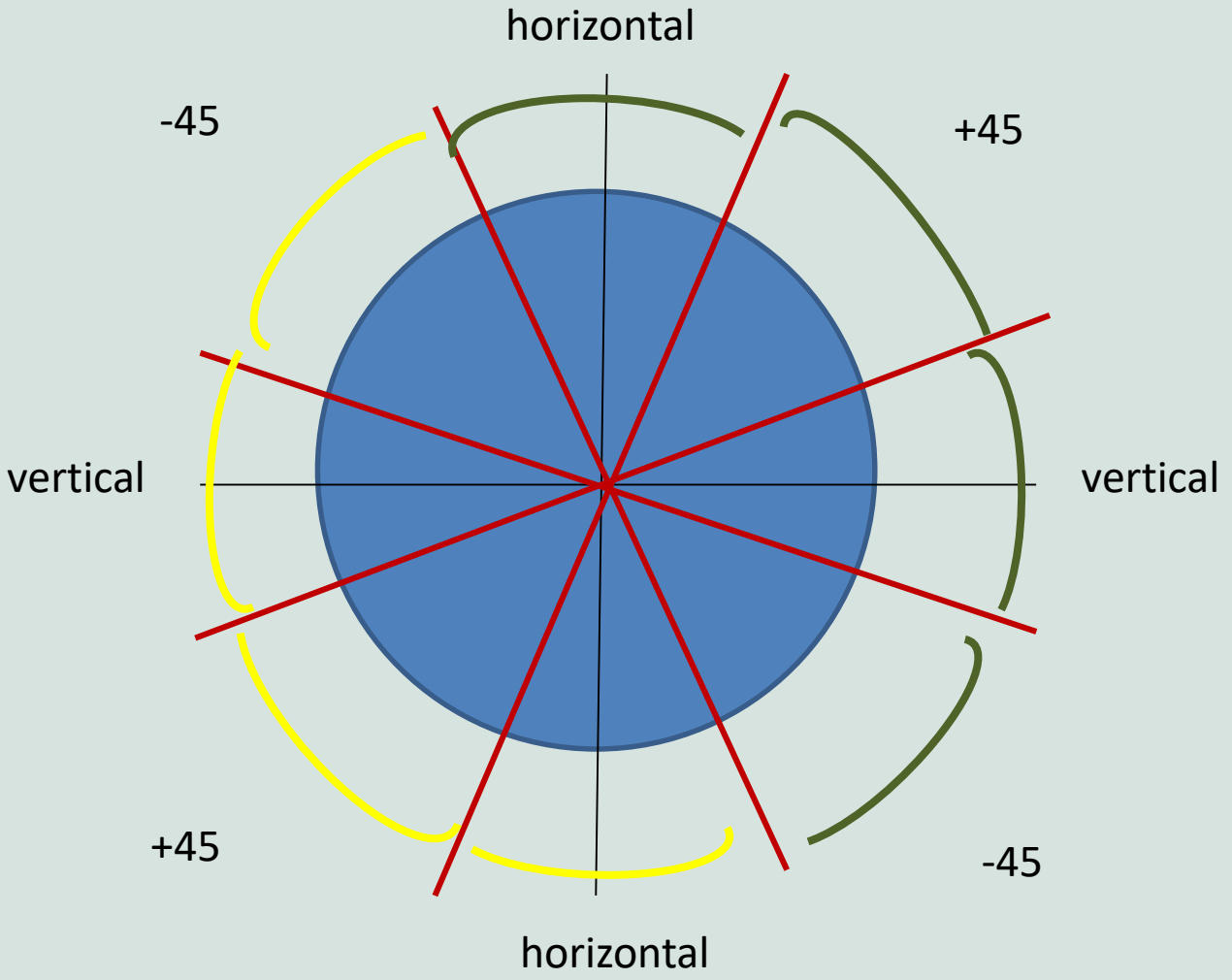
Canny Edge Detector

- Steps
  1. Smooth the image using Gaussian filters
  2. Compute first-order derivatives along main directions (i.e. X, Y, or Z)
  3. Compute magnitude matrix and angle matrix
  4. Suppress non-maximal points
     - Label angle matrix using main directions
     - Remove a point if it is less than one of its neighbours in magnitude along the edge normal (also known as gradient vector)

Non-maximum Suppression







Canny Edge Detector (cont'd)

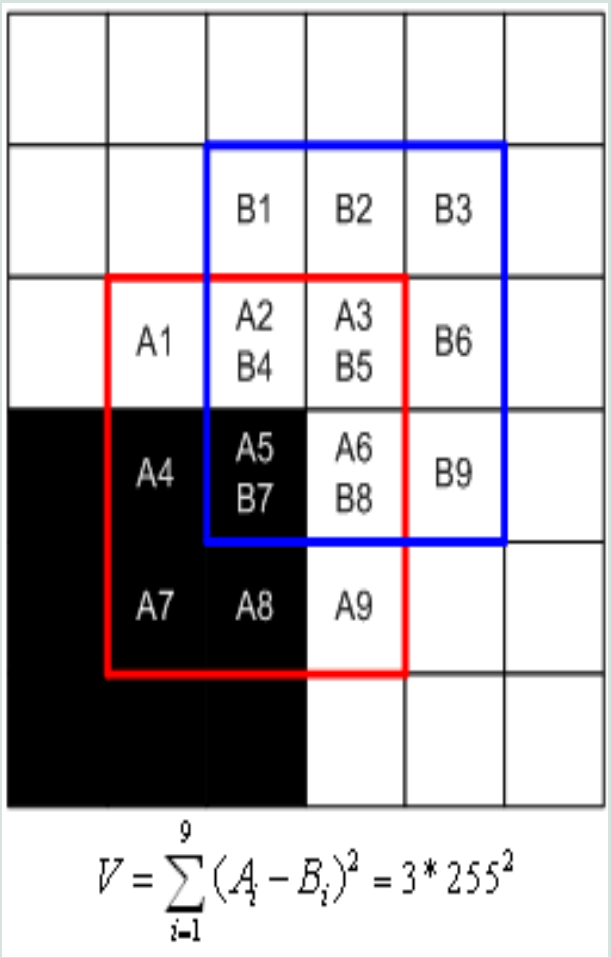
5. Clear false edge points using double thresholding
   - Remove points with magnitude less than the low-level threshold
   - For each point with magnitude greater than the high-level threshold perform connected component labeling
   - Remove any weak edge point that is not connected to a strong point
6. Perform morphological thin operation to make thick edges thinner

Point/Corner Detectors

- Harris corner detector
- SUSAN detector
- USAN detector
- Hessian detector
- Shi and Tomasi
- FAST

Harris Corner Detector

- Depends on Moravec detector (Moravec,1979)
  - Computes local intensity differences in 8 main directions
  - Cornerness measure is the least one of the summed intensity differences
  - Removes points weak in cornerness value
  - Points greater than their neighbours in a window W are final corners



Harris Corner Detector

- Harrris corner detector algorithm (Harris ve Stephens, 1988)
  - Unlike Moravec not limited to only 8 direction
    - Interest points with higher repeatability
  - Searches for points with high gradients in two main directions in local neighborhoods
    - Not totally independent of rotations (Schmid vd., 1998)
  - Scale-space representation may be adapted for scale independence
    - Harris-Laplace is a widely known variation

Harris Corner Detector

X-direction

A1	A2 B1	A3 B2	B3
A4	A5 B4	A6 B5	B6
A7	A8 B7	A9 B8	B9

Yatay Moravec yoğunluk değişimi

$$V_x = \sum_{i=1}^{9} (A_i - B_i)^2 = \sum_{i=1}^{9} (B_i - A_i)^2 \approx \sum_{i=1}^{9} \left(\frac{\partial I_i}{\partial x} \right)^2$$

öyle ki: $\frac{\partial I_i}{\partial x} \equiv \mathrm{I}_i \otimes (-1, 0, 1) \approx B_i - A_i$

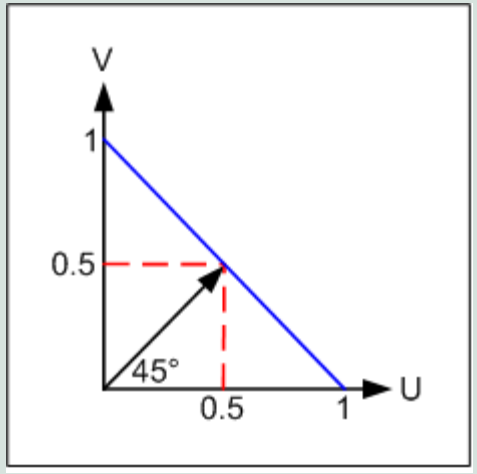
H-direction

	B1	B2	B3
A1	A2 B4	A3 B5	B6
A4	A5 B7	A6 B8	B9
A7	A8	A9	

Köşegensel Moravec yoğunluk değişimi

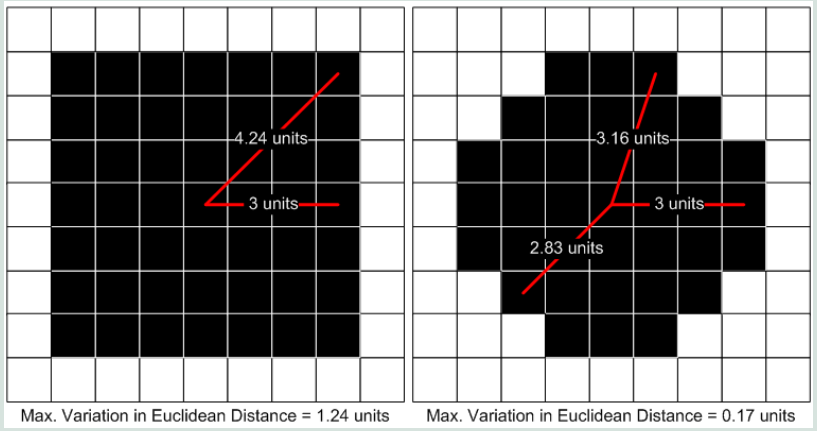
$$V_h = \sum_{i=1}^{9} (A_i - B_i)^2 = \sum_{i=1}^{9} (B_i - A_i)^2 \approx \sum_{i=1}^{9} \left(\frac{\partial I_i}{\partial h} \right)^2$$

öyle ki: $\frac{\partial I_i}{\partial h} \equiv I_i \otimes \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \approx B_i - A_i$



$$V_{u,v}(x,y) = \sum_{(x,y)\ merkezli\ pencerede\ \forall i\ icin} \left(u\frac{\partial I_i}{\partial x} + v\frac{\partial I_i}{\partial y} \right)^2$$

Harris Corner Detector



$$V_{u,v}(x,y) = \sum_{(x,y)\ merkezli\ pencerede\ \forall i\ icin} w_i \left(u\frac{\partial I_i}{\partial x} + v\frac{\partial I_i}{\partial y} \right)^2$$

$$= \sum_{(x,y)\ merkezli\ pencerede\ \forall i\ icin} w_i \left(u^2 \frac{\partial I_i}{\partial x}^2 + 2uv\frac{\partial I_i}{\partial x}\frac{\partial I_i}{\partial y} + v^2 \frac{\partial I_i}{\partial y}^2 \right)$$

$$= Au^2 + 2Cuv + Bv^2$$

$$A = \left(\frac{\partial I_i}{\partial x}\right)^2 \otimes w$$

$$B = \left(\frac{\partial I_i}{\partial y}\right)^2 \otimes w$$

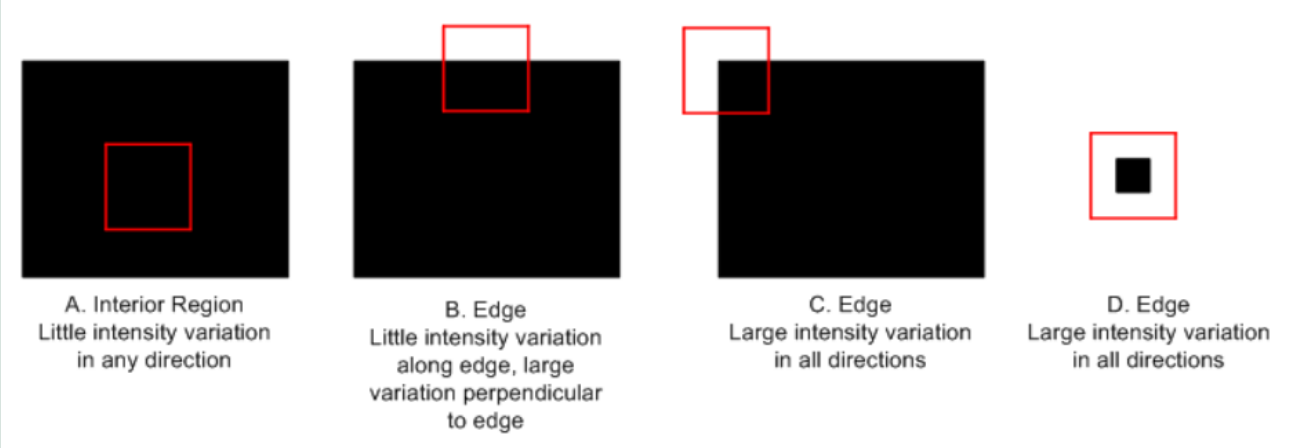
$$C = \left(\frac{\partial I_i}{\partial x}\frac{\partial I_i}{\partial y}\right) \otimes w$$

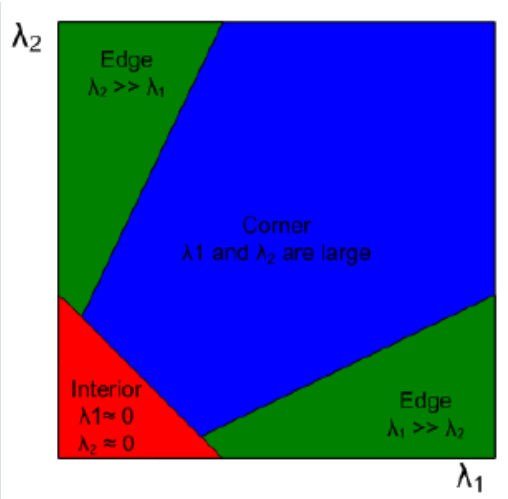
$$V_{u,v}(x,y) = Au^2 + 2Cuv + Bv^2 = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix},$$

$$ki\ M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

Harris Corner Detector

- Eigenvalues of autocorrelation matrix M indicate how a point is strong in gradient magnitude, especially in two main directions

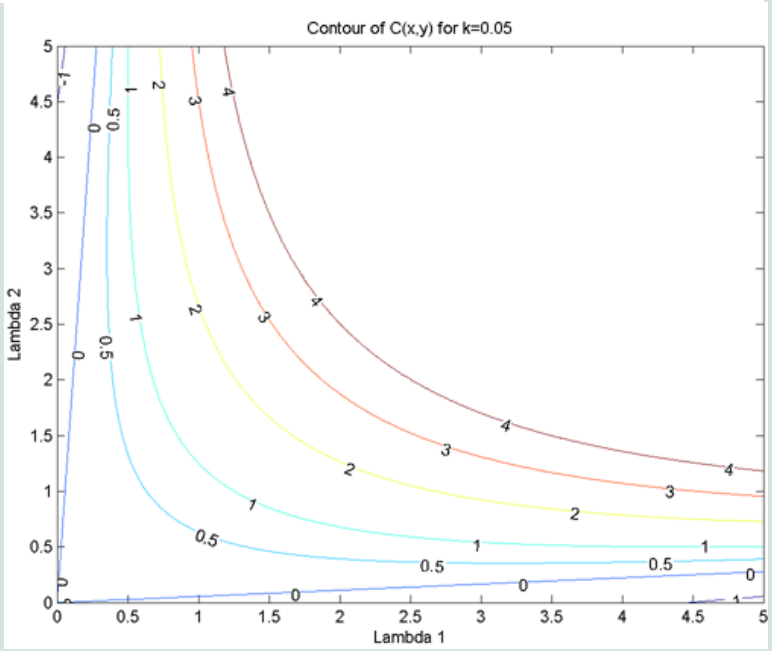




Harris Corner Detector

- Practically without explicitly computing the eigenvalues of M cornerness measure can be computed according to:
- Experiments showed best results when k is in between 0.04 and 0.06
- A threshold needs to be applied over cornerness measures

$$C(x,y) = \det(M) - k(\mathrm{trace}(M))^2$$
$$\det(M) = \lambda_1\lambda_2 = AB - C^2$$
$$\mathrm{trace}(M) = \lambda_1 + \lambda_2 = A + B$$
$$k = sabit$$



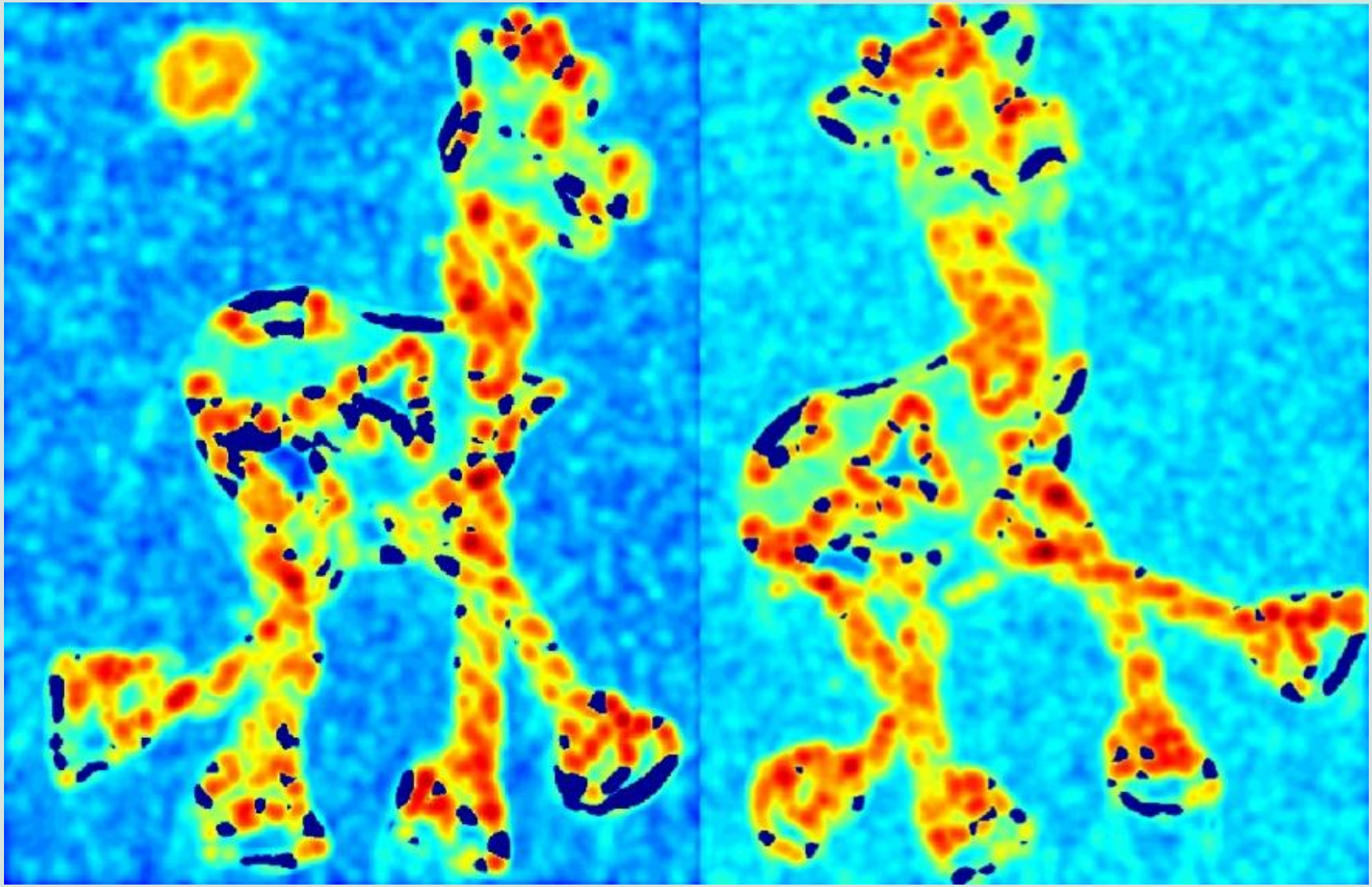
Harris Corner Detector: Algorithm

- Input:
  - Gray-scale I, Gaussian window variance, constant k, threshold T over cornerness measures, window W
- Output: corner coordinates in the image
- Algorithm:
  1. For each pixel (x, y) compute autocorrelation matrix M
  2. For each points compute cornerness measure
  3. Apply threshold T over cornerness measures C(x, y)
  4. Determine points higher in cornerness measure with respect to their neighbours in window W

Harris Detector: Workflow

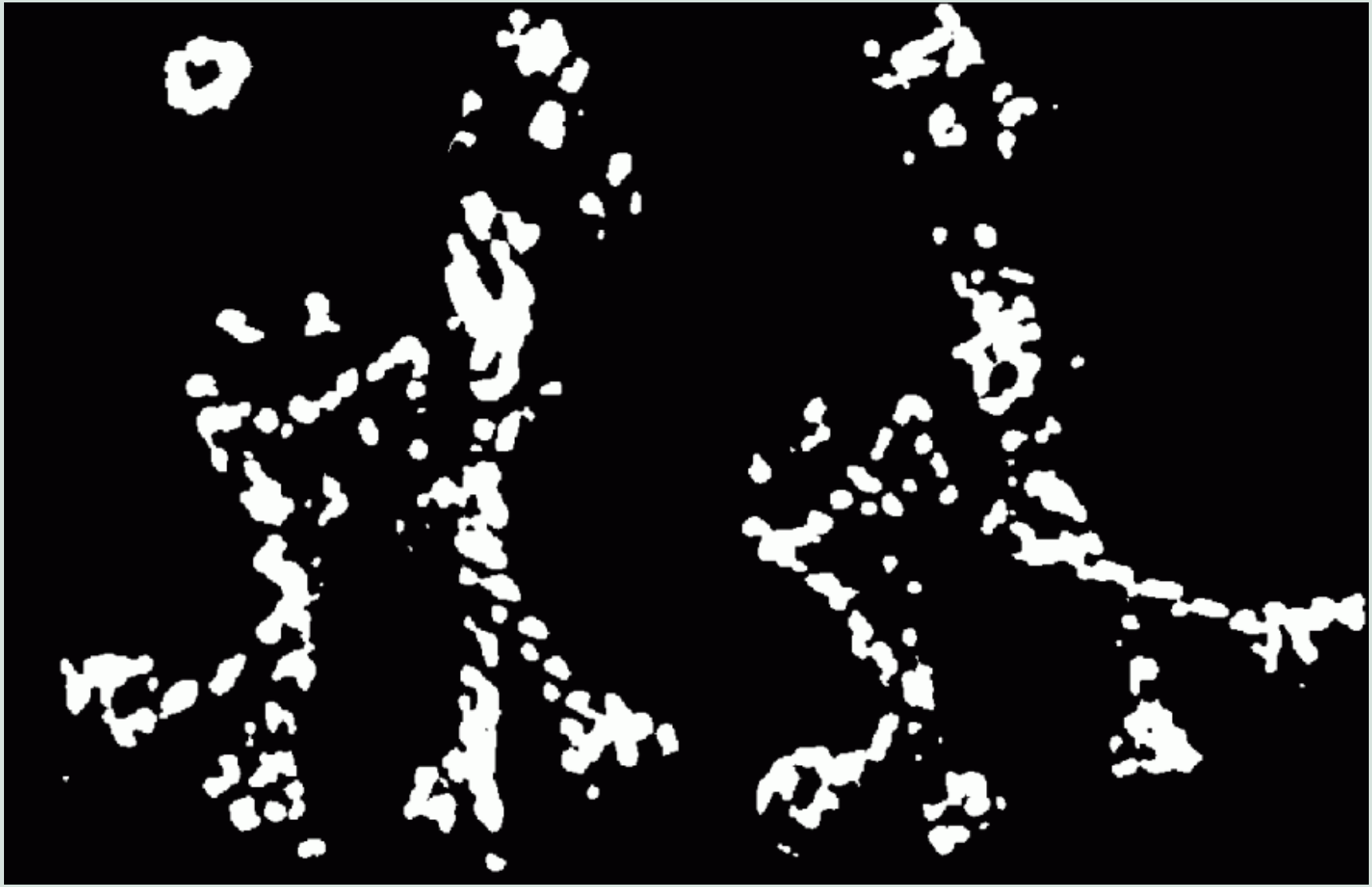
Harris Detector: Workflow

Compute corner response C



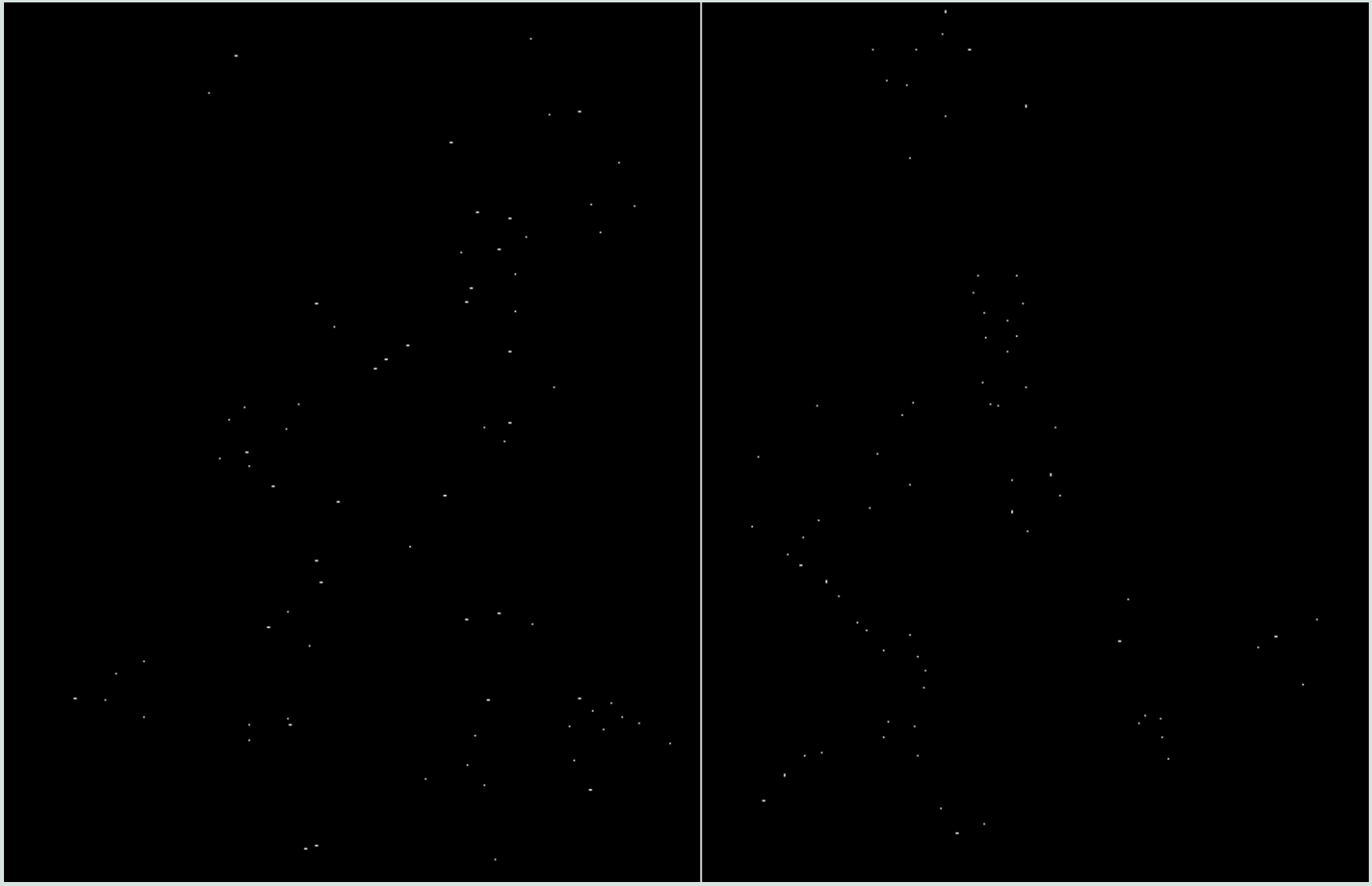
Harris Detector: Workflow

Find points with large corner response: C>threshold



Harris Detector: Workflow

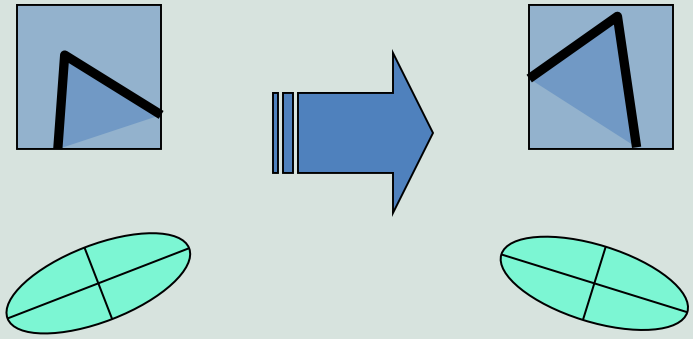
Take only the points of local maxima of C



Harris Detector: Workflow

Harris Detector: Properties

- Rotation invariance

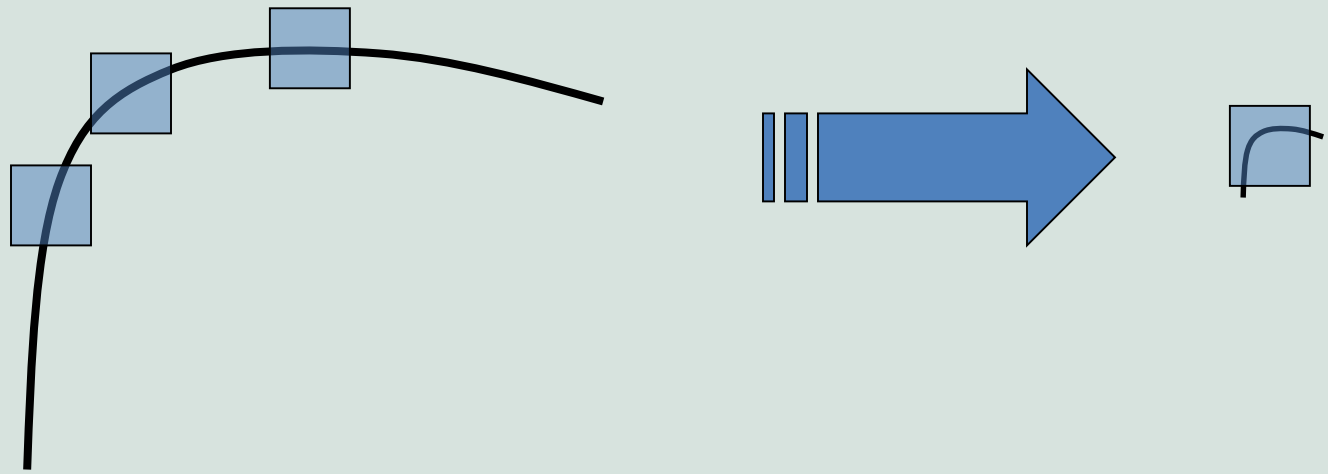


Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response C is said to be invariant to image rotation

Harris Detector: Properties

- Not invariant to image scale



All points will be classified as edges

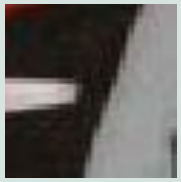
Corner !

Scale Invariance

- How can we detect **scale invariant** interest points?
- How to cope with transformations?
  - Exhaustive search
  - Invariance
  - Robustness

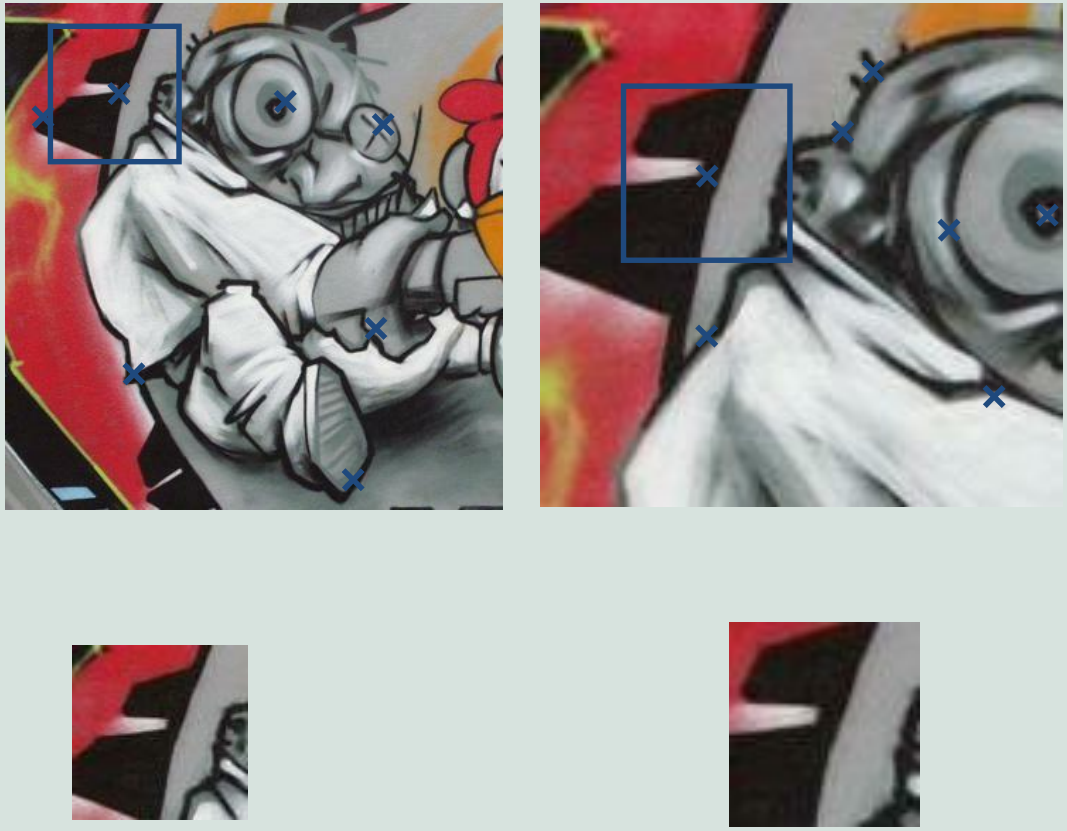
Exhaustive search

- Multi-scale approach



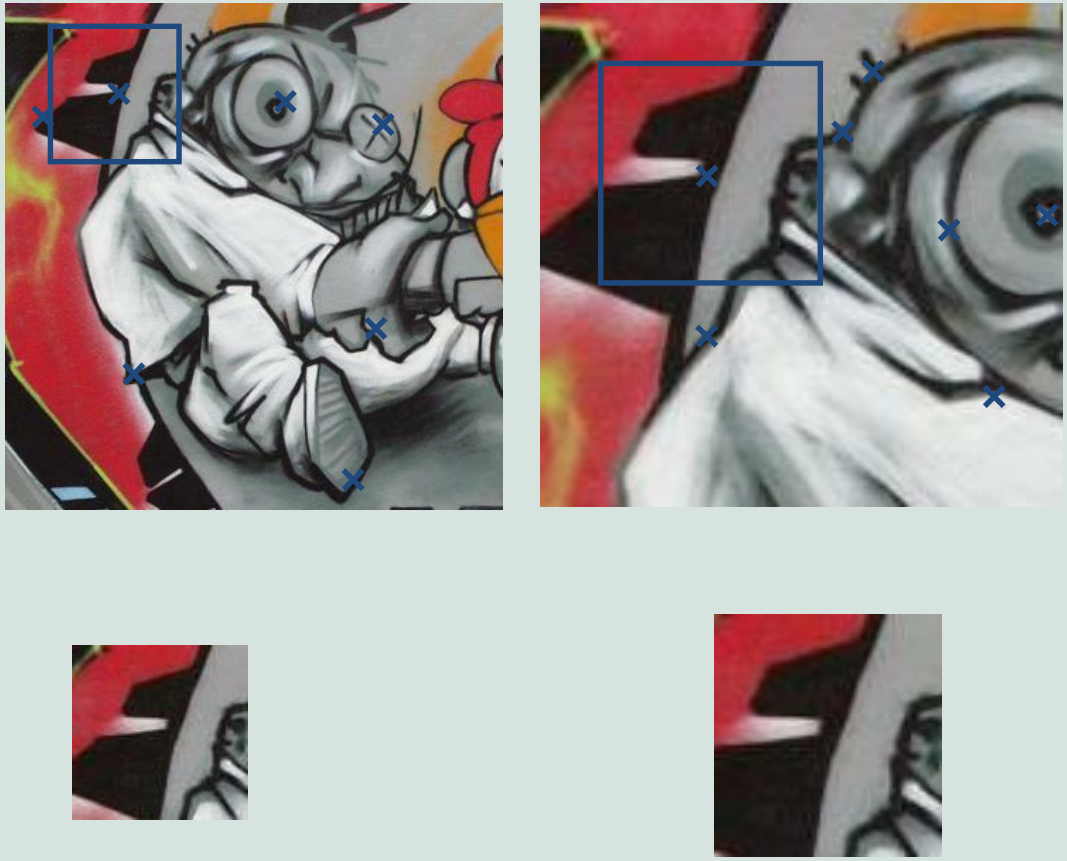
Exhaustive search

- Multi-scale approach



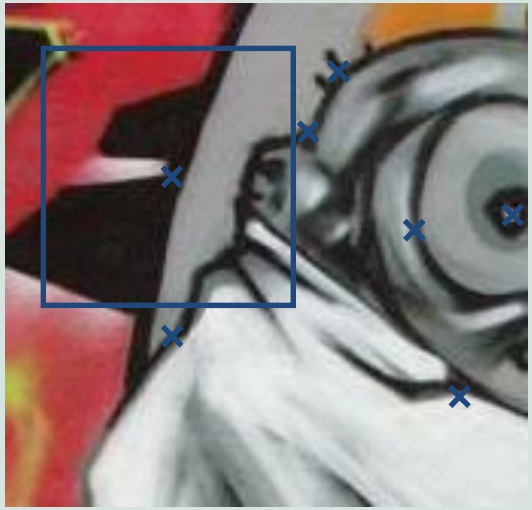
Exhaustive search

- Multi-scale approach



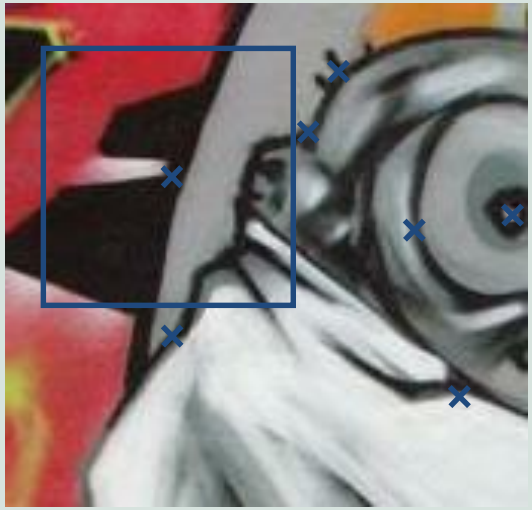
Exhaustive search

- Multi-scale approach



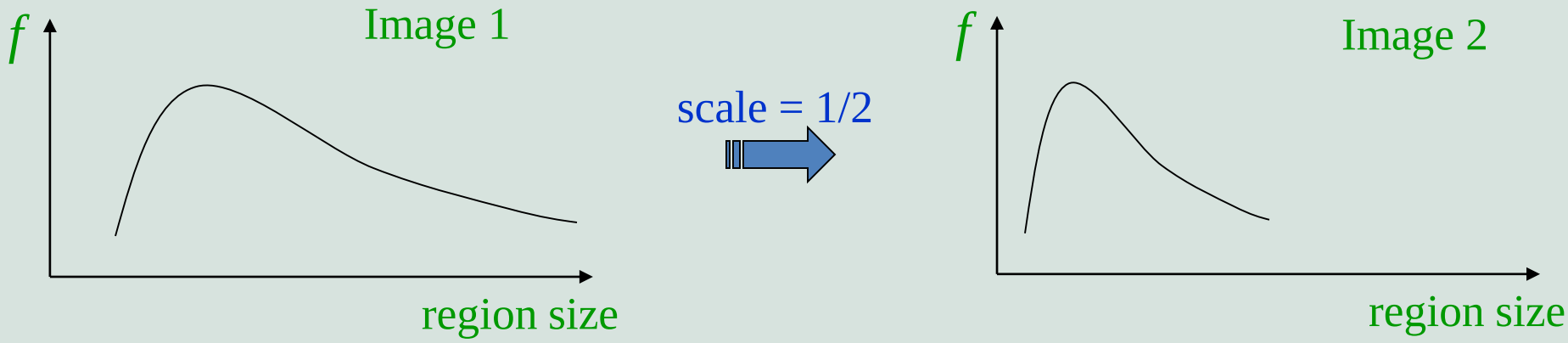
Invariance

- Extract patch from each image individually



Automatic scale selection

- Solution:
  - Design a function on the region, 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 (patch width)



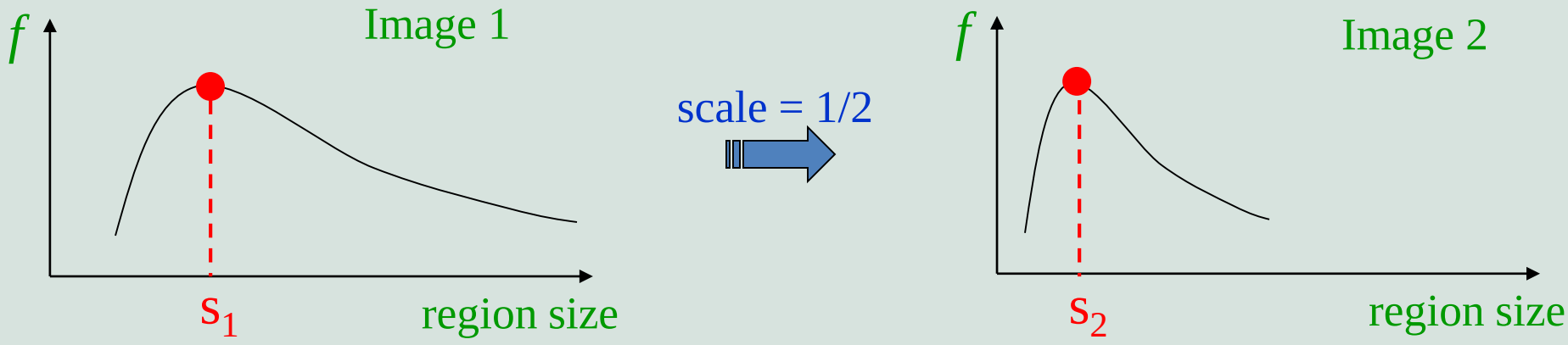
Automatic scale selection

- Common approach:

  Take a local maximum of this function

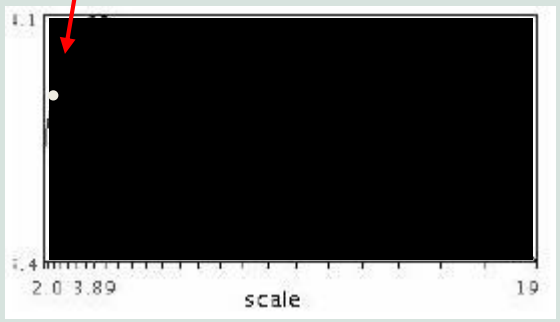
  Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

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

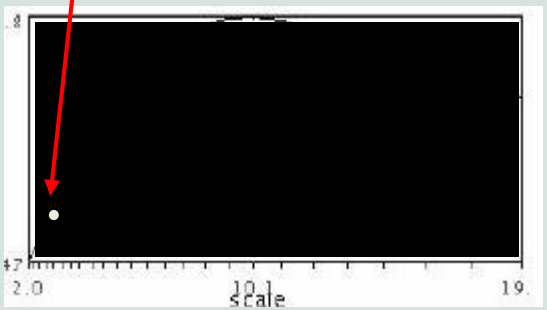


Automatic Scale Selection

- Function responses for increasing scale (scale signature)



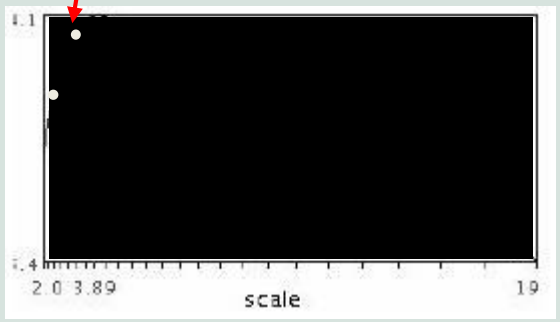
$f(I_{i_1 \ldots i_m}(x,\sigma))$



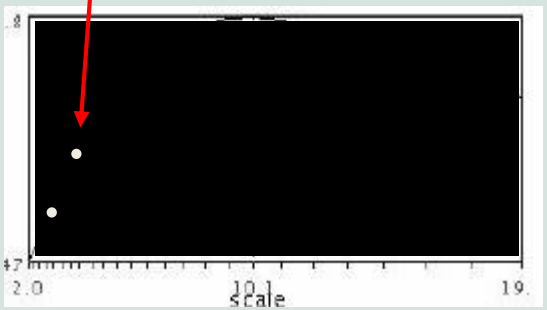
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



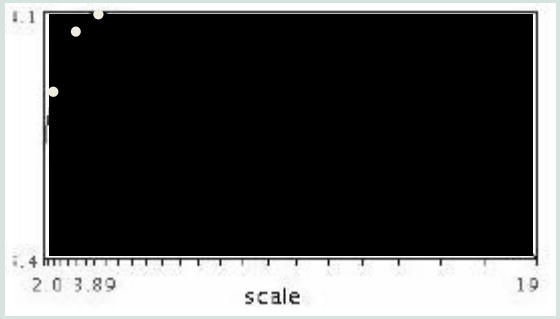
$f(I_{i_1 \ldots i_m}(x,\sigma))$



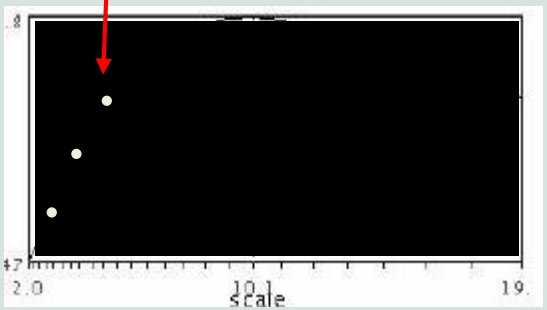
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



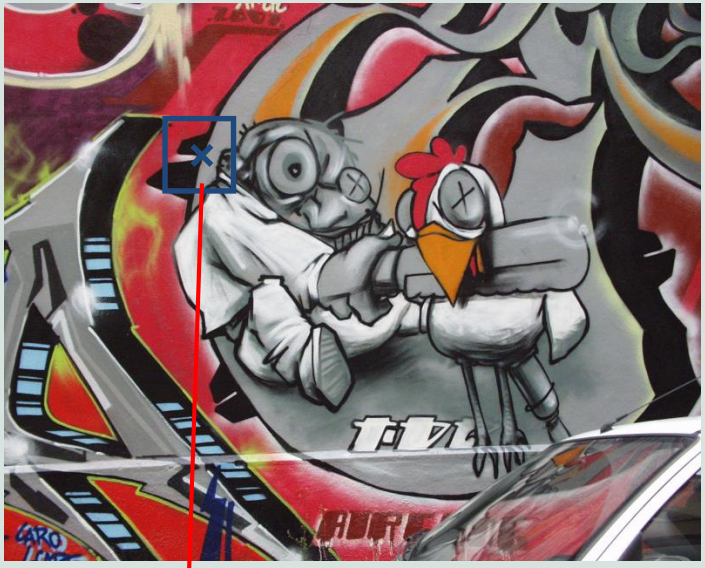
$f(I_{i_1 \ldots i_m}(x,\sigma))$

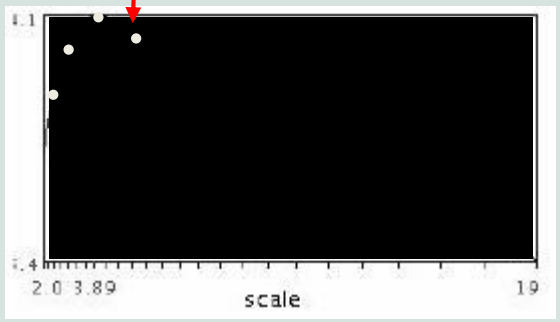


$f(I_{i_1 \ldots i_m}(x',\sigma))$

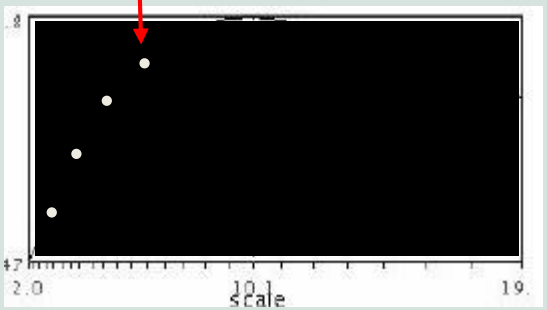
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





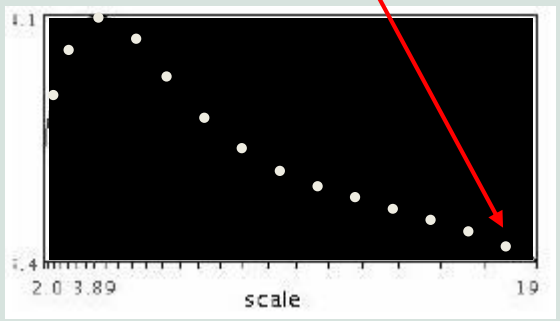
$f(I_{i_1 \ldots i_m}(x,\sigma))$



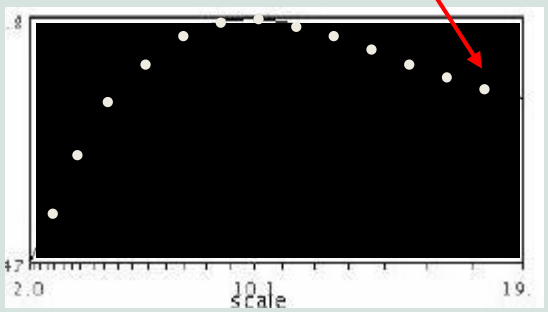
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



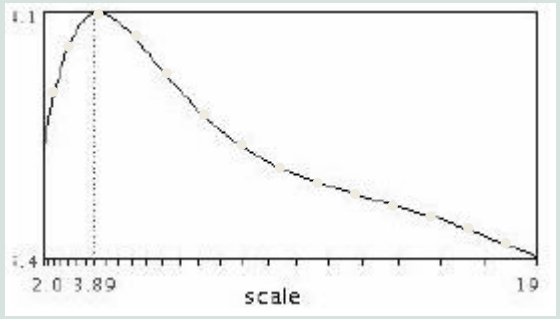
$f(I_{i_1 \ldots i_m}(x,\sigma))$



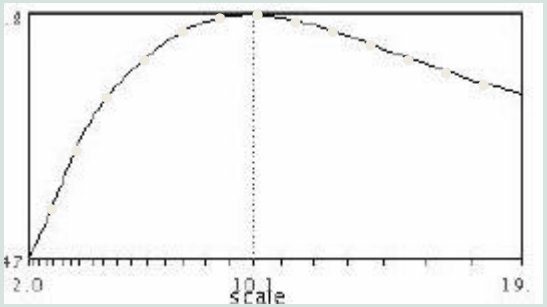
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



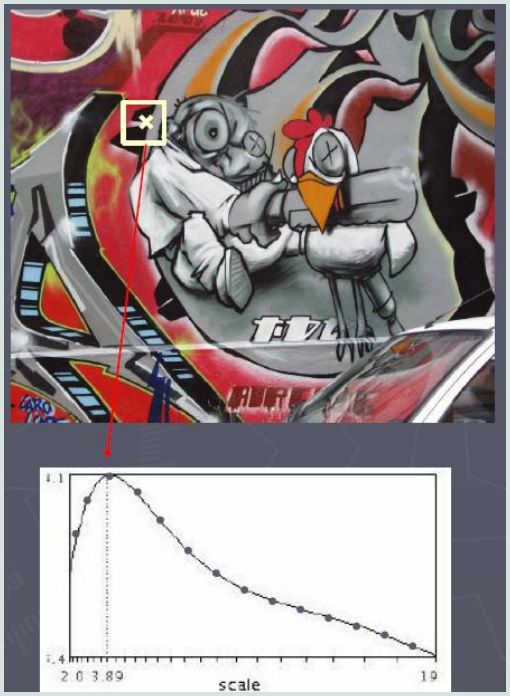
$f(I_{i_1 \ldots i_m}(x,\sigma))$

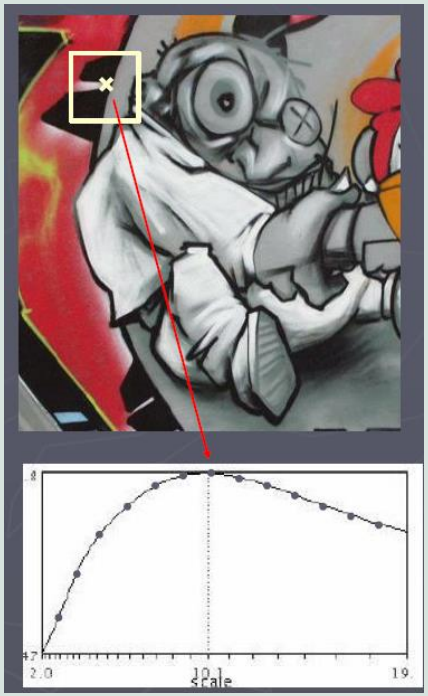


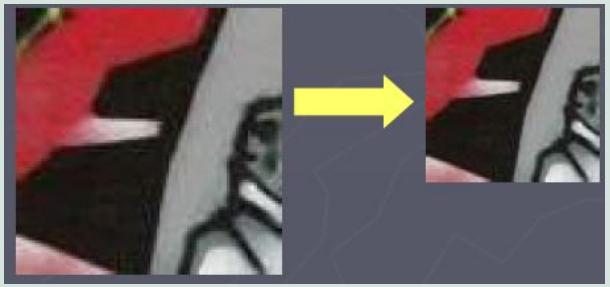
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Scale selection

- Use the scale determined by detector to compute descriptor in a normalized frame

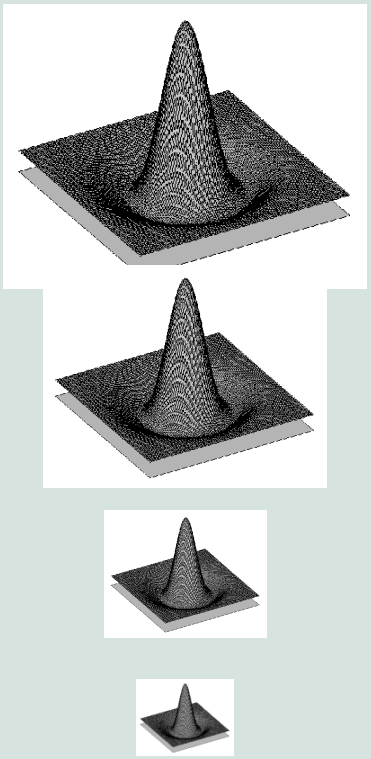


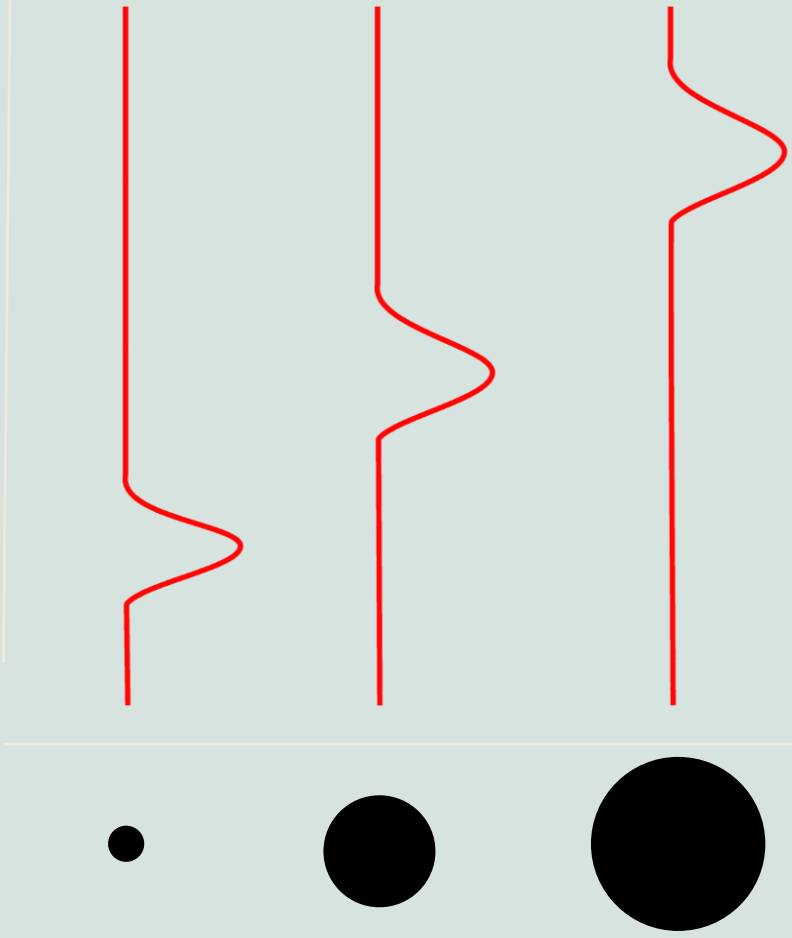




What Is A Useful Signature Function?

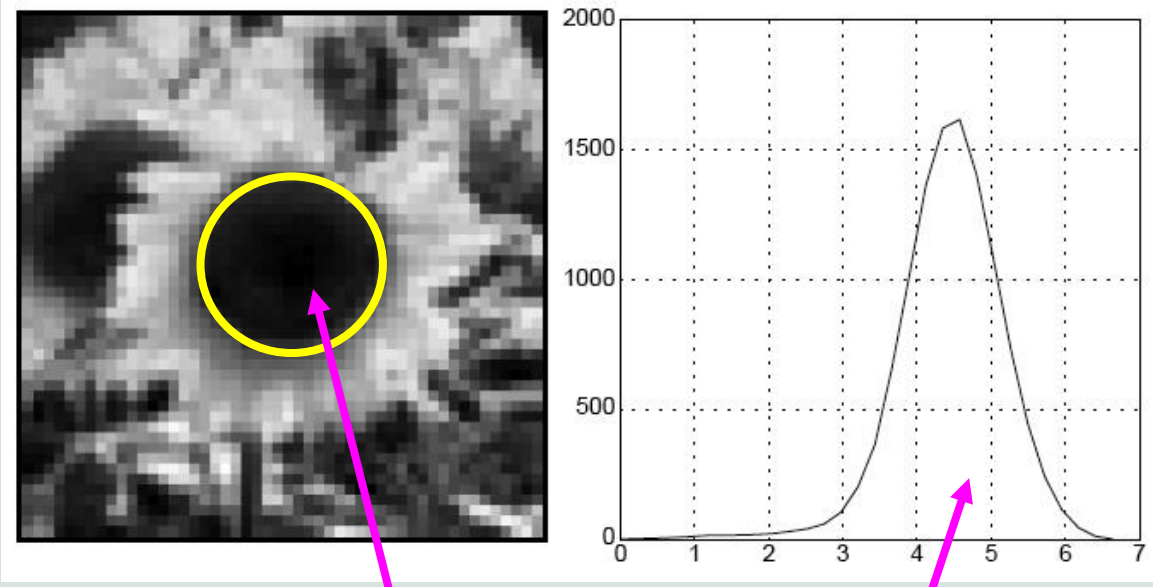
- Laplacian-of-Gaussian = “blob” detector





Characteristic scale

- We define the *characteristic scale* as the scale that produces peak of Laplacian response



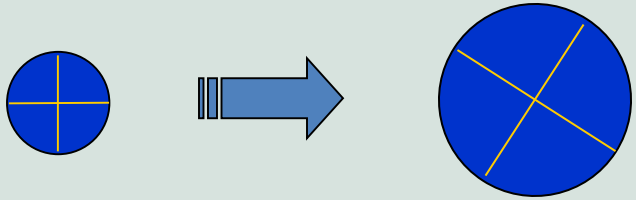
T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

Scale Invariant Detection: Summary

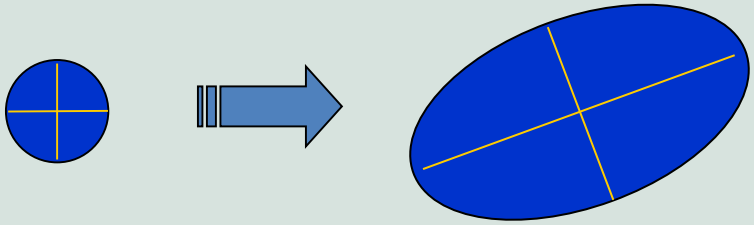
- **Given:** two images of the same scene with a large *scale difference* between them
- **Goal:** find *the same* interest points *independently* in each image
- **Solution:** search for *maxima* of suitable functions in *scale* and in *space* (over the image)

Affine Invariant Detection

- Above we considered:
  Similarity transform (rotation + uniform scale)



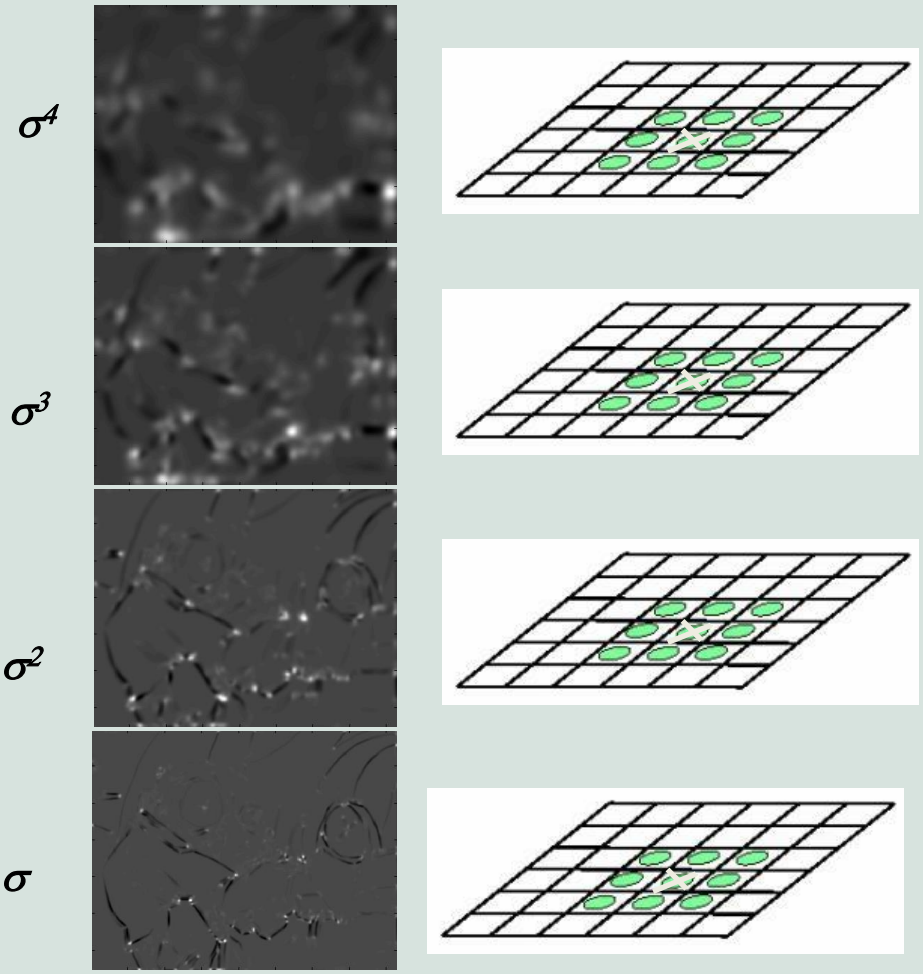
- Now we go on to:
  Affine transform (rotation + non-uniform scale)



- Affine invariance is a proxy for invariance to perspective transformations

Mikolajczyk: Harris Laplace

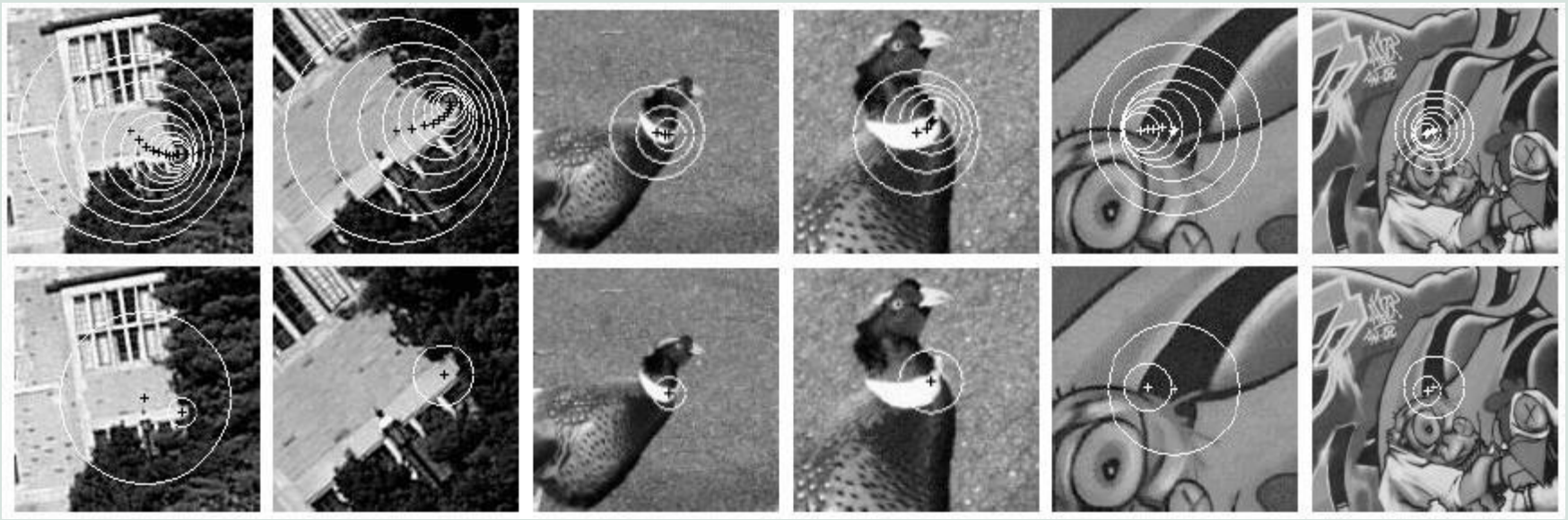
1. Initialization: Multiscale Harris corner detection



Mikolajczyk: Harris Laplace

- Initialization: Multiscale Harris corner detection
- Scale selection based on Laplacian

Harris points



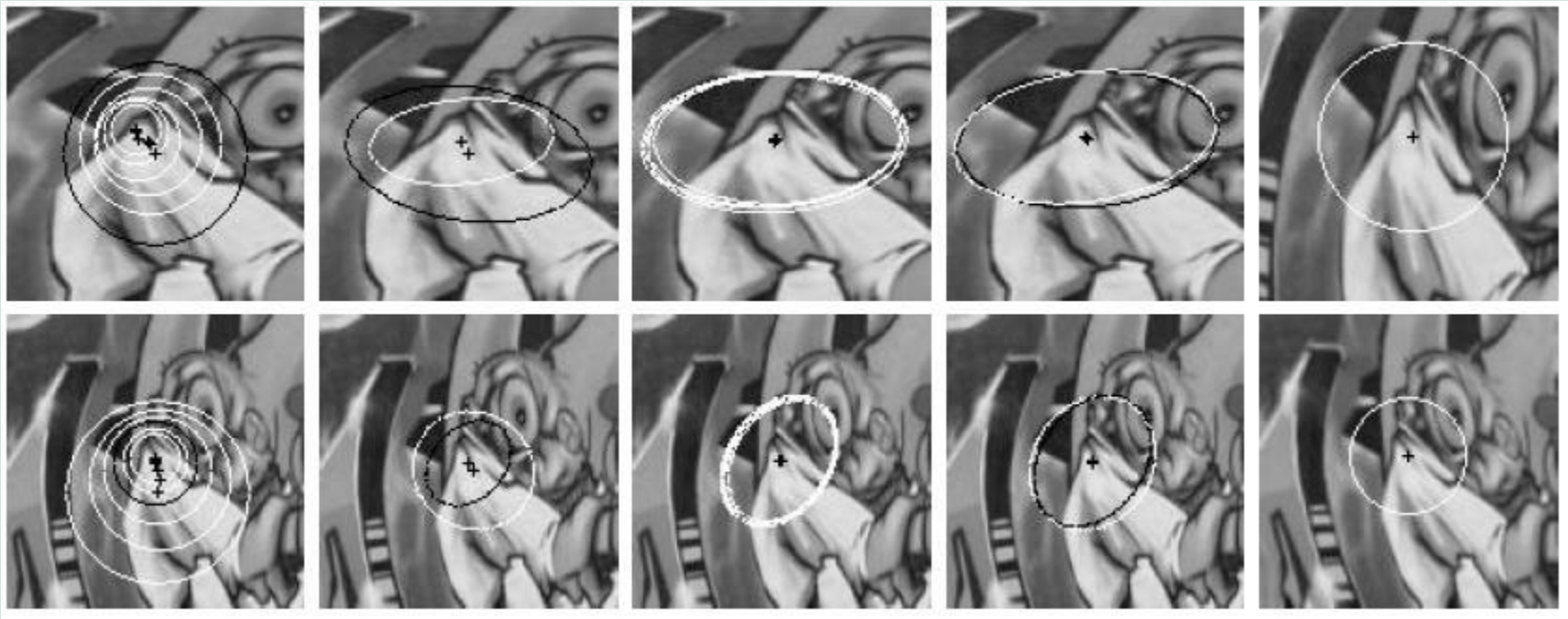
Harris-Laplace points

Mikolajczyk: Harris Affine

- Based on Harris Laplace
- Using normalization / deskewing

Mikolajczyk: Harris Affine

1. Detect the initial region with the Harris-Laplace detector
2. Estimate the affine shape with the second moment matrix
3. Normalize the affine region (ellipsis) to a circular one
4. Refine point location

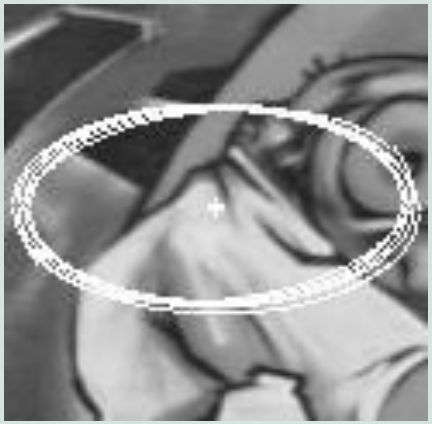


Mikolajczyk: Harris Affine Refinement

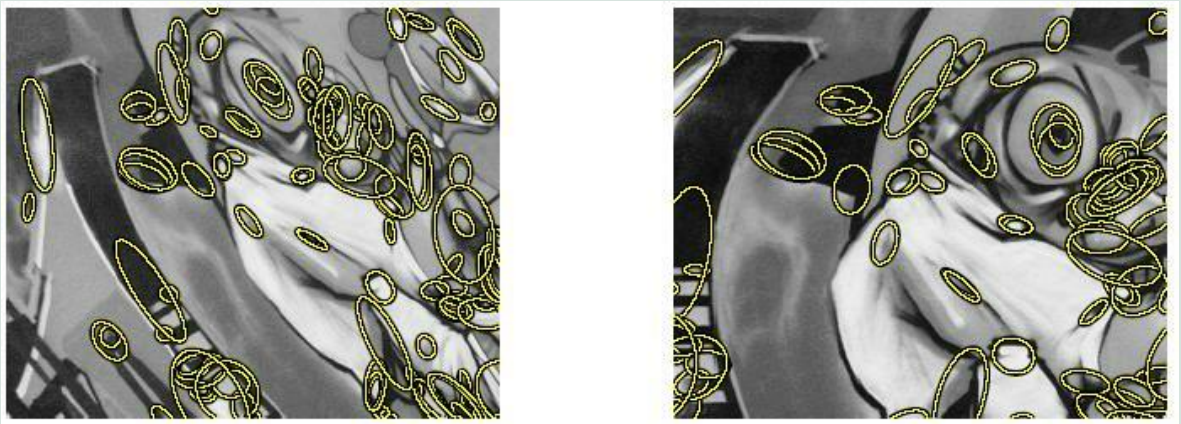
- Second order moment matrix
  - $L_x \rightarrow$ First order derivative in x

$$\sigma_D^2 G(\sigma_I) * \begin{bmatrix} L_x^2(x,\sigma_D) & L_x L_y(y,\sigma_D) \\ L_x L_y(y,\sigma_D) & L_y^2(y,\sigma_D) \end{bmatrix}$$

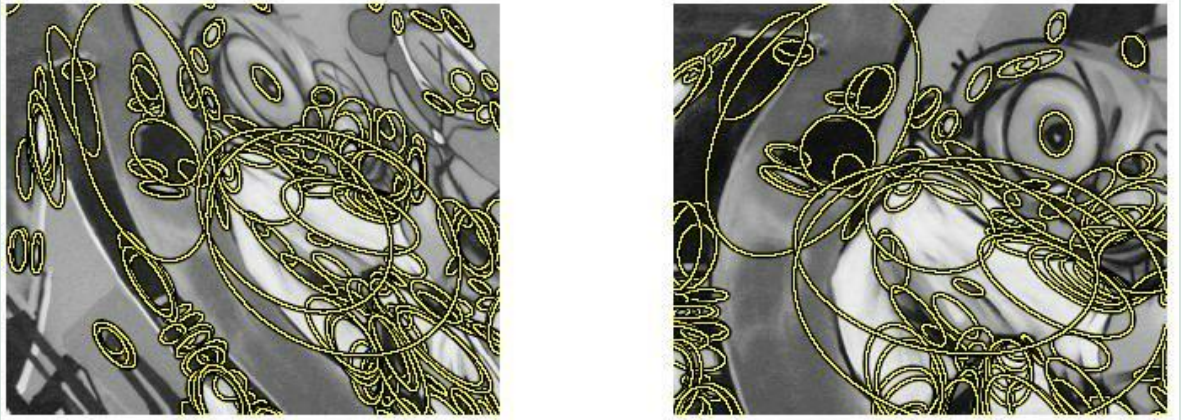
- Iterative algorithm
  1. Normalize window (deskewing)
  2. Select integration scale σ_I (max. of LoG)
  3. Select differentiation scale σ_D (max. $\lambda_{min}/\lambda_{max}$)
  4. Detect spatial localization (by Harris corner detector)
  5. Compute new affine transformation (μ)
  6. Go to step 2. until $1 - (\lambda_{min}/\lambda_{max}) < \in_C$



Harris Affine



(a) **Harris-Affine**



(b) **Hessian-Affine**

Affine Invariant Detection : Summary

- Under affine transformation, we do not know in advance shapes of the corresponding regions
- Ellipse given by geometric **covariance matrix** of a region robustly approximates this region
- For corresponding regions ellipses also correspond

Other Methods:

1. Search for extremum along rays *[Tuytelaars, Van Gool]*
2. Maximally Stable Extremal Regions *[Matas et.al.]*

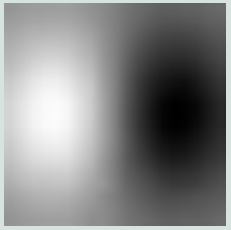
Region Detectors

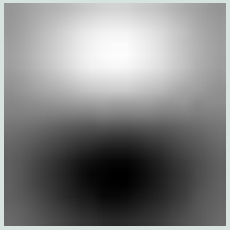
- Salient regions detector
  - computes entropy information in local regions
  - designates clusters of salient points as regions using k-nearest neighbor algorithm
- MSER (Maximally Stable Extremal Region)
  - finds stable connected component of some level sets of the image
  - a detector with higher repeatability rates but lower efficiency

Specific Object Detectors

- Template matching can be a solution
- Filters as **templates**:

  Note that filters look like the effects they are intended to find --- "matched filters"



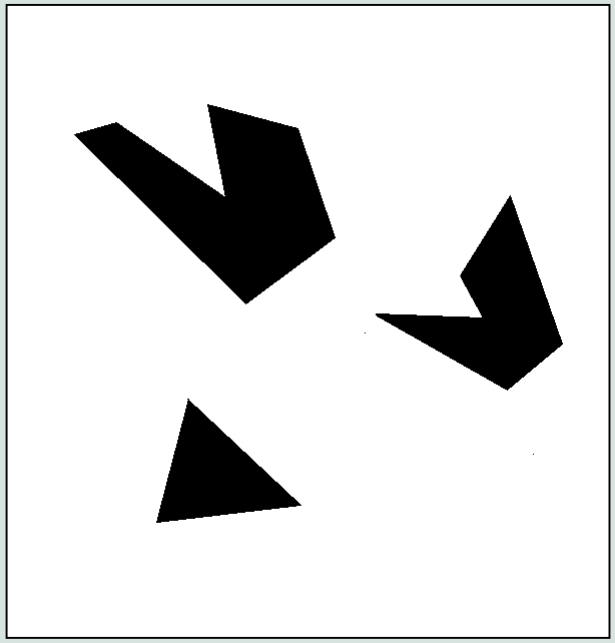


- Use normalized cross-correlation score to find a given pattern (template) in the image.

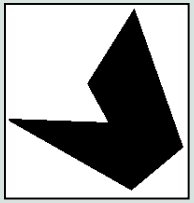
  – $E_{NCC}(\boldsymbol{u}) = \frac{\sum_i [I_0(\boldsymbol{x}_i) - \bar{I_0}][I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \bar{I_1}]}{\sqrt{\sum_i [[I_0(\boldsymbol{x}_i) - \bar{I_0}]]^2} \sqrt{\sum_i [[I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \bar{I_1}]]^2}}$

- Normalization needed to control for relative brightness

Template matching



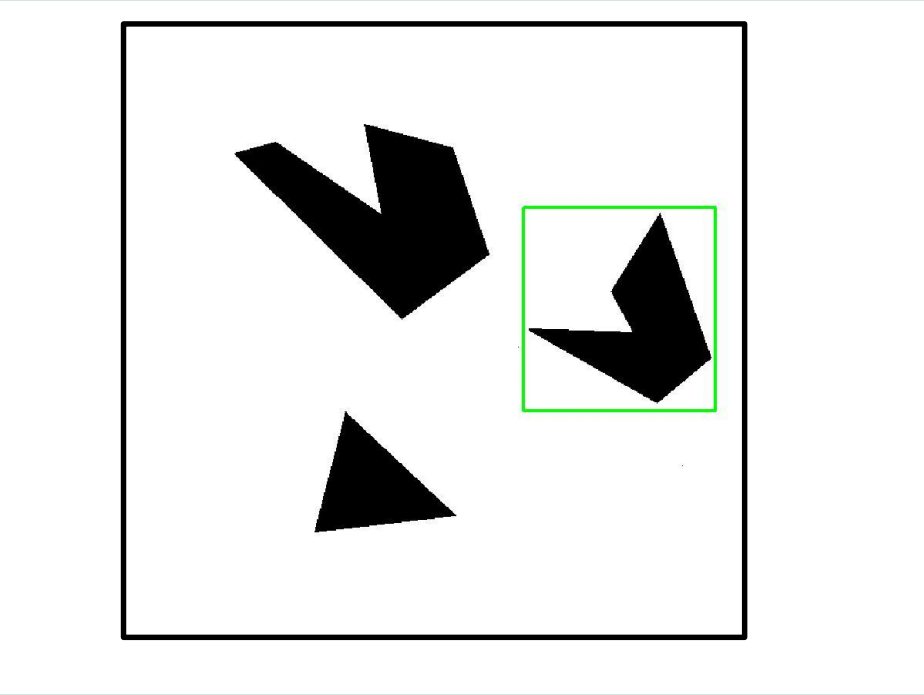
Scene



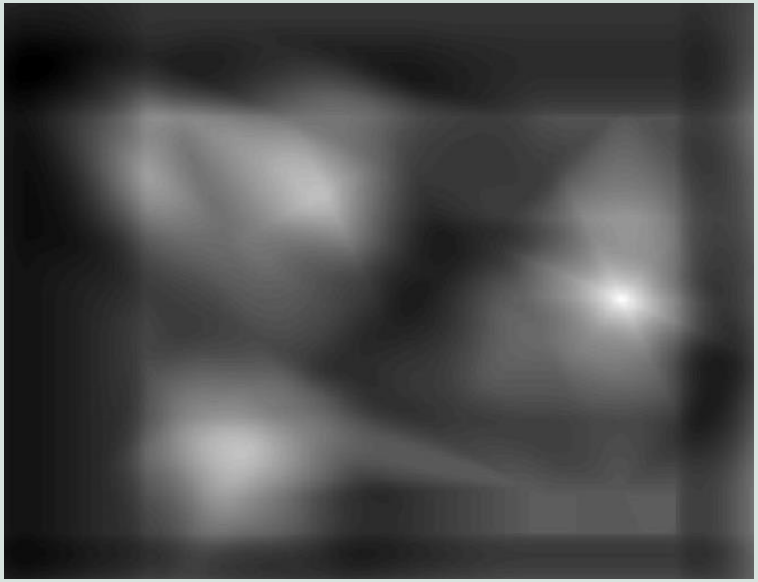
Template (mask)

A toy example

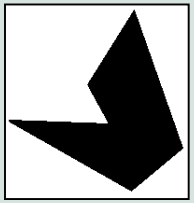
Template matching



Detected template



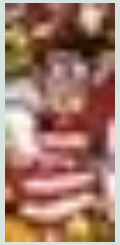
Correlation map



Template

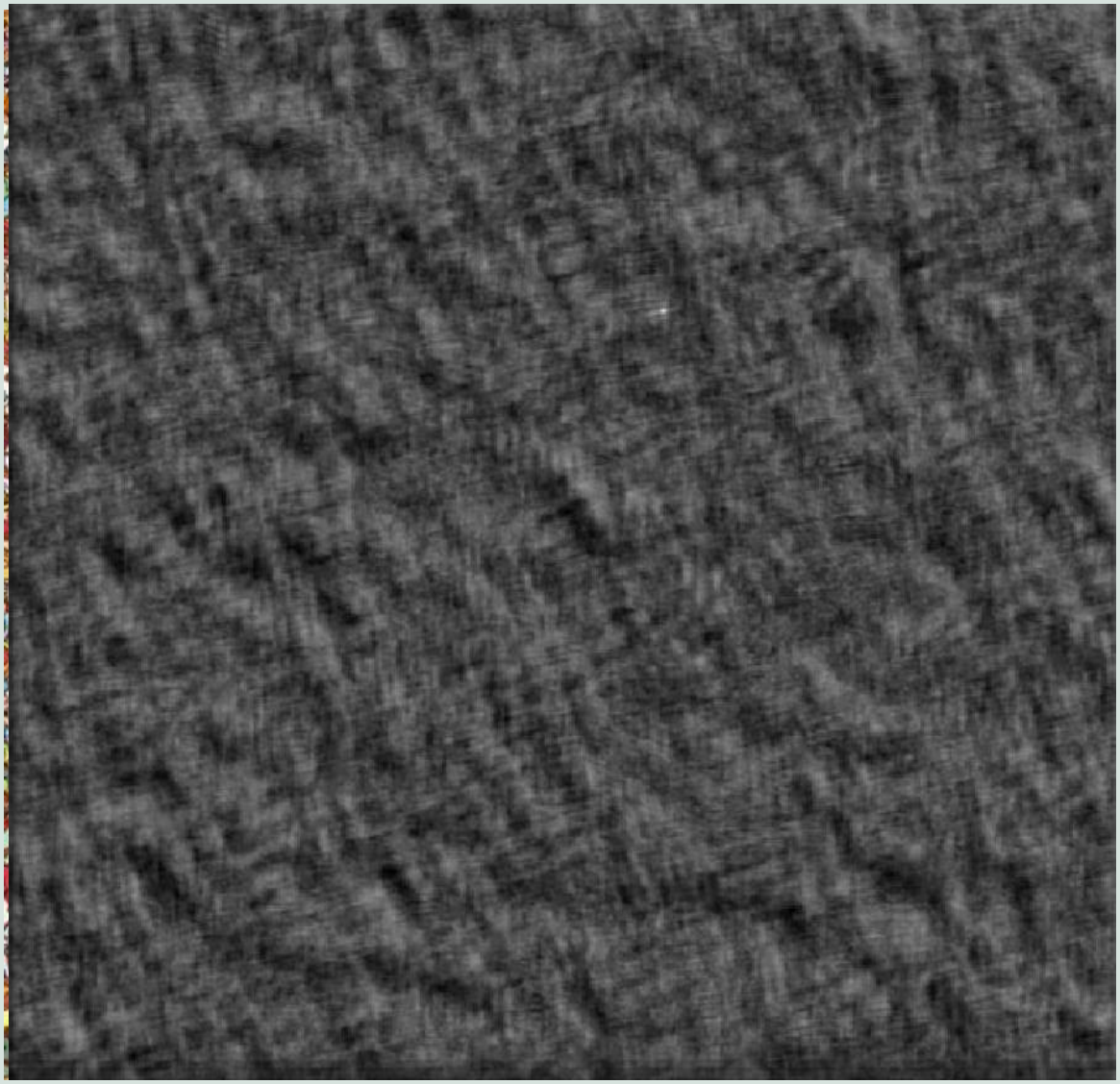
Where’s Waldo?

Scene

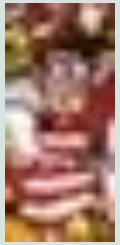


Template

Where’s Waldo?

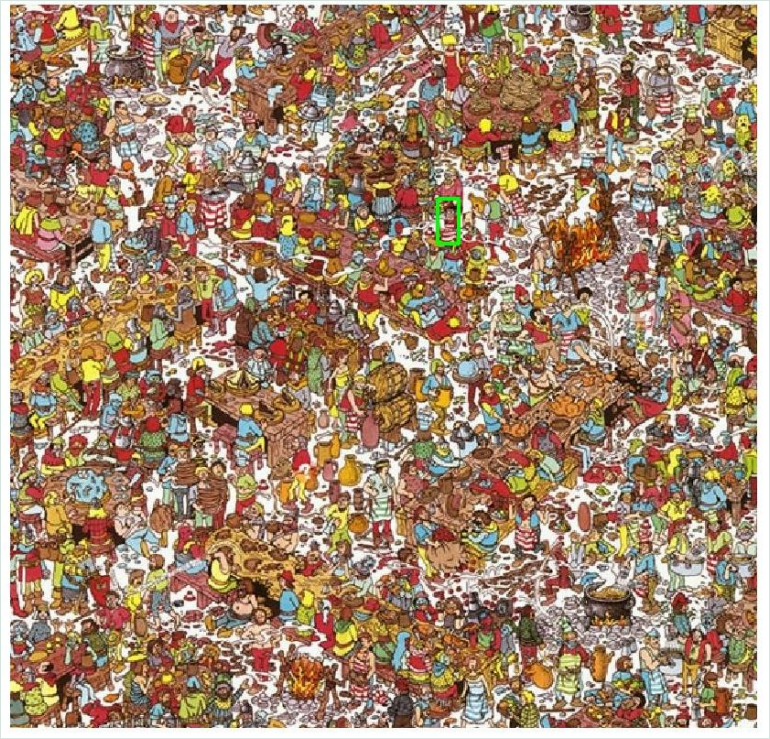


Scene

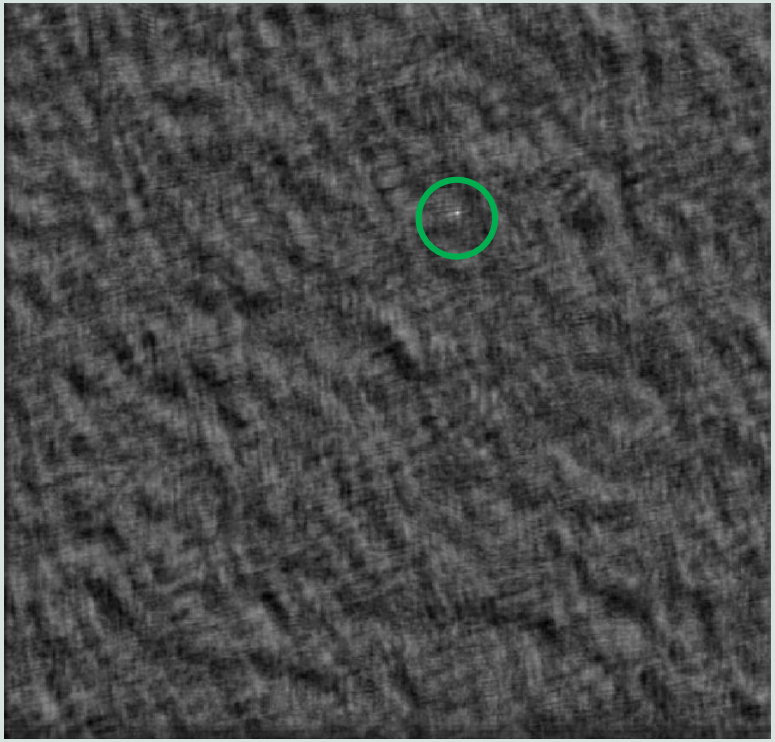


Template

Where’s Waldo?



Detected template



Correlation map

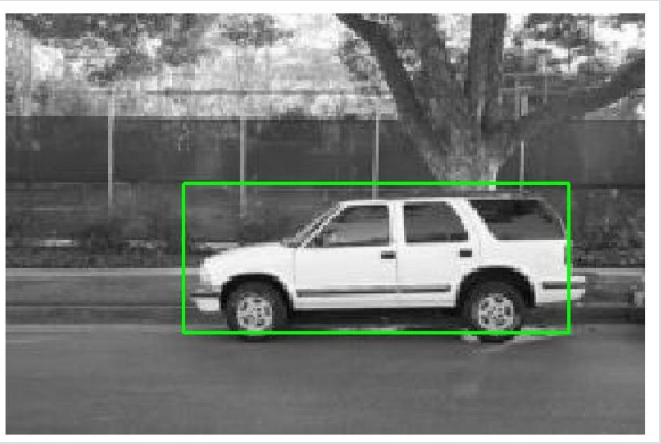
Template matching

Scene

Template

What if the template is not identical to some subimage in the scene?

Template matching



Detected template

Template

Match can be meaningful, if scale, orientation, and general appearance is right.