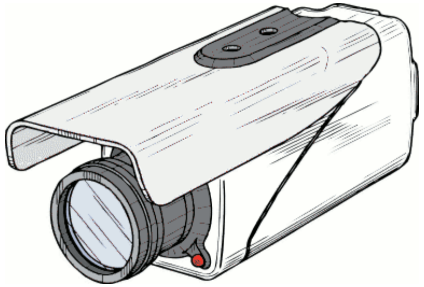
Univerza *v Ljubljani*

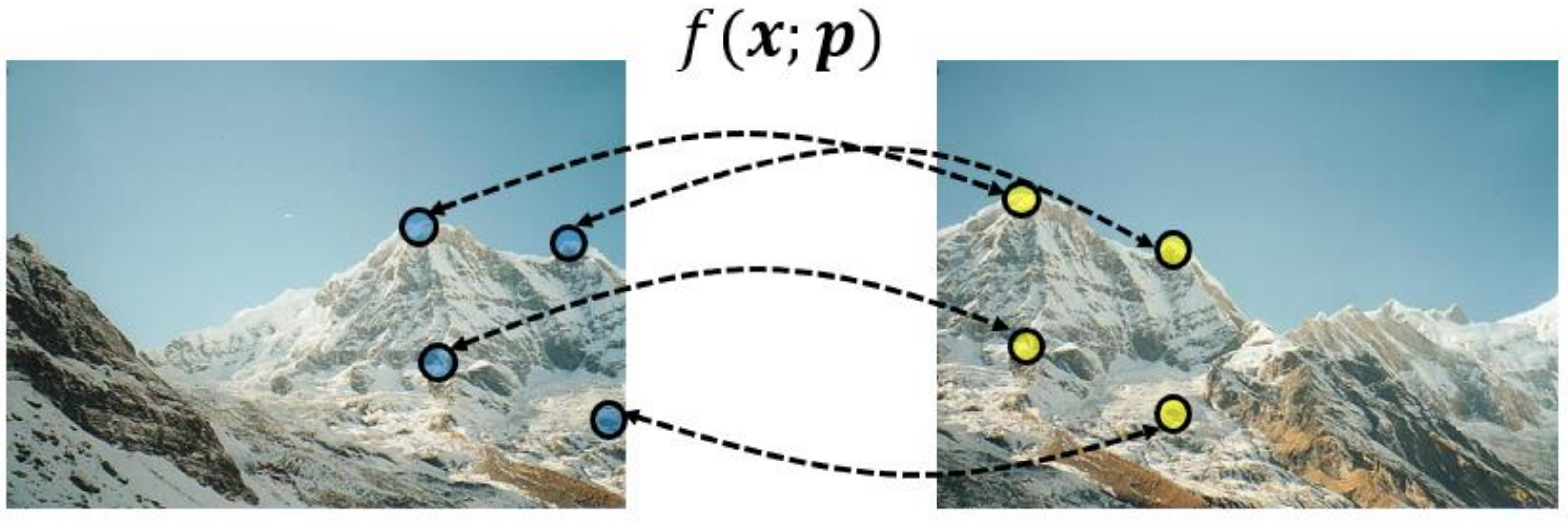
Machine Perception
Key-points and correspondences

Matej Kristan



Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

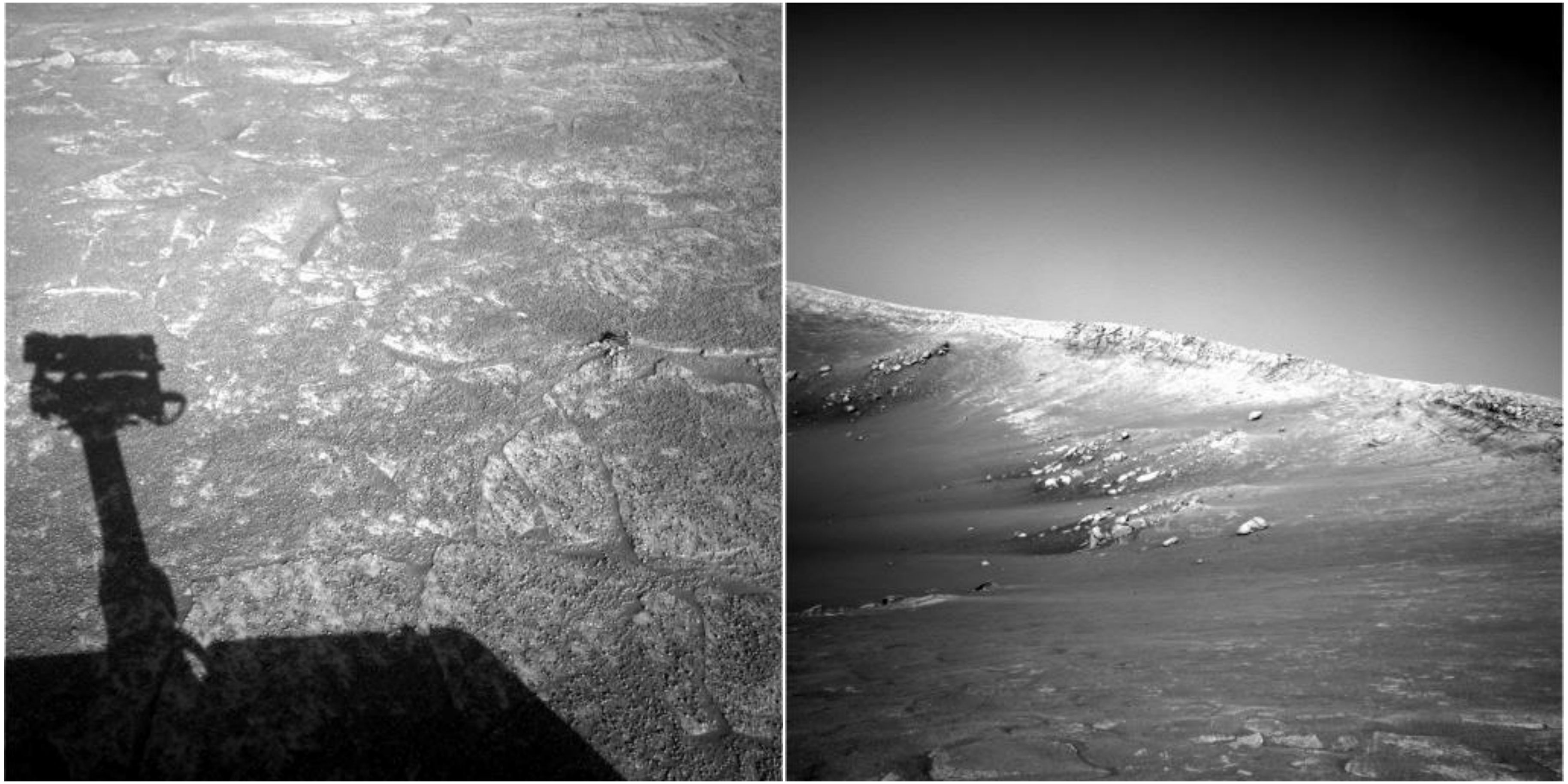
Recall the panorama creation process



Identification of the corresponding "key points" required!

Corresponding key points selection

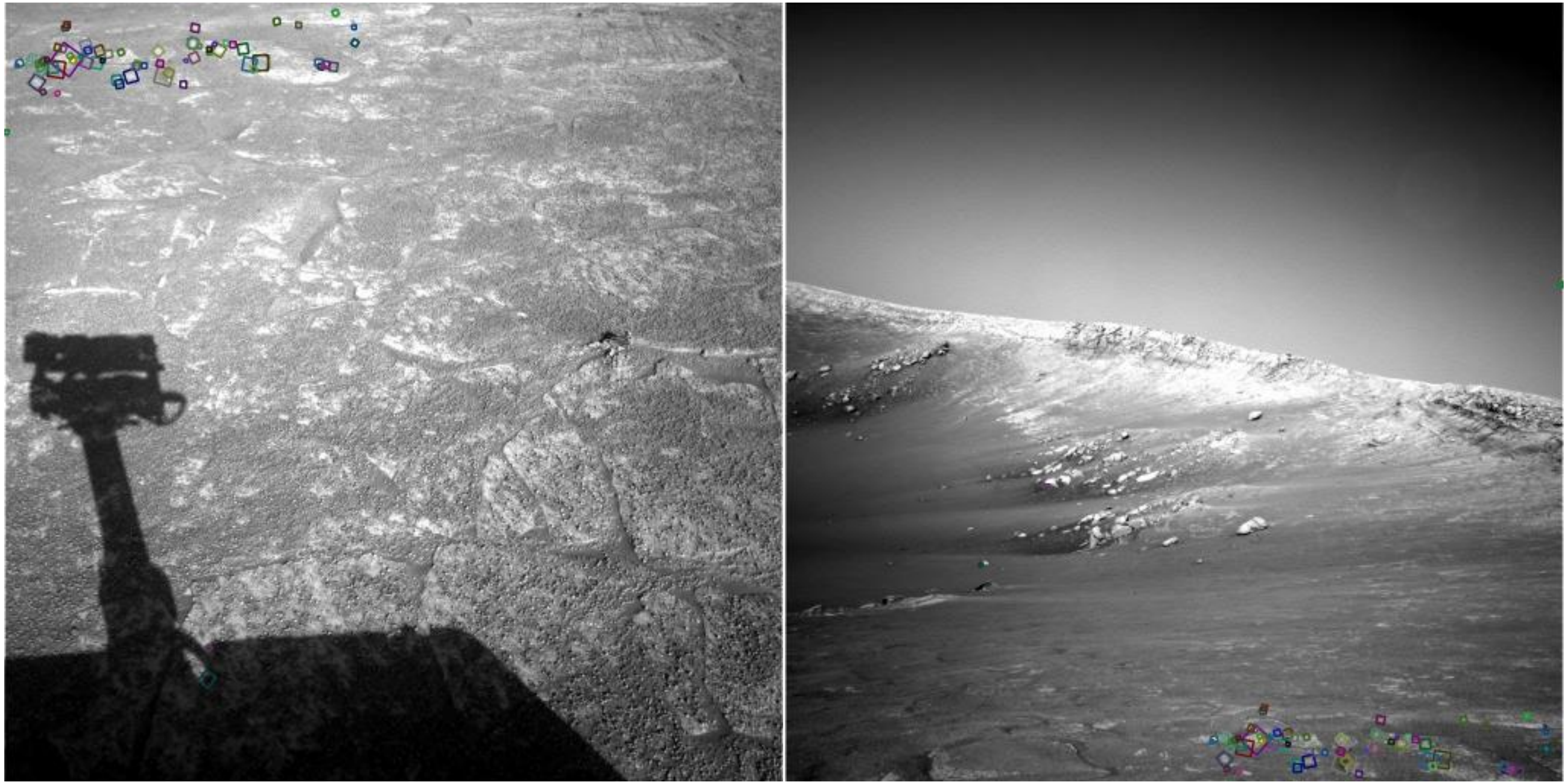
Manual selection often nontrivial



NASA Mars Rover images
(Figure by Noah Snavely)

Corresponding key points selection

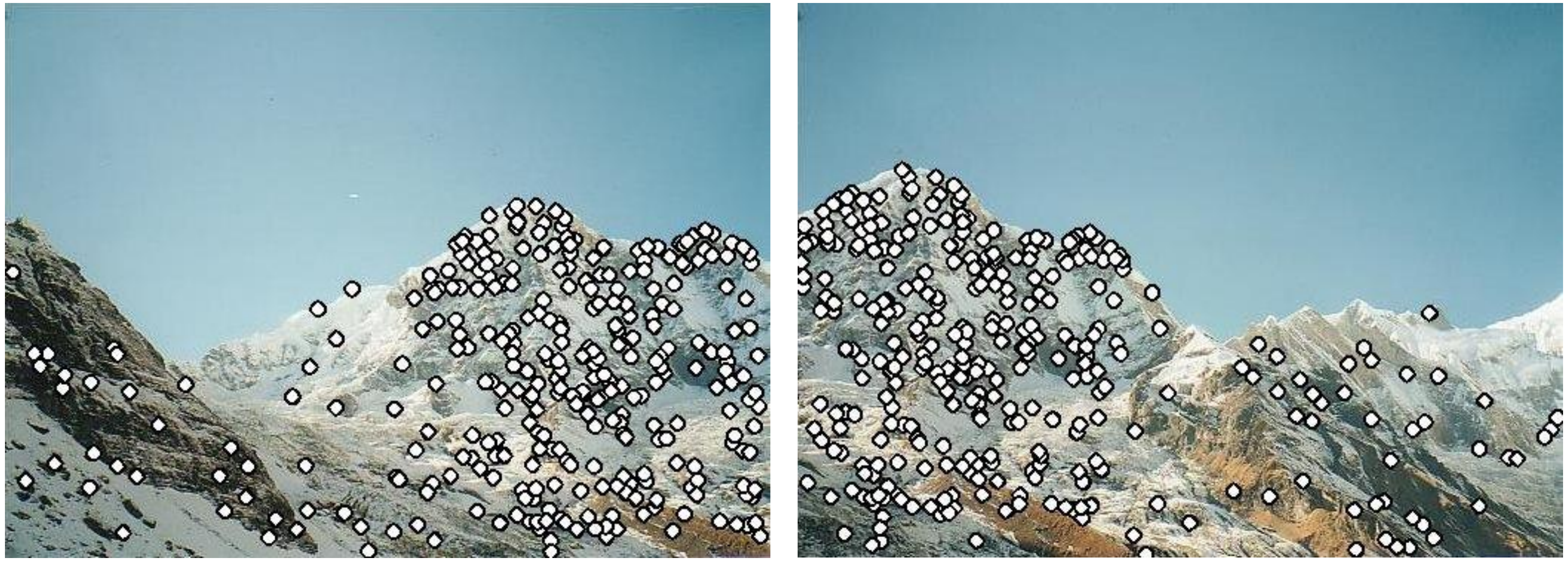
Manual selection often nontrivial



NASA Mars Rover images
with SIFT feature matches
(Figure by Noah Snavely)

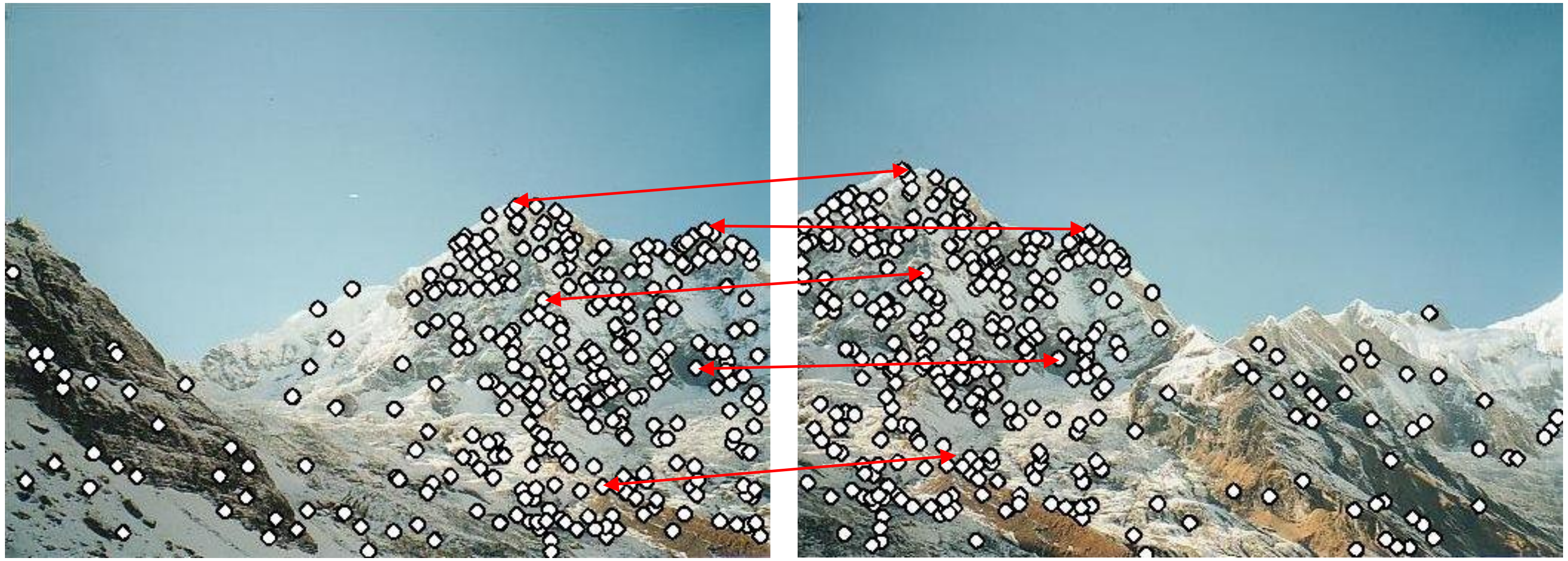
A case study: Automatic panorama creator

A case study: Automatic panorama creator



- Standard procedure:
  - Detect interest points (key-points) in both images

A case study: Automatic panorama creator



- Standard procedure:
  - Detect interest points (key-points) in both images
  - Find pairs of corresponding points

A case study: Automatic panorama creator

- Standard procedure:
  - Detect interest points (key-points) in both images
  - Find pairs of corresponding points
  - Use these pairs for image registration ← (e.g., RANSAC/least-squares transformation model estimation)

Efficient keypoint detector requirements

- Requirement 1:
  - Detect the same structure *independently* in each image.

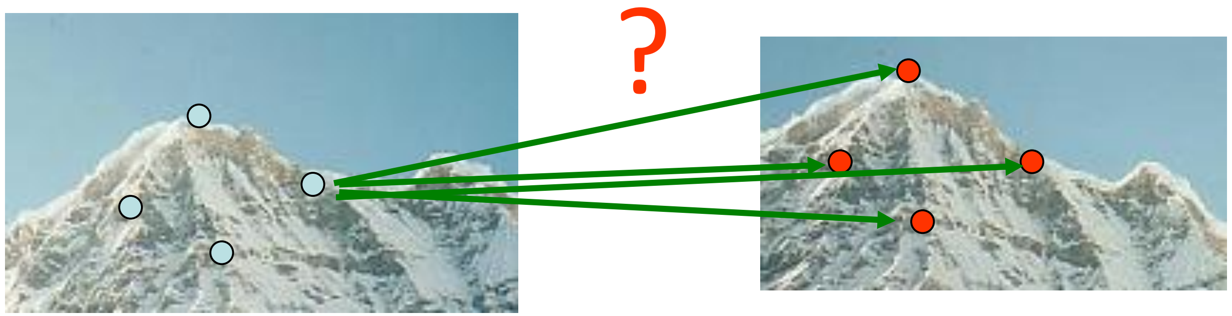
Try random sampling?

Bad idea: By random sampling in each image,
we will not likely detect the same points.

A *detector* with a high detection repeatability is required!

Efficient keypoint detector requirements

- Requirement 1:
  - Detect the same structure *independently* in each image.
- Requirement 2:
  - For each point find a corresponding point in the other image.



A reliable and distinctive *descriptor* is required!

Outline of this lecture

1. Keypoint DETECTION

2. Keypoint DESCRIPTION

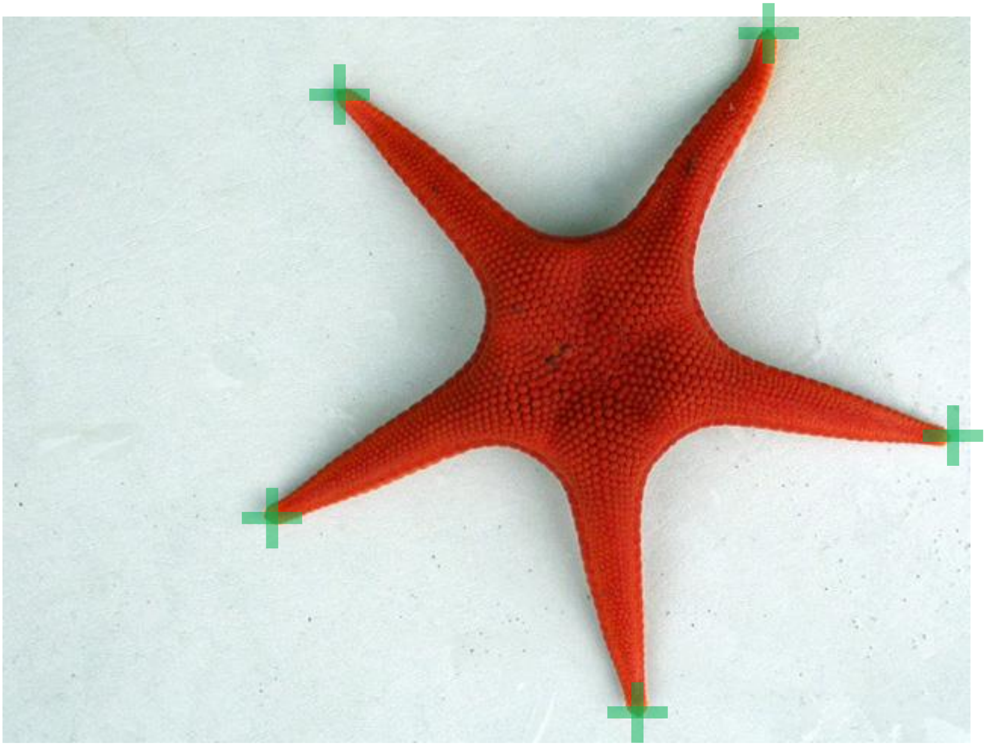
3. Keypoint MATCHING

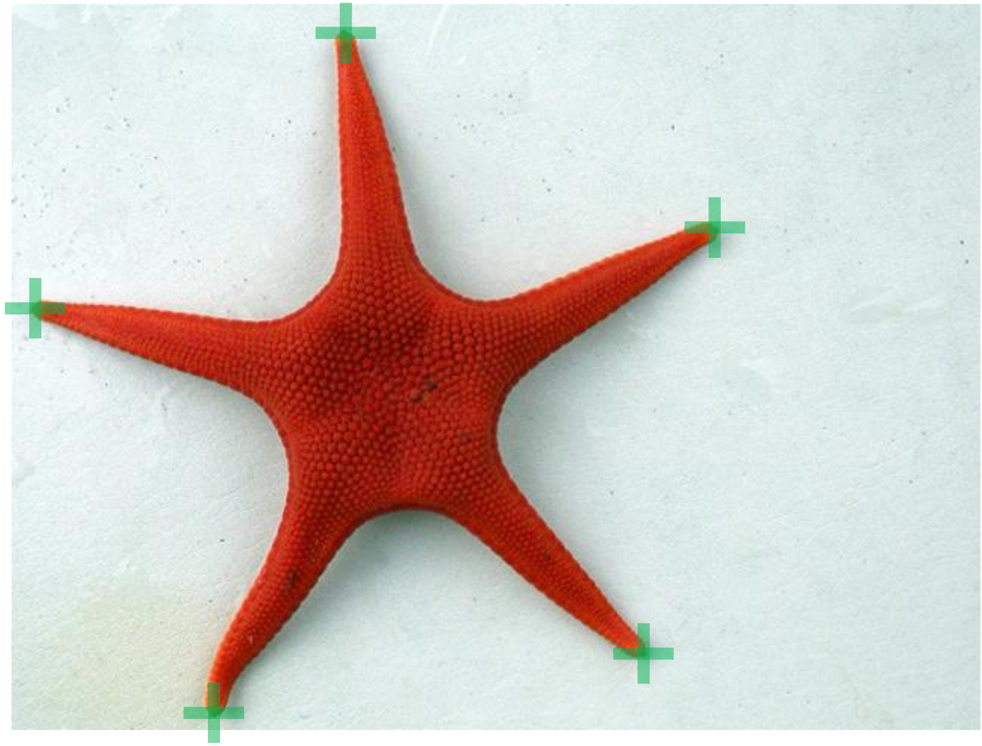
Machine Perception

SINGLE SCALE KEY-POINT DETECTION

Corners as keypoints

- Distinctive and repeatedly occurring on the same structures even if the structure changes pose in 3D





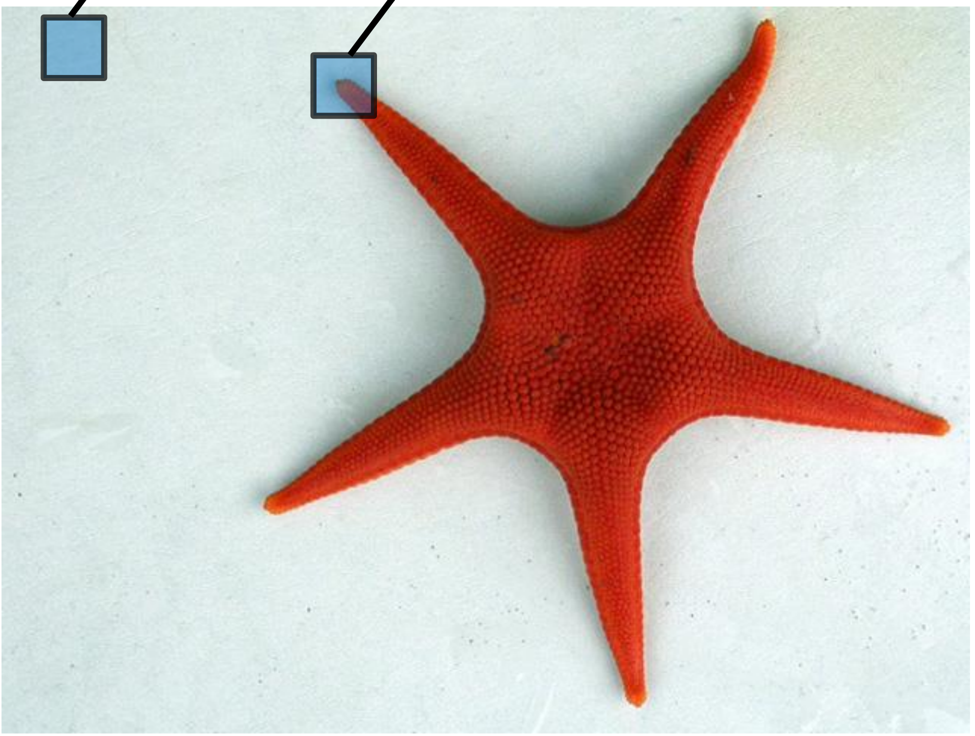
C.Harris and M.Stephens. "A Combined Corner and Edge Detector.“ *Proceedings of the 4th Alvey Vision Conference*, 1988.

Require a corner response function – CRF

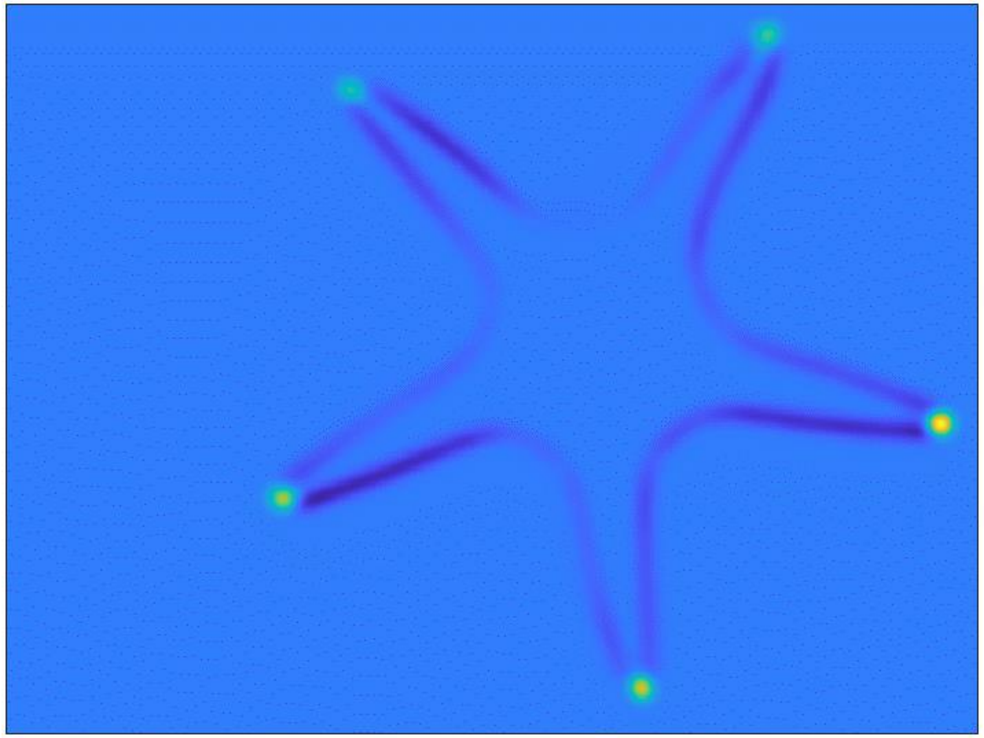
- An operator that gives a strong response on the corner structure

No corner: Low value

Corner: High value

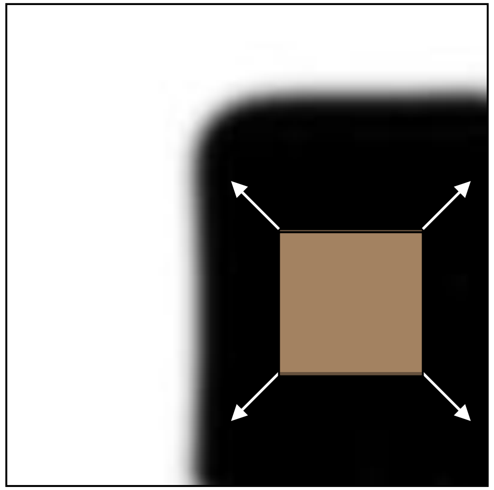


Corner response function (CRF)

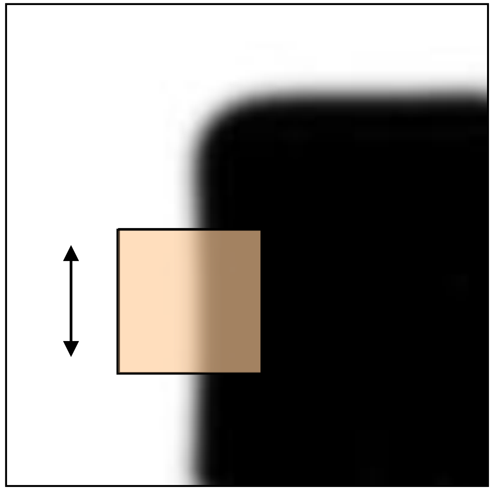


Corner response function: Intuition

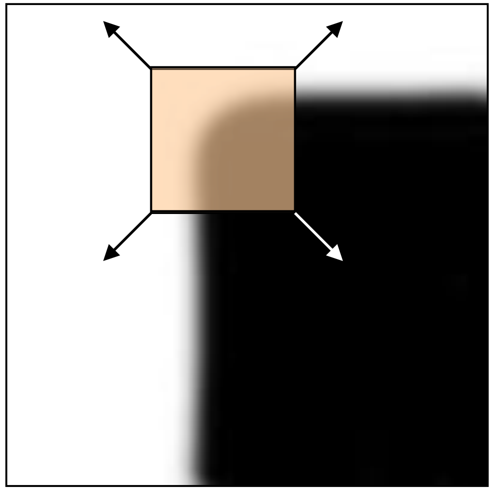
- A good corner detector criteria: Self similarity
  - Observe a small window R around a potential corner (locality).
  - A small shift in window in any direction results in a large intensity change (good localization)



"Flat" region:
A small shift in any direction does not cause an intensity change.



"Edge":
No change when shifting along the edge, otherwise there is a change.



"Corner":
A shift in any direction significantly changes the local intesity.

Slide credit: Alexej Efros

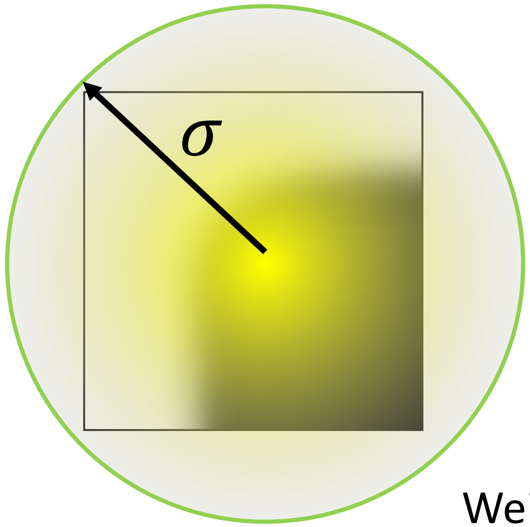
Harris corner detector

- The intensity change for a shift [u,v]:

  (weighted autocorrelation function)

$E_R(u,v)$ should be large *for any* small displacement (u,v)

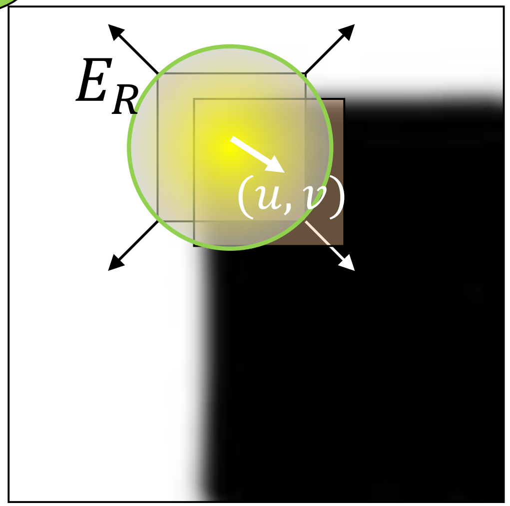
$$E_R = \sum_{x,y \in R} w(x,y)\left(I(x,y) - I(x-u,y-v)\right)^2$$



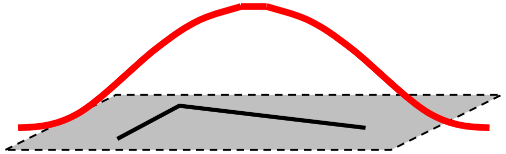
Weight function

Intensity before the shift

Intensity afer the shift



Weight function $w(x,y)$ =



Gaussian kernel $G(\sigma)$

The σ specifies the region R size!

Harris corner detector

- Linearize for small shifts (u, v):

$$\mathrm{I}(x-u, y-v) \approx I(x, y) + [I_x(x, y), I_y(x, y)] \begin{bmatrix} u \\ v \end{bmatrix}$$

$$I_x(x, y) = \frac{\partial I(x, y)}{\partial x}, I_y(x, y) = \frac{\partial I(x, y)}{\partial y}$$

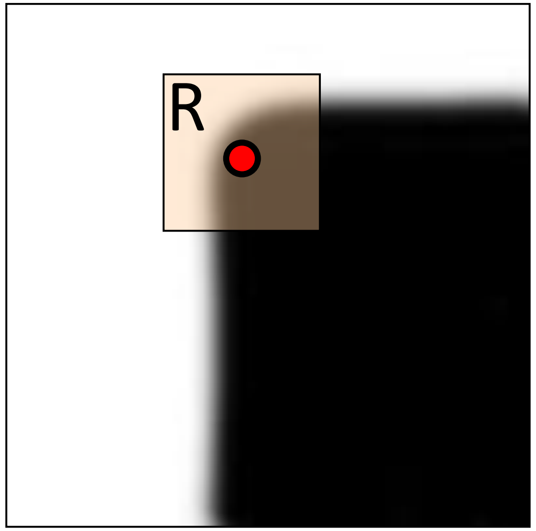
- Plug into the weighted autocorrelation

$$E_R \approx \sum_{x,y \in R} w(u, v)([I_x(x, y), I_y(x, y)] \begin{bmatrix} u \\ v \end{bmatrix})^2$$

$$\approx [u, v] \left(\sum_{x,y \in R} w(x, y) \begin{bmatrix} I_x(x, y) \\ I_y(x, y) \end{bmatrix} [I_x(x, y), I_y(x, y)] \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

Harris corner detector

$$E_R(u,v) = \sum_{x,y \in R} w(x,y)\left(I(x,y) - I(x-u,y-v)\right)^2$$



- For small shifts (u, v) E can be linearly approximated by:

$$E_R \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

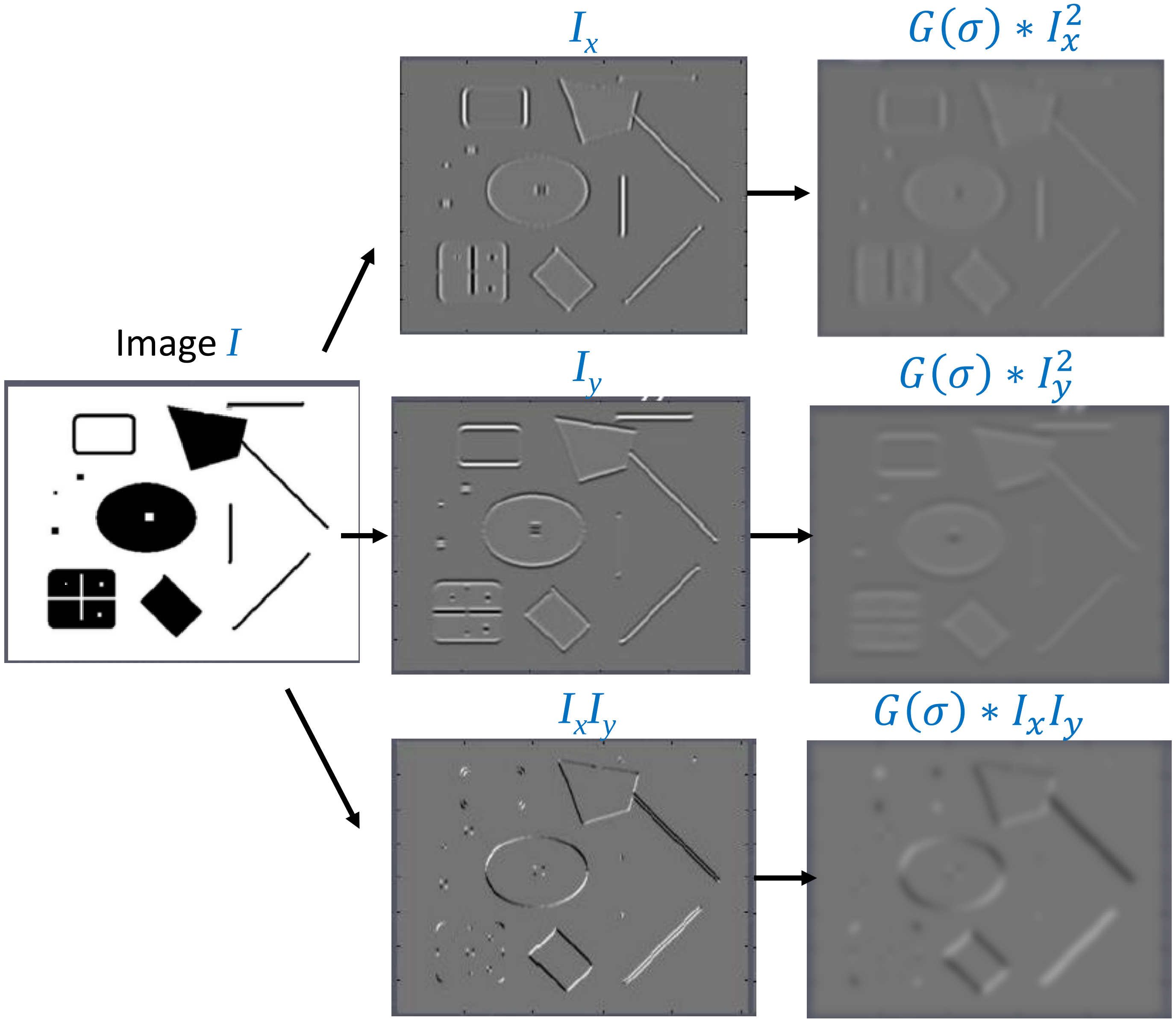
with M 2x2 matrix of image derivatives:

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

A weighted sum over the region R centered at (x,y)
in which we are verifying a corner

Construction of M can be made more efficient!

Construction of M



Note, M is different for each location

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

$$M = \begin{bmatrix} G(\sigma) * I_x^2 & G(\sigma) * I_x I_y \\ G(\sigma) * I_x I_y & G(\sigma) * I_y^2 \end{bmatrix}$$

Harris corner detector

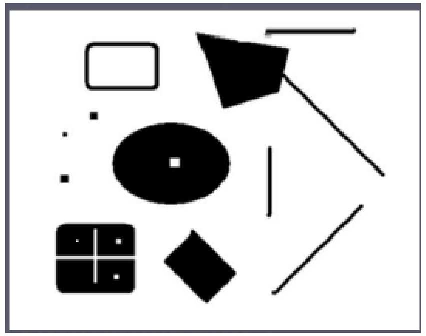
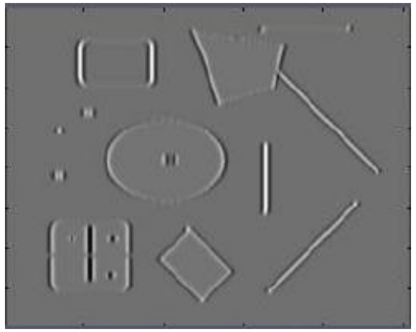
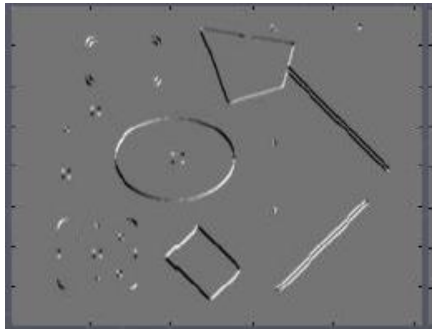


Image I



I_x

I_y



I_xI_y

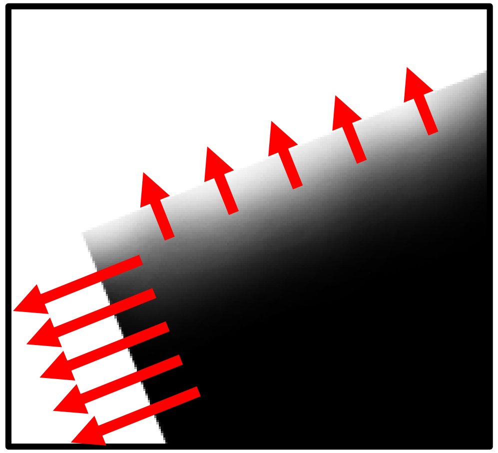
- Matrix M is the covariance matrix of region gradients:

$$M = \begin{bmatrix} G(\sigma) * I_x^2 & G(\sigma) * I_xI_y \\ G(\sigma) * I_xI_y & G(\sigma) * I_y^2 \end{bmatrix}$$

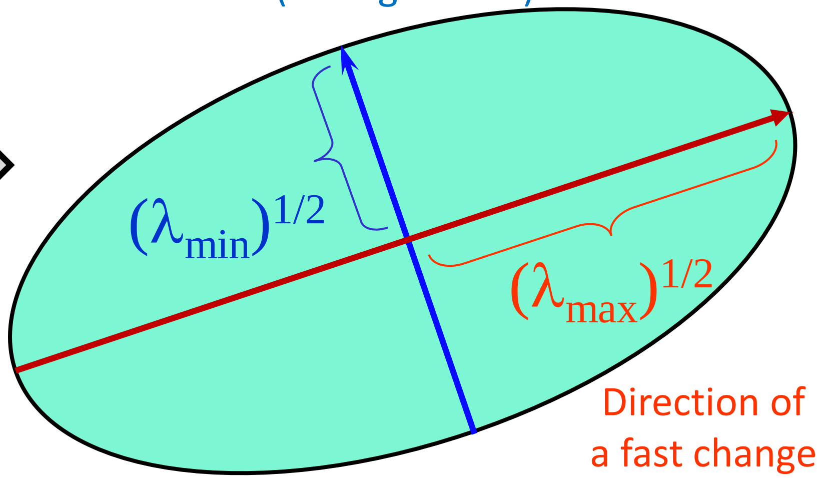
- A corner is detected by analyzing the gradient covariance matrix

The Covariance matrix analysis

- Visualize the covariance matrix as an ellipse...



$\Rightarrow$ ***M*** $\Rightarrow$



- Decompose into eigenvectors and eigenvalues:

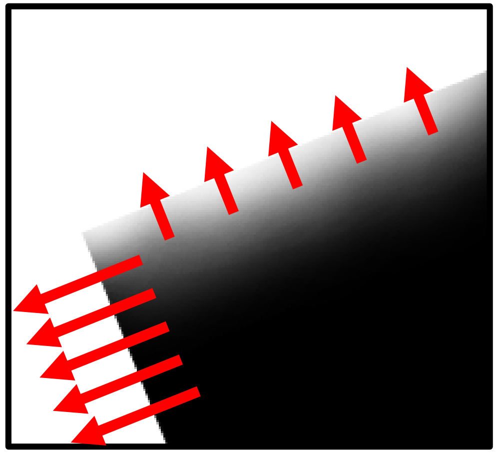
$$M = R \begin{bmatrix} \lambda_{\max} & 0 \\ 0 & \lambda_{\min} \end{bmatrix} R^T$$

eigen values (ellipse SCALING)

eigen vectors
(ellipse ROTATION)

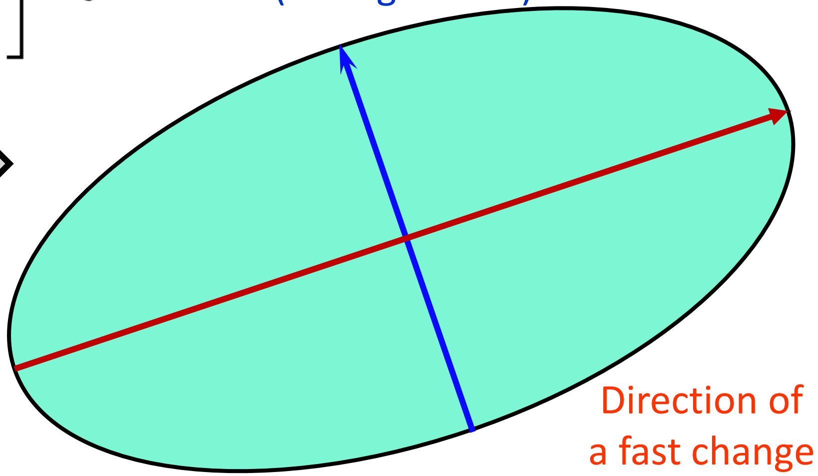
The Covariance matrix analysis

- Visualize the covariance matrix as an ellipse...



$$M = R \begin{bmatrix} \lambda_{\max} & 0 \\ 0 & \lambda_{\min} \end{bmatrix} R^T$$

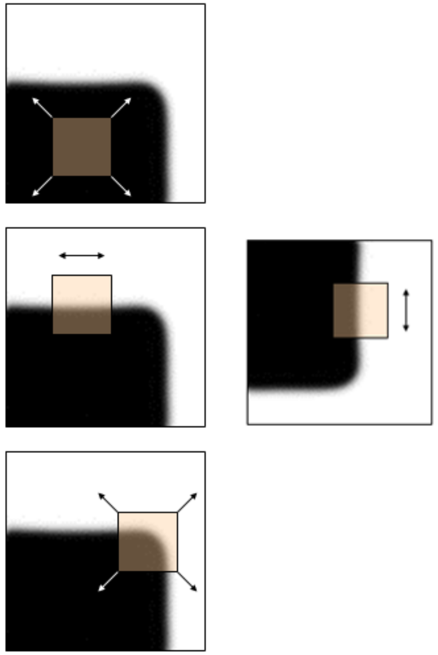
M



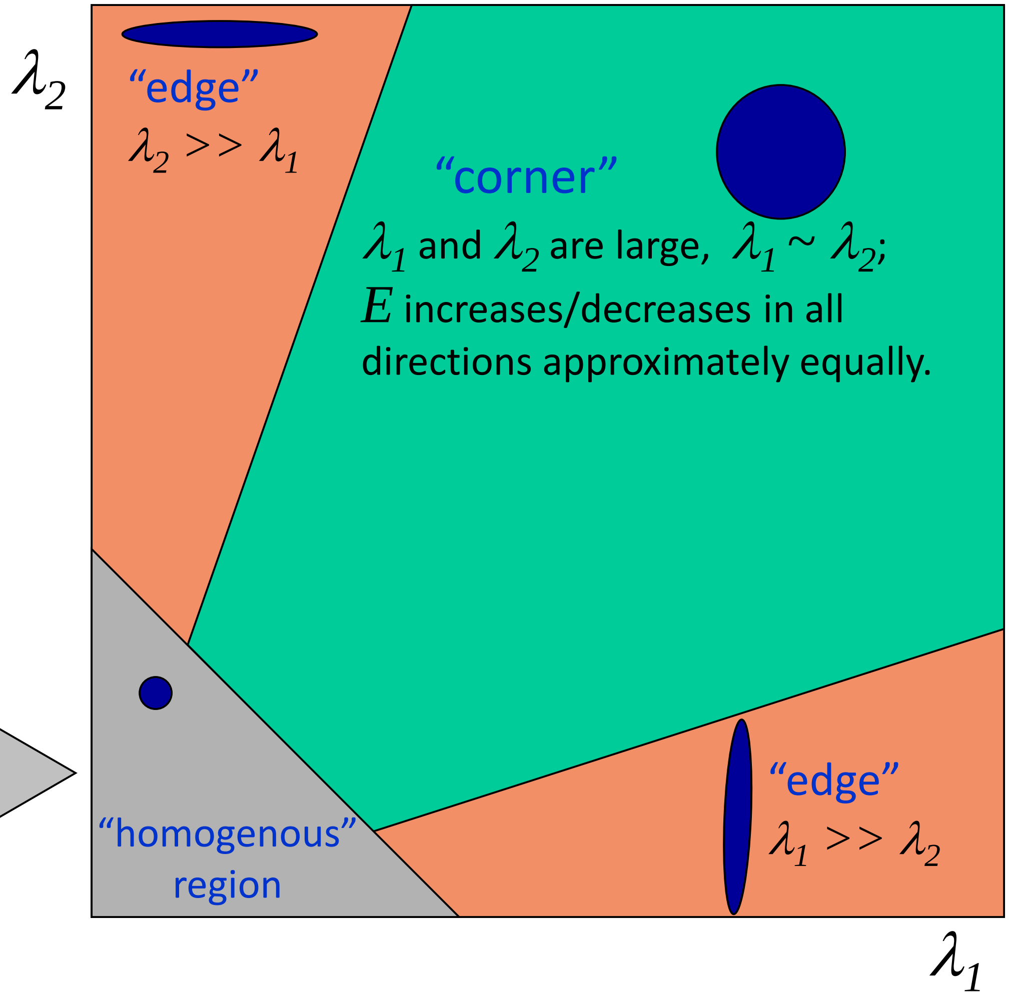
- Decompose into eigenvectors and eigenvalues:
  - A corner has a strong gradient in both major directions!
  - A corner is present when both eigenvalues are large.

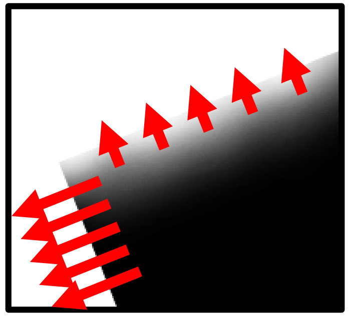
Eigen values: Interpretation

- Corner detection by eigenvalues of ***M***:

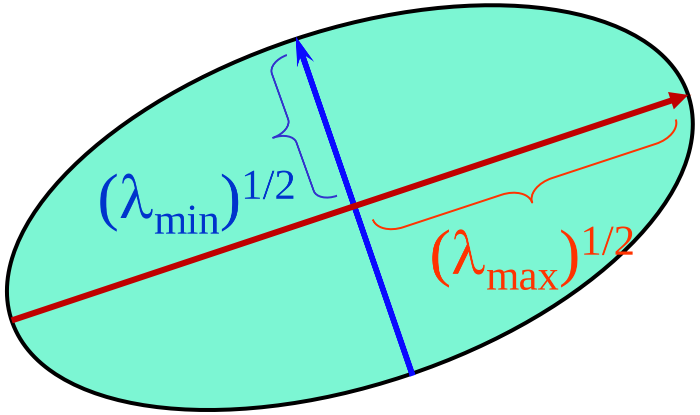


λ_1 and λ_2 small;
E almost constant in all directions.





Direction of a slow change (smal gradient)



Direction of a fast change (large gradient)

Eigen values: Interpretation

- Problem: Calculating the eigenvalues at each pixel is computationally intensive!
- Solution: We are after the ratio between the two eigenvalues and a rough estimate of their magnitude.

$$r = \frac{\lambda_1}{\lambda_2}$$

Standard results: $\det(\mathbf{M}) = \lambda_1 \lambda_2, \mathrm{trace}(\mathbf{M}) = \lambda_1 + \lambda_2$

$$\frac{\mathrm{trace}^2(\mathbf{M})}{\det(\mathbf{M})} = \frac{(\lambda_1+\lambda_2)^2}{\lambda_1 \lambda_2} = \frac{(r+1)^2}{r} = \alpha^{-1}$$

$$\boxed{\det(\mathbf{M}) - \alpha \mathrm{trace}^2(\mathbf{M})} = 0$$

This is the corner response function!

The corner response function

- In practice, fix α and check if corner response function exceeds a threshold

$$\det(\mathbf{M}) - \alpha \mathrm{trace}^2(\mathbf{M}) > t$$

- We can calculate the *Determinant* and *Trace* directly:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \qquad \begin{aligned} \det(\mathbf{M}) &= AB - C^2 \\ \mathrm{trace}(\mathbf{M}) &= A + B \end{aligned}$$

$$AB - C^2 - \alpha(A+B)^2 > t$$

In practice, α: (0.04 to 0.06)

$$M = \begin{bmatrix} G(\sigma) * I_x^2 & G(\sigma) * I_x I_y \\ G(\sigma) * I_x I_y & G(\sigma) * I_y^2 \end{bmatrix}$$

Harris corner detector: Summary

- Calculate the covariance matrix

  (by virtue of autocorrelation)

$$M = \begin{bmatrix} G(\sigma) * I_x^2 & G(\sigma) * I_x I_y \\ G(\sigma) * I_x I_y & G(\sigma) * I_y^2 \end{bmatrix}$$

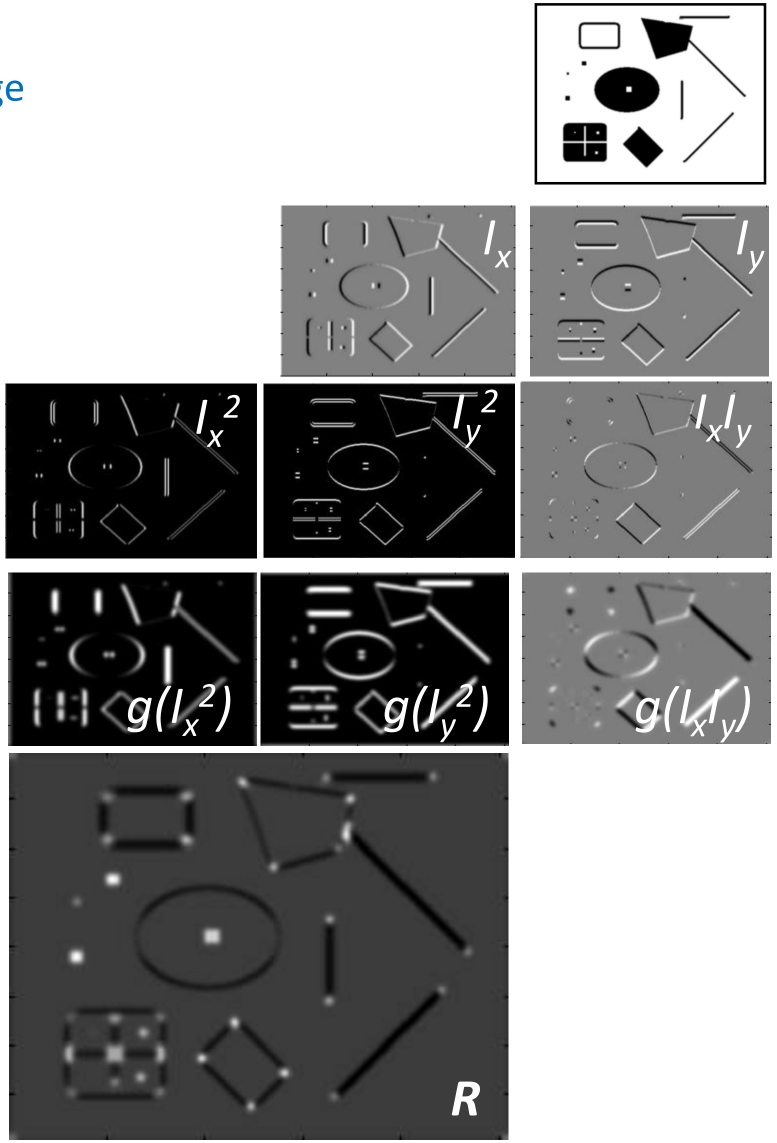
$$\det(\mathbf{M}) - \alpha \mathrm{trace}^2(\mathbf{M}) > t$$

0. The source image

1. Image derivatives

2. Squared derivatives

3. Gaussian filtered squared derivatives $g(s_I)$



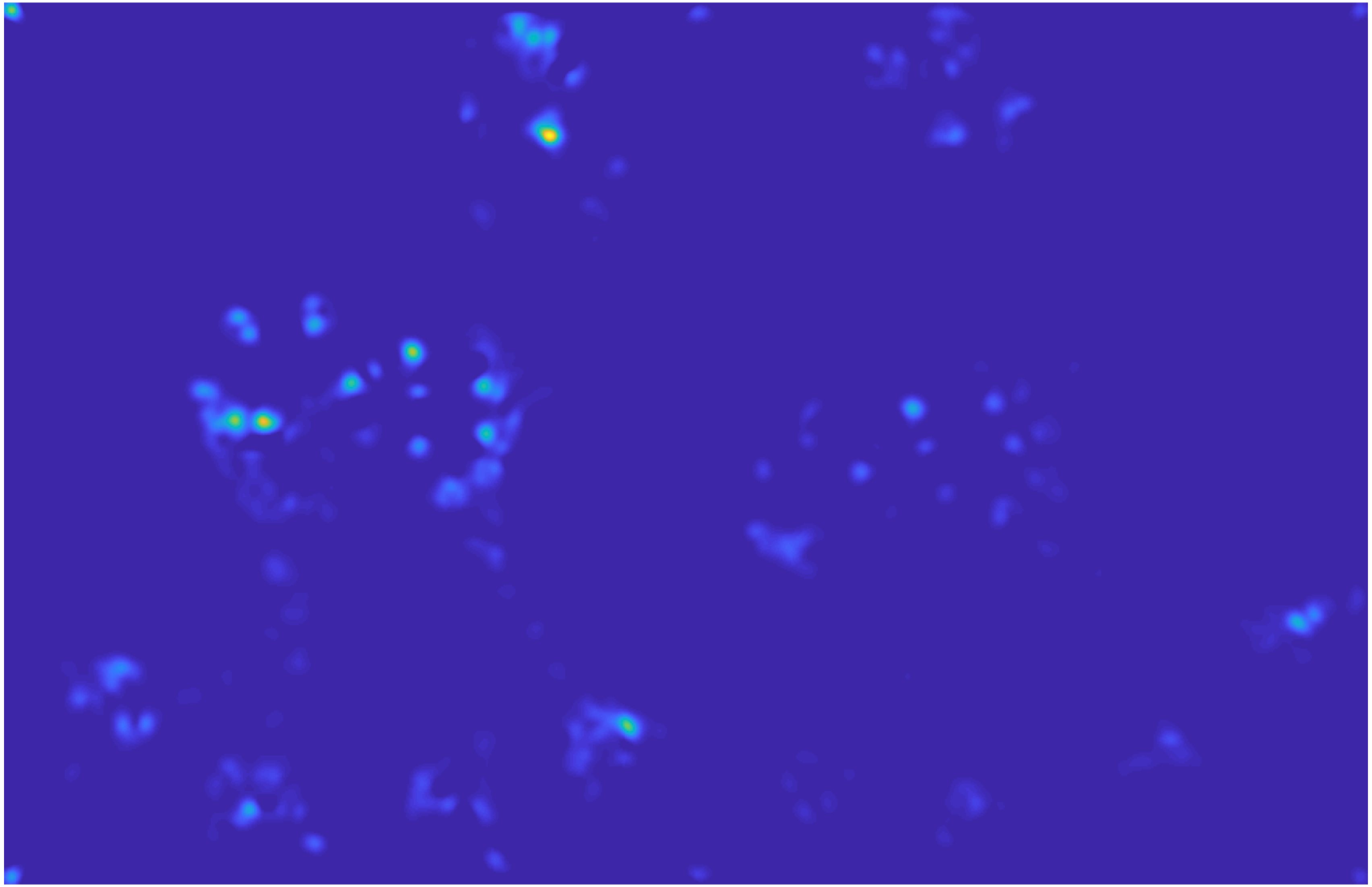
4. Corner presence– two strong eigen values

$$\mathrm{c(I)} = \det[M] - \alpha[\mathrm{trace}^2(M)]$$
$$= g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Apply a non-maxima suppression

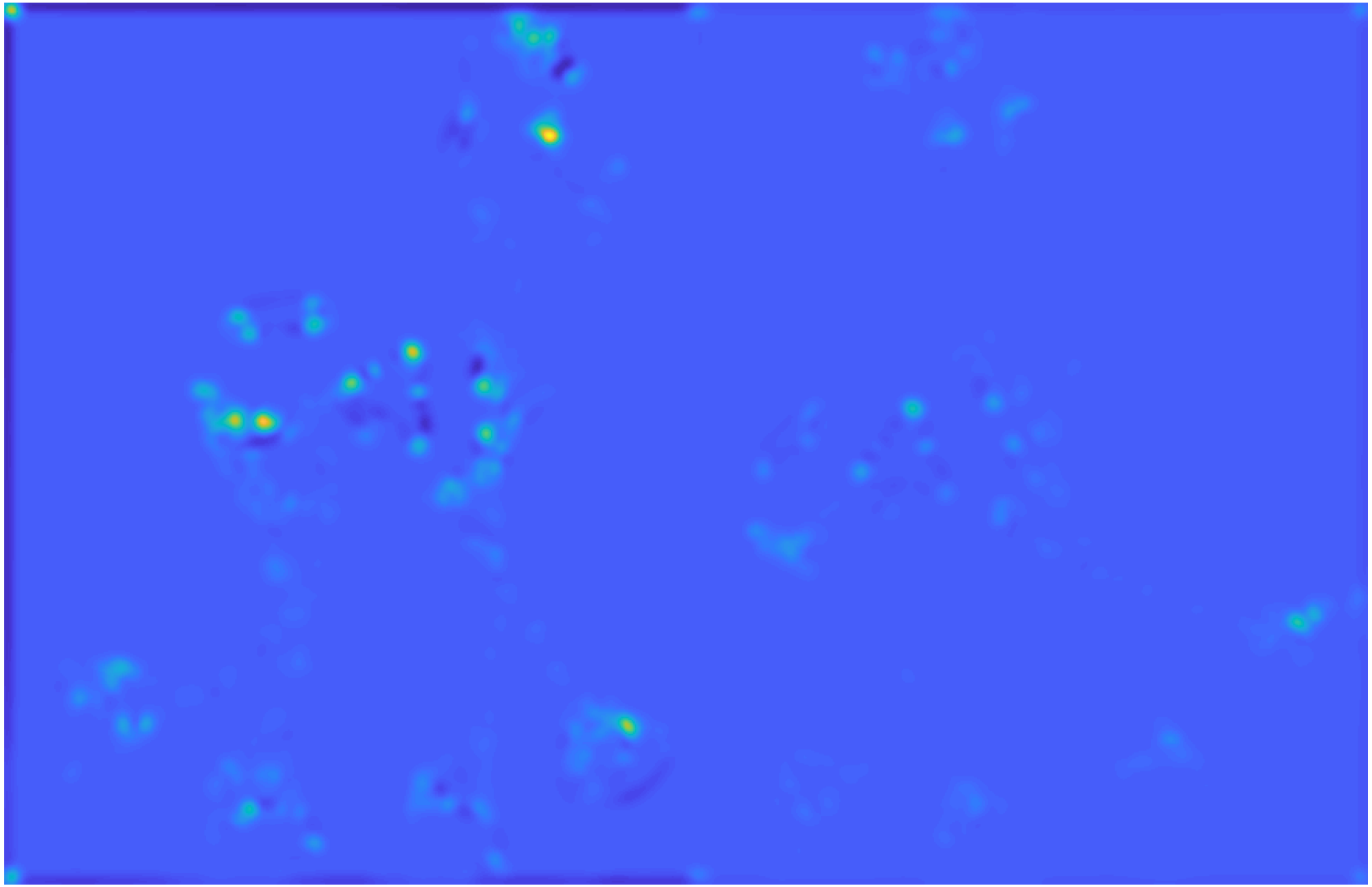
Harris corner detector: Summary

Harris corner detector: Summary



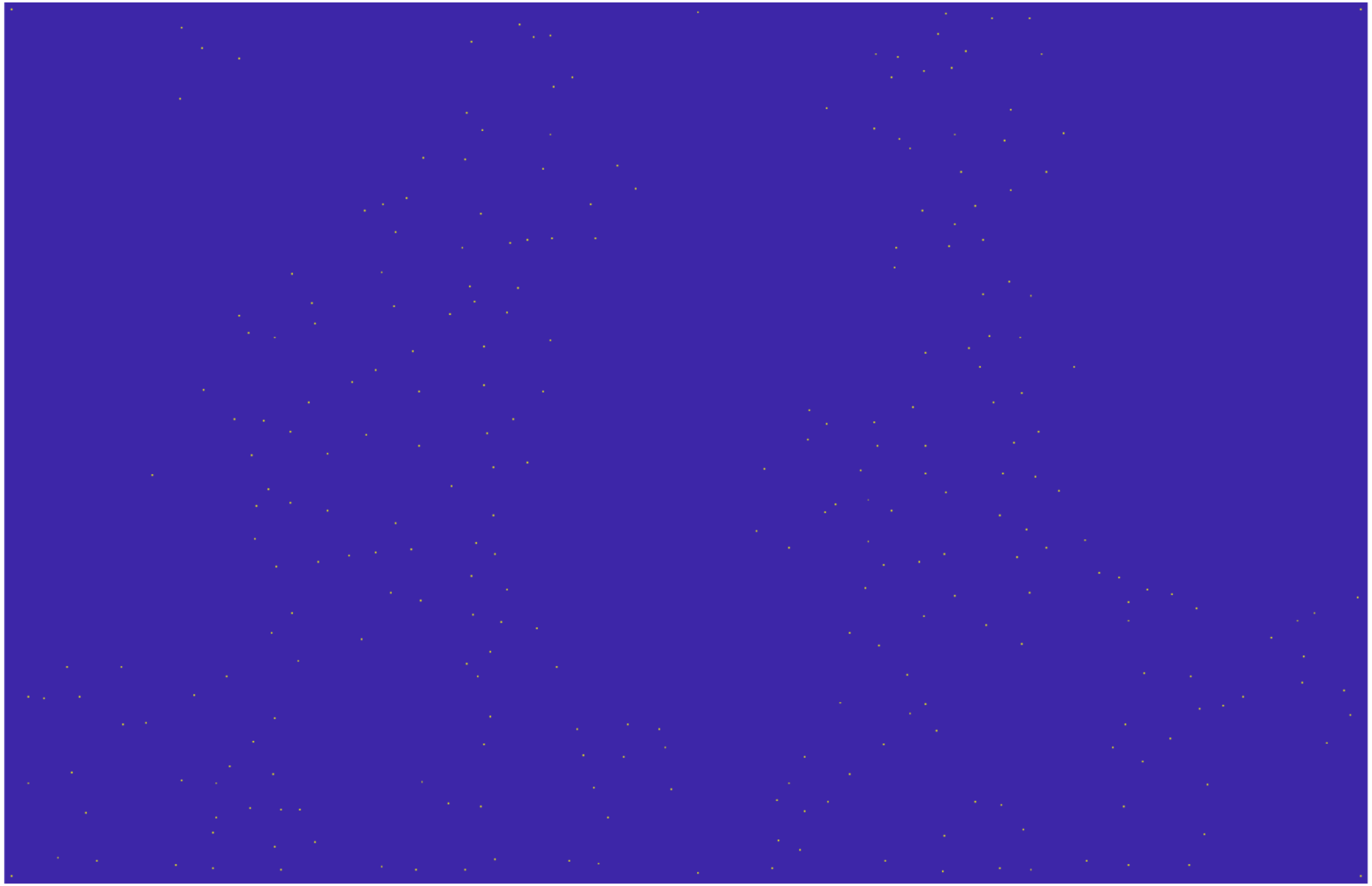
- The Corner response function: $c(I) = g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$

Harris corner detector: Summary



- Set values lower than a threshold to zero: $c(c < \text{threshold}) = 0$

Harris corner detector: Summary

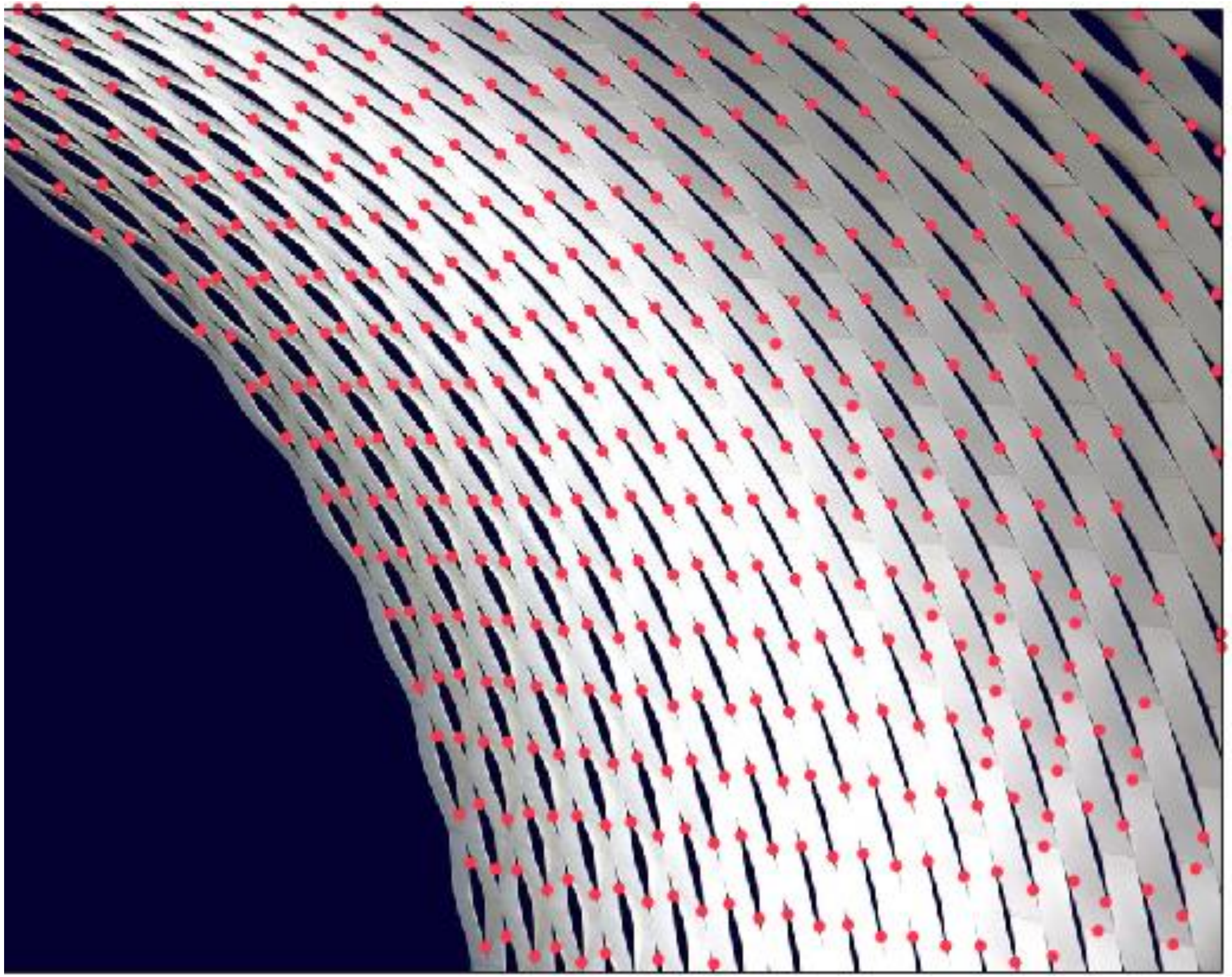


- Find the local maxima in *c(I)*

Harris corner detector: Summary

- Detected Harris corners

Harris detector

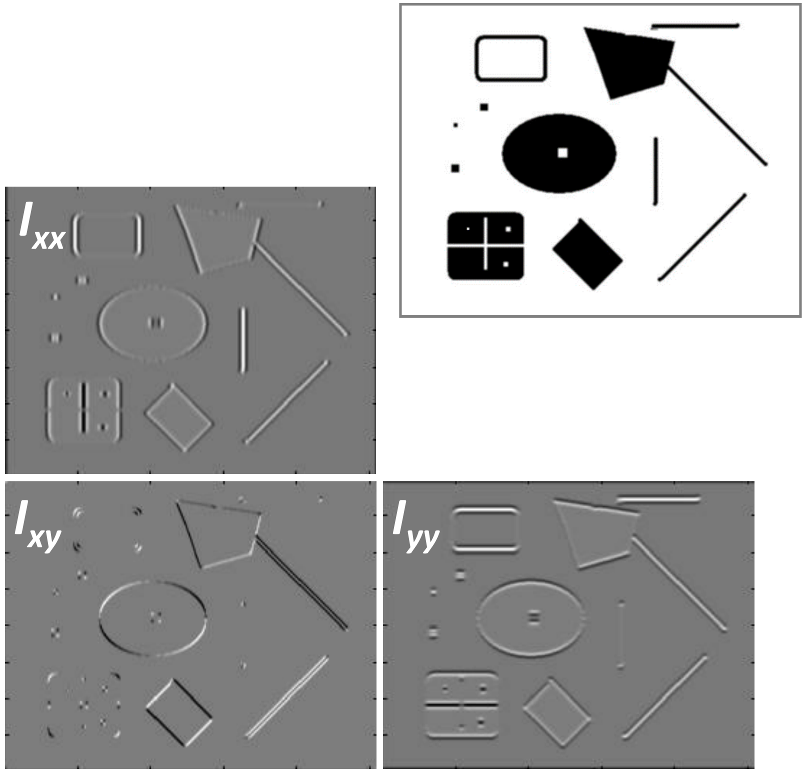


Local curvature as a key-point presence measure

The Hessian corner detector

- Determinant of a Hessian

$$\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

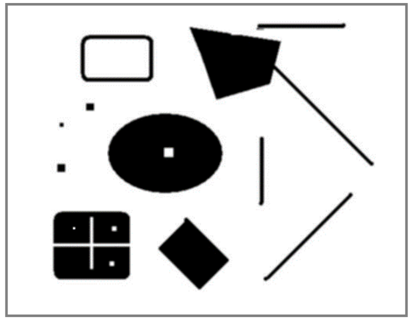


Note: these are second order derivatives!
(Recall what Hessian means
→ a measure of local curvature)

Intuition: Find strong gradients in two orthogonal directions

The Hessian corner detector

- Determinant of a Hessian

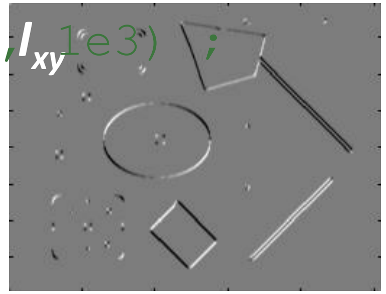


$$\mathrm{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

Note: these are second order derivatives!

(Recall what Hessian means → a measure of local curvature)

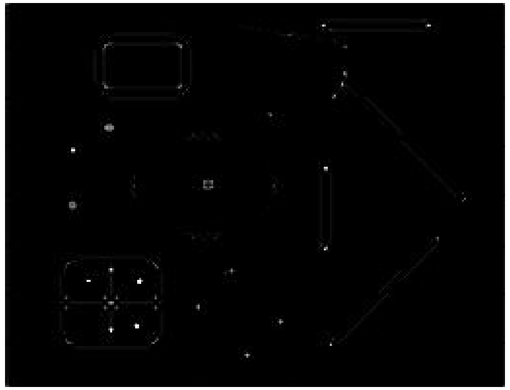
```
[P, H] = harris(I0, 1, 1e3) ;
```



$$\det(\mathrm{Hessian}(I)) = I_{xx} I_{yy} - I_{xy}^2$$

In Matlab:

$$I_{xx} .* I_{yy} - (I_{xy}).\wedge 2$$



The Hessian corner detector

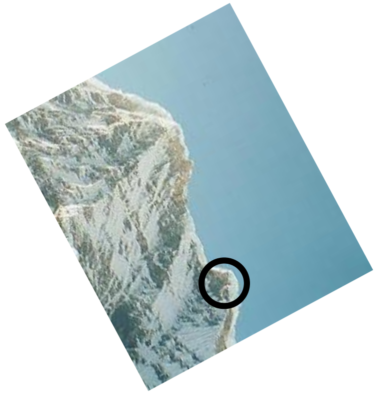
Result: responses on corners and blobs.

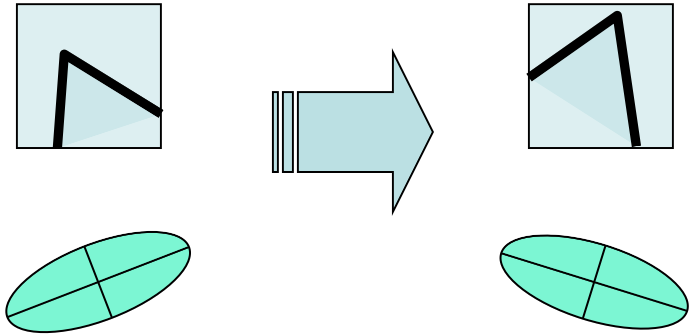
A number of keypoint detectors exist

- Hessian & Harris [Beaudet '78], [Harris '88]
- Laplacian, DoG [Lindeberg '98], [Lowe 1999]
- Harris-/Hessian-Laplace [Mikolajczyk & Schmid '01]
- Harris-/Hessian-Affine [Mikolajczyk & Schmid '04]
- MSER [Matas '02]
- FAST , and lots of others
- A very good tutorial ECCV 2012.
- *Learning-based detectors introduced in the last five years.*
- *These detector have become building blocks of numerous computer vision applications!*

Harris/Hessian detector: properties

- Is it rotation invariant?



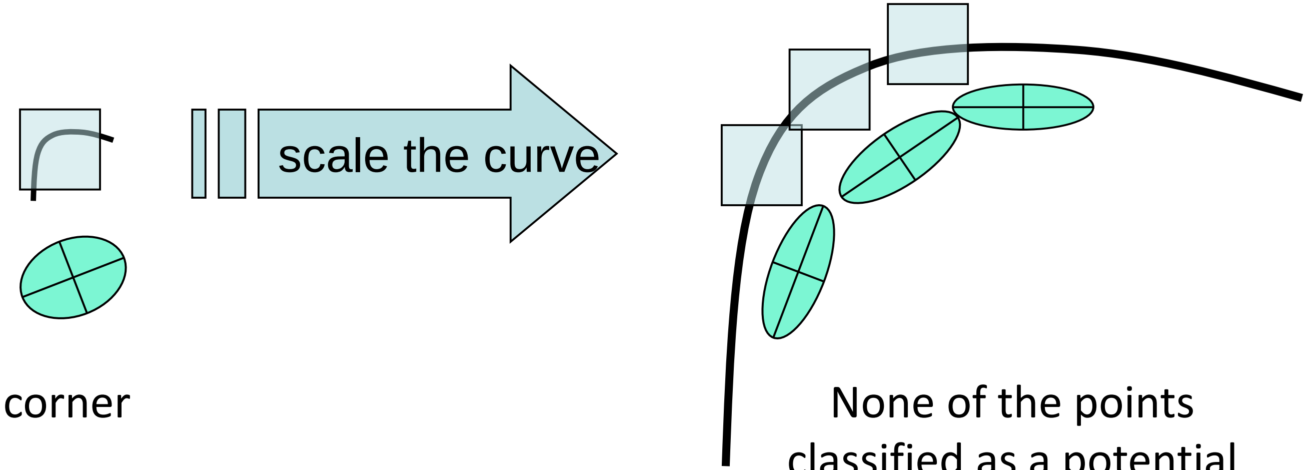


Ellipse rotates, but its shape (e.g., eigenvalues)
remains unchanged!

The corner response function is rotation invariant!

Harris/Hessian detector: properties

- Rotation invariance
- Is it invariant to scale change?



corner

None of the points classified as a potential corners!

NOT invariant to scale change!

Machine Perception

SCALE INVARIANCE

Key point neighborhood

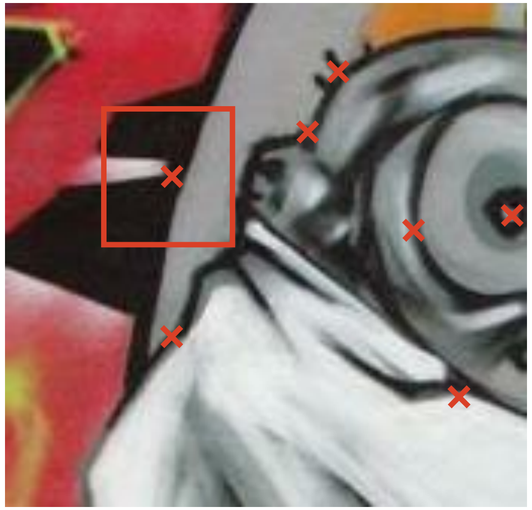
- Harris and Hessian determine the key-point location

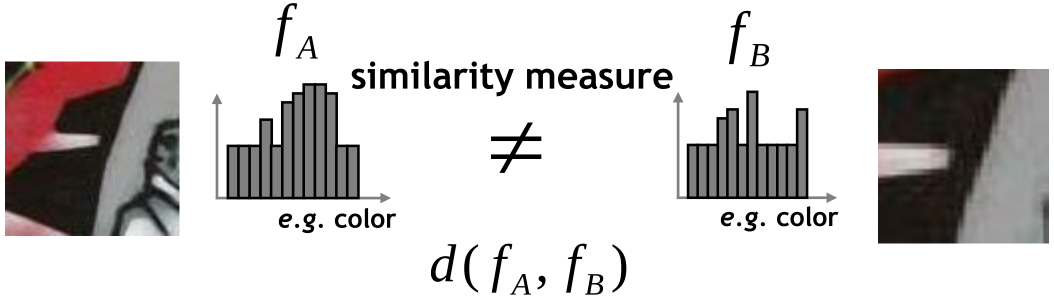
- Later we will see that keypoints are matched by comparing their neighborhood patches.

- For practical applications, the scale (local size) of the keypoint has to be estimated as well.

Naive approach: check all scales

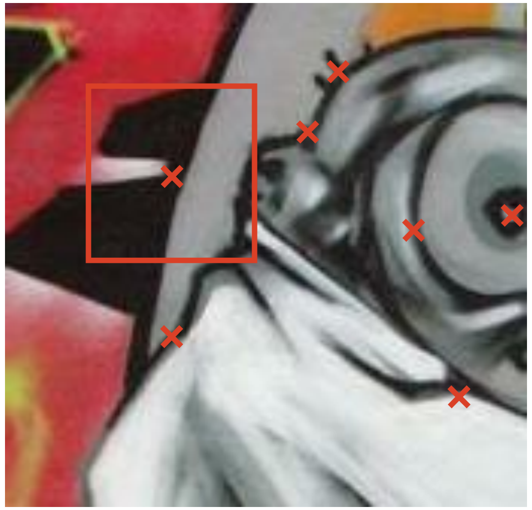
- Check all scales exhaustively
  - Vary the region size and compare the descriptors...

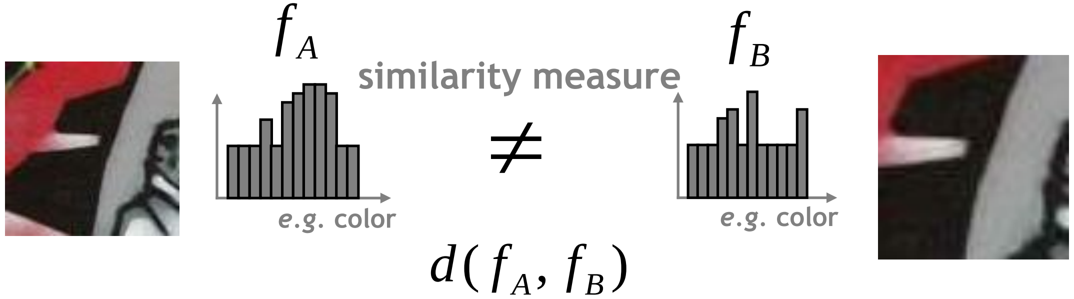




Naive approach: check all scales

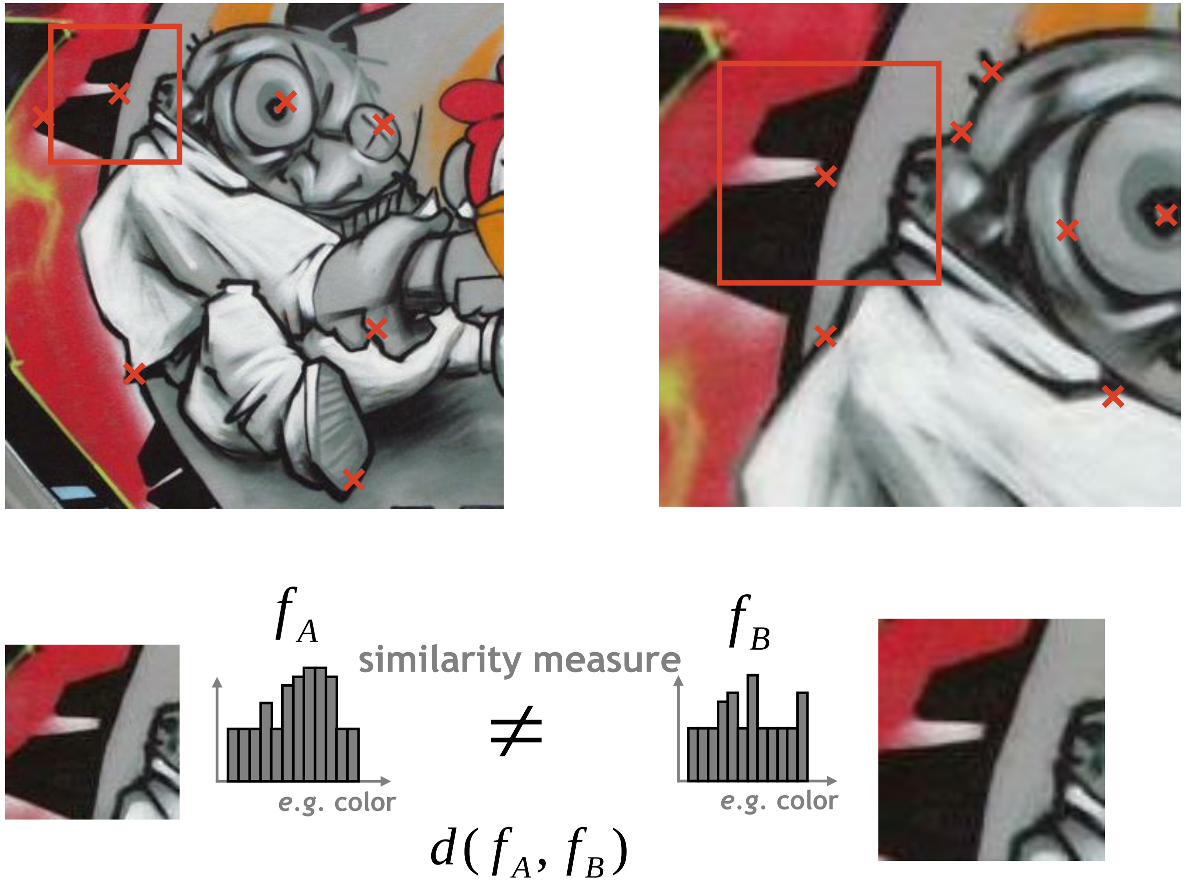
- Check all scales exhaustively
  - Vary the region size and compare the descriptors...





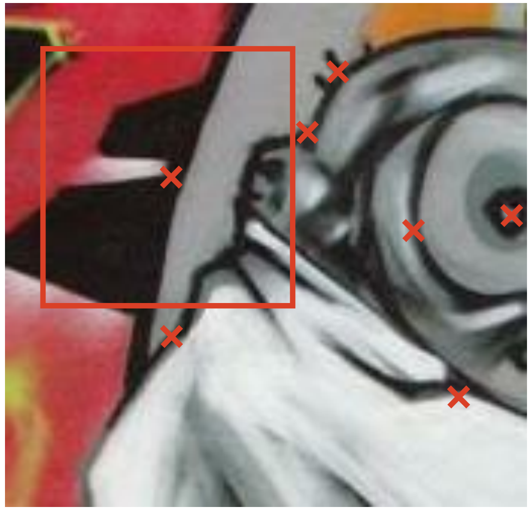
Naive approach: check all scales

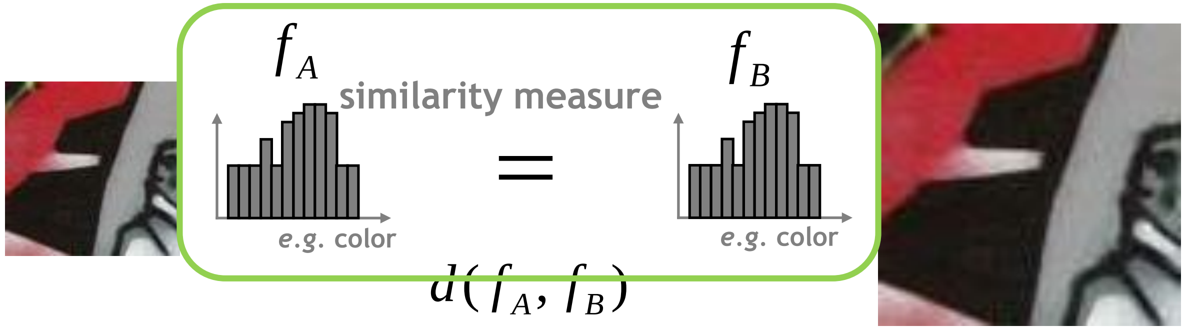
- Check all scales exhaustively
  - Vary the region size and compare the descriptors...



Naive approach: check all scales

- Check all scales exhaustively
  - Vary the region size and compare the descriptors...
  - Very inefficient!
- Need to identify the scale at each point independently from the other images



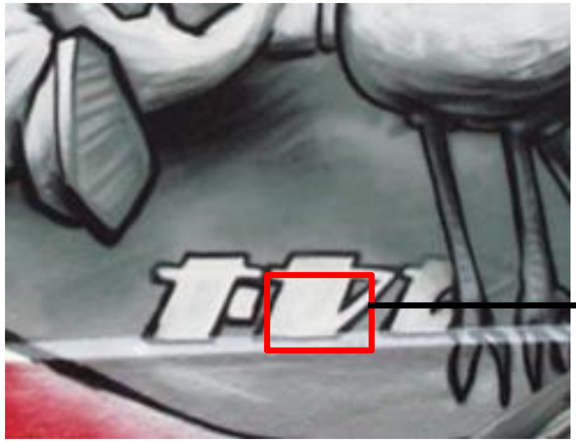


Automatic scale selection

- Solution: construct a scale invariant function on a selected region
    - Outputs the same value for regions with the same content, even if the regions are located at different scales.

Example: *Average intensity of the gray-scale region.*
Even if two corresponding regions are at different scales, we will get the same output.

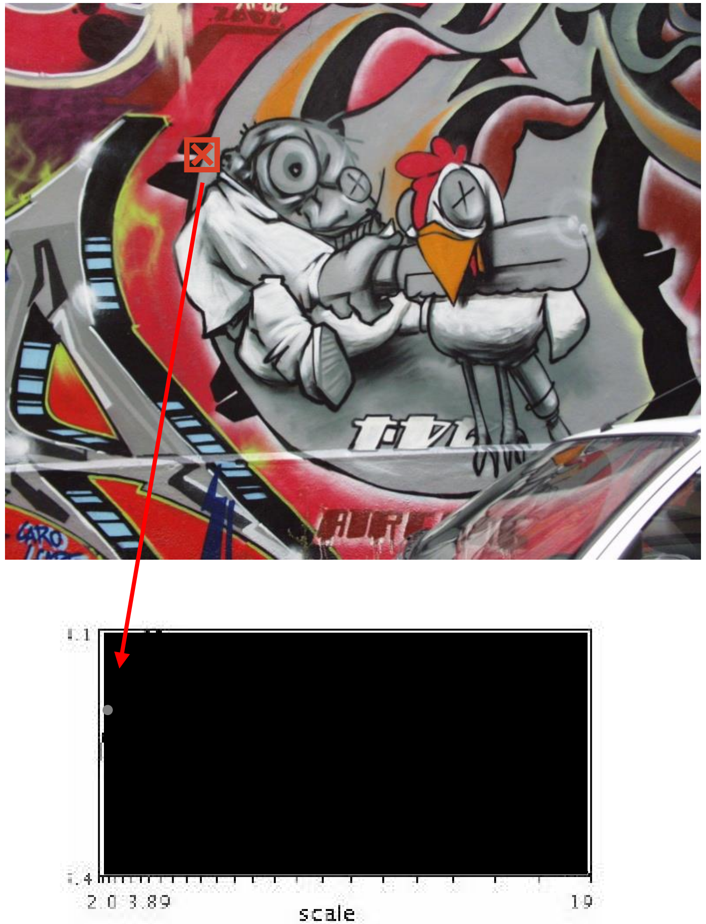
241.3



241.3

Automatic scale selection

- Function responses to different scales (scale signatures)

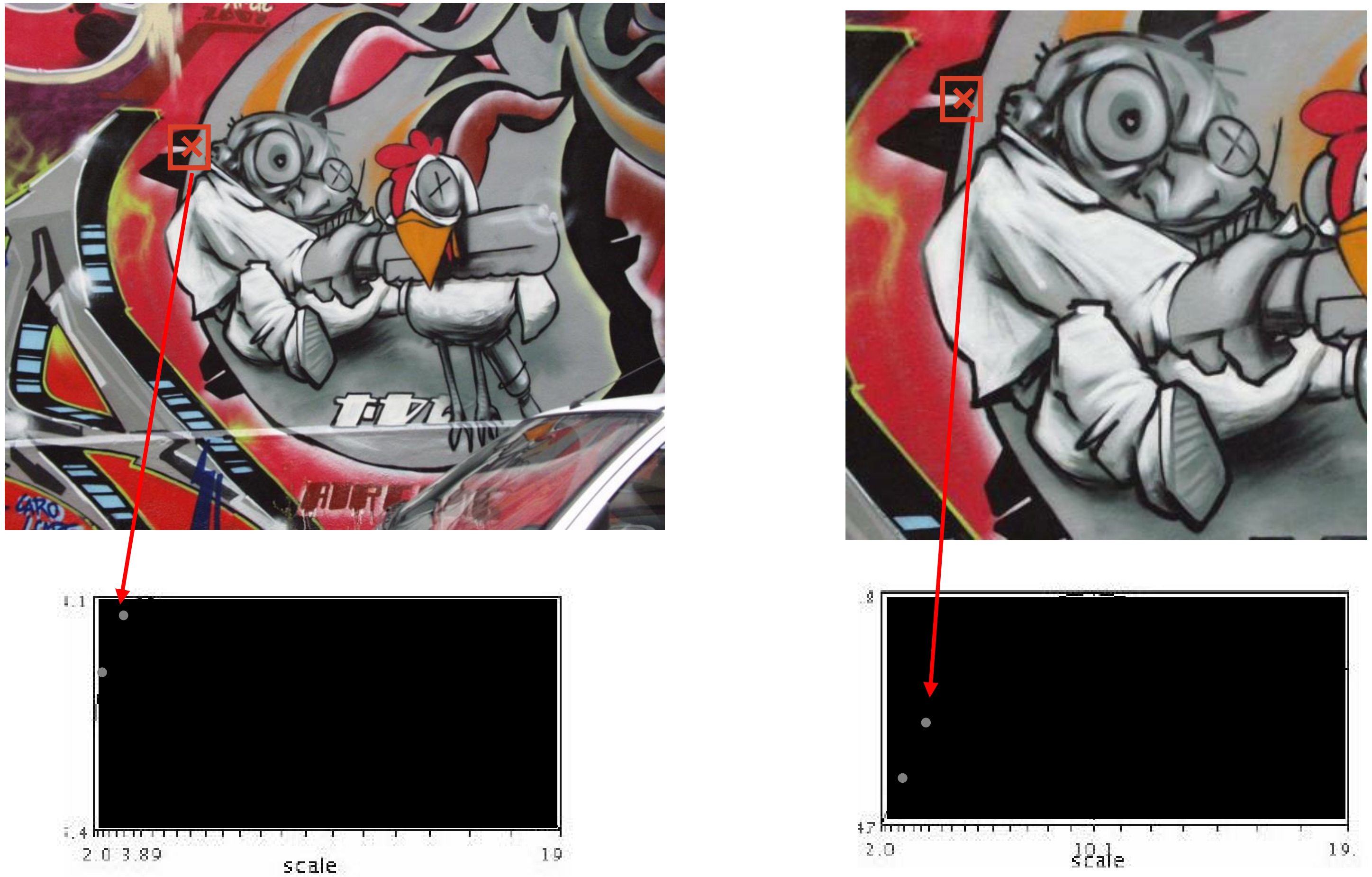


$f(I_{i_1 \ldots i_m}(x,\sigma))$

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

Automatic scale selection

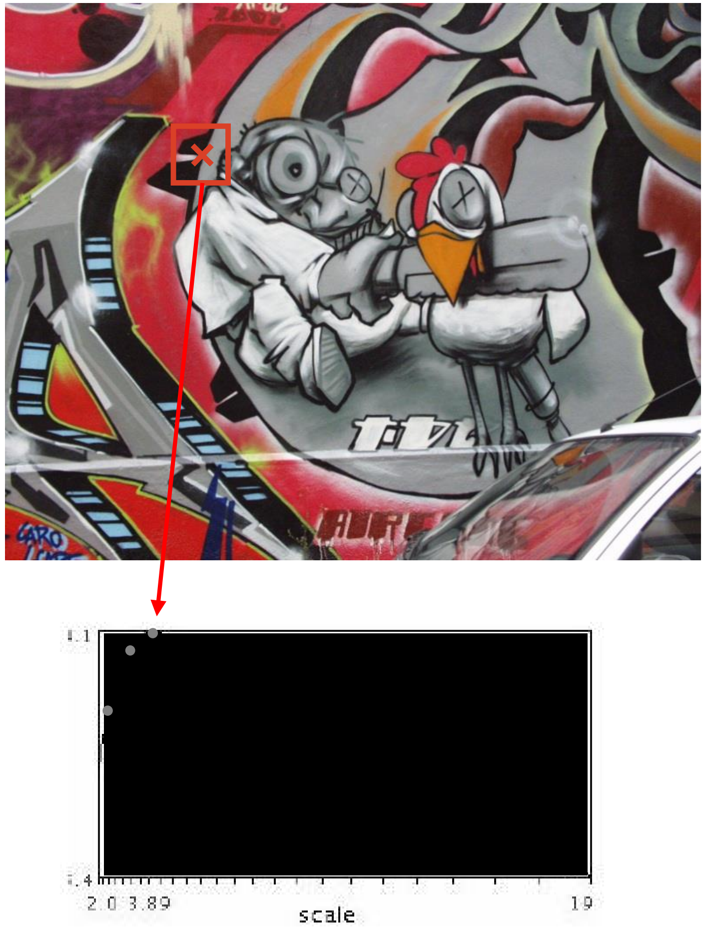
- Function responses to different scales (scale signatures)



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

Automatic scale selection

- Function responses to different scales (scale signatures)

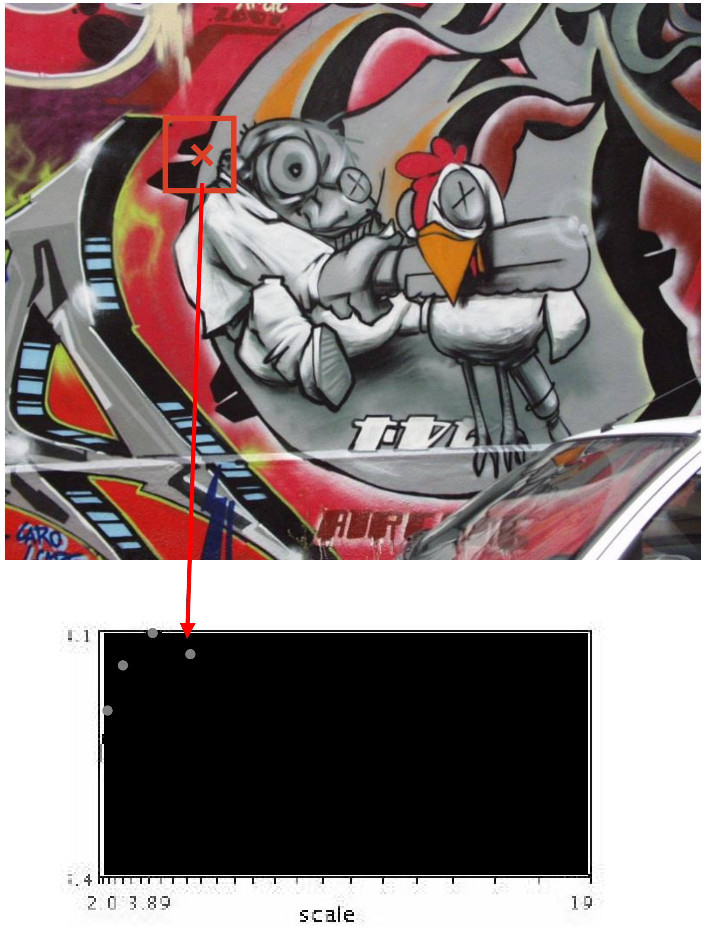


$f(I_{i_1 \ldots i_m}(x, \sigma))$

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

Automatic scale selection

- Function responses to different scales (scale signatures)

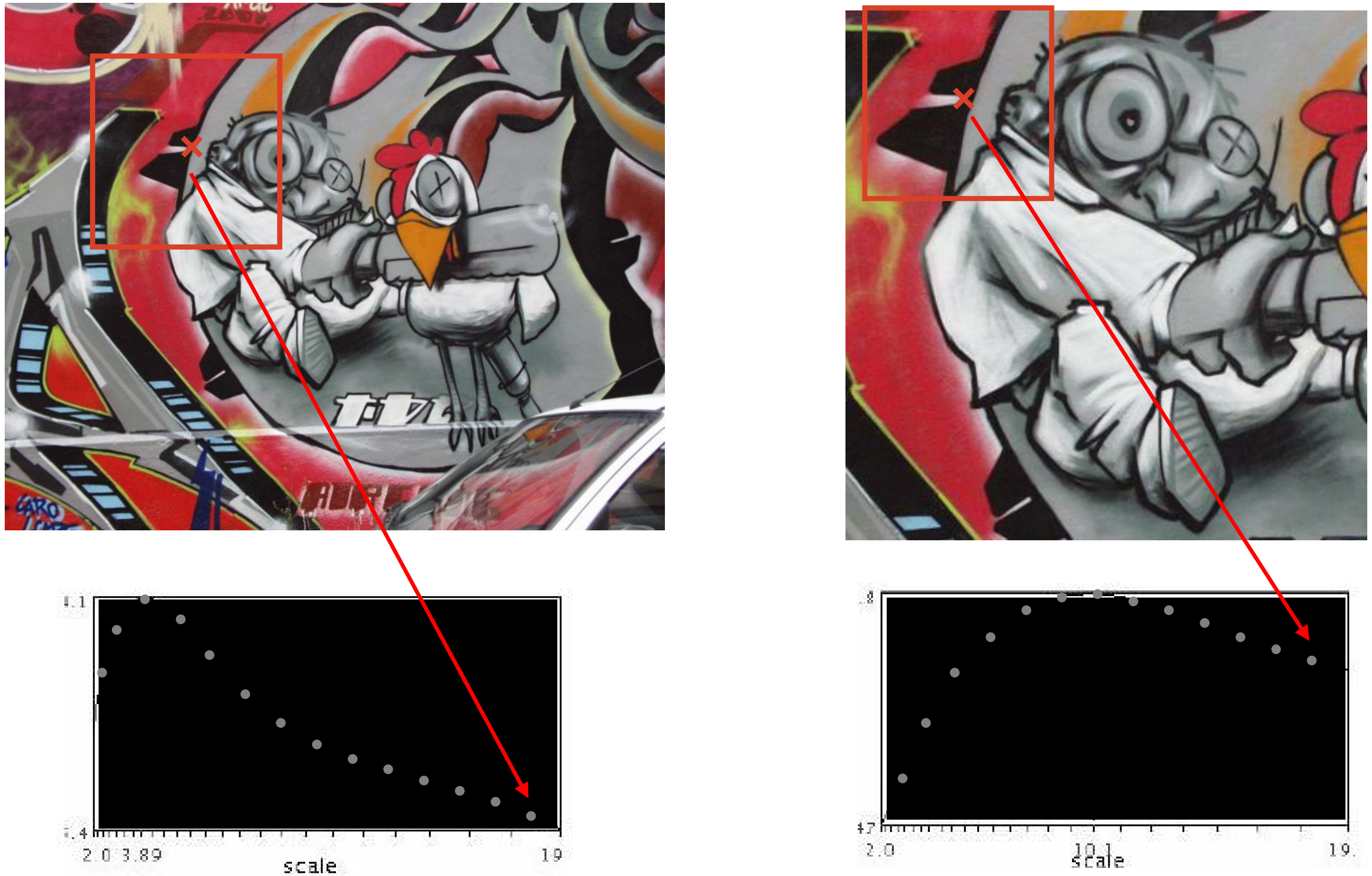


$f(I_{i_1 \ldots i_m}(x, \sigma))$

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

Automatic scale selection

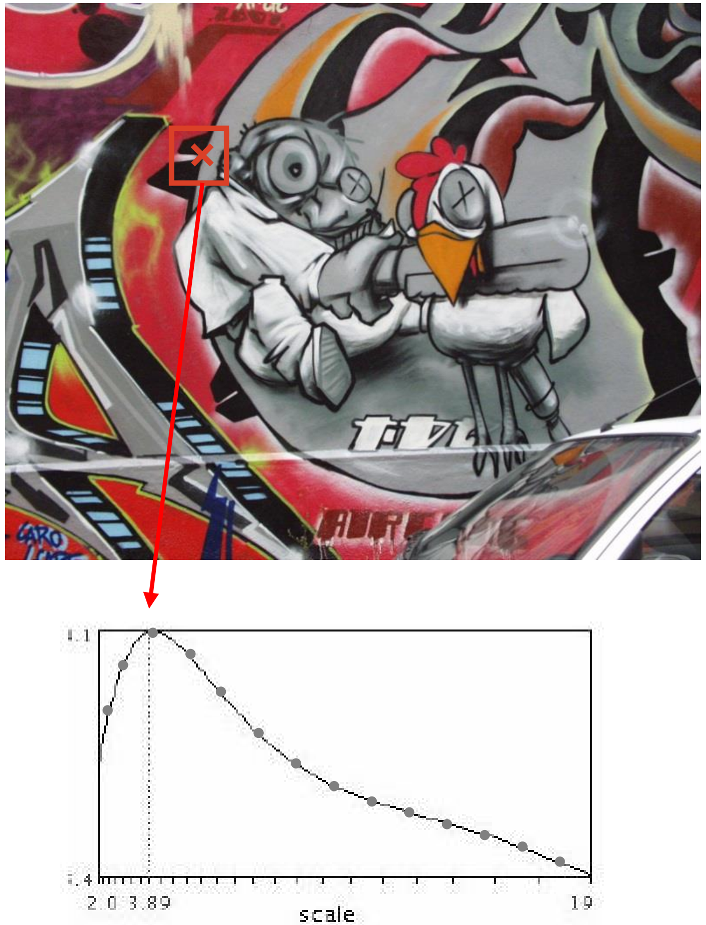
- Function responses to different scales (scale signatures)



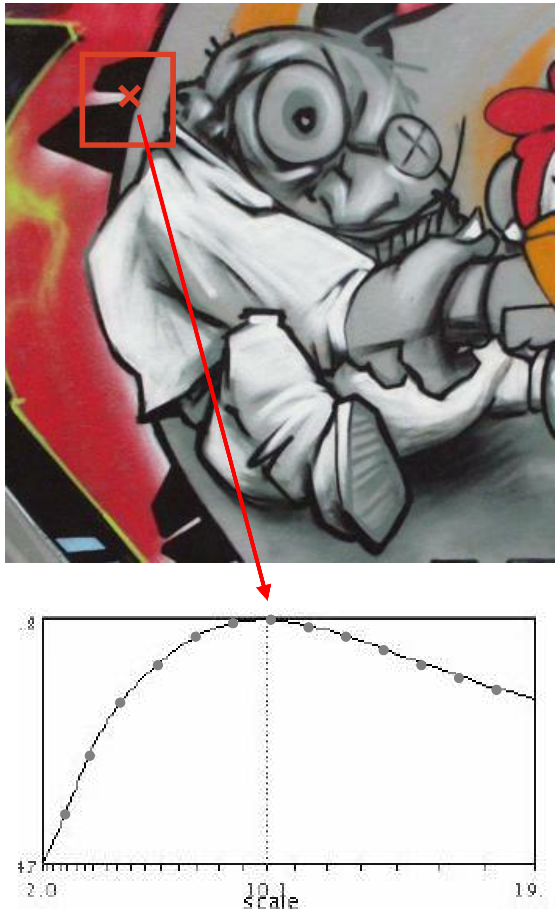
$f(I_{i_1 \ldots i_m}(x, \sigma))$ $\qquad$ $f(I_{i_1 \ldots i_m}(x', \sigma))$

Automatic scale selection

- Function responses to different scales (scale signatures)



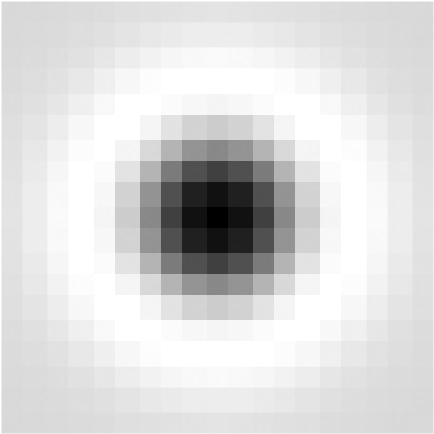
$f(I_{i_1 \dots i_m}(x, \sigma))$

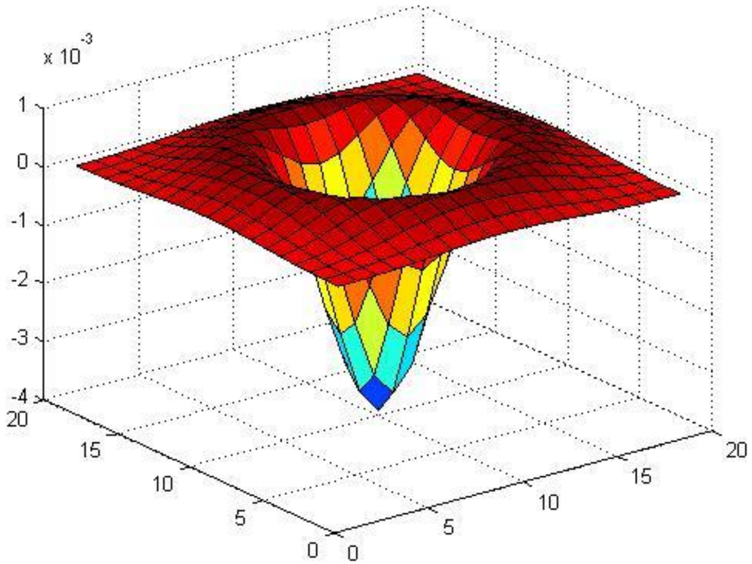


$f(I_{i_1 \dots i_m}(x', \sigma'))$

What is a useful keypoint signature function?

- Natural images abundantly contain blob-like features
- Blob detection – find regions that locally look like “spots”
- Laplacian of Gaussian (LoG):
  Circular-symmetric operator for blob detection...



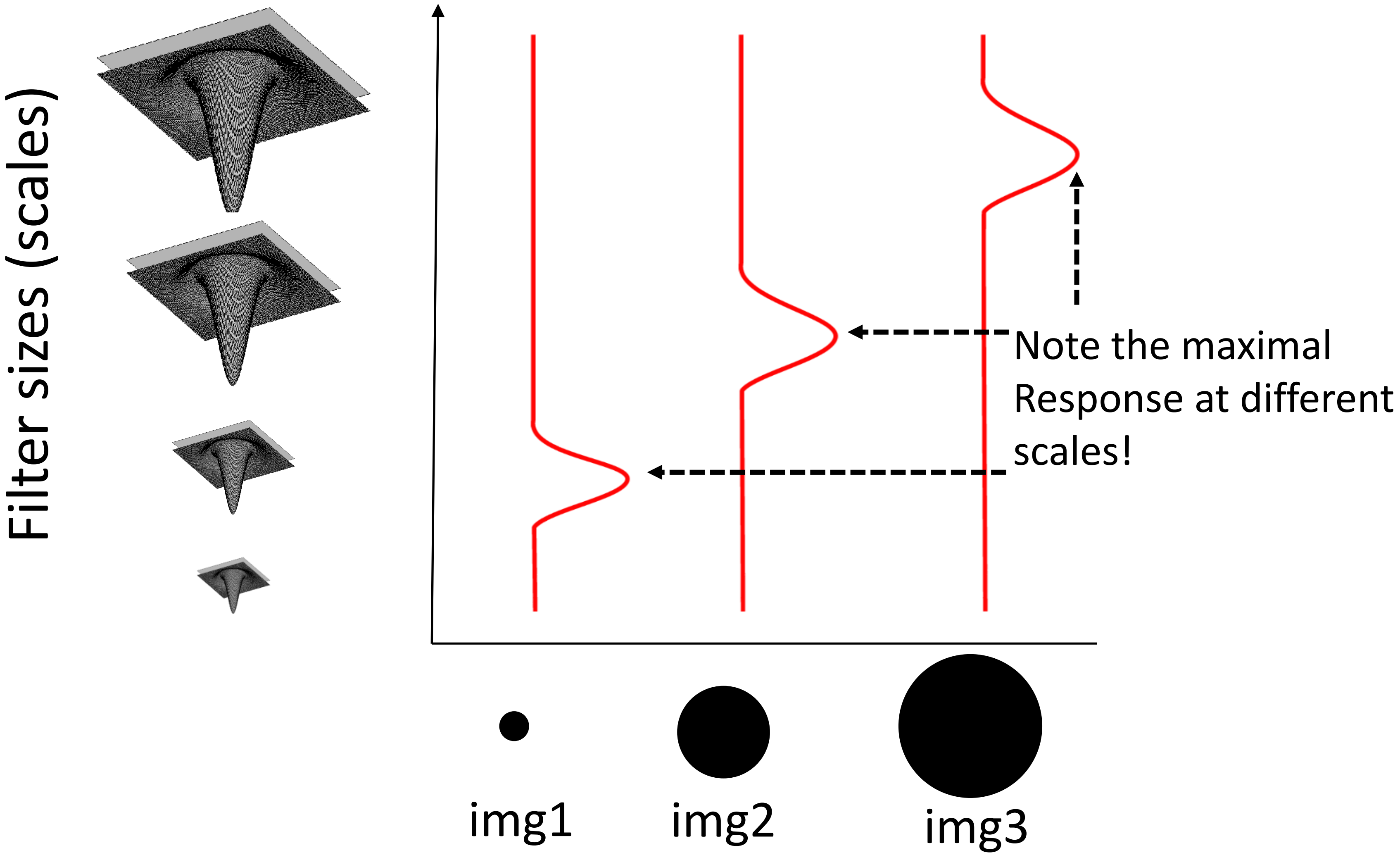


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Blob size matching == scale selection

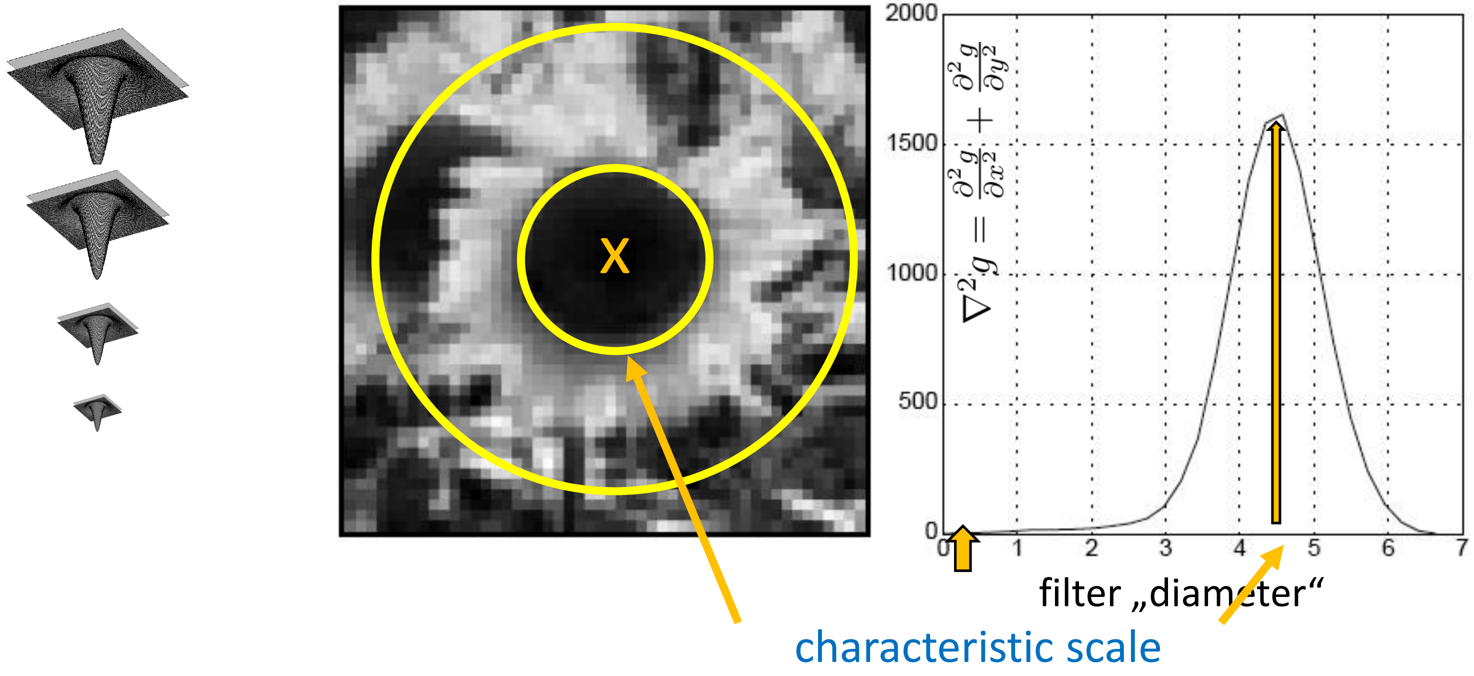
- Laplacian of Gaussian = blob detector

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



Detecting characteristic scale

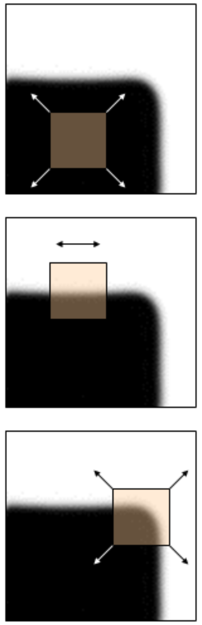
- The characteristic scale is the scale at which the LoG filter yields a maximum response.

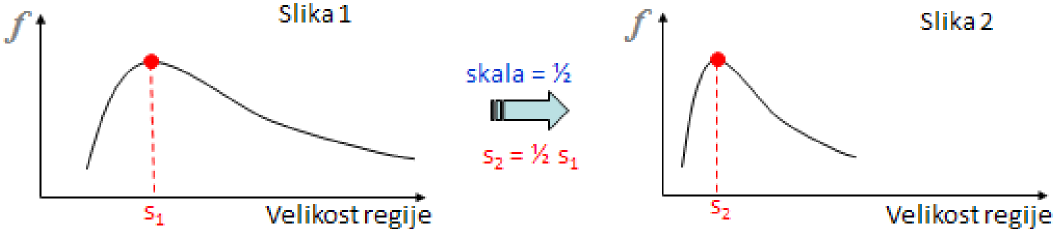


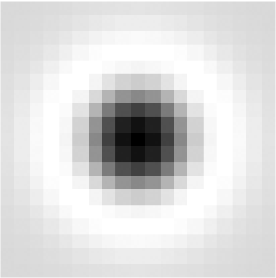
T. Lindeberg "Feature detection with automatic scale selection." *International Journal of Computer Vision* 30 (2), 1998.

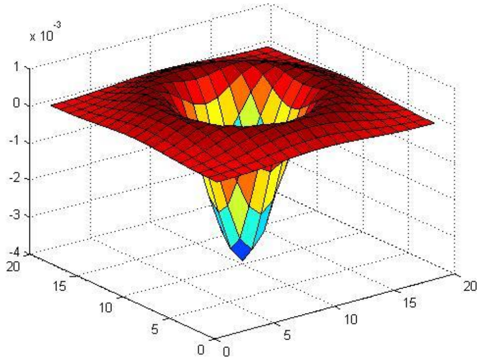
Previously at MP...

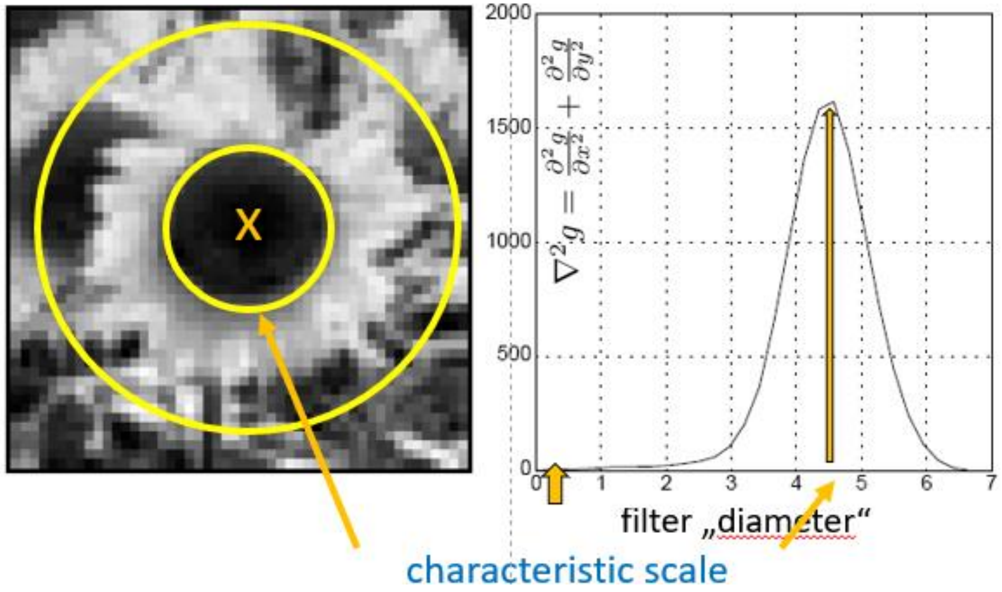
- Key-point detection
  - Analysis of gradient distribution
  - Harris, Hessian
- Scale (region size) selection





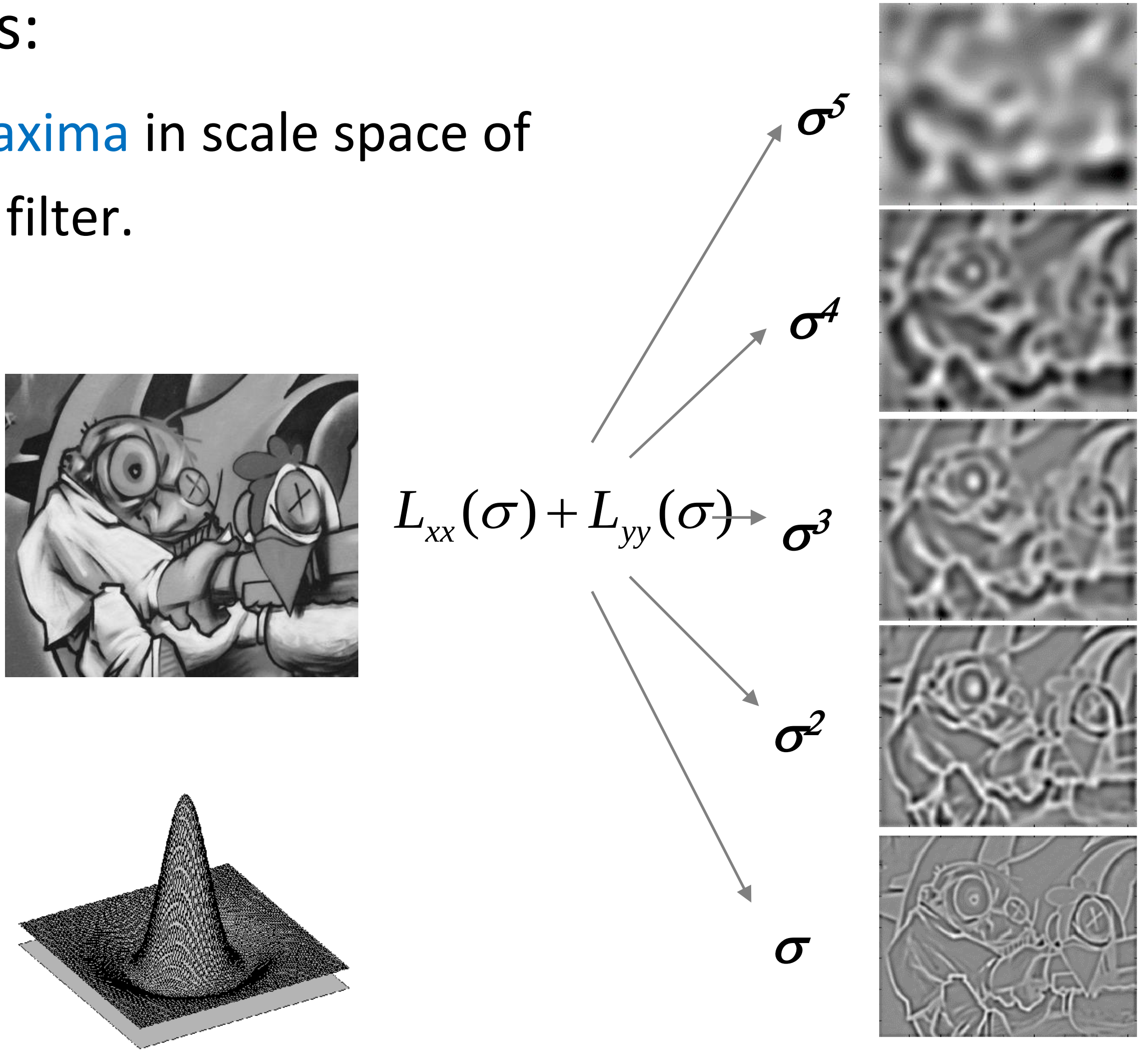






Laplacian pyramid implementation

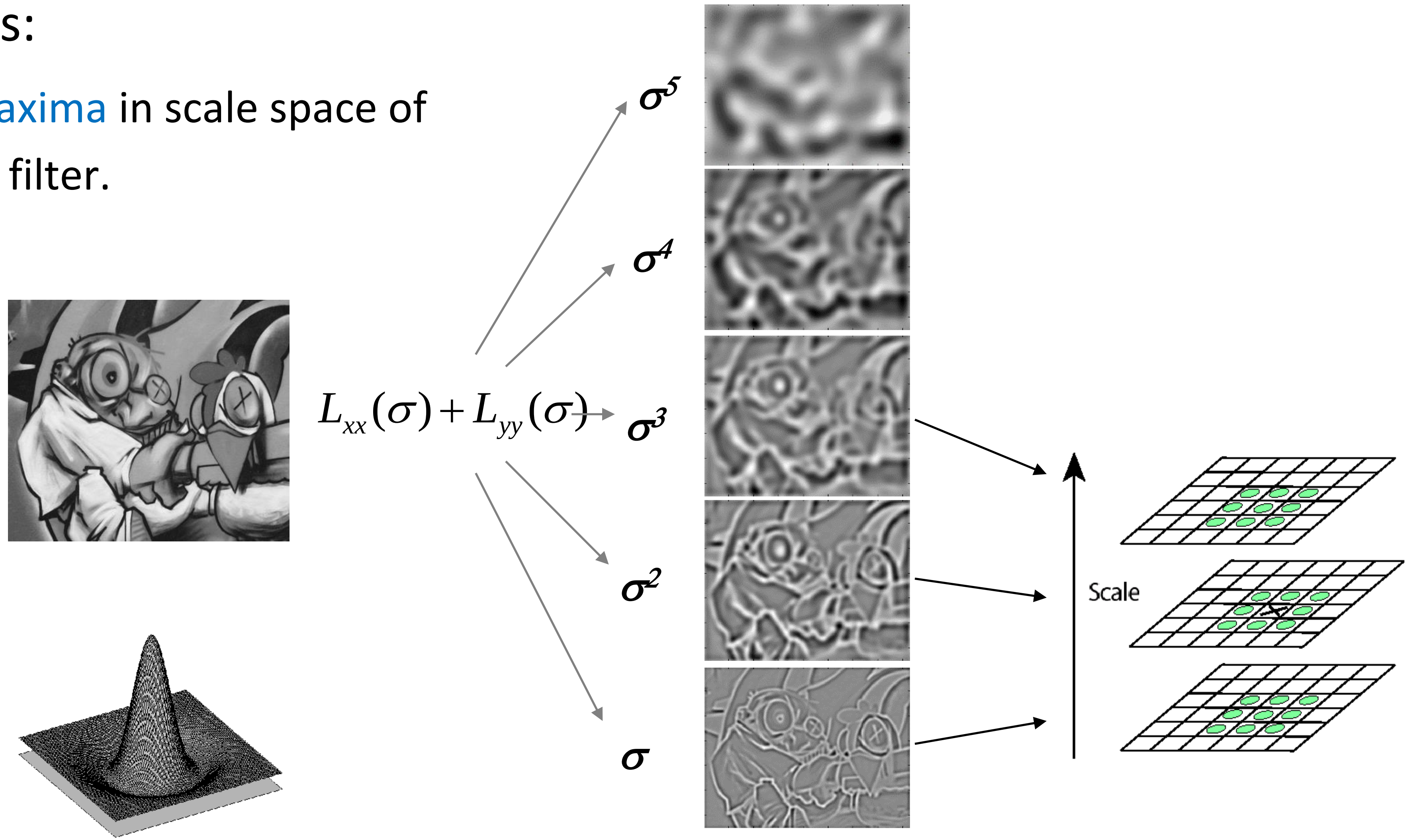
- Key-points:
  - Local maxima in scale space of the LoG filter.



Laplacian pyramid!

Laplacian pyramid implementation

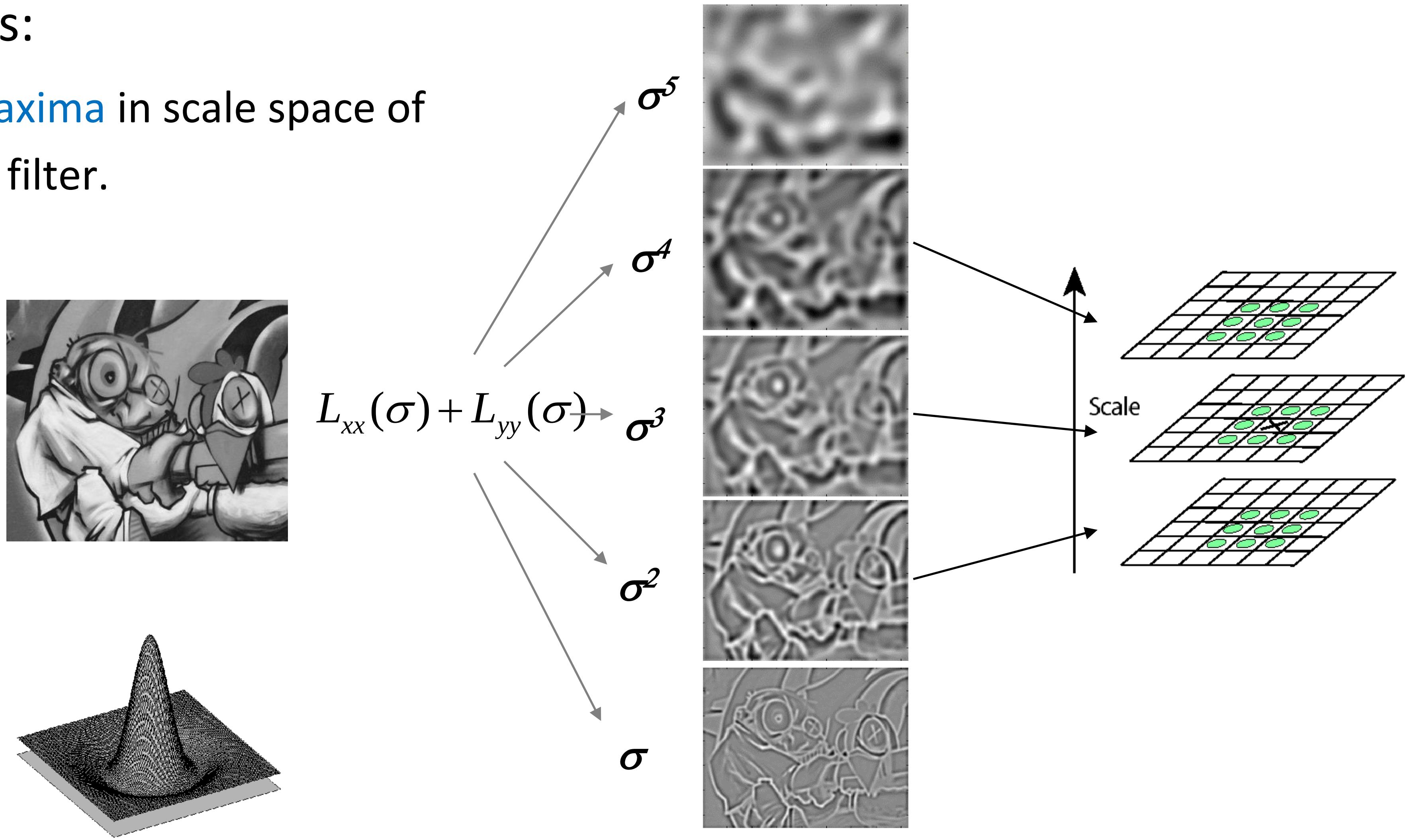
- Key-points:
  - Local maxima in scale space of the LoG filter.



Compare LoG at each point to its 8+9 × 2 neighbors (same scale + upper/lower scale.)

Laplacian pyramid implementation

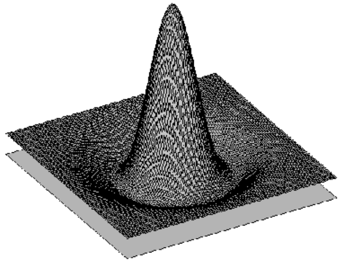
- Key-points:
  - Local maxima in scale space of the LoG filter.



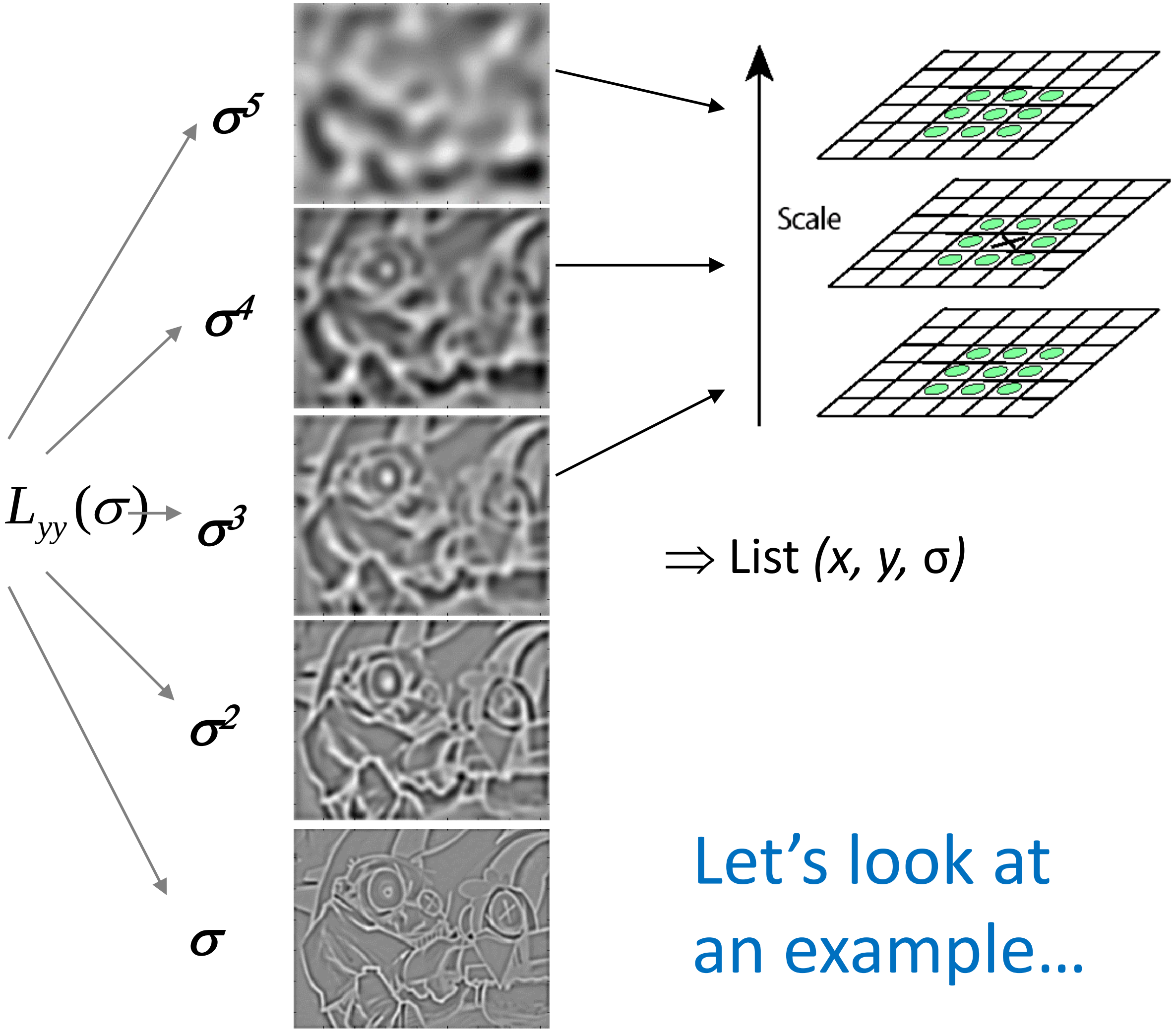
Compare LoG at each point to its 8+9 neighbors (same scale + upper/lower scale.)

Laplacian pyramid implementation

- Key-points:
  - Local maxima in scale space of the LoG filter.



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



$\Rightarrow$ List *(x, y, σ)*

Let's look at an example…

Compare LoG at each point to its 8+9 neighbors (same scale + upper/lower scale.)

LoG detector in action

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

Input image

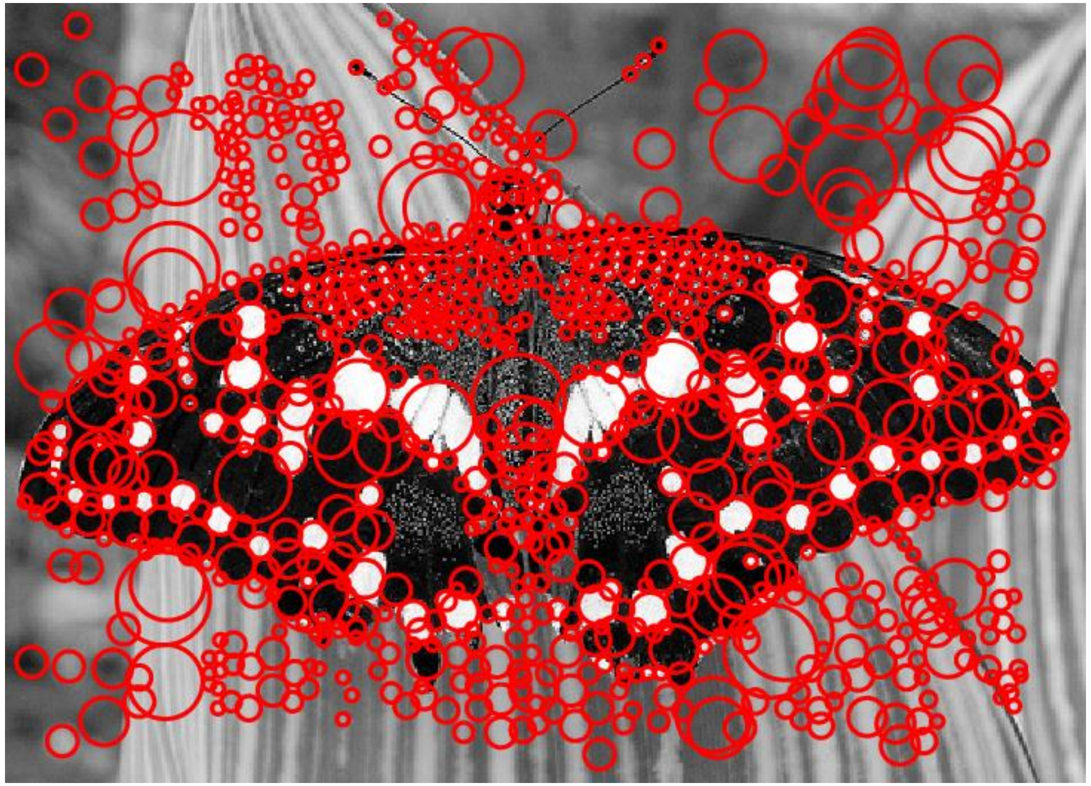
LoG detector in action

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

sigma = 11.9912

LoG filtered image (varying sigma)

LoG detector in action



Local maxima across scales

LoG approximation by difference of Gaussians

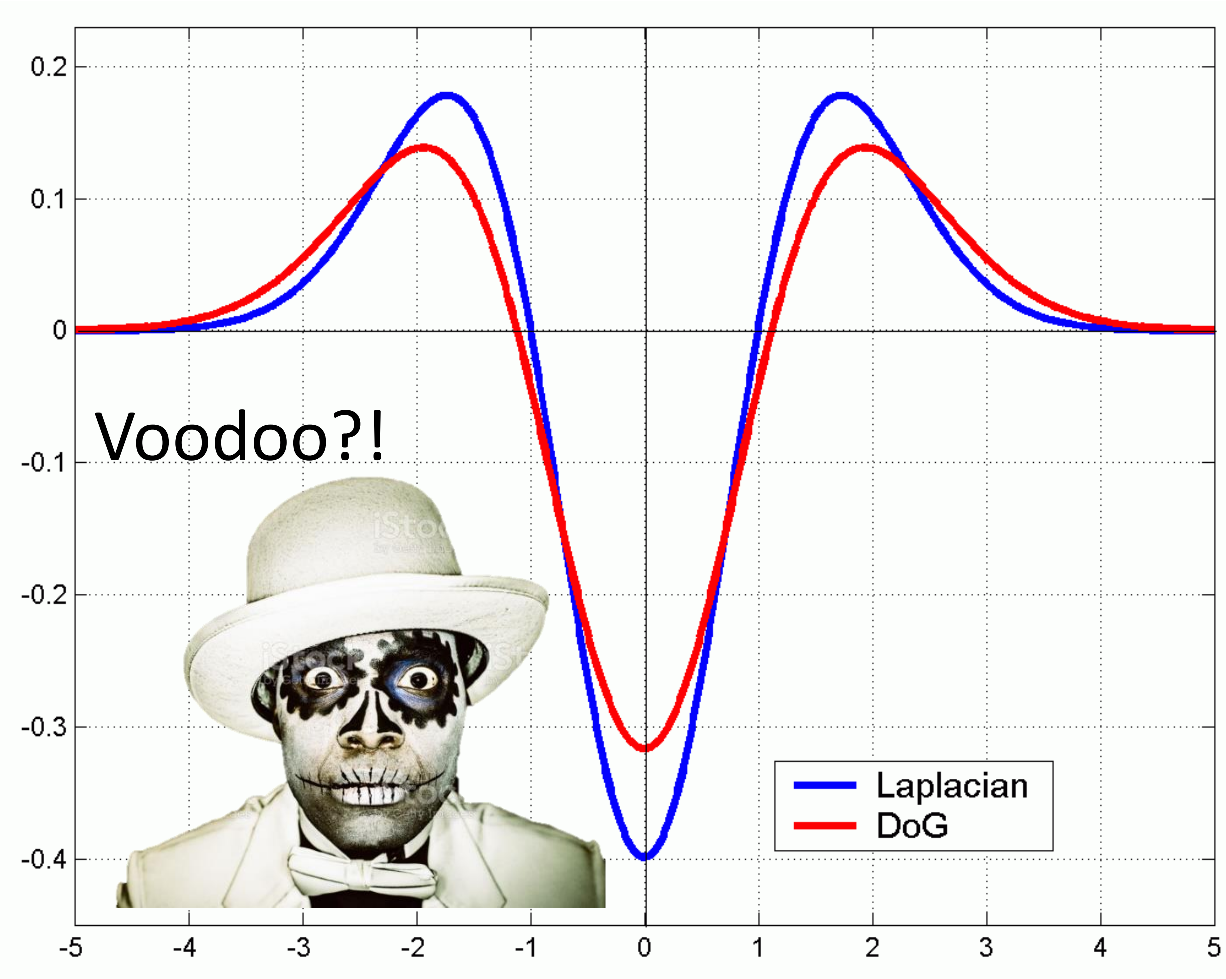
- The LoG can be well approximated with a difference of Gaussians at different values of σ.

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

(normalized Laplacian of Gaussian)

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

(Difference of Gaussians)

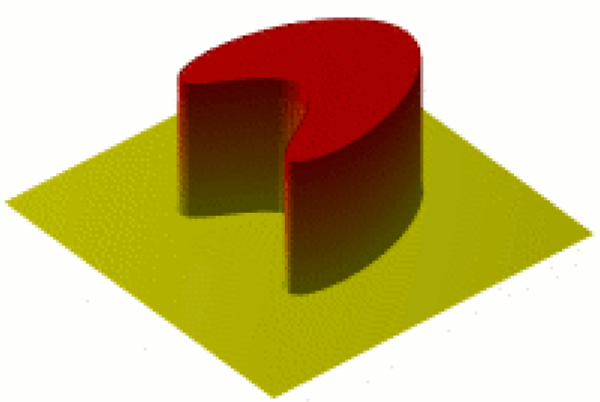


Not Voodoo – Technical details

- Let *u*(*x*,*y*,*t*) be a density of diffusion material (eg., heat) at location (*x*,*y*) at time *t*.
- Then this holds:

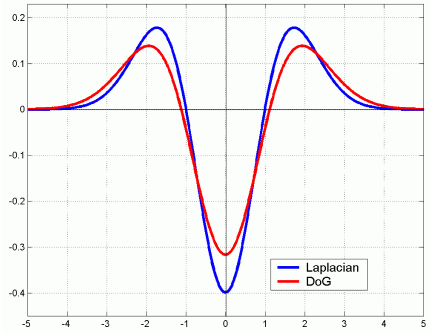
$$\frac{\partial u}{\partial t} - \alpha\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0$$

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u$$



- Instead *u* take a Gaussian *g*, instead *t*, use its variance σ^2.
- Using the finite differences we get the following approximation:

$$\sigma \nabla^2 g = \frac{\partial g}{\partial \sigma} \approx \frac{g(x,y,k\sigma) - g(x,y,\sigma)}{k\sigma - \sigma}$$



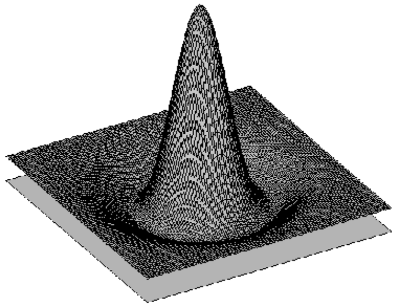
- Which yields:

$$(k-1)\sigma^2 \nabla^2 g \approx g(x,y,k\sigma) - g(x,y,\sigma)$$

Difference of Gaussians (DoG)

- Difference of Gaussians is an approximation of the LoG

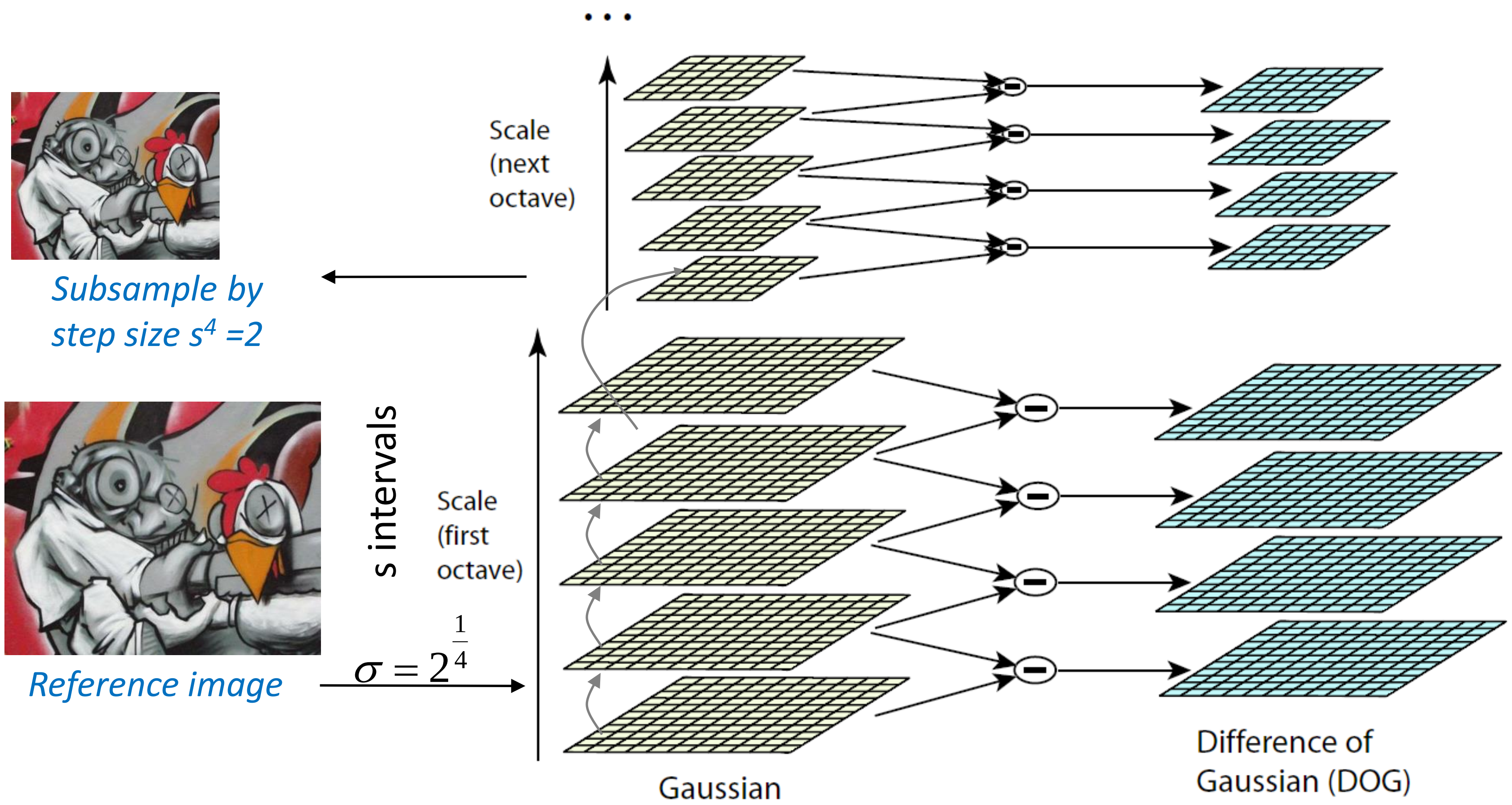
- Advantages
  - Does not require computation of second derivatives
  - Results of Gaussian filtering already calculated during calculation of image resizing (Gaussian Pyramid!).



How to efficiently localize the key-points using DoG?

DoG pyramid– Efficient calculation

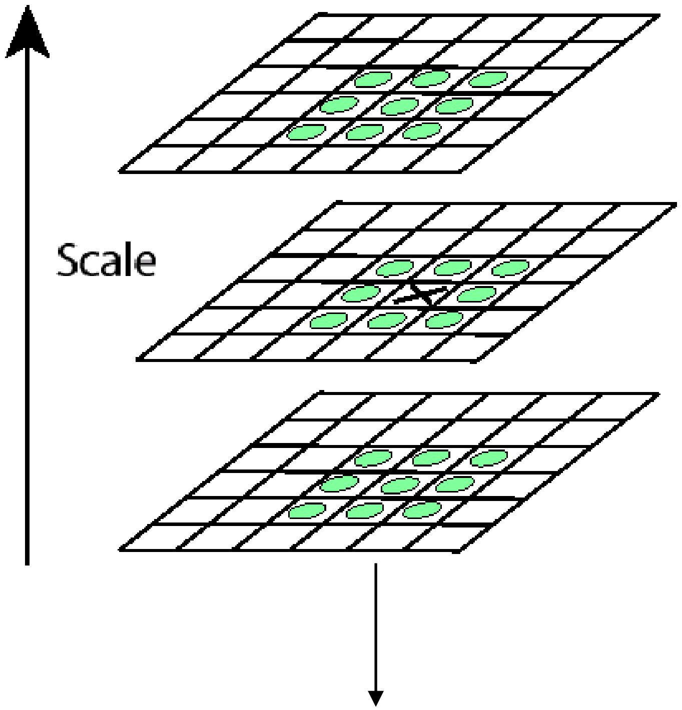
- Calculated from a Gaussian pyramid (sequential octaves equivalent to filtering with $\sigma_{next} = 2\sigma_{prev}$)



David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* 60, 2 (2004), pp. 91-110

Key-point localization using DoG

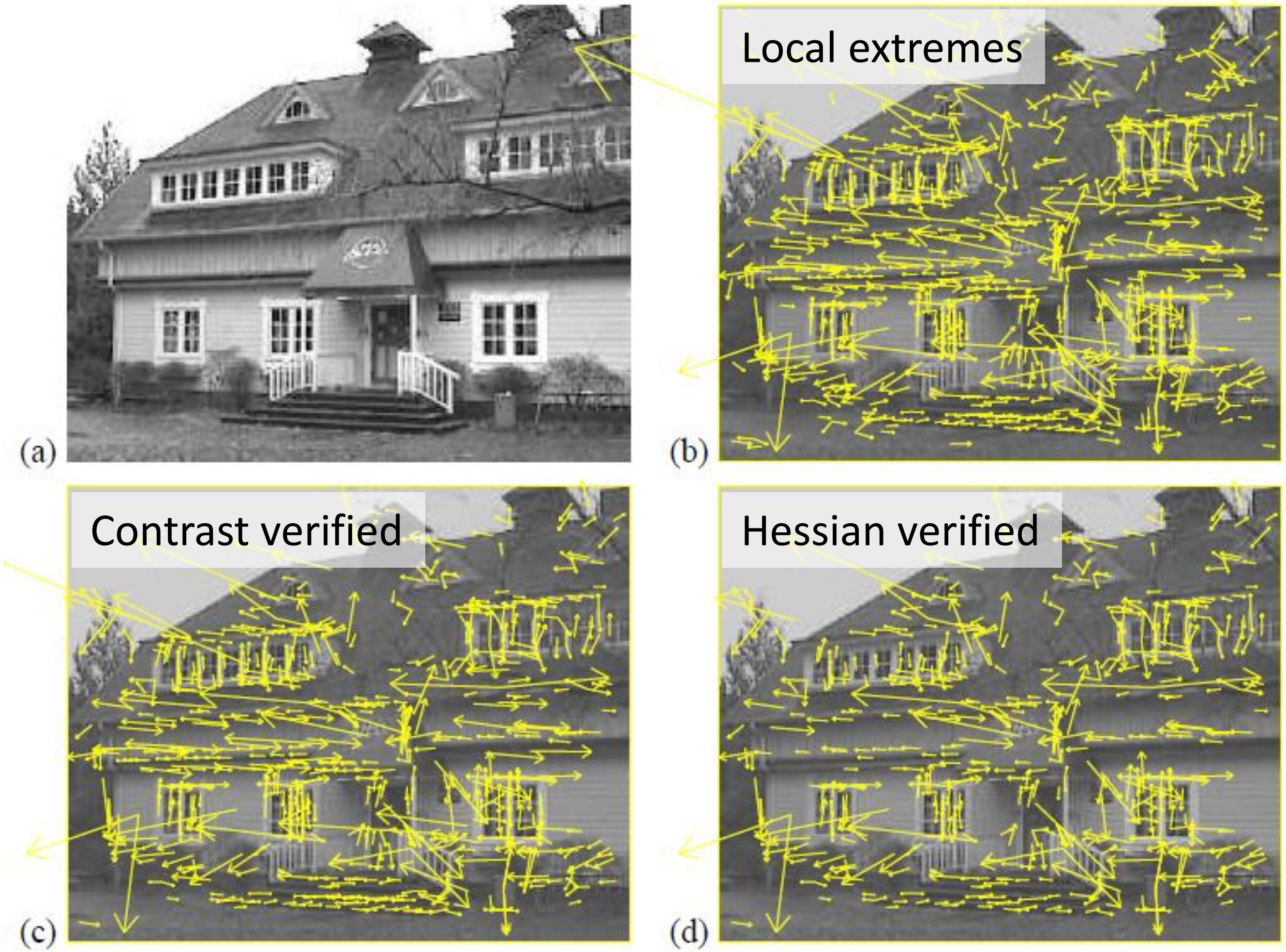
- Find local maxima of DoG in the scale-space.
  - Check 8+2*9=26 neighbors
- Remove the low contrast points (*threshold dependent*)
  - If local change in response is small compared to neighbors.
- Remove points detected at the edges
  - Test using the Hessian matrix.



Key-point candidates:
List of triplets (x,y,σ)

Fit a quadratic function to each key-point and its neighbors to improve localization of the maxima (x,y ,σ).

Results: Lowe's DoG-based detector



(a) 233x189 image

(b) 832 extremes in DoG

(c) 729 remain after contrast verification

(d) 536 remain after verification of the Hessian matrices.

David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* 2004

Results: Lowe's DoG-based detector

Summary: scale-invariant key-point detection

- Input: Image of some scene taken at unknown scale.
- Goal: Find stable key-points *independently* in each image.
- Solution:
  Find *maxima* of specialized functions in scale-space and image coordinates.
- Two strategies
  - Laplacian of Gaussian (LoG)
  - Difference of Gaussians (DoG) as an efficient approximation

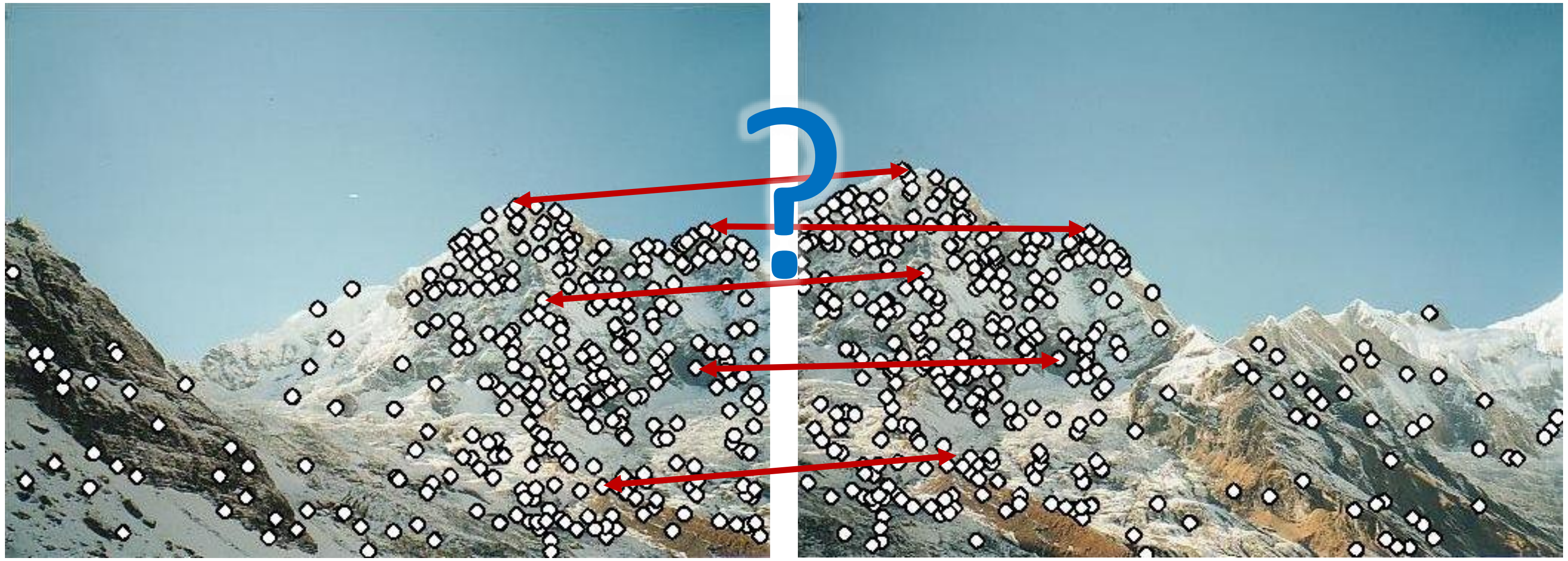
Machine Perception

LOCAL DESCRIPTORS

Local descriptors

- Now we know how to detect the key-points
- Next question:

How to describe them?

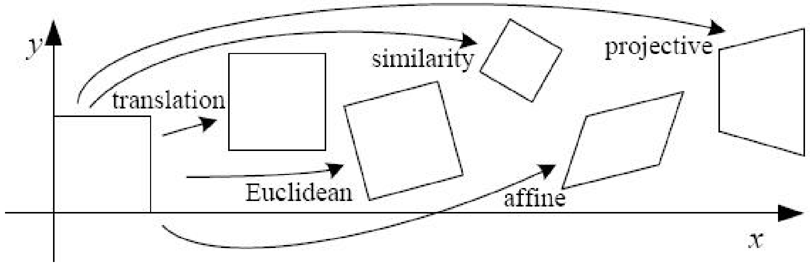


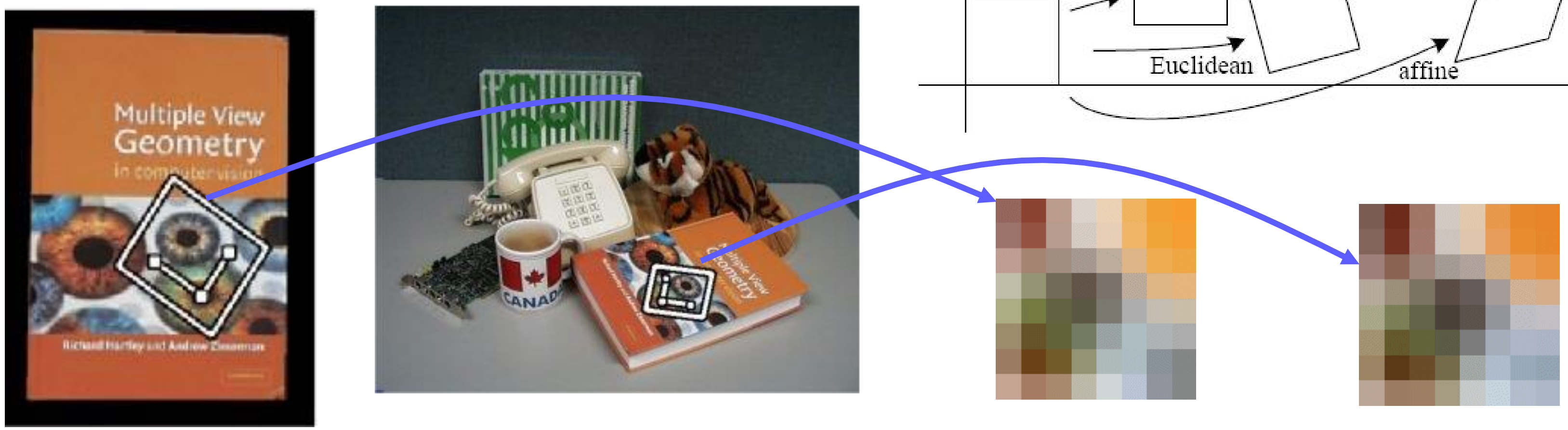
Key-point descriptors should be:

1. Distinctive (be different for keypoints on different structures)
2. Invariant to ambiental changes

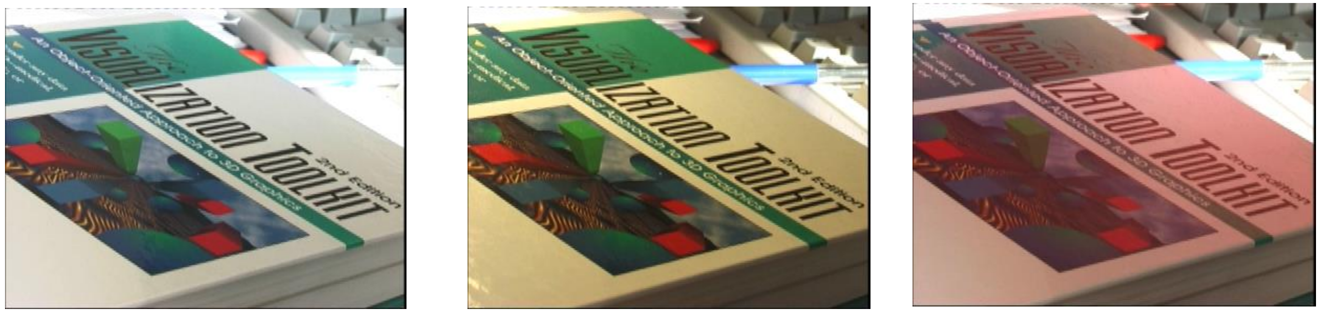
Invariance of descriptor

- Geometric transformations





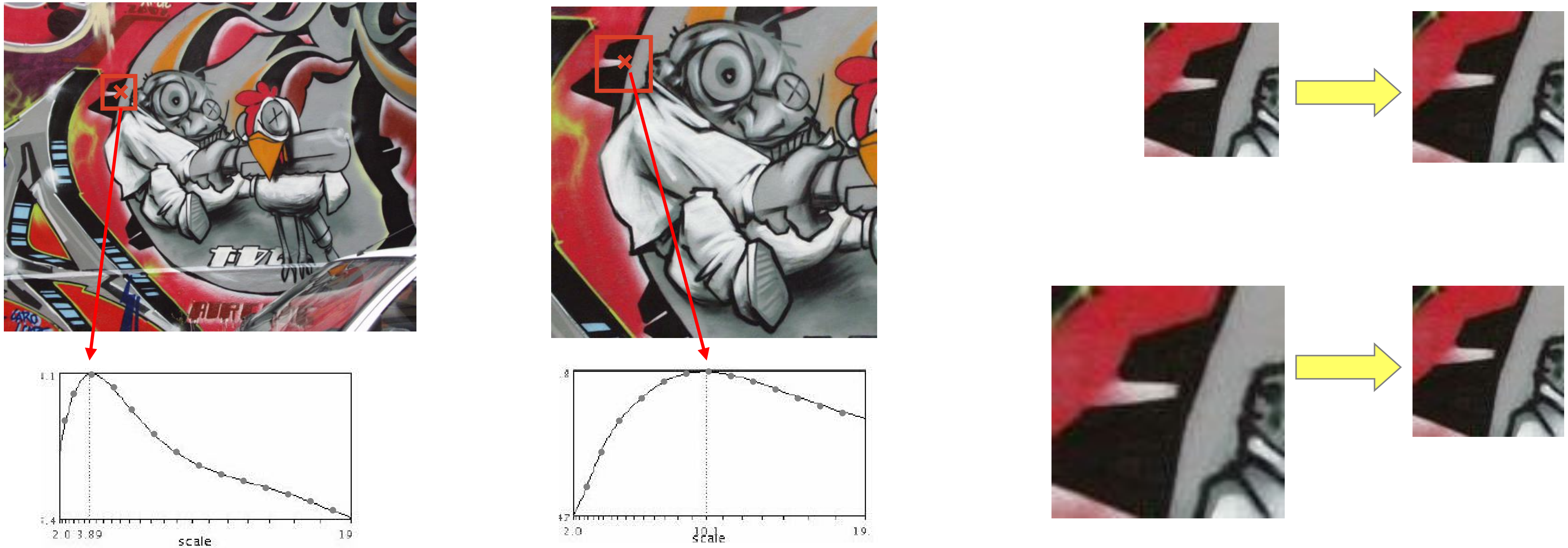
- Photometric transformations
  - Often modeled by
    intensity scaling and translation



Scale invariant detection (already covered)

- For comparing regions, normalize: Rescale to a predefined size

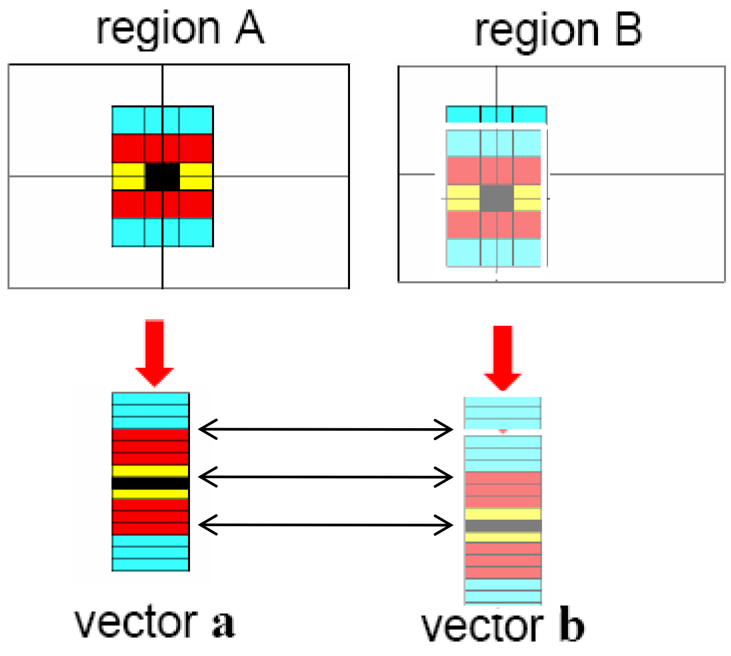
Important: the region location and size (scale) is determined independently in each image for each key-point!



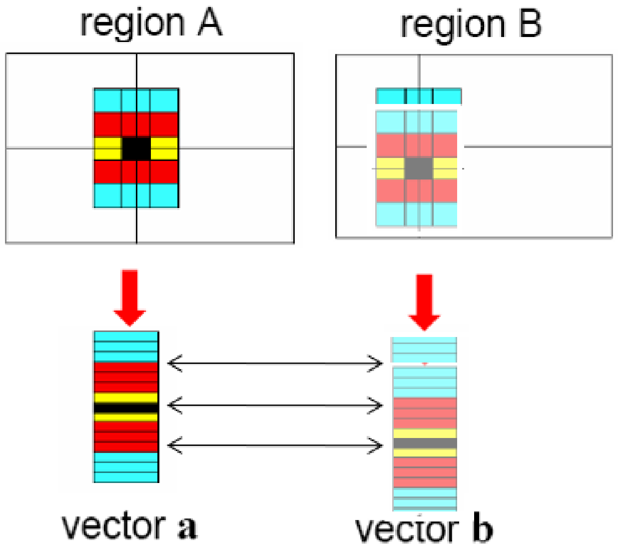
$f(I_{i_1 \ldots i_m}(x, \sigma))$ $f(I_{i_1 \ldots i_m}(x', \sigma'))$

Local descriptors

- The simplest descriptor: a vector of region intensities.
- Analyze the invariance of such descriptor…
- Small shifts may cause a large change in the descriptor.
- Sensitive to photometric changes.

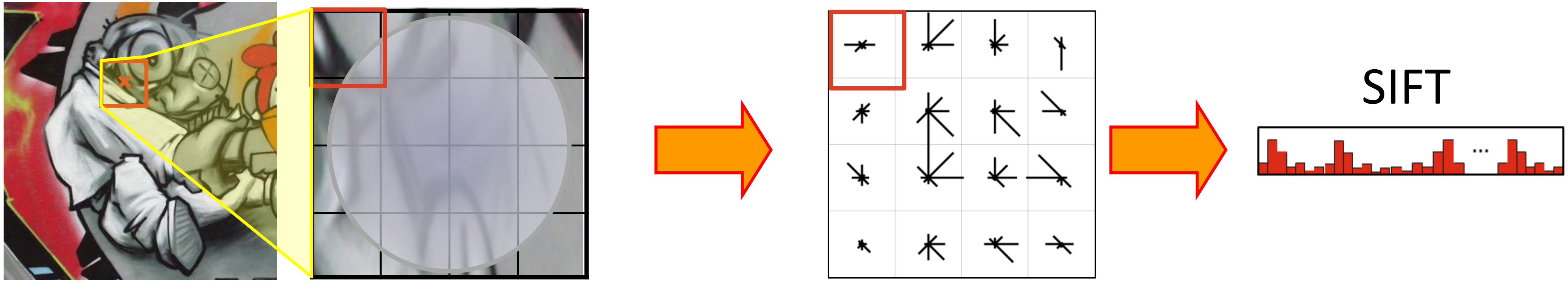


Invariances



Descriptor: SIFT

- Scale Invariant Feature Transform:
  - Split region into 4x4 sub-regions: 16 cells
  - Calculate gradients on each pixel and smooth over a few neighbors.
  - In each cell calculate a histogram of gradient orientations (8 directions)
    - Each point contributes with a weight proportional to its gradient magnitude
    - The contribution is weighted by a Gaussian centered at the region center
  - Descriptor (Stack histograms into a vector and normalize): 4x4x8 = 128 dim



Actually, there are a few important suttle details:

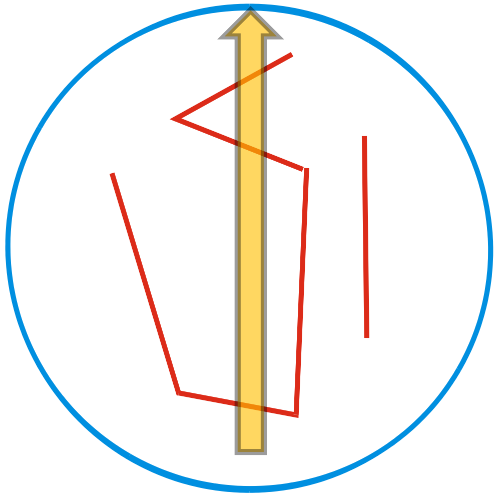
David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Invariance: Orientation normalization

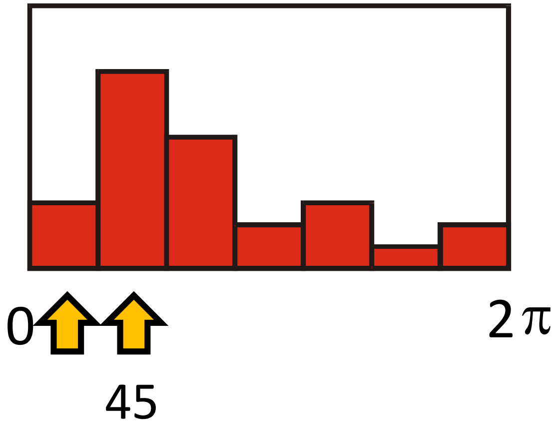
- The SIFT from previous slide is not rotation-invariant.
- Calculate the histogram of orientations
  - 36 bins by angle, each point contributes proportionally to its gradient magnitude and distance from the center.
- Determine the dominant orientation from histogram
- Normalize: rotate gradients into a rectified orientation

Calculate the SIFT using these rectified gradients.

Some keypoint
Region considered



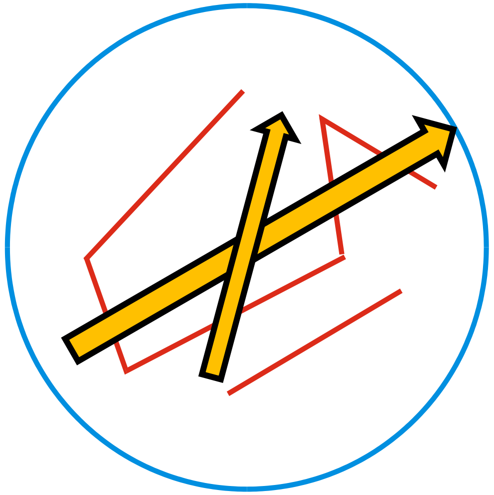
Gradient orientation histogram

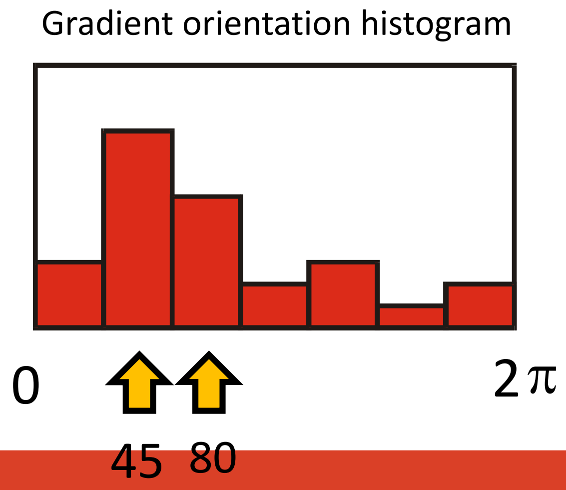


Invariance: Orientation normalization

- The SIFT from previous slide is not rotation-invariant.
- Normalize: rotate gradients into a rectified orientation
- Find all orientations in histogram, whose amplitude is, e.g., 80% of the strongest bin.
- Form a separate SIFT for each detected orientation.

Some keypoint
Region considered





Summary: SIFT

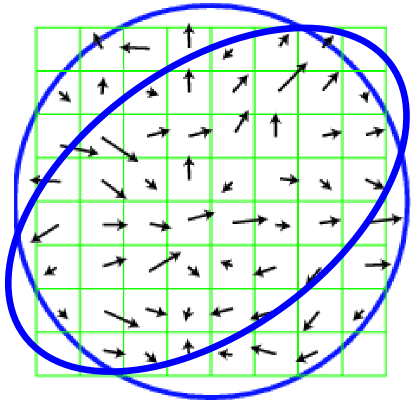
- A surprisingly robust key-point descriptor
  - Allows ~60 degrees of out-of-plane rotation
  - Robust to significant intensity changes
  - Fast (lots of real-time implementations)

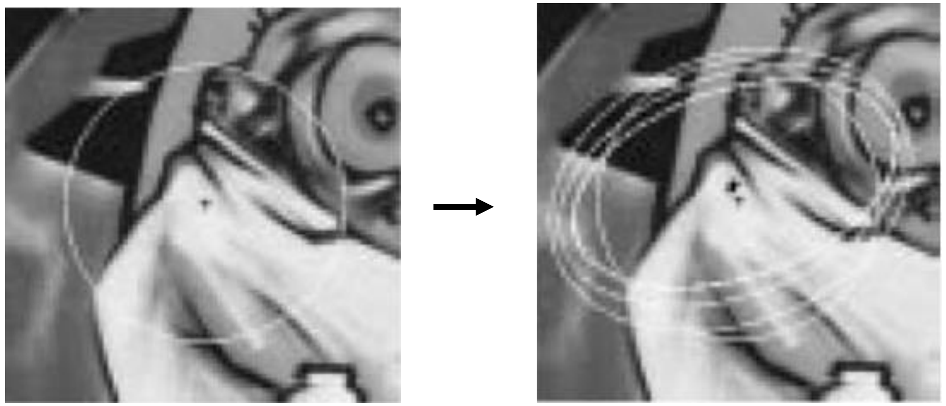
Affine adaptation

- We have addressed invariance to
  - Translation
  - Scale
  - Rotation
- But that’s not enough for very large changes in viewpoint
  - We require an affine adaptation!

Affine adaptation

- Problem:
  - Determine the characteristic shape of local region.
  - Assumption: shape described by an affine local window.
- Solution: iterative approach
  - In circular window calculate a gradient covariance matrix (similar to Harris)
  - Estimate an ellipse from the covariance matrix
  - Using the new window calculate the new covariance matrix and iterate.

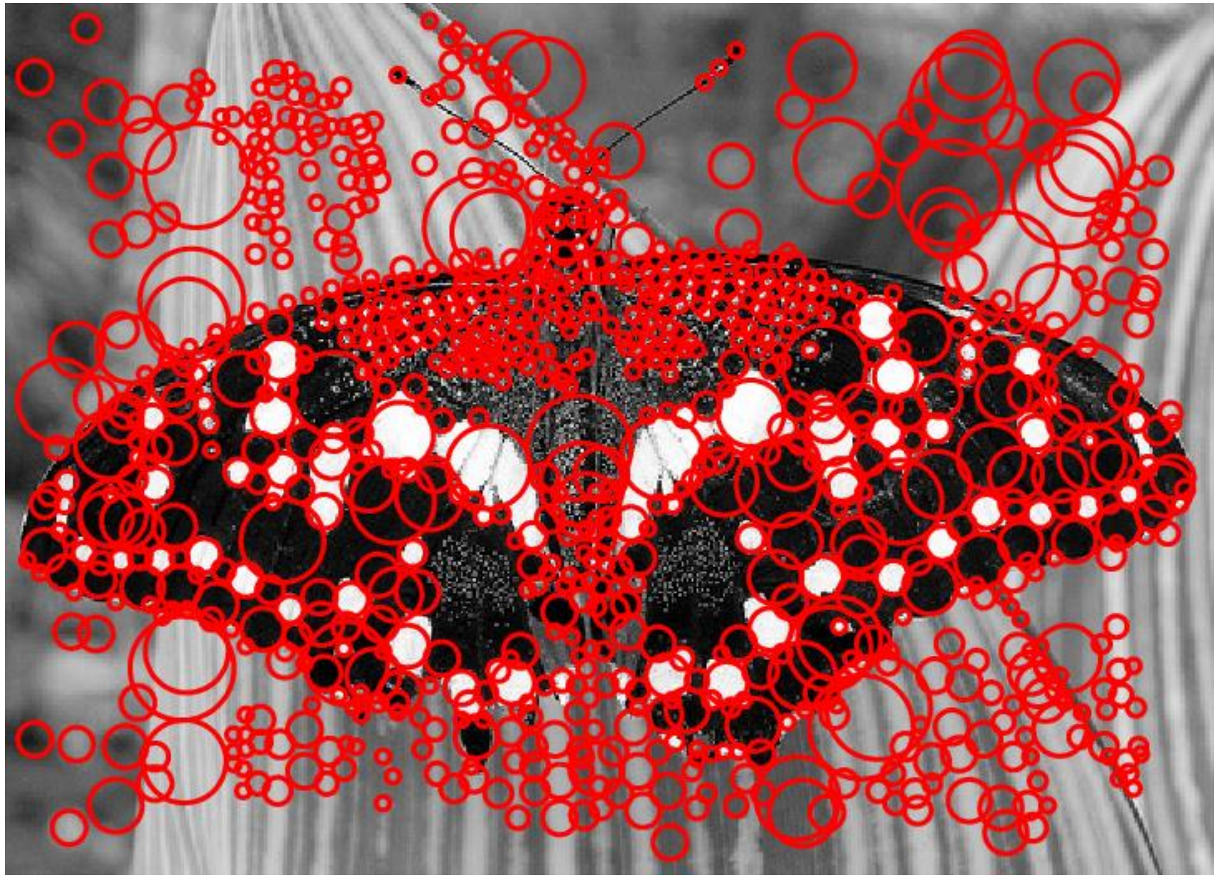




K. Mikolajczyk and C. Schmid, Scale and affine invariant interest point detectors, IJCV 60(1):63-86, 2004.

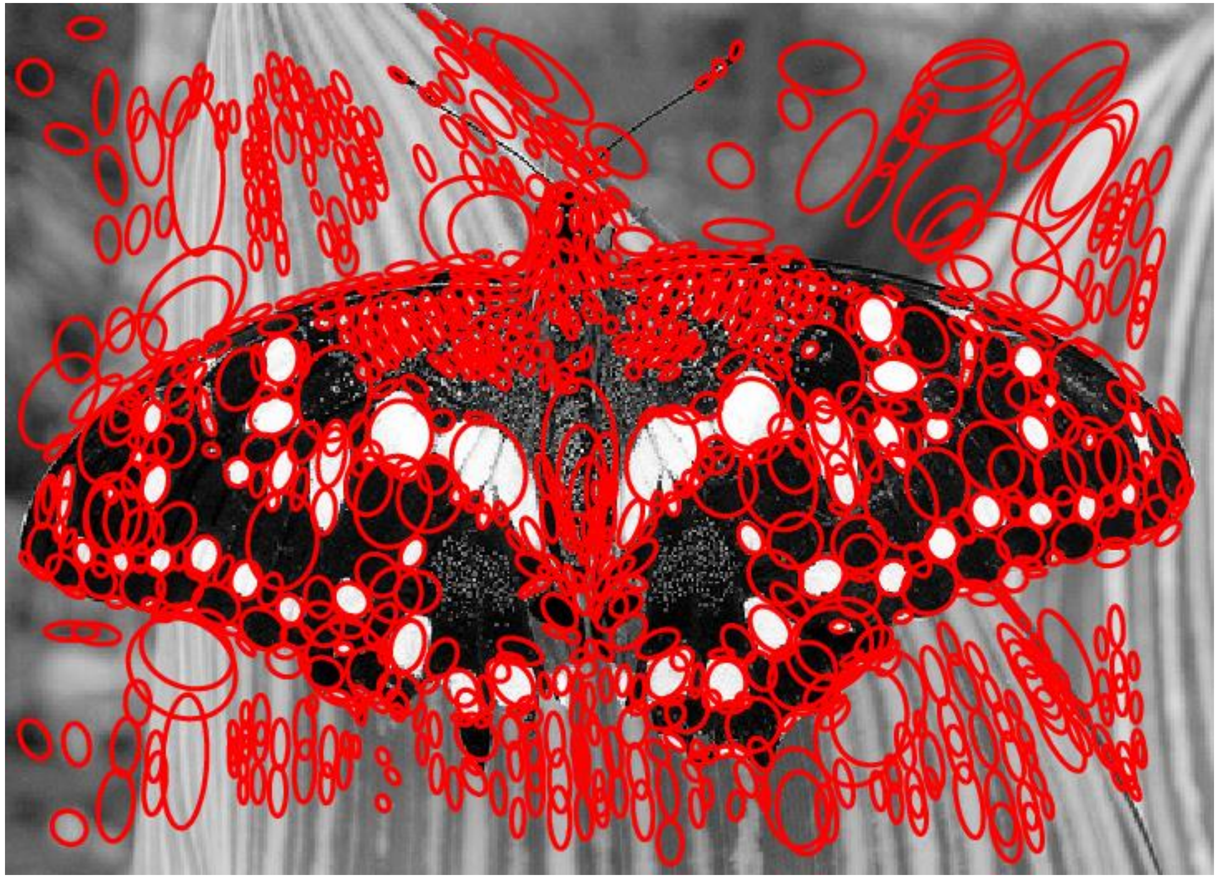
Slide credit: Svetlana Lazebnik

Affine adaptation: Example



Detect blobs accross scales

Affine adaptation: Example



Affine-adapted regions

Affine patch normalization

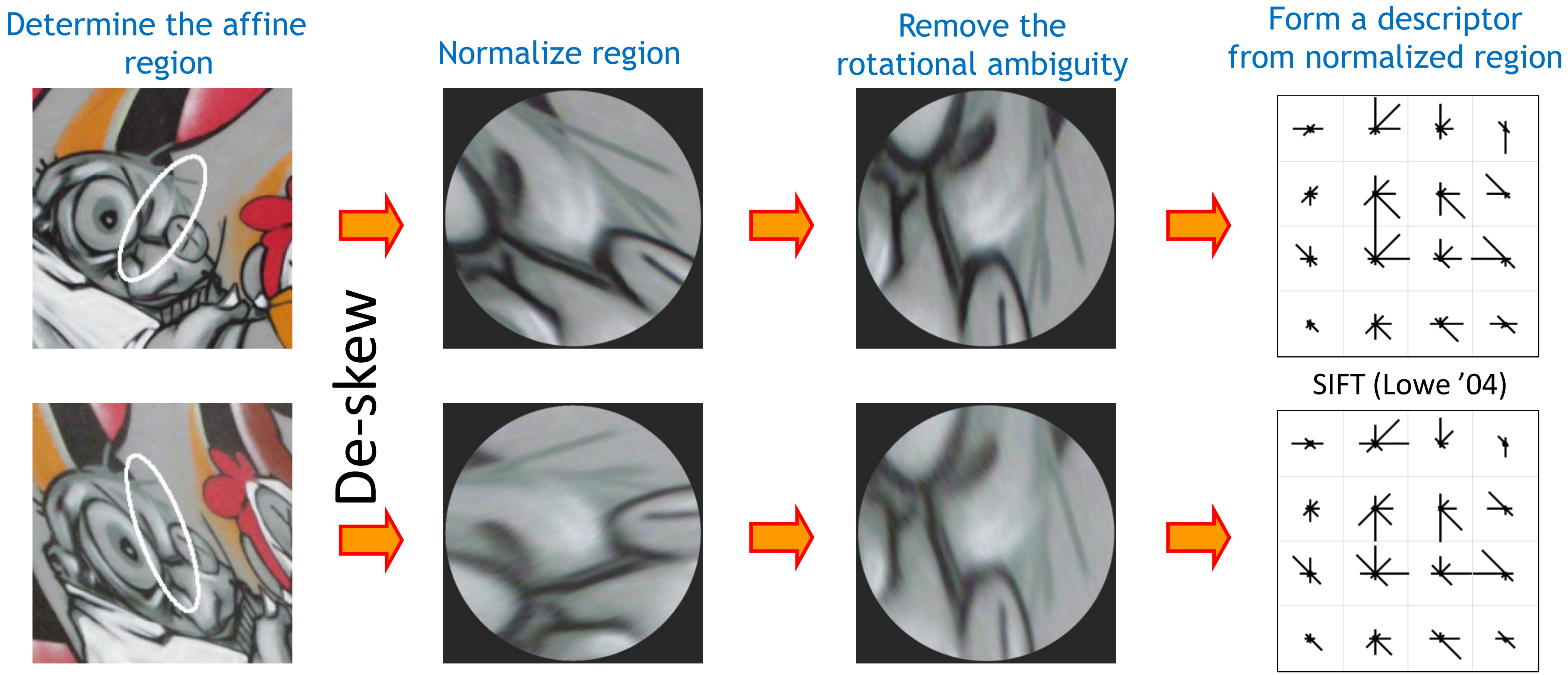
- Transform the patch such that the ellipse becomes as circle.
- Rotate the region such that the ellipse rotates into a horizontal position
- Scale the region such that the ellipse transforms into a circle

Note: Rotation + Scaling computed from the (ellipse, Σ) eigen vectors and eigen values

$$\Sigma = USU^T$$

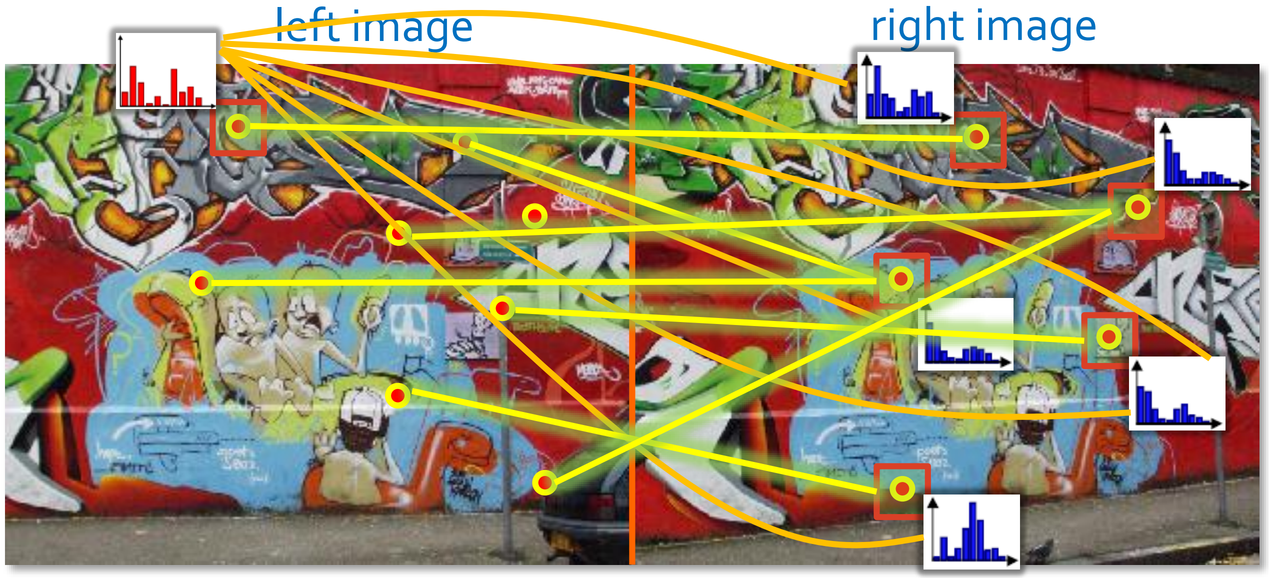
Summary: Affine invariance

- For each key-point determine the affine adaptation, and calculate the descriptor from the rectified region.



Correspondences using keypoints

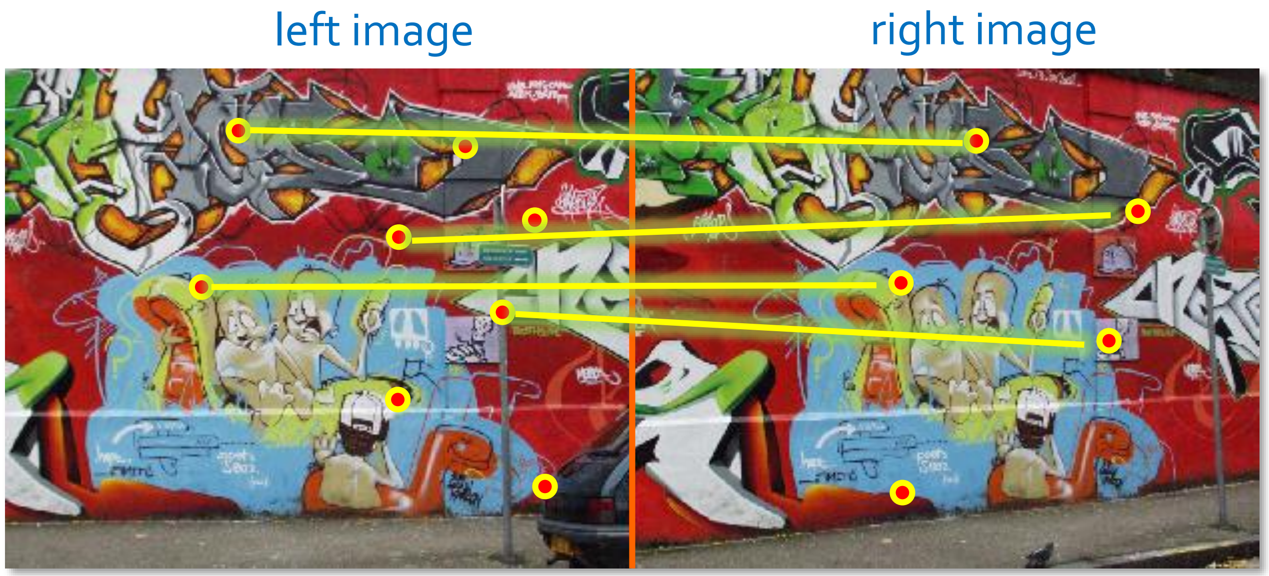
- Compare keypoints by calculating the Euclidean distance (L_2 norm) between descriptors.



- Strategy 1: For each keypoint in the left image identify the most similar keypoint in the right image.
- Result: potential (putative) matches/correspondences

Correspondences using keypoints

- Strategy 2: Keep only symmetric matches

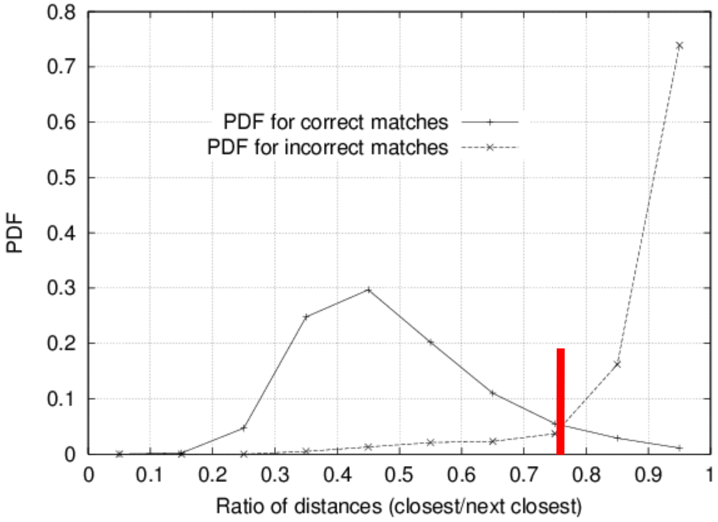


Definition of a symmetric match:

- “Let point A be a point in the left image and point B its match in the right image. If B is most similar to A among all points in the right image and vice versa, they form a symmetric match.”

Correspondences using keypoints

- Strategy 3: *Calculate the similarity of A to the second-most similar keypoint and the most similar keypoint and in the right image.*
  - Ratio of these two similarities will be low for distinctive key-points and high for non-distinctive ones.
  - Threshold ~0.8 gives good results with SIFT.



David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

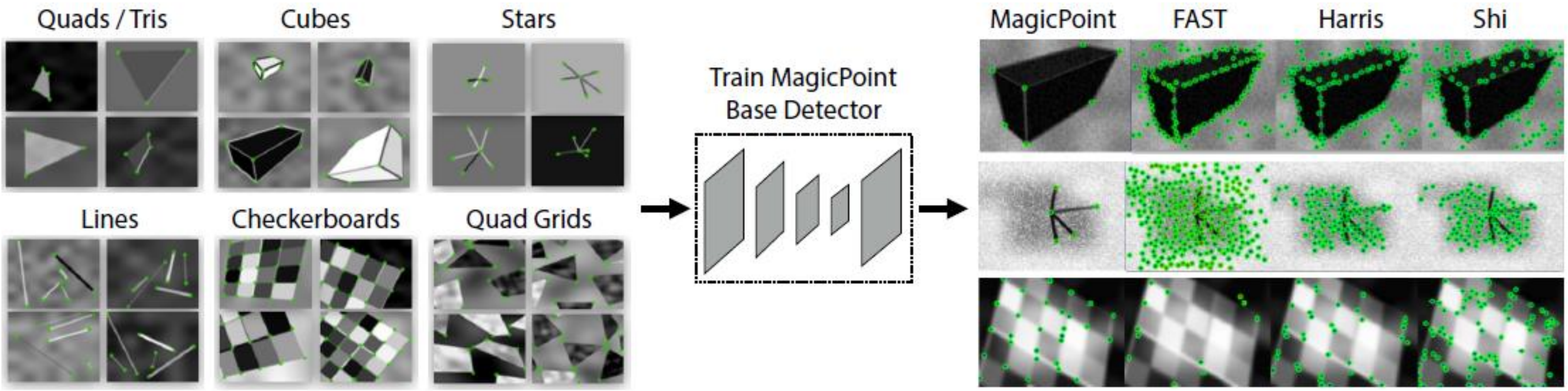
Finally stitching can be fully automated

- Detect key-points independently in each image
- Determine potential correspondences
- Reject improbable correspondences by strategy 1,2, or 3
- Perform RANSAC to find the inliers and fit the model

All correspondences + filtering by strategy 1,2,3 +RANSAC:

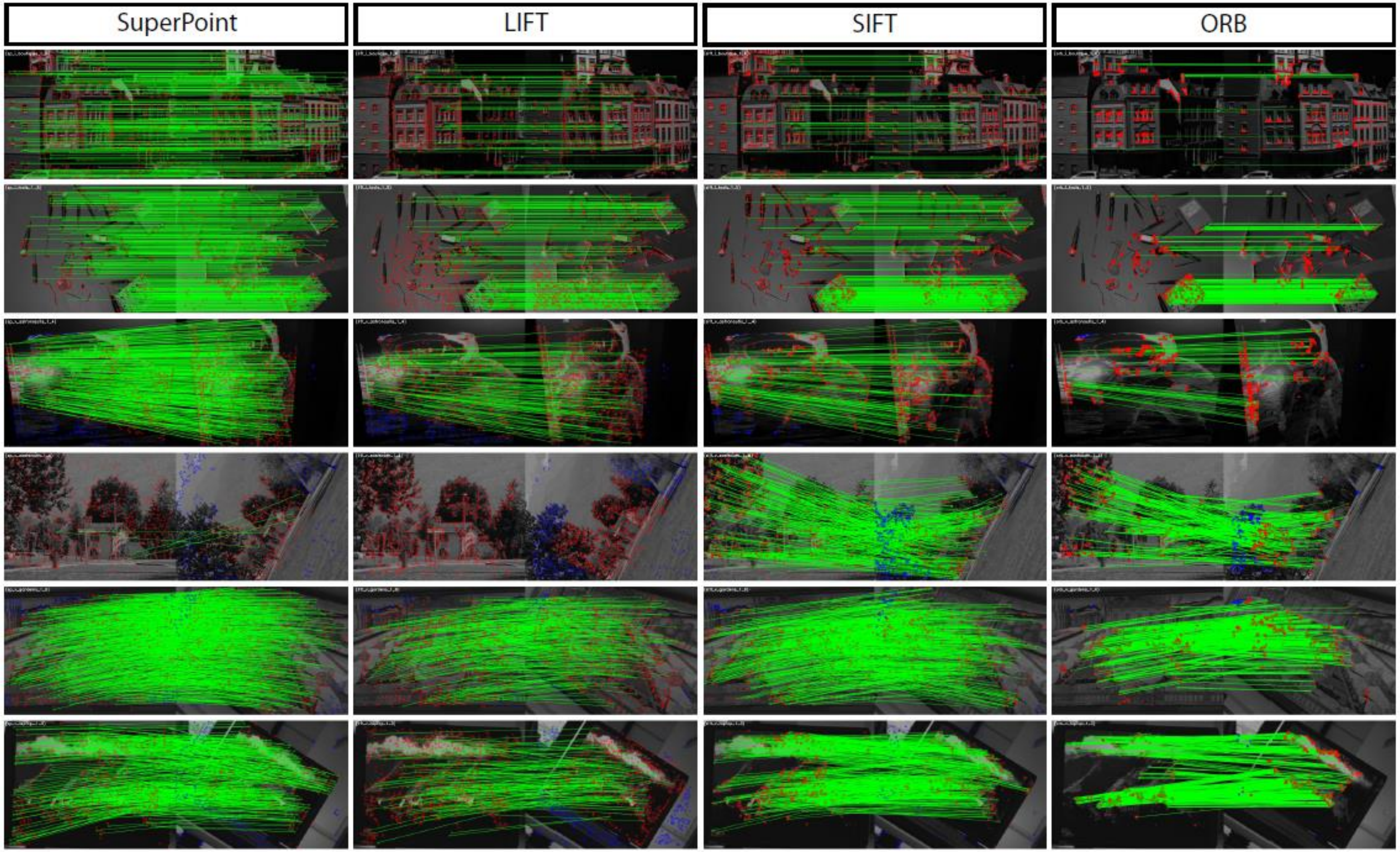
Recent work on keypoint detection

- SuperPoint – a convolutional neural network trained to "fire" on a key point
- Keypoints trained on simulated data, adapted to real data, re-trained for joint extraction of keypoints and descriptors



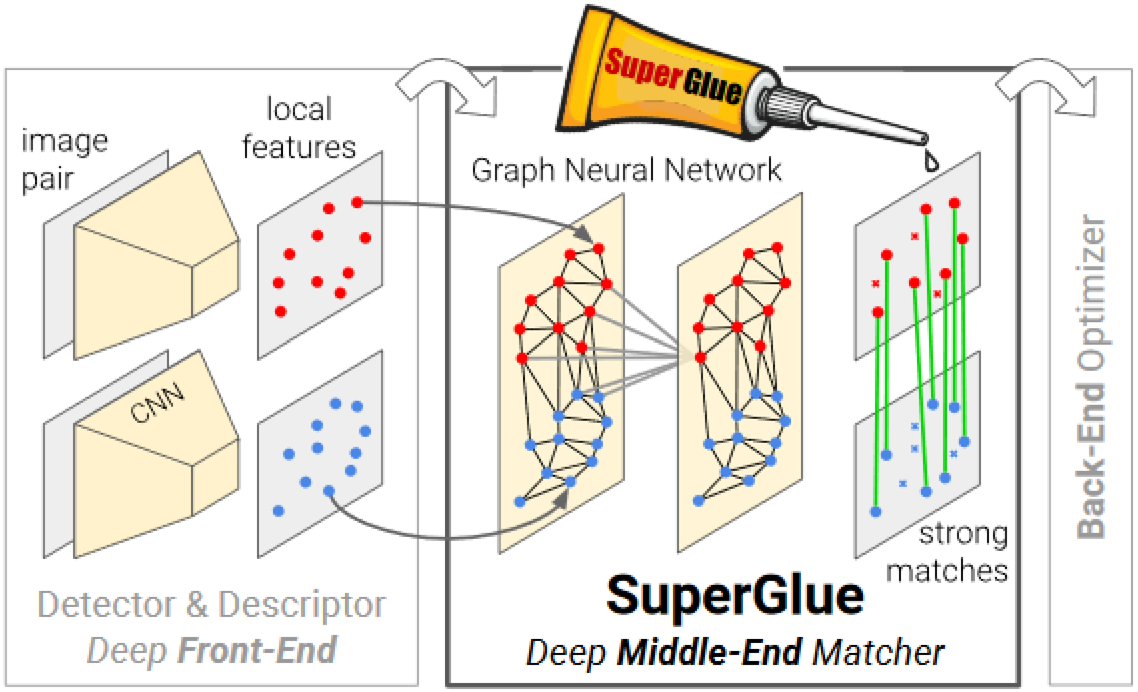
DeTone et al., SuperPoint: Self-Supervised Interest Point Detection and Description, CVPR2018

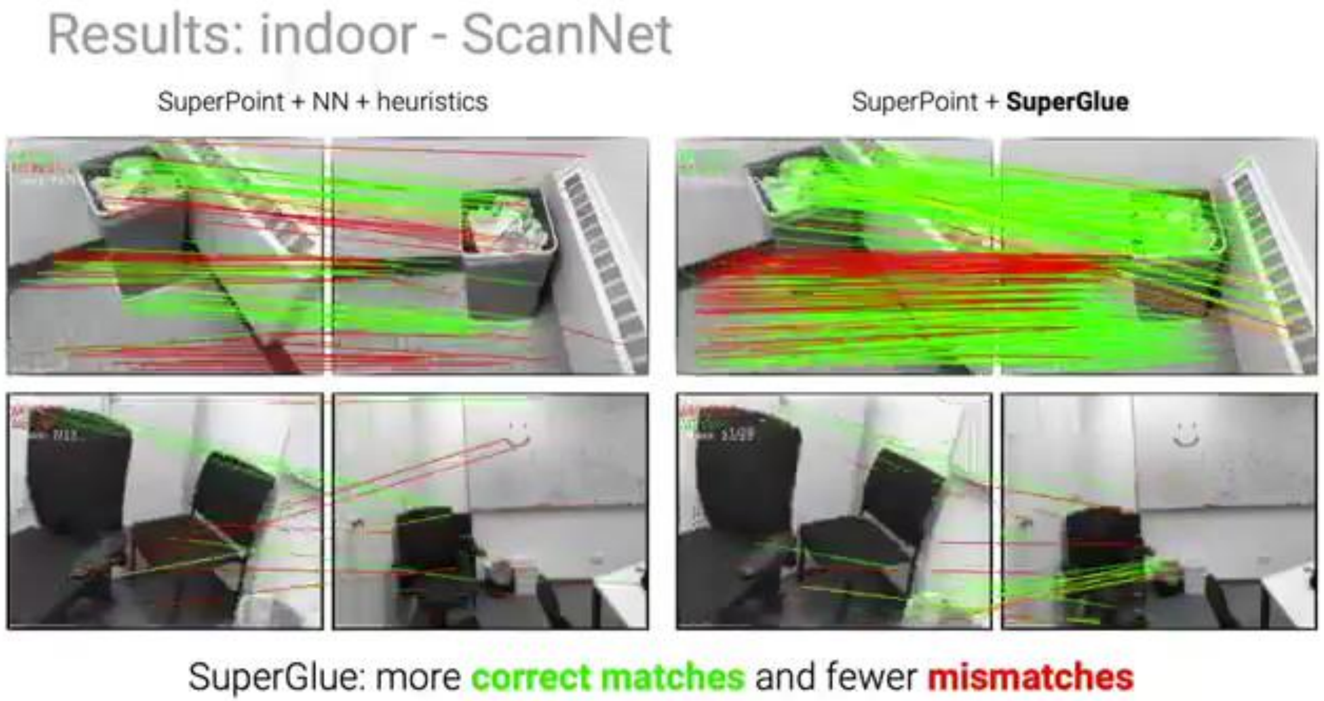
Recent work on keypoint detection

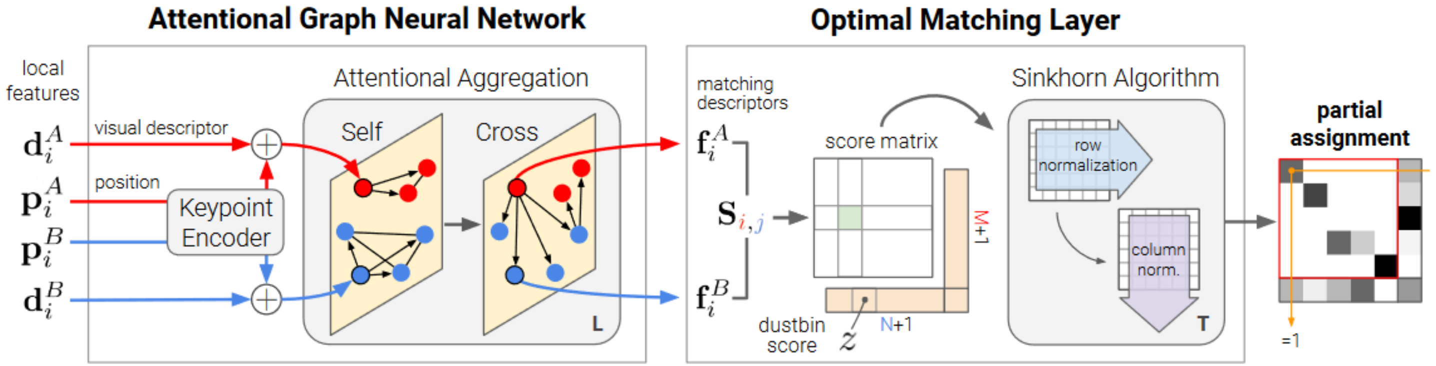


DeTone et al., [SuperPoint: Self-Supervised Interest Point Detection and Description](), CVPR2018

Recent work on correspondences matching







Sarlin et al. SuperGlue: Learning Feature Matching with Graph Neural Networks, CVPR2020 (video)

Numerous detectors/descriptors exist

- We have only considered a most popular descriptor (SIFT, Lowe2004)
  - Note that Lowe proposed DoG for keypoint detection and SIFT for descriptor – don't mix these!
- Many efficient and really fast descriptors have been presented since.

- Significant research currently invested into end-to-end learning the keypoint detection, description and matching process by convolutional neural nets.

References

- David A. Forsyth, Jean Ponce, Computer Vision: A Modern Approach (2nd Edition), (prva izdaja dostopna na spletu)
- R. Szeliski,Computer Vision: Algorithms and Applications, 2010
- R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, 2004
- D. Lowe, Distinctive image features from scale-invariant keypoints, IJCV 60(2), pp. 91-110, 2004
- T. Tuytelaars, K. Mikolajczyk, Local Invariant Feature Detectors: A Survey, Foundations and Trends in Computer Graphics and Vision, Vol. 3, No. 3, pp 177-280, 2008.