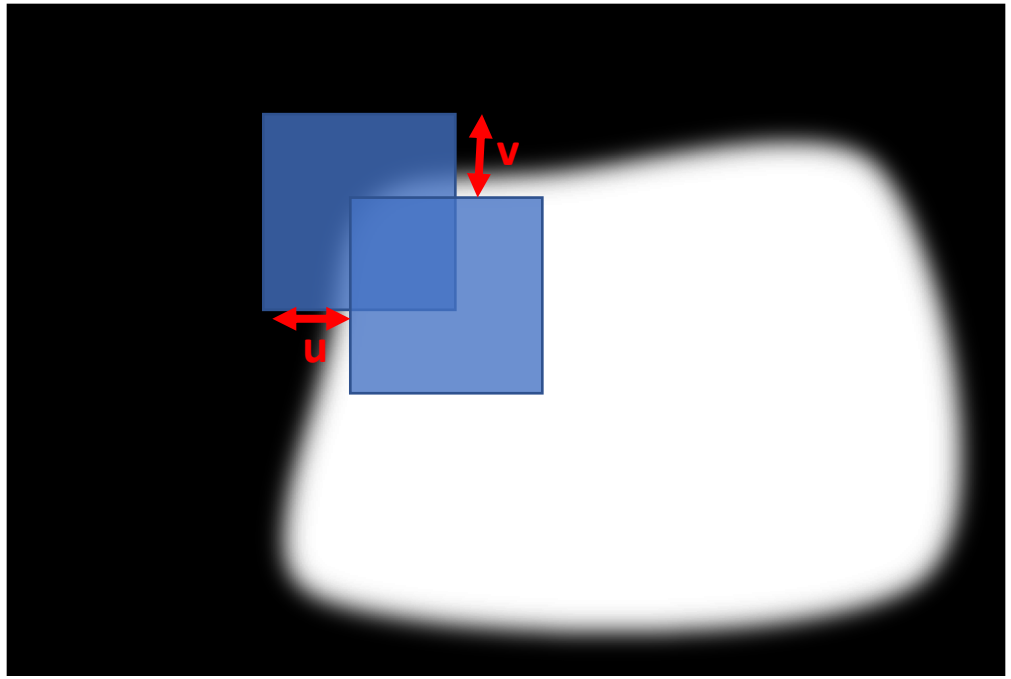
Invariance in Feature Detection

Harris corner detection - recap

- Key idea: *distinctiveness*
- We want patches that are unique compared to neighbors
- E(u,v) is appearance change of a window W when moved in the x-direction by u and y direction by v



$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} \sum_{x,y \in W} w(x,y) I_x^2 & \sum_{x,y \in W} w(x,y) I_x I_y \\ \sum_{x,y \in W} w(x,y) I_x I_y & \sum_{x,y \in W} w(x,y) I_y^2 \end{bmatrix}}_{\text{Second Moment matrix}} \begin{bmatrix} u \\ v \end{bmatrix}$$

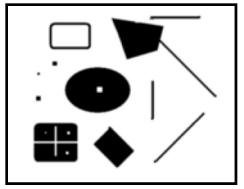
Computing the second moment matrix

- Consider first element of second moment matrix: $\sum_{x,y \in W} w(x,y) I_x^2$
- Need to compute this for every patch!
- Idea: First compute I_x^2 for every pixel
  - Might need to blur image with gaussian when computing derivatives. Why?
- Then convolve with w!

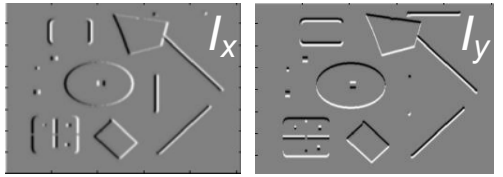
Harris Detector [Harris88]

- Second moment matrix

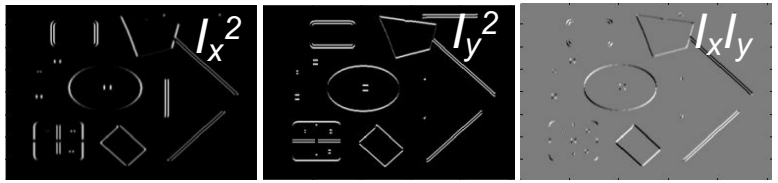
$$\mu(\sigma_I,\sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$



1. Image derivatives (optionally, blur first)



2. Square of derivatives

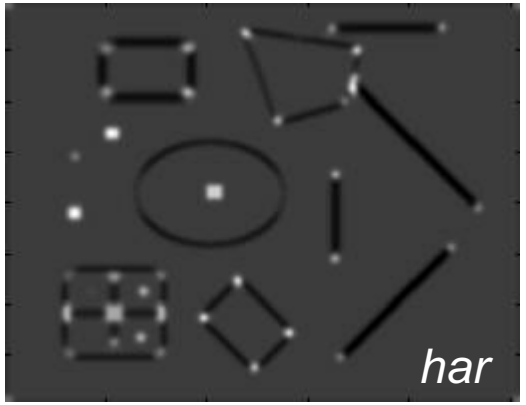


3. Gaussian filter $g(\sigma_I)$

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

4. Cornerness function – both eigenvalues are strong

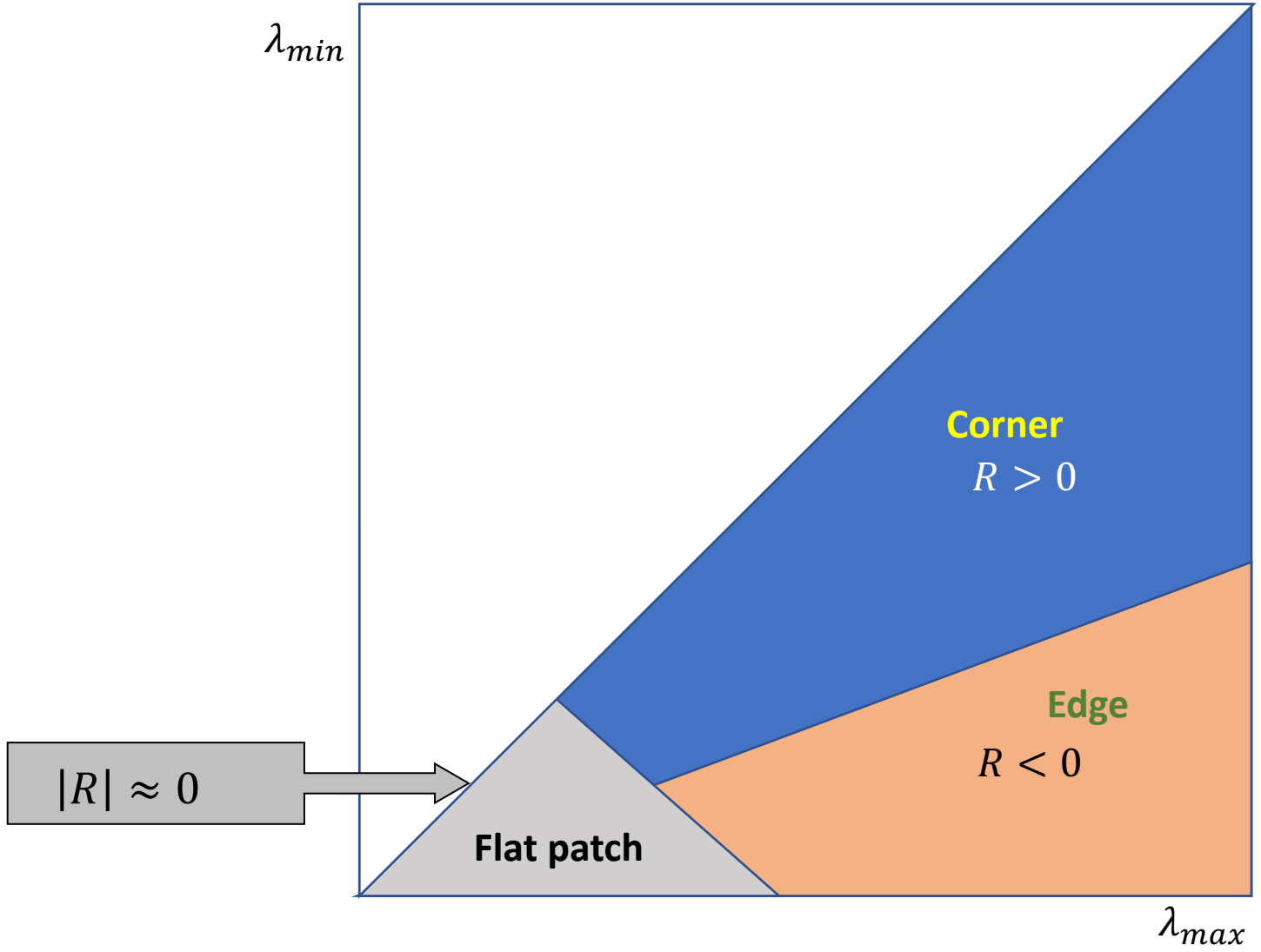
$$har = \det[\mu(\sigma_I,\sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I,\sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$



5. Non-maxima suppression

Corner response function

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



Final step: Non-maxima suppression

- Pick a pixel as corner if cornerness at patch centered on it > cornerness of neighboring pixels
- And if cornerness exceeds a threshold

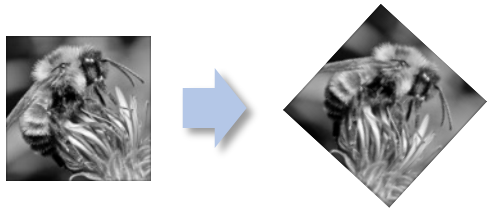
Invariance and equivariance

- Second criterion of corner detection: repeatability
- Suppose Image is transformed in some way
  - Image translation
  - Image rotation
  - Scaling of intensity
- How *should* the corners change?
  - Location?
  - Whether a corner is detected or not?
- How *does* the output of Harris corner detector change?

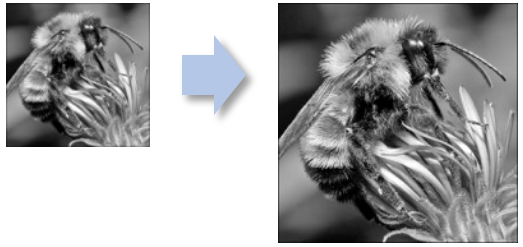
Image transformations

- Geometric

Rotation

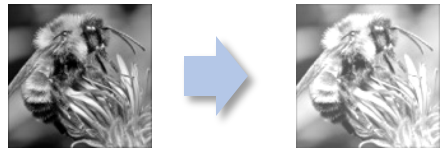


Scale



- Photometric

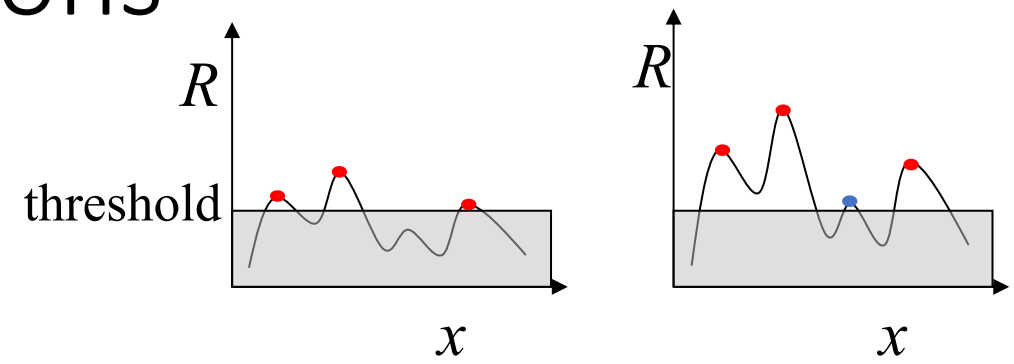
Intensity change



How should *photometric transformations* affect corner detection?

- Corner location:
  - Should *not* change!
- Probability of detection
  - Should *not* change

Harris corner detector invariance to photometric transformations



- Let us assume affine intensity change
- $I' = aI + b$
- What happens to image derivatives?
  - $I'_x = aI_x$
  - $I'_y = aI_y$
- What happens to the second moment matrix?
  - $M' = a^2 M$
- What happens to eigenvalues & cornerness response function?
  - $\lambda'_i = a^2\lambda_i, R' = a^2 R$
- Does this change probability of detecting a corner?
  - Yes, because of thresholding (last step)!
- What happens if no scaling (i.e., a = 1)?

Harris corner detector invariance to photometric transformations

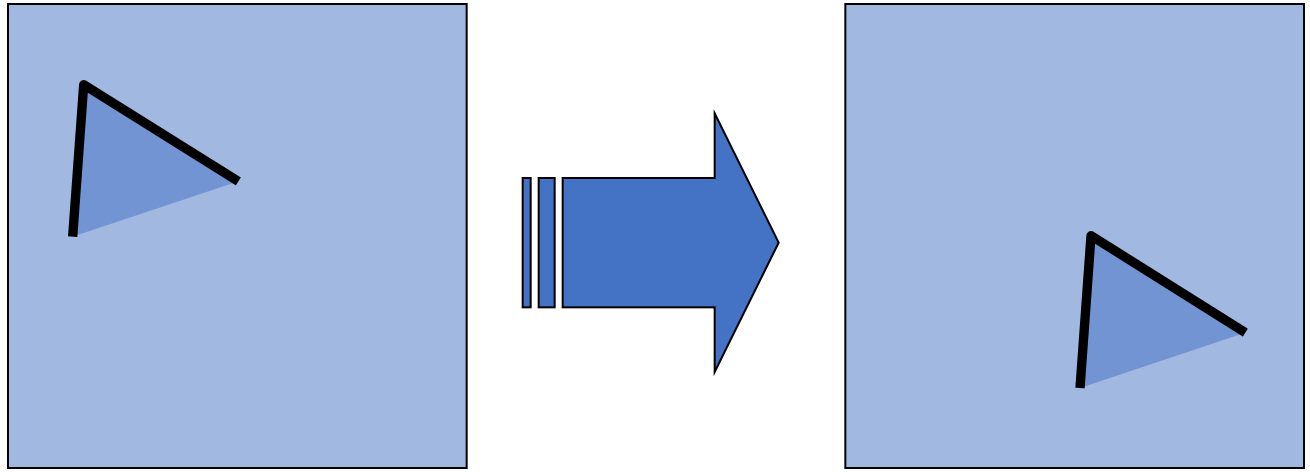
- Harris corner detector (both locations and probability of corner detection) is invariant to *additive* changes in intensity
  - changes in overall “Brightness”
- It is *not invariant* to scaling of intensity
  - Changes in “Contrast”

How should *geometric transformations* affect corner detection?

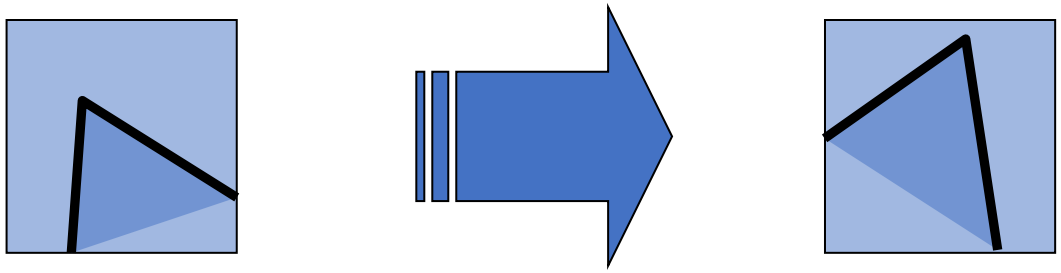
- Corner *location* should move as the underlying pixel moves
  - Thus corner location should be *equivariant* to geometric transformations
- Corner detection probability should be unaffected by geometric transformations
  - Thus *invariant* to geometric transformations

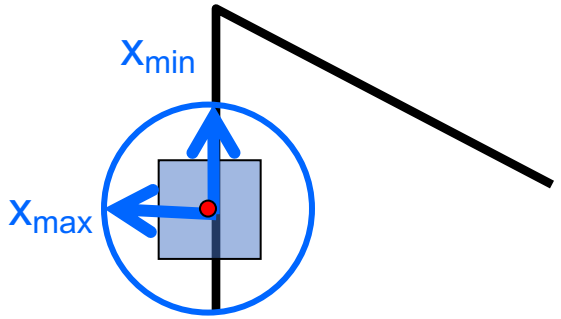
Harris corner detection and translation

- What happens if image is translated?
- Derivatives, second moment matrix obtained through convolution, which is *translation equivariant*
- Eigenvalues based only on derivatives so cornerness is *invariant*
- Thus Harris corner detection location is *equivariant to translation,* and response is *invariant to translation*



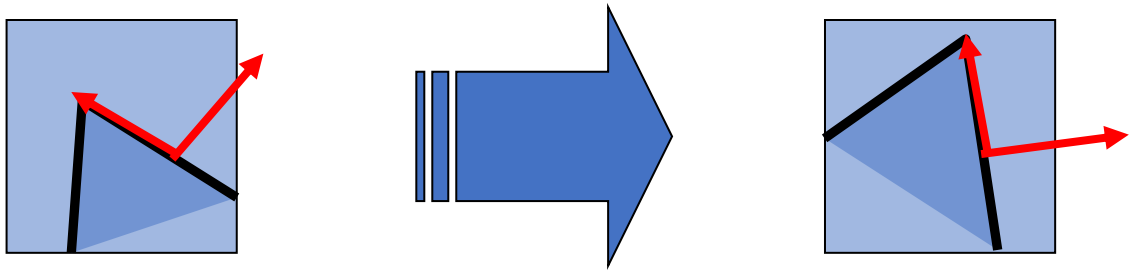
What about rotation?





- Now every patch is rotated, so problem?
- Recall properties of second moment matrix
- Eigenvalues and eigenvectors of M
  - Define shift directions with the smallest and largest change in error
  - x_{max}= direction of largest increase in E (across the edge)
  - λ_{max} = amount of increase in direction x_{max}
  - x_{min}= direction of smallest increase in E (along the edge)
  - λ_{min} = amount of increase in direction x_{min}

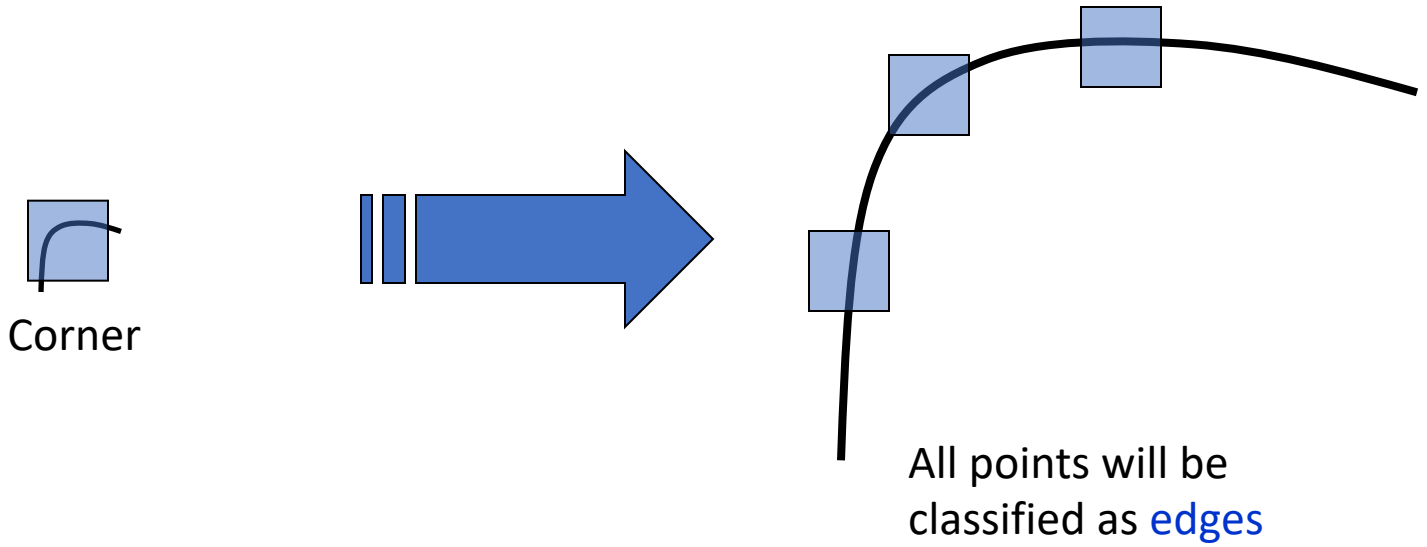
What about rotation?



- What happens to eigenvalues and eigenvectors when a patch rotates?
- Eigenvectors represent the *direction* of maximum / minimum change in appearance, so they rotate *with the patch*
- Eigenvalues represent the corresponding *magnitude* of maximum/minimum change so they *stay constant*
- Corner response is only dependent on the eigenvalues so is *invariant to* rotation
- Corner location is as before equivariant to rotation.

What about scaling?

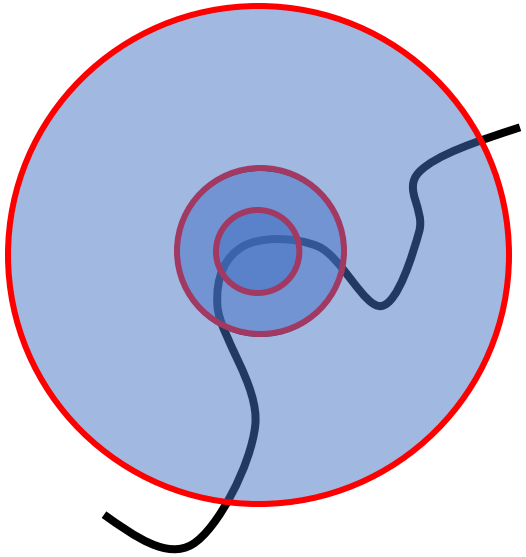
- What was one patch earlier is now many



Not invariant to scaling

Scale invariant detection

Suppose you're looking for corners

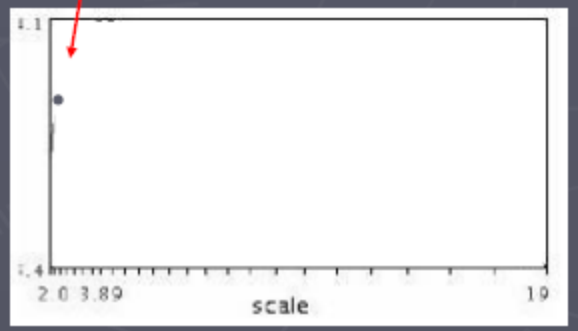


Key idea: find scale that gives local maximum of *cornerness*

- in both position and scale
- One definition of *cornerness*: the Harris operator

Automatic scale selection

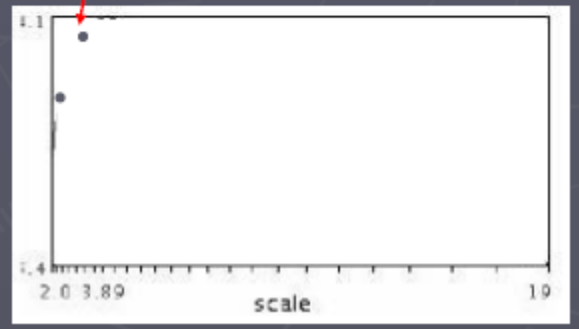
Lindeberg et al., 1996



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

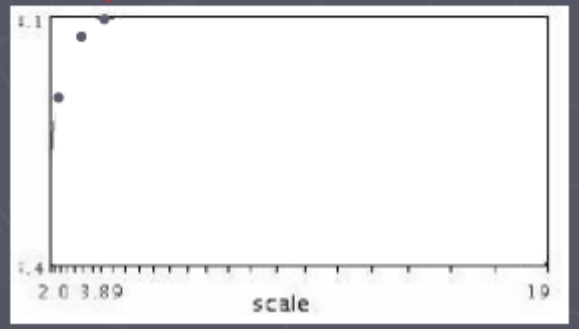
Slide from Tinne Tuytelaars

Automatic scale selection



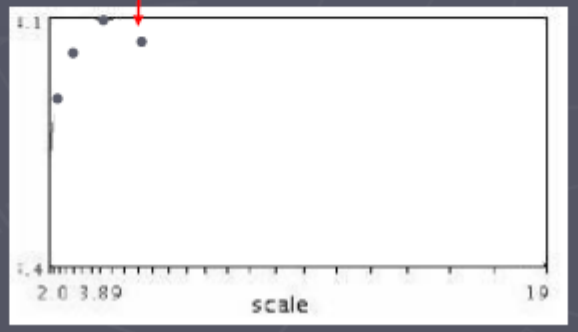
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection



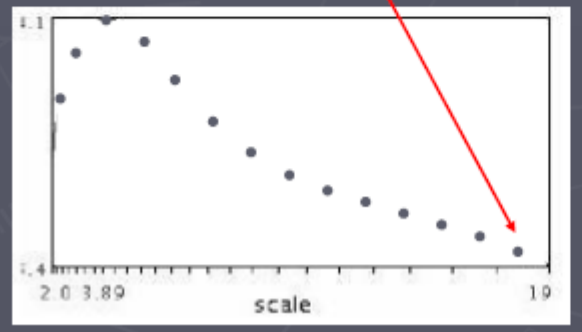
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection



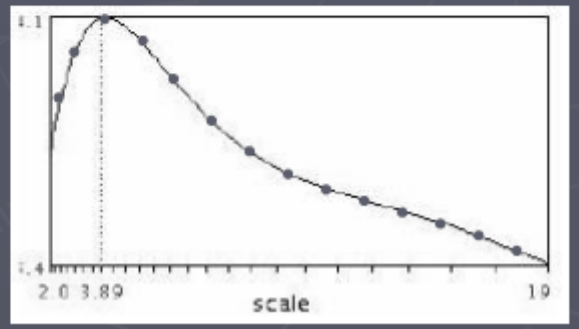
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection



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

Automatic scale selection



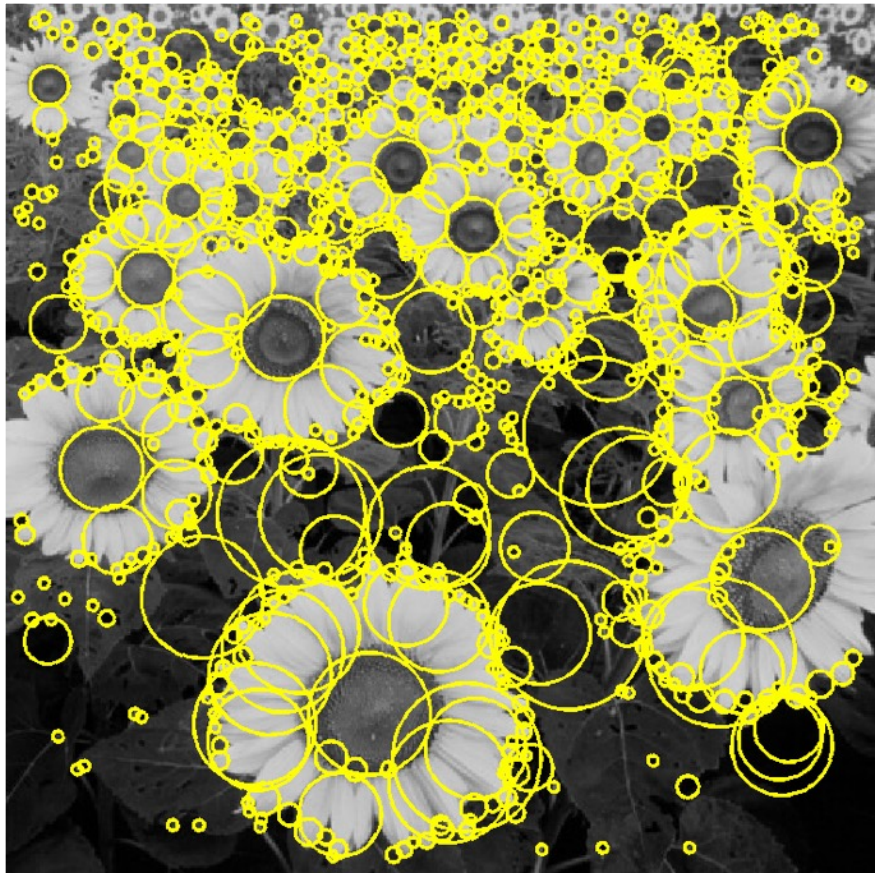
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Implementation

- Instead of computing *f* for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid

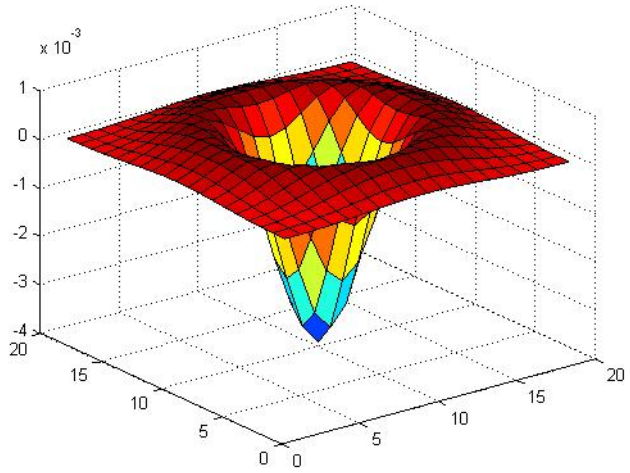
Another kind of feature: blob

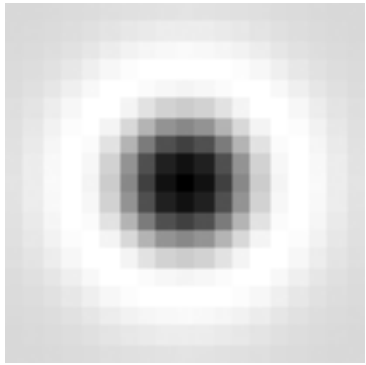
- “Peak” or “Valley” in intensity



Detecting peaks and valleys

- The *Laplacian of Gaussian* (*LoG*)



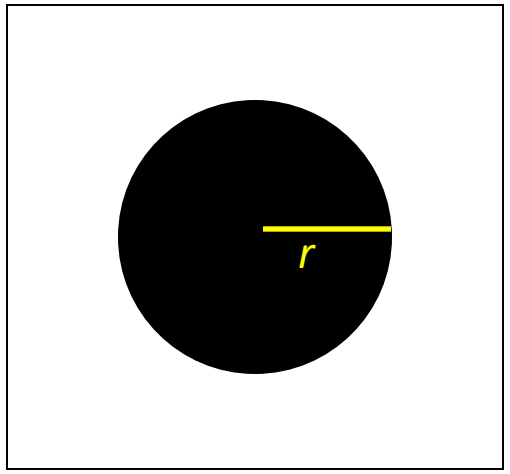


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

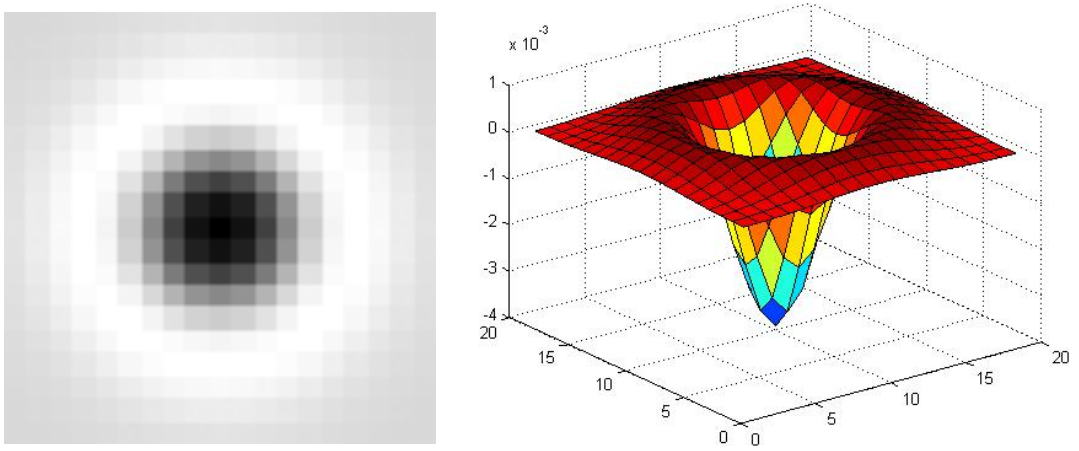
(very similar to a Difference of Gaussians (DoG) – i.e. a Gaussian minus a slightly smaller Gaussian)

Scale selection

- At what scale does the Laplacian achieve a maximum response for a binary circle of radius r?



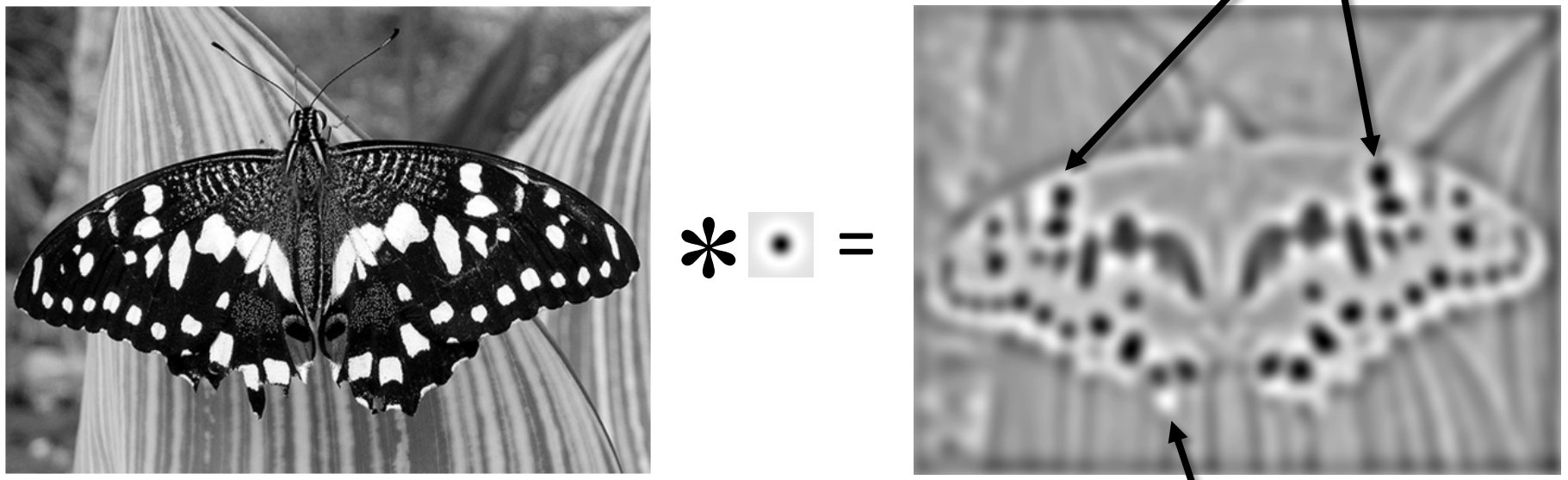
image



Laplacian

Laplacian of Gaussian

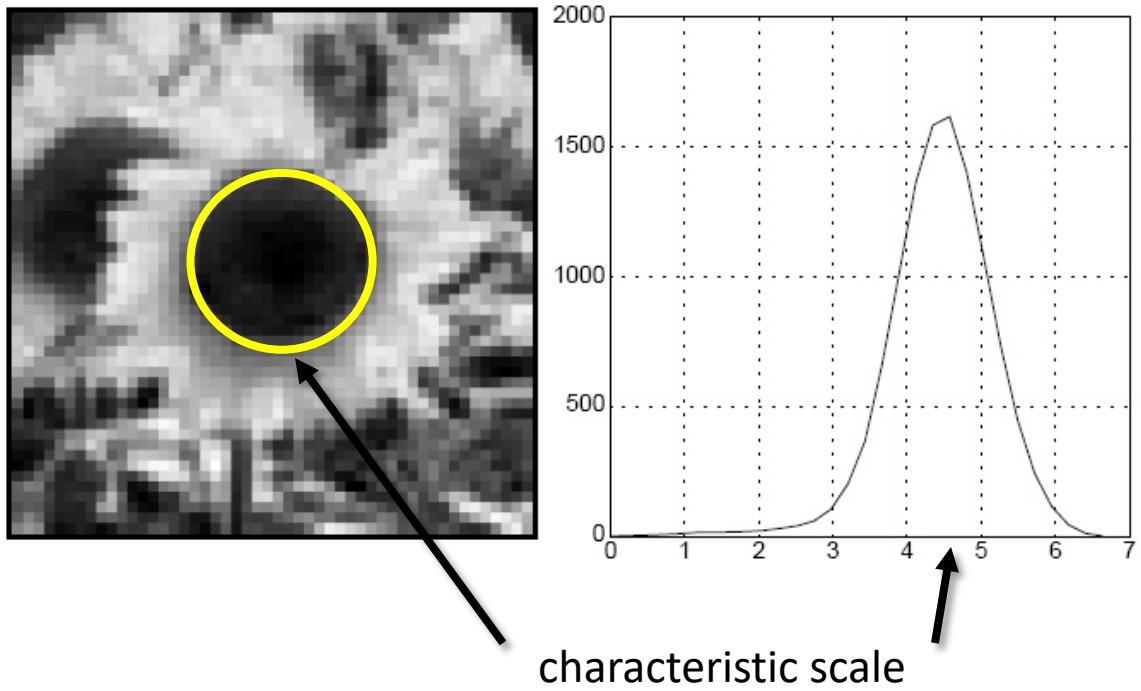
- “Blob” detector



- Find maxima *and minima* of LoG operator in space and scale

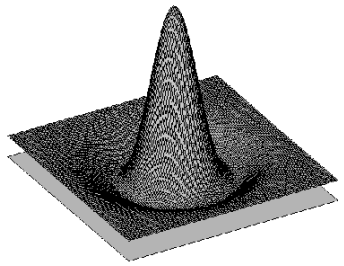
Characteristic scale

- The scale that produces peak of Laplacian response

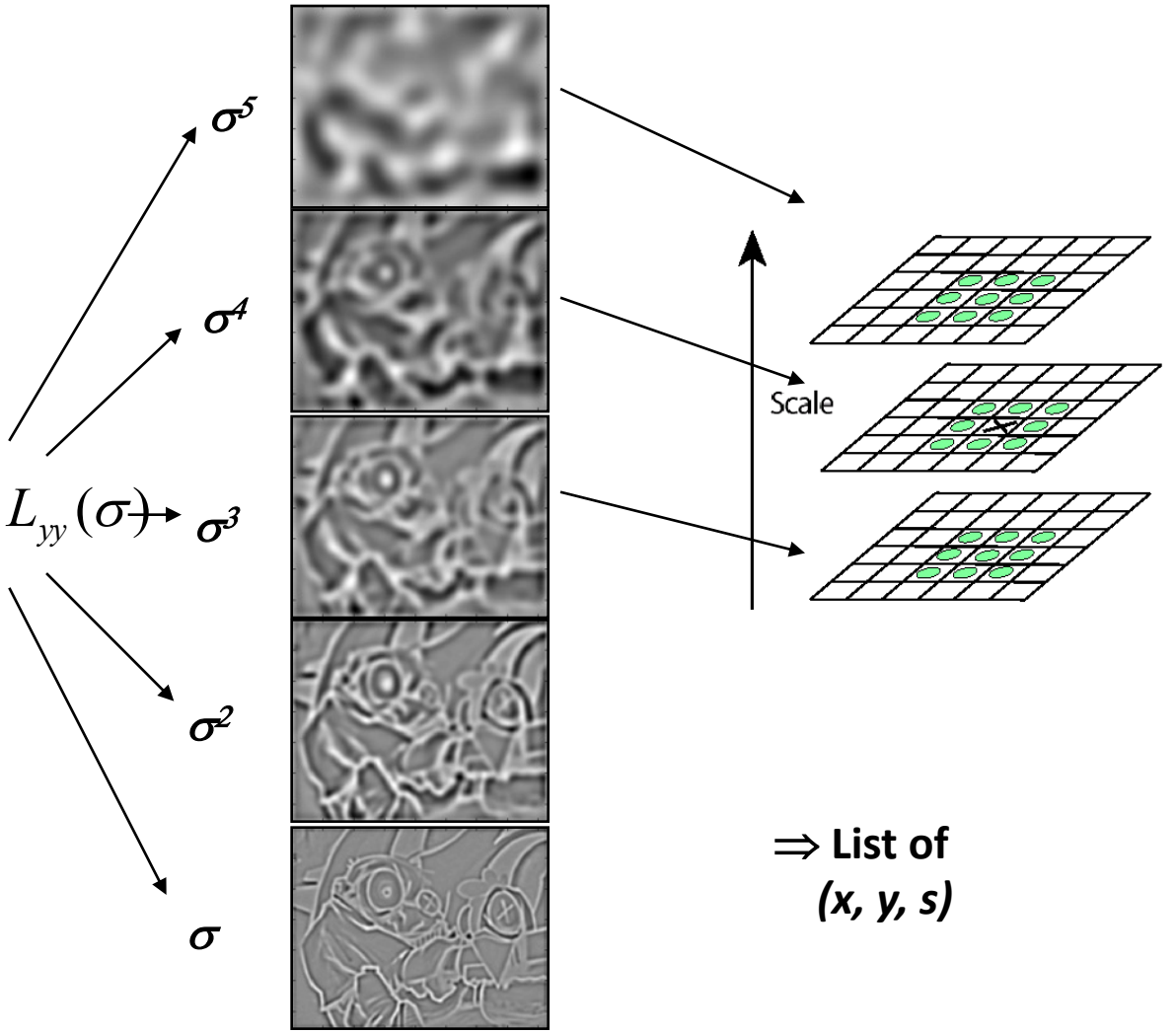


T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

Find local maxima in position-scale space



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



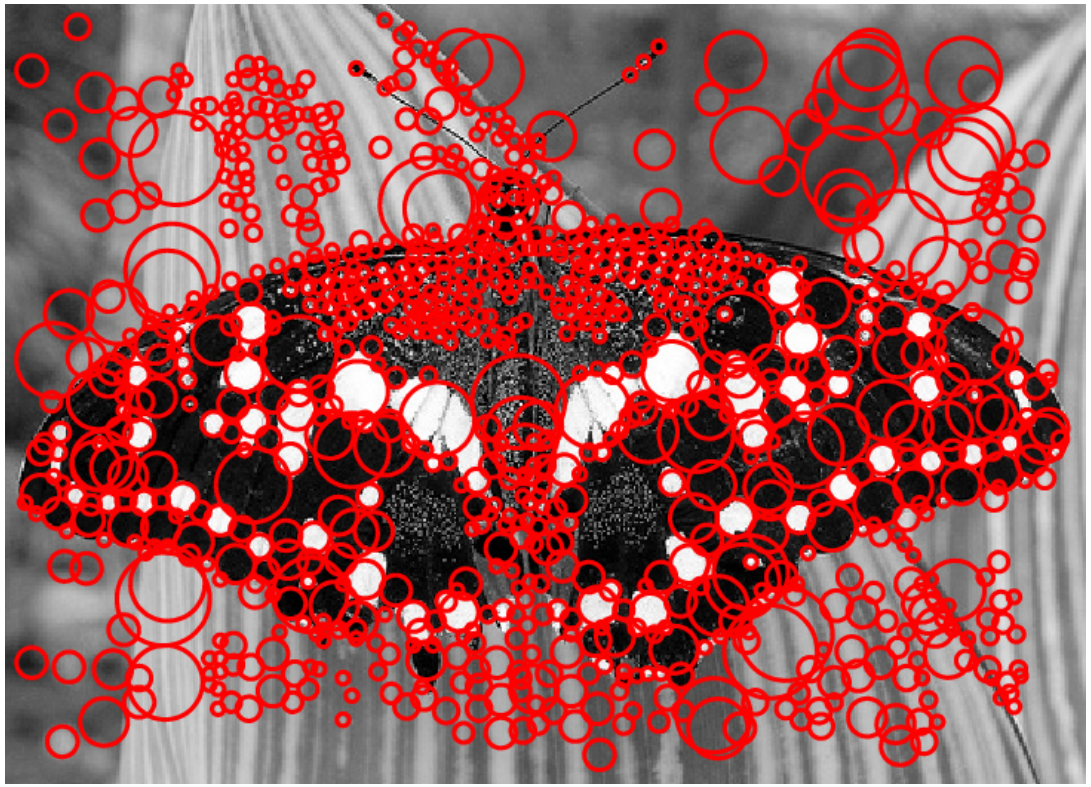
⇒ **List of**
(x, y, s)

Scale-space blob detector: Example

Scale-space blob detector: Example

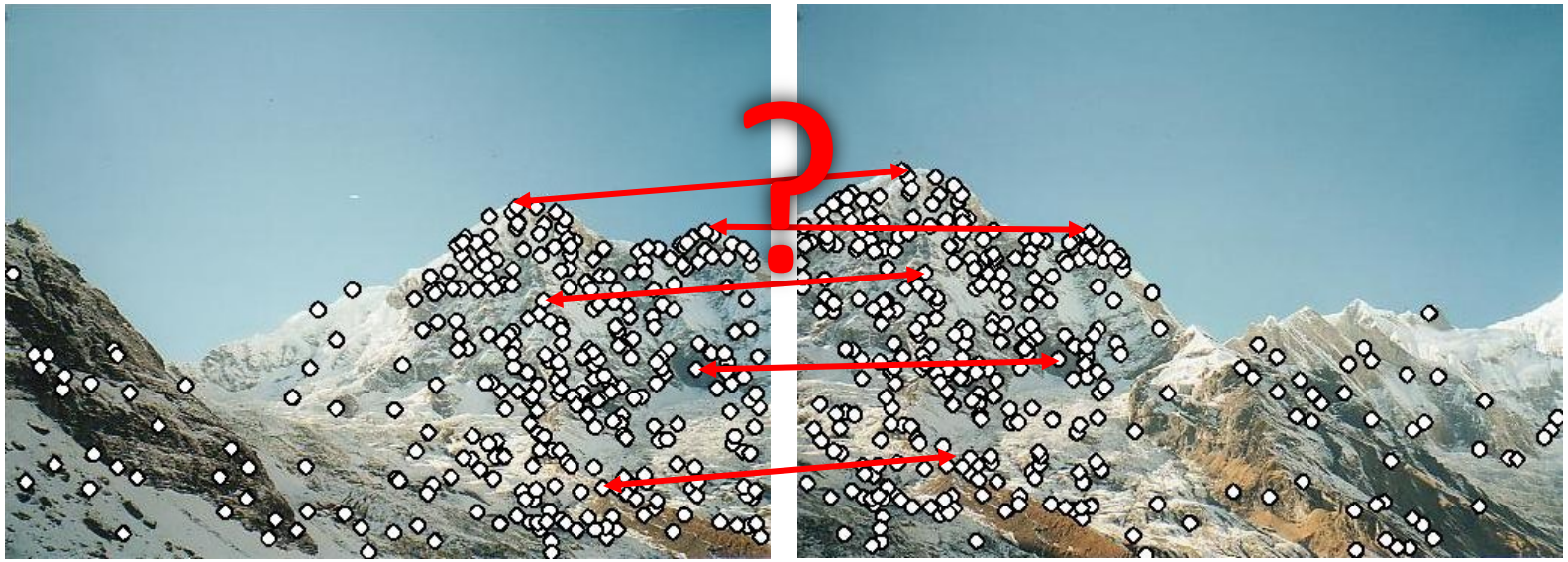
sigma = 11.9912

Scale-space blob detector: Example



Matching feature points

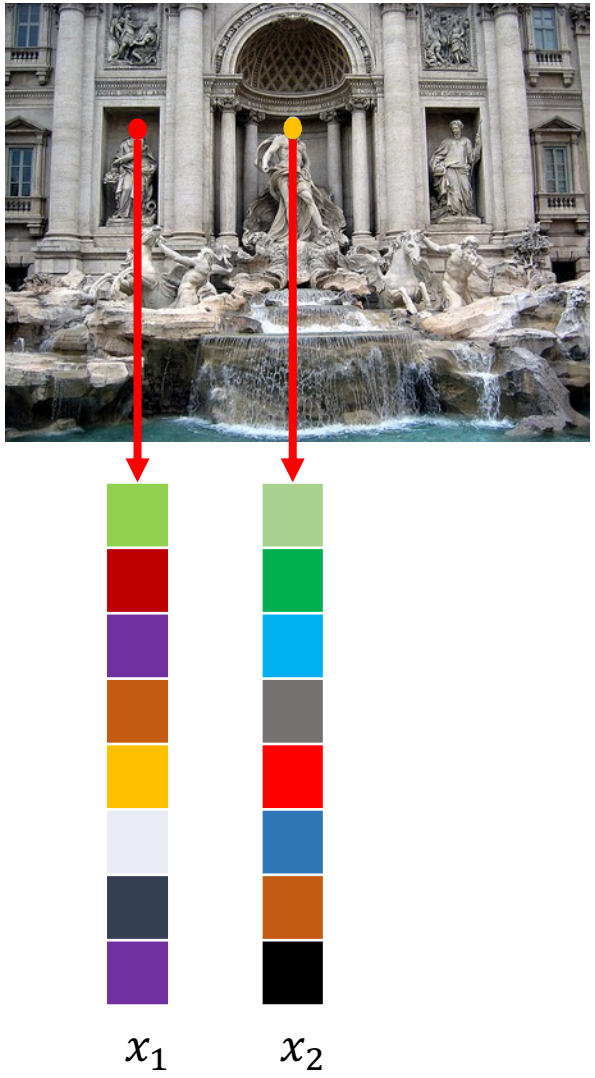
We know how to detect good points
Next question: **How to match them?**

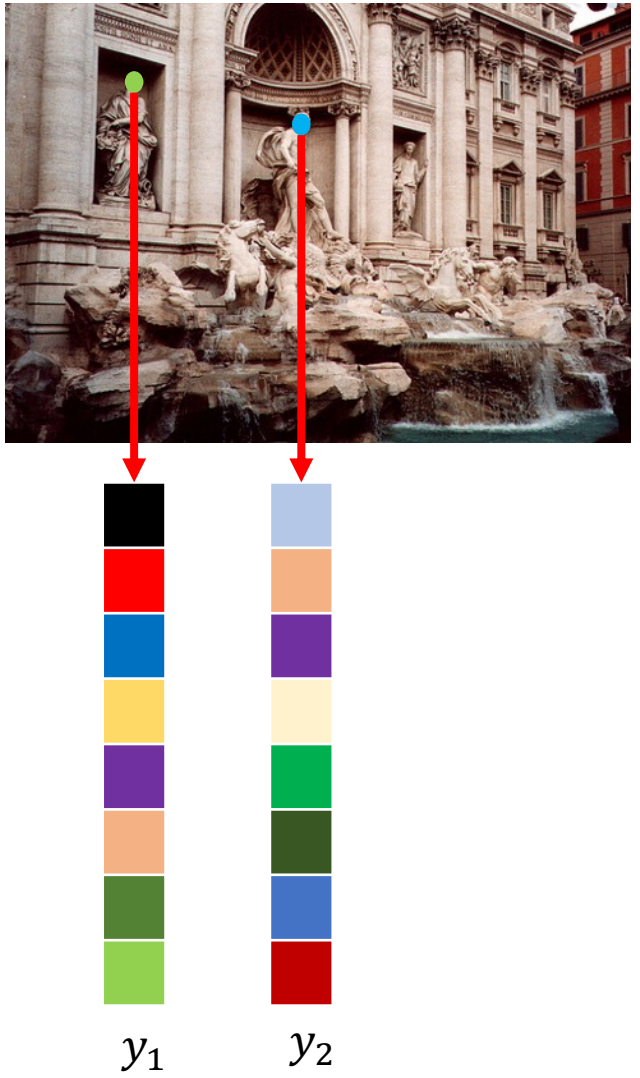


Two interrelated questions:

1. How do we *describe* each feature point?
2. How do we *match* descriptions?

Feature descriptor





Feature matching

- Measure the distance between (or similarity between) every pair of descriptors

	y_1	y_2
x_1	$d(x_1, y_1)$	$d(x_1, y_2)$
x_2	$d(x_2, y_1)$	$d(x_2, y_2)$

Invariance vs. discriminability

- Invariance:
  - Distance between descriptors of corresponding points should be small even if image is transformed

- Discriminability:
  - Descriptor for a point should be highly unique for each point (far away from other points in the image)

Invariance

- Most feature descriptors are designed to be invariant to
  - Translation, 2D rotation, scale

- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 60 degrees)
  - Limited affine transformations (some are fully affine invariant)
  - Limited illumination/contrast changes

Simple baseline descriptors

Design an invariant feature descriptor

- Simplest descriptor: a single 0
  - What's this invariant to?
  - Is this discriminative?
- Next simplest descriptor: a single pixel
  - What's this invariant to?
  - Is this discriminative?