CAP 4453
Robot Vision

Dr. Gonzalo Vaca-Castaño

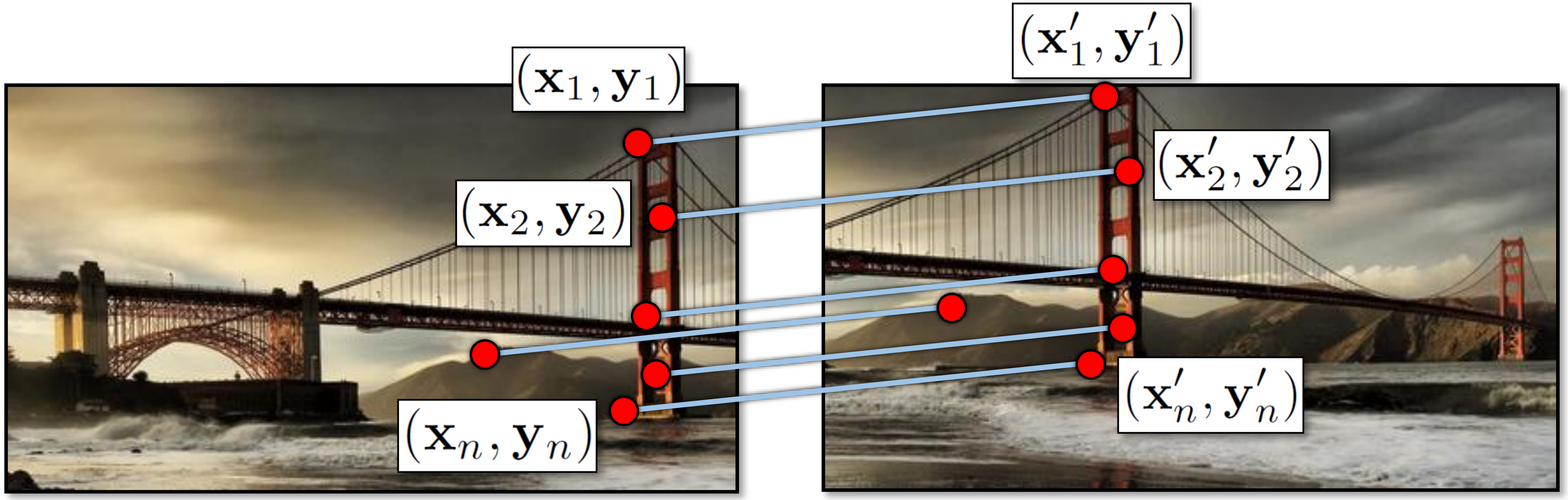
gonzalo.vacacastano@ucf.edu

Credits

- Some slides comes directly from these sources:
  - Ioannis (Yannis) Gkioulekas (CMU)
  - Kris Kitani.
  - Fredo Durand (MIT).
  - James Hays (Georgia Tech).
  - Yogesh S Rawat (UCF)
  - Noah Snavely (Cornell)
  - Trym Vegard Haavardsholm (Unik)

Short Review from last class

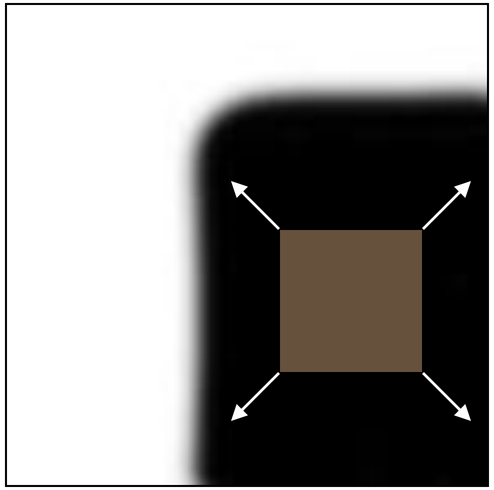
Image warping



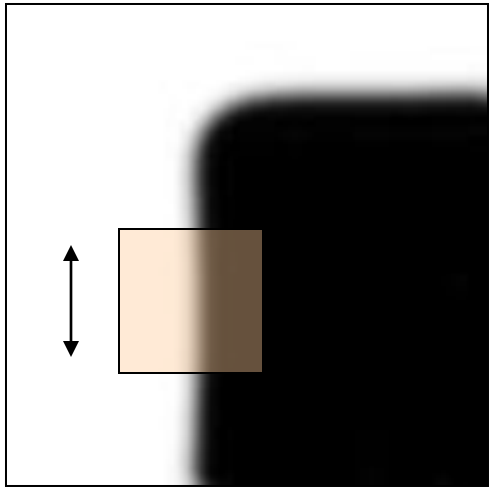
How do we find point correspondences automatically?

Corner Detection: Basic Idea

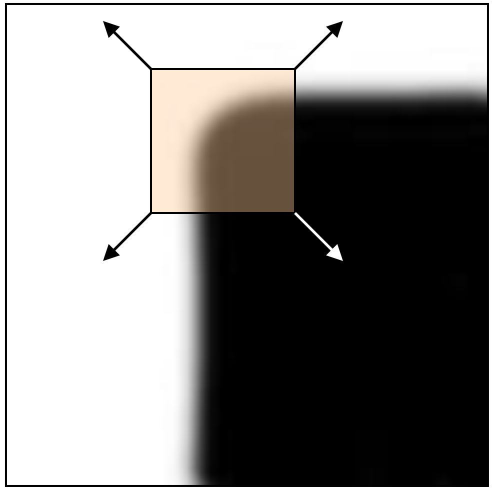
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

Source: A. Efros

C.Harris and M.Stephens. “A Combined Corner and Edge Detector.”1988.

Harris Detector

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \qquad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x^2} = I_x \cdot I_x \qquad I_{y^2} = I_y \cdot I_y \quad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{xy} = G_{\sigma'} * I_{xy} \qquad S_{y^2} = G_{\sigma'} * I_{y^2} \qquad S_{x^2} = G_{\sigma'} * I_{x^2}$$

4. Define the matrix at each pixel

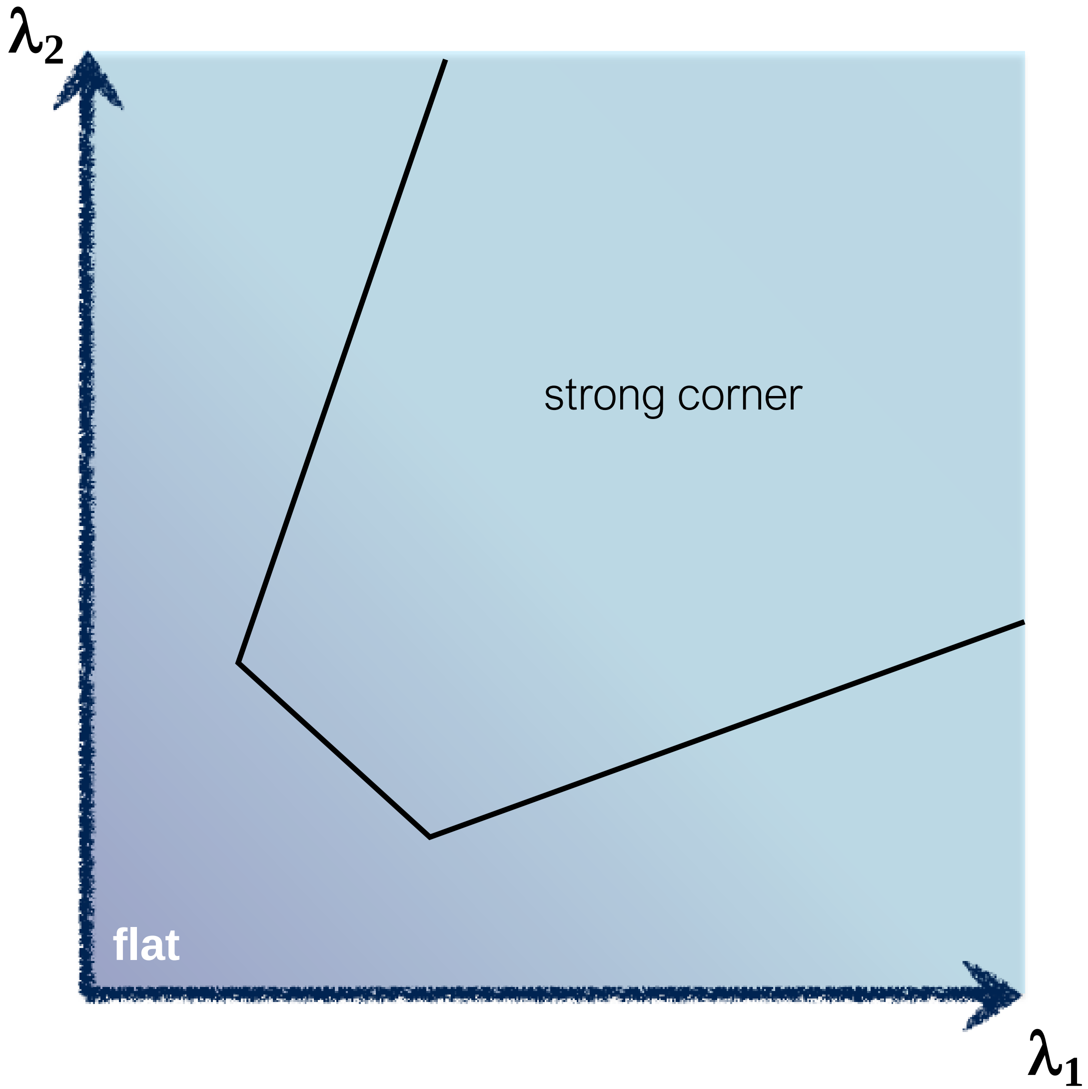
$$M(x, y) = \begin{bmatrix} S_{x^2}(x, y) & S_{xy}(x, y) \\ S_{xy}(x, y) & S_{y^2}(x, y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

$$R = \det M - k(\operatorname{trace} M)^2$$

6. Threshold on value of R; compute non-max suppression.

Use threshold on eigenvalues to detect corners



Think of a function to score 'cornerness'

Harris & Stephens (1988)

$$R = \det(M) - \kappa \operatorname{trace}^2(M)$$

Kanade & Tomasi (1994)

$$R = \min(\lambda_1, \lambda_2)$$

Nobel (1998)

$$R = \frac{\det(M)}{\operatorname{trace}(M) + \epsilon}$$

$$\det M = \lambda_1 \lambda_2$$

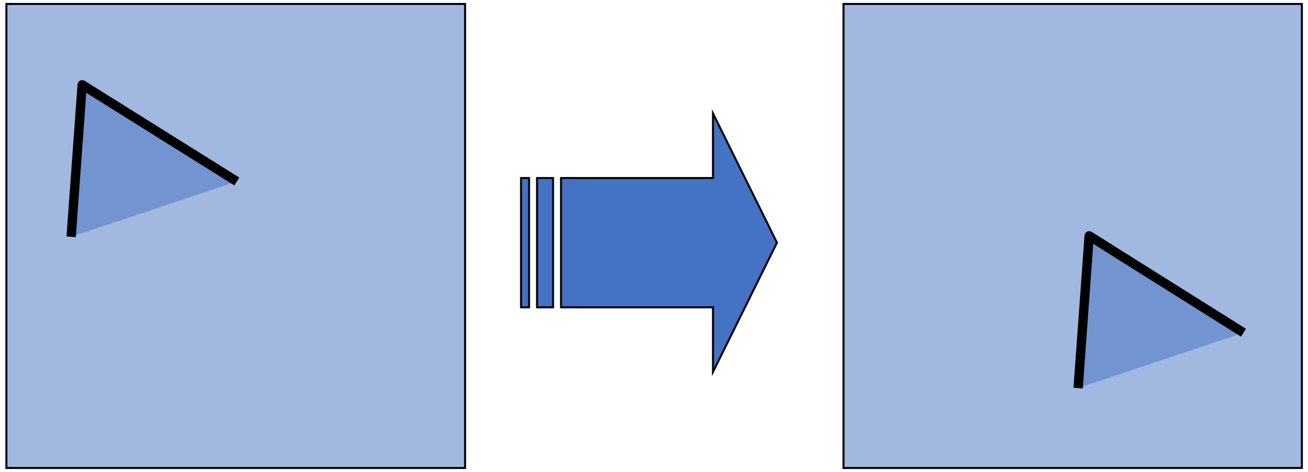
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

$$det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

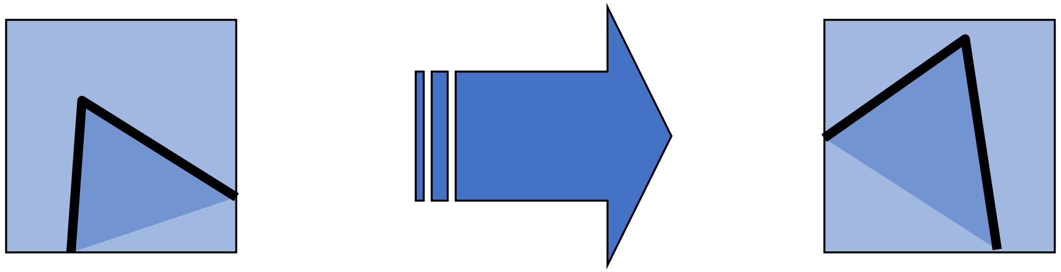
$$trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$$

Harris corner detection and translation

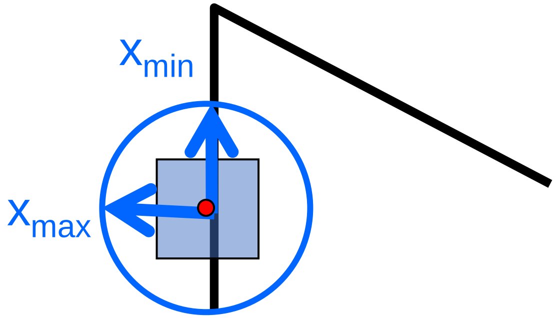
- What happens if image is translated?
- Derivatives, second moment matrix obtained through convolution, which is *translation equivariant*
- Eigenvalues based only on derivatives so cornerness is *invariant*
- Thus Harris corner detection location is *equivariant to translation,* and response is *invariant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



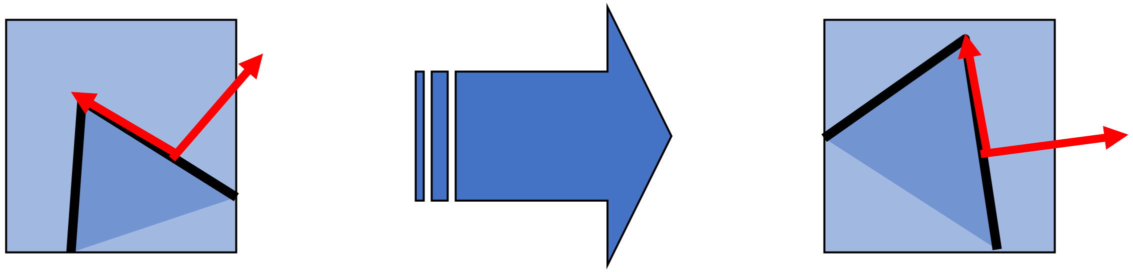
What about rotation?



- 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}



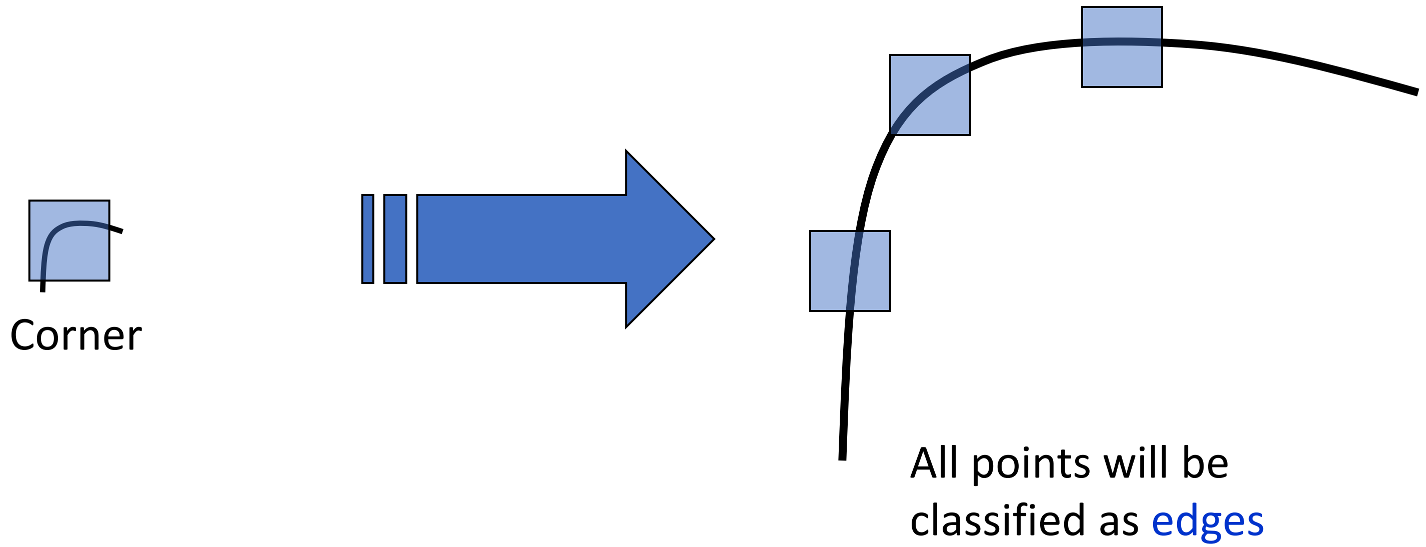
What about rotation?



- 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

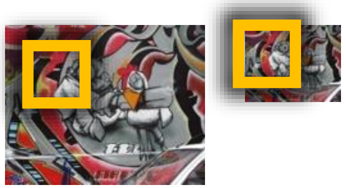
implementation

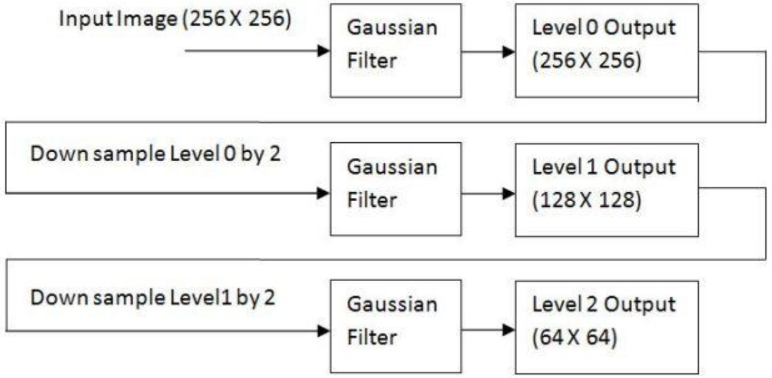
```
For each level of the Gaussian pyramid

    compute feature response (e.g. Harris, Laplacian)


For each level of the Gaussian pyramid

    if local maximum and cross-scale

    save scale and location of feature
```

(x, y, s)

Implementation

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

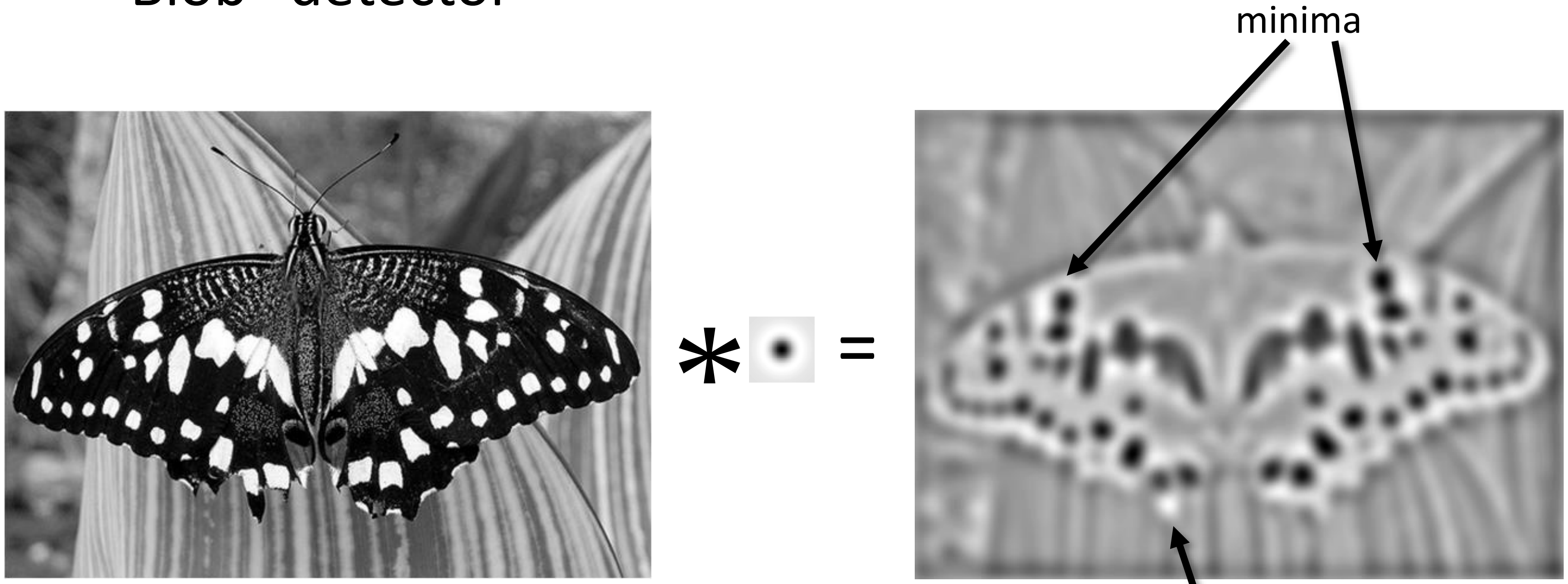




Blob detection

Laplacian of Gaussian

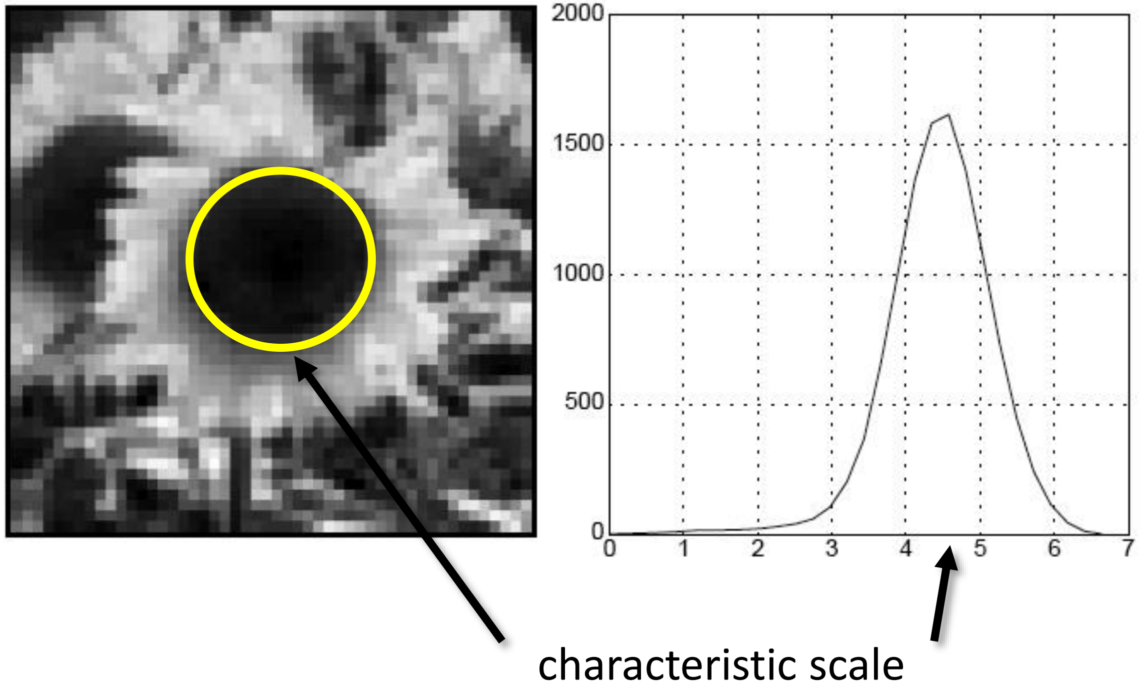
- “Blob” detector



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

Characteristic scale

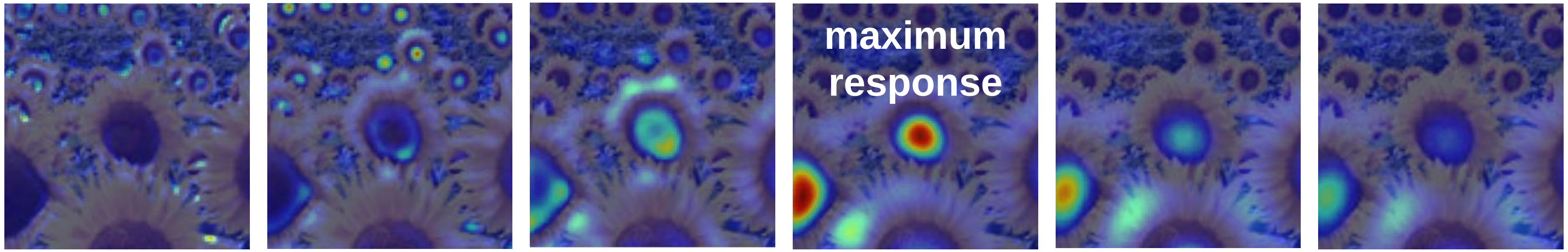
- We define the characteristic scale as the scale that produces peak of Laplacian response



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

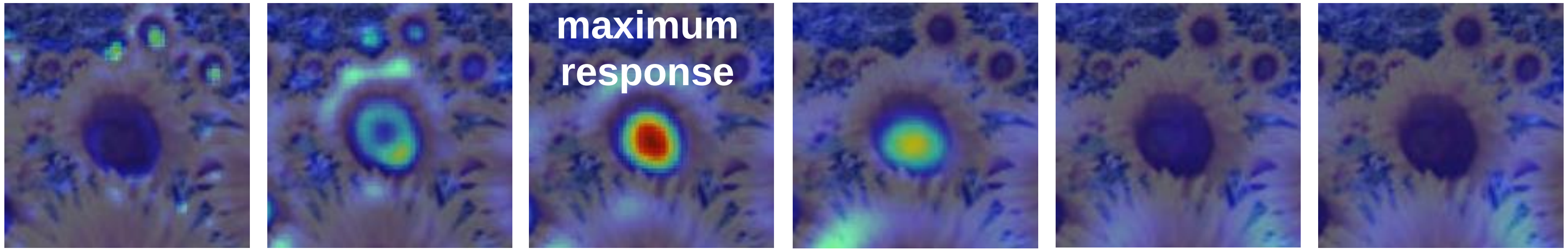
optimal scale

2.1 4.2 6.0 9.8 15.5 17.0

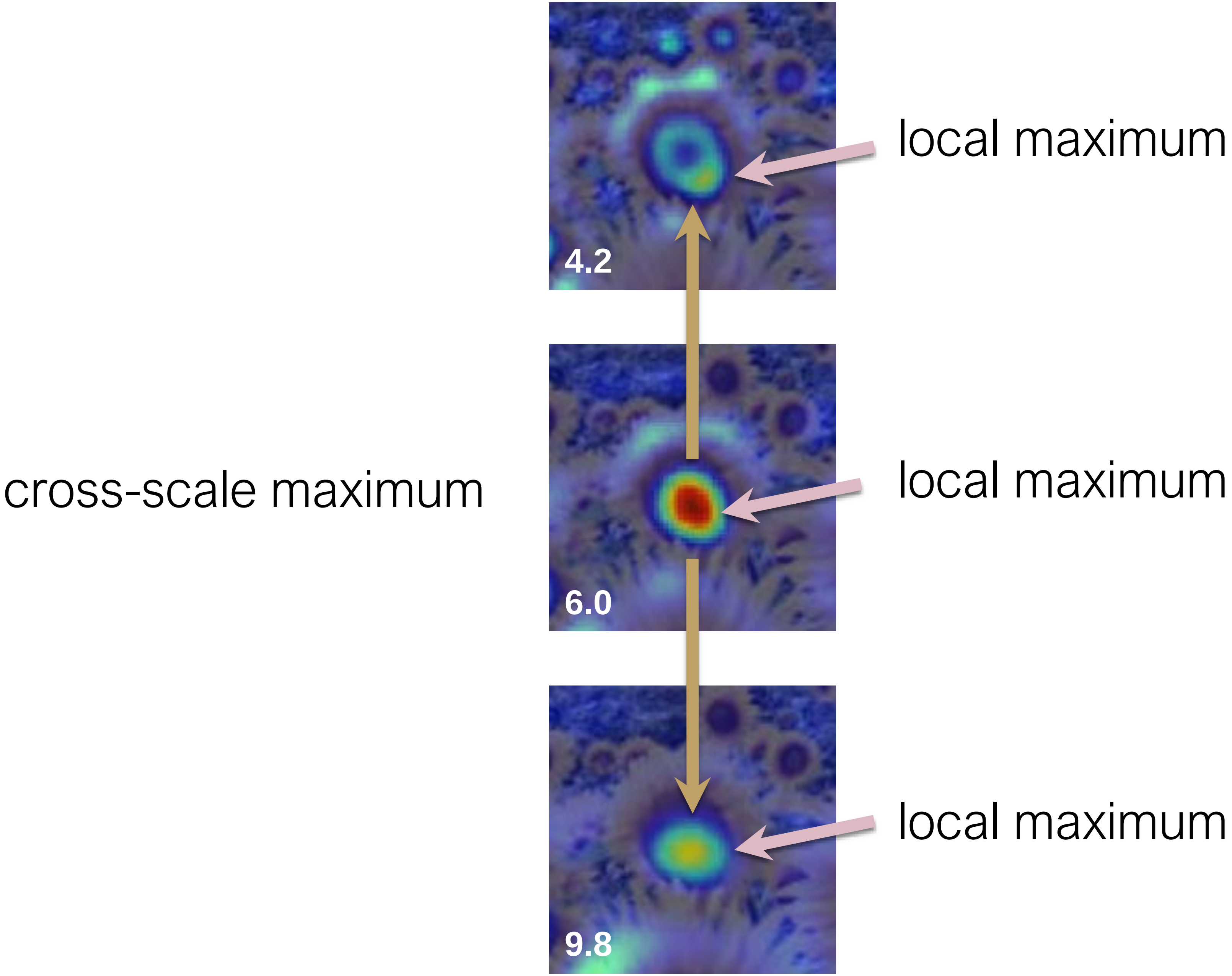


Full size image

2.1 4.2 6.0 9.8 15.5 17.0



3/4 size image



local maximum
4.2
cross-scale maximum
local maximum
6.0
local maximum
9.8

Robot Vision

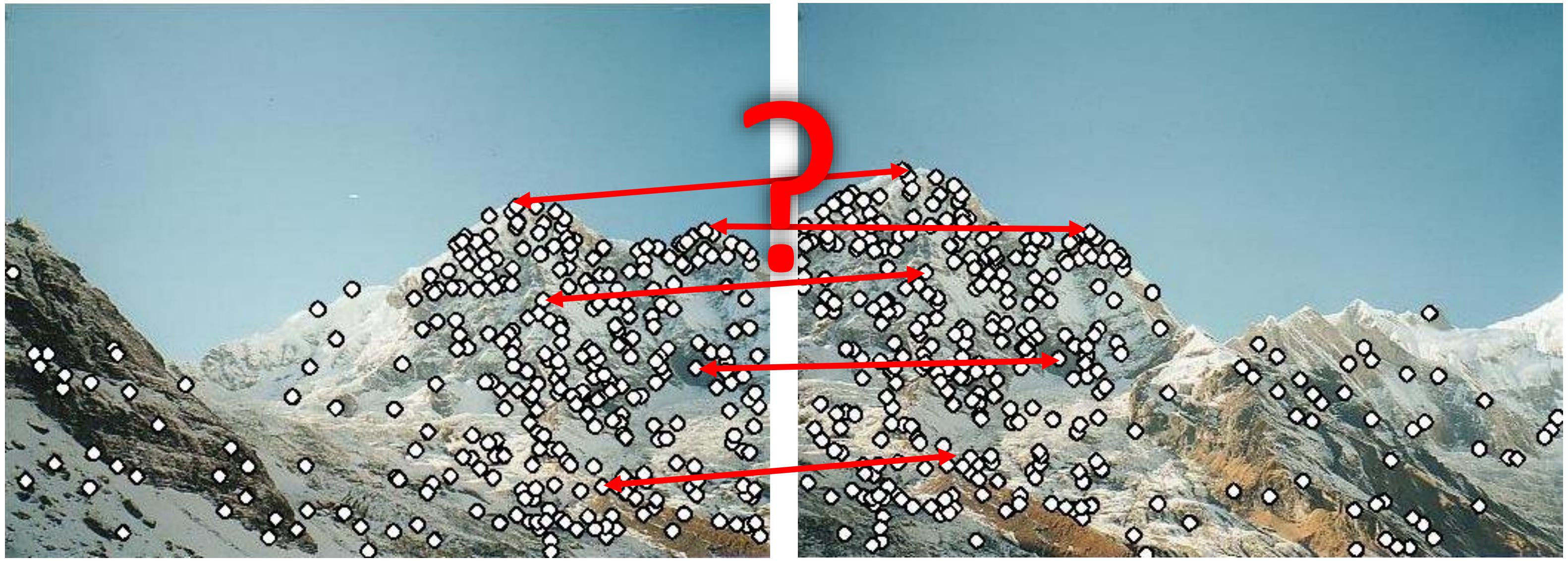
11. Feature points description

Outline

- Motivation
- Detecting key points
  - Harris corner detector
  - Blob detection
- Feature descriptors
  - HOG
  - MOPS
- SIFT

Matching feature points

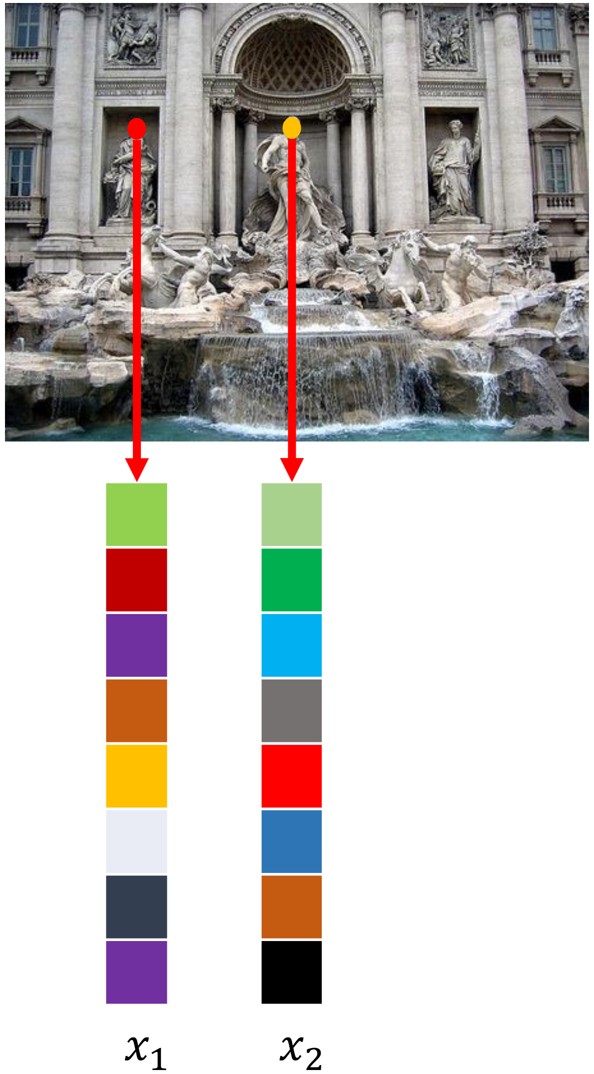
We know how to detect good points
Next question: **How to match them?**

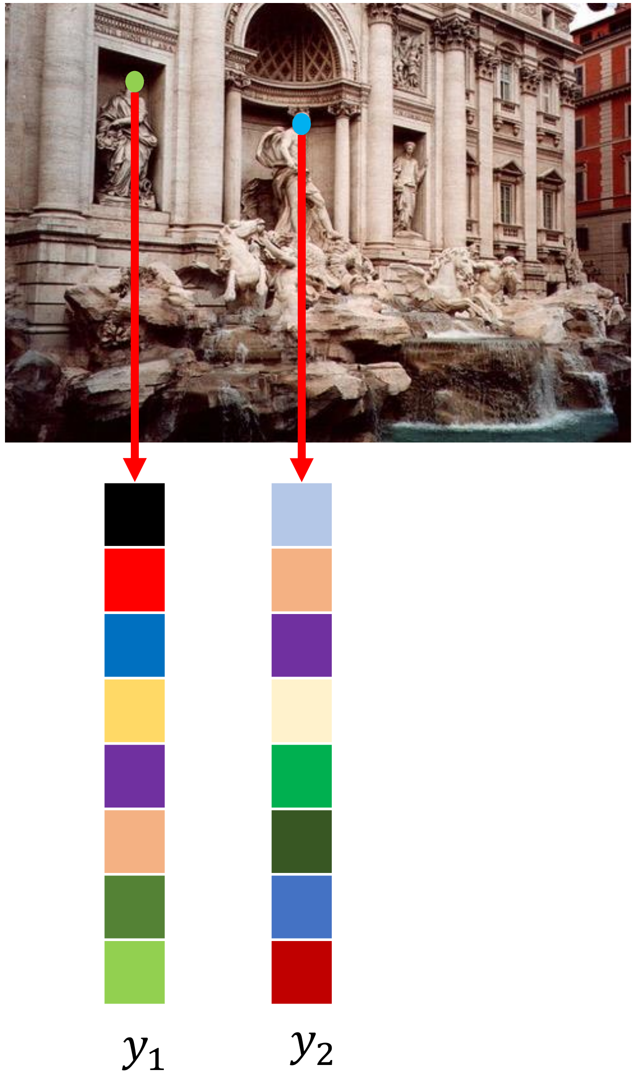


Two interrelated questions:

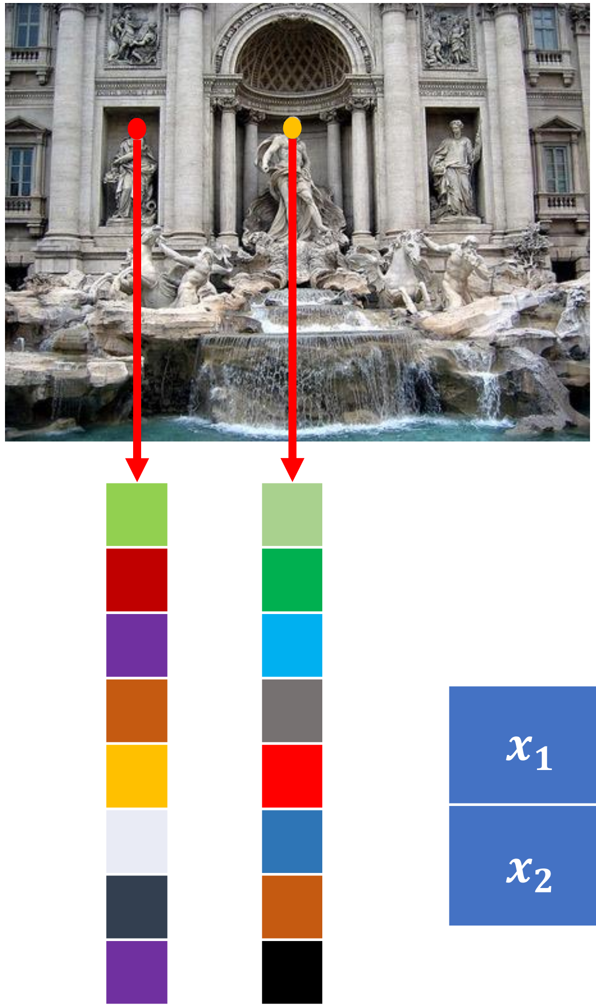
1. How do we *describe* each feature point?
2. How do we *match* descriptions?

Feature descriptor



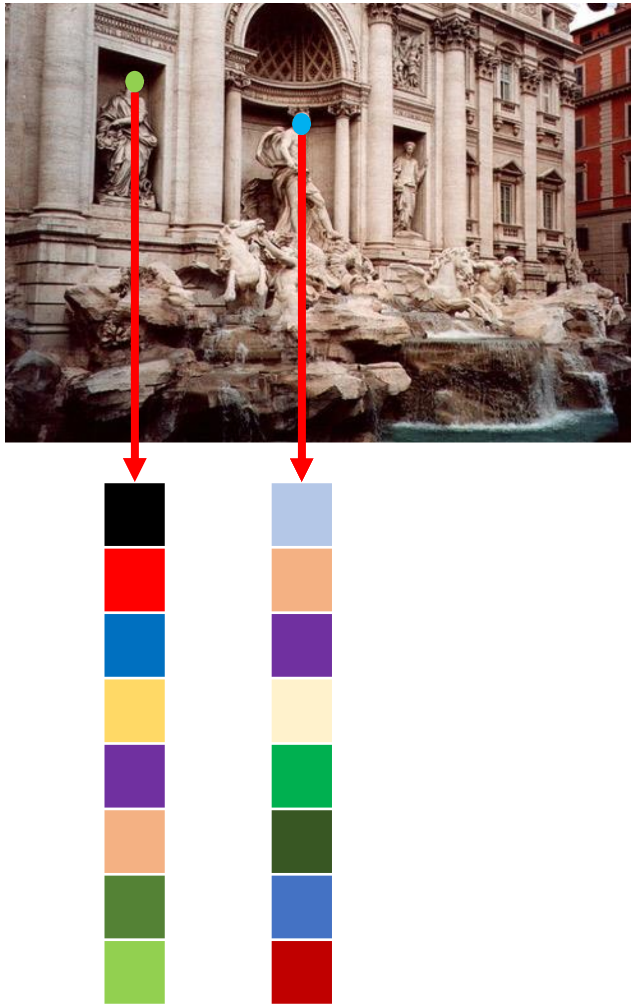


Feature matching



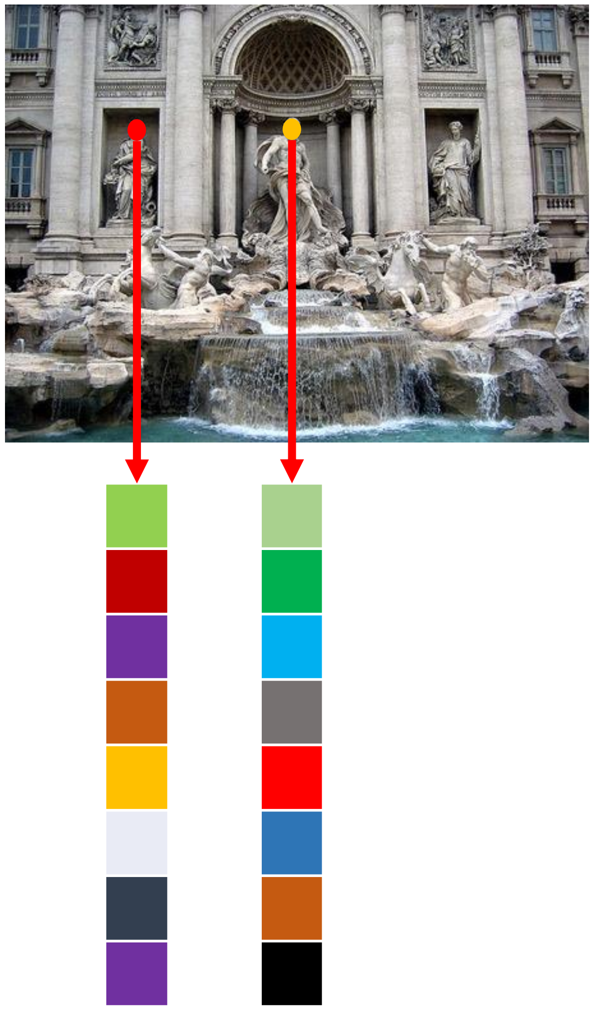
x_1 x_2

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



y_1 y_2

Feature Descriptor



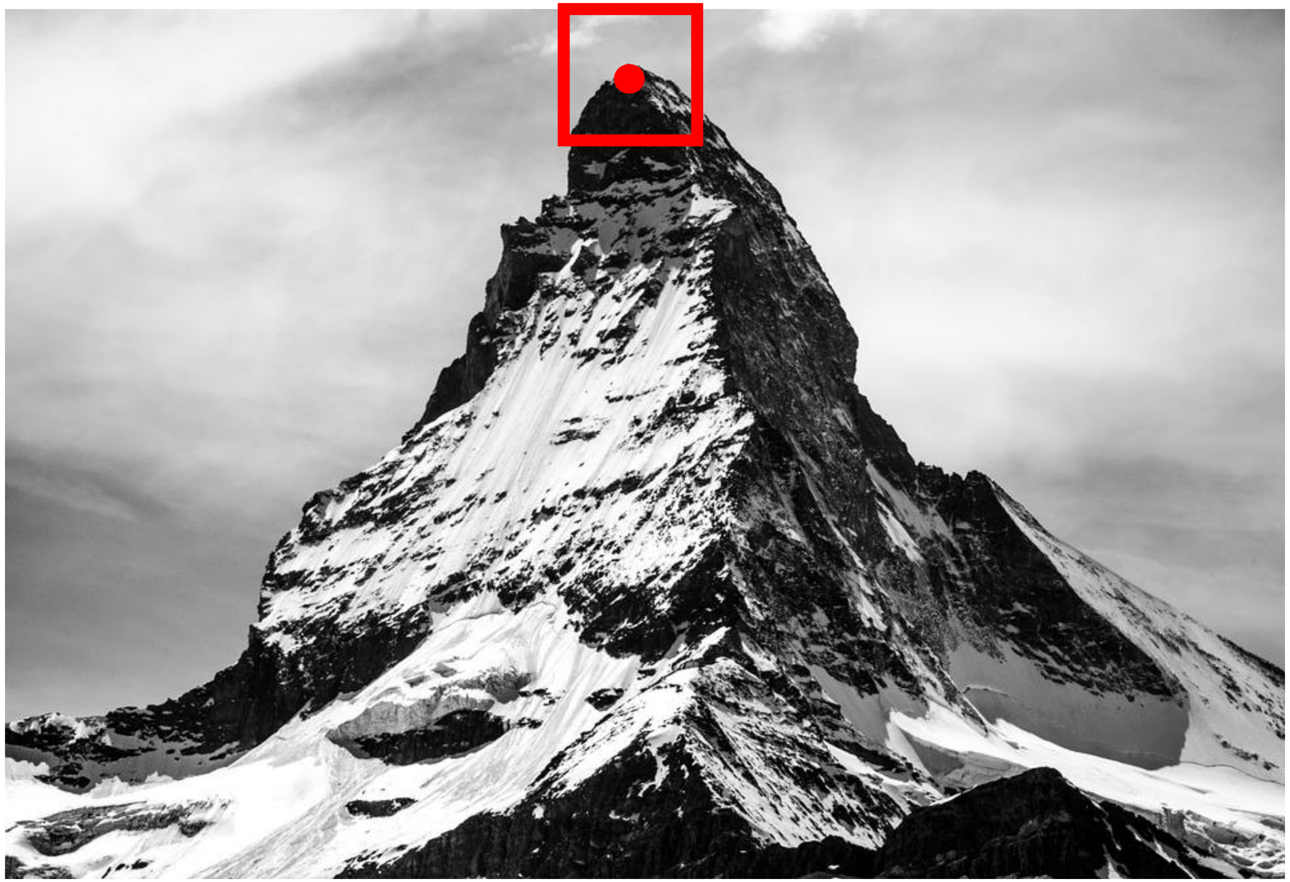
x_1 x_2

Feature detection and description

- Harris corner detection gives:
  - Location of each detected corner
  - Orientation of the corner (given by $\mathbf{x}_{max}$)
  - Scale of the corner (the image scale which gives the maximum response at this location)
- Want feature descriptor that is
  - Invariant to photometric transformations, translation, rotation, scaling
  - Discriminative

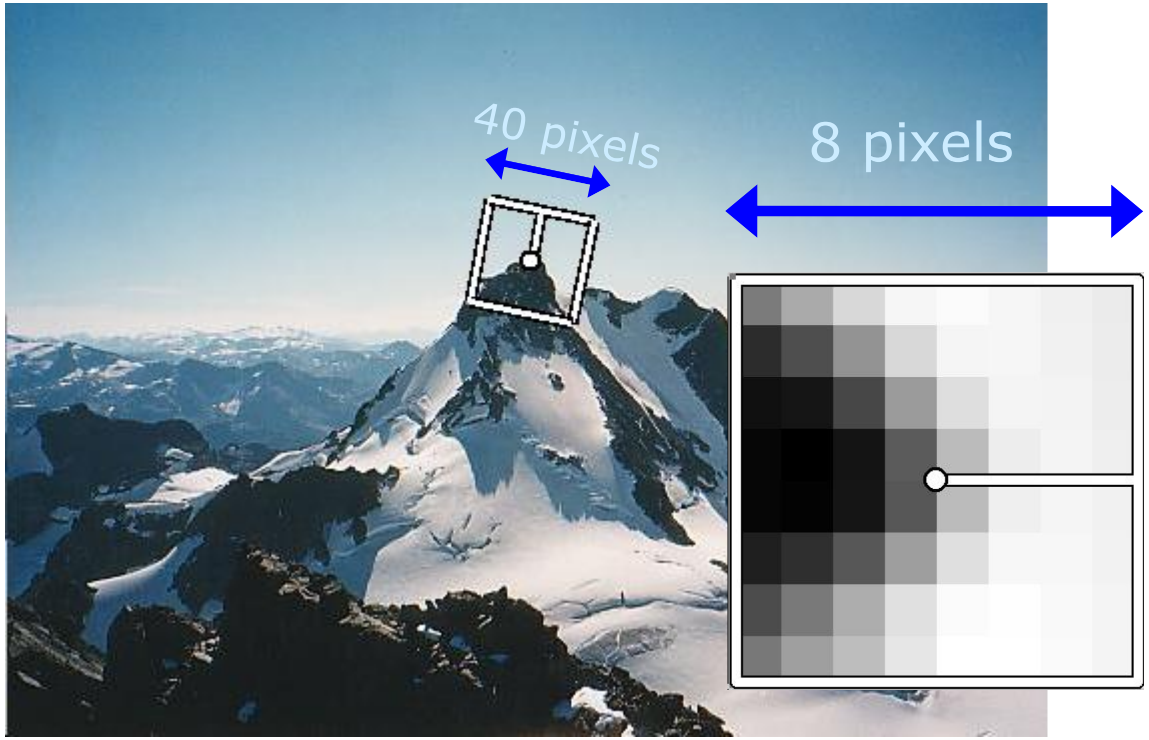
Multiscale Oriented PatcheS descriptor

- Describe a corner by the patch around that pixel
- Scale invariance by using scale identified by corner detector
- Rotation invariance by using orientation identified by corner detector
- Photometric invariance by subtracting mean and dividing by standard deviation



Multiscale Oriented PatcheS descriptor

- Take 40x40 square window around detected feature at the right scale
- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



Adapted from slide by Matthew Brown

MOPS

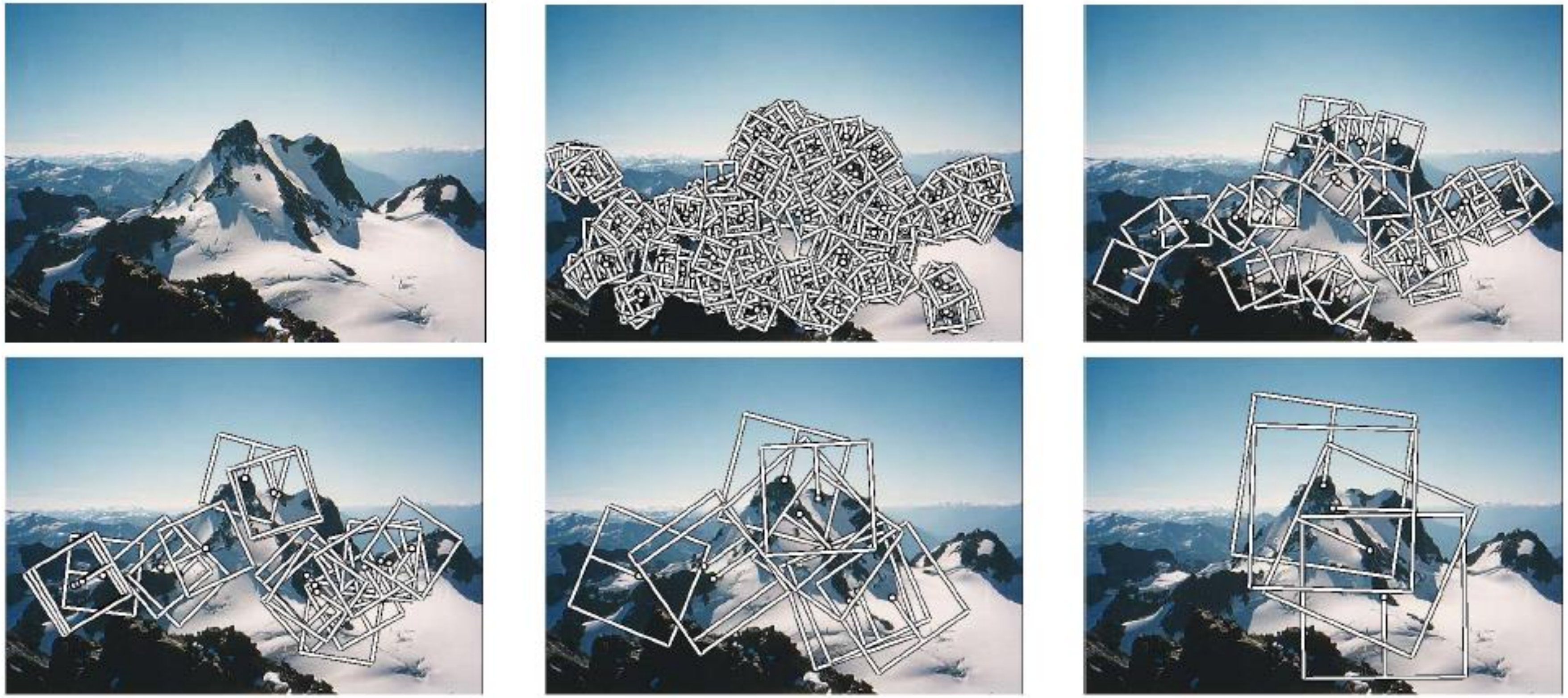


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

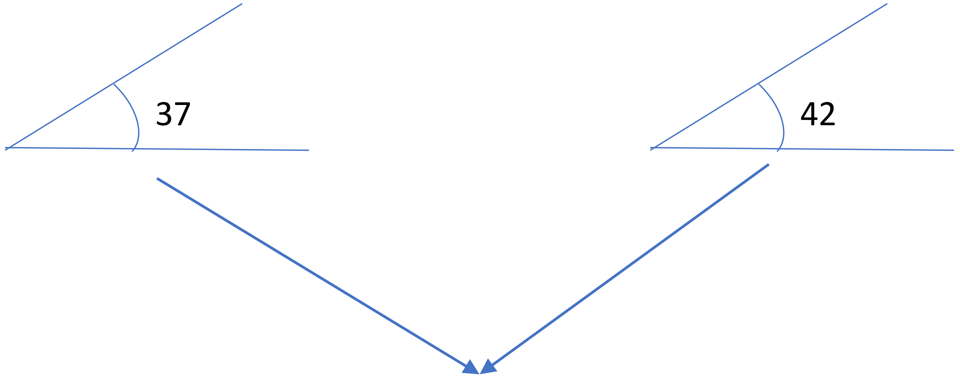
Towards a better feature descriptor

- Match *pattern of edges*
  - Edge orientation – clue to shape
  - Invariant to almost all photometric transformations

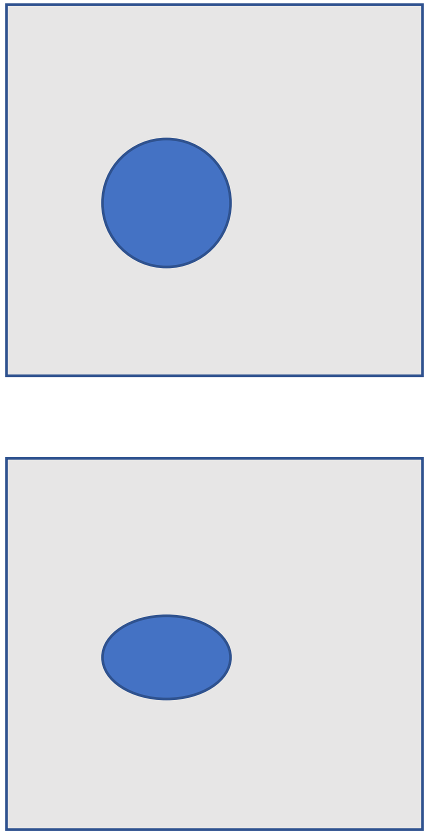
- Be resilient to *small deformations*
  - Deformations might move pixels around, but slightly
  - Deformations might change edge orientations, but slightly

Invariance to deformation

- Precise edge orientations are not resilient to out-of-plane rotations and deformations
- But we can *quantize* edge orientation: only record rough orientation



Between 30 and 45

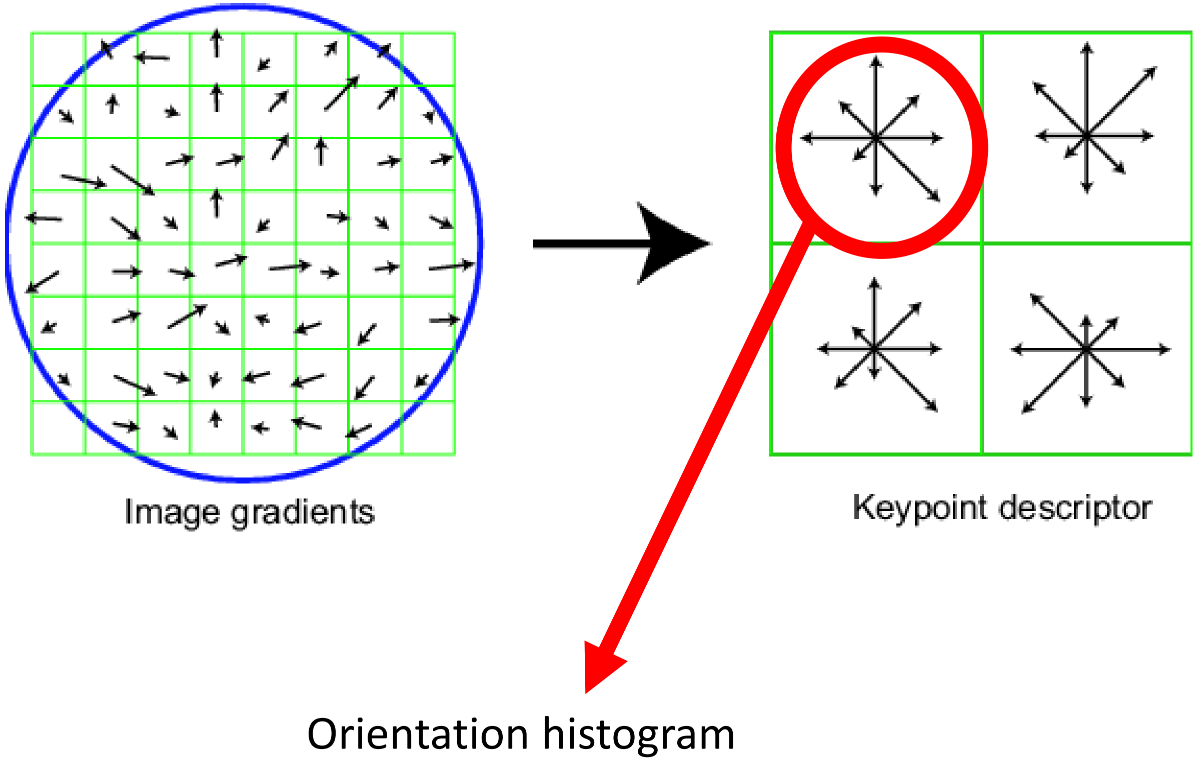


Invariance to deformation

$$g(\theta) = \begin{cases} 0 & \text{if } 0 < \theta < 2\pi/N \\ 1 & \text{if } 2\pi/N < \theta < 4\pi/N \\ 2 & \text{if } 4\pi/N < \theta < 6\pi/N \\ & \dots \\ N-1 & \text{if } 2(N-1)\pi/N \end{cases}$$

Invariance to deformation

- Deformation can also move pixels around
- Again, instead of precise location of each pixel, only want to record rough location
- Divide patch into a grid of *cells*
- Record *counts* of each orientation in each cell: *orientation histograms*

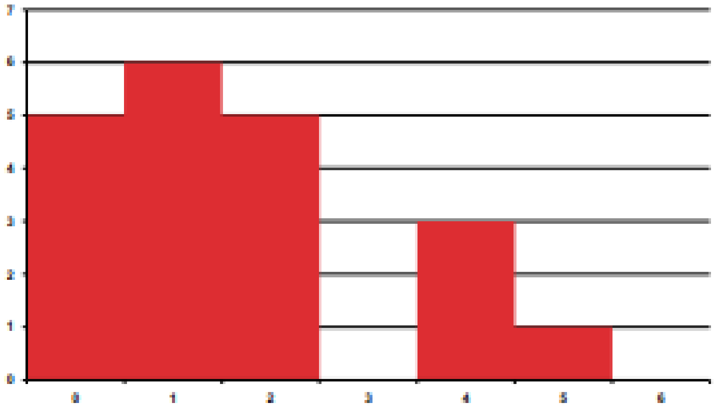


Histogram of Oriented Gradients (HOG)

- Revisiting histogram

0	1	1	2	4
2	1	0	0	2
5	2	0	0	4
1	1	2	4	1

image



histogram

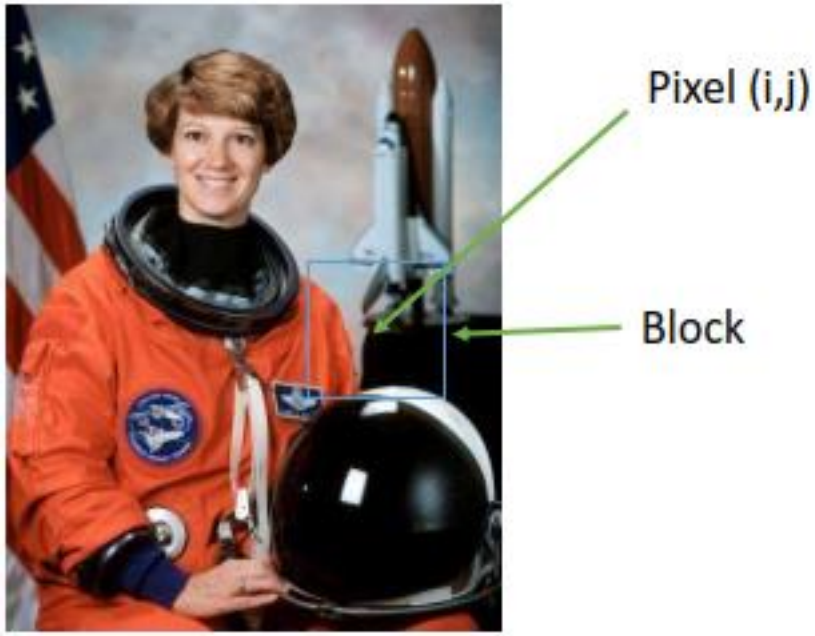
Histogram of Oriented Gradients (HOG)

- Given an image I, and a pixel location (i,j).
- We want to compute the HOG feature for that pixel.
- The main operations can be described as a sequence of five steps.

Pixel (i,j)

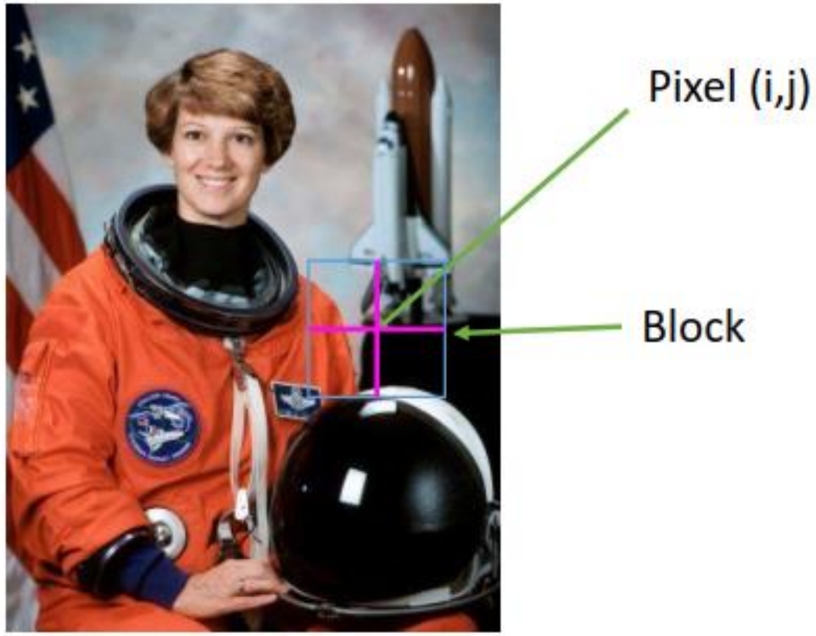
Histogram of Oriented Gradients (HOG)

- Step 1: Extract a square window (called “block”) of some size.



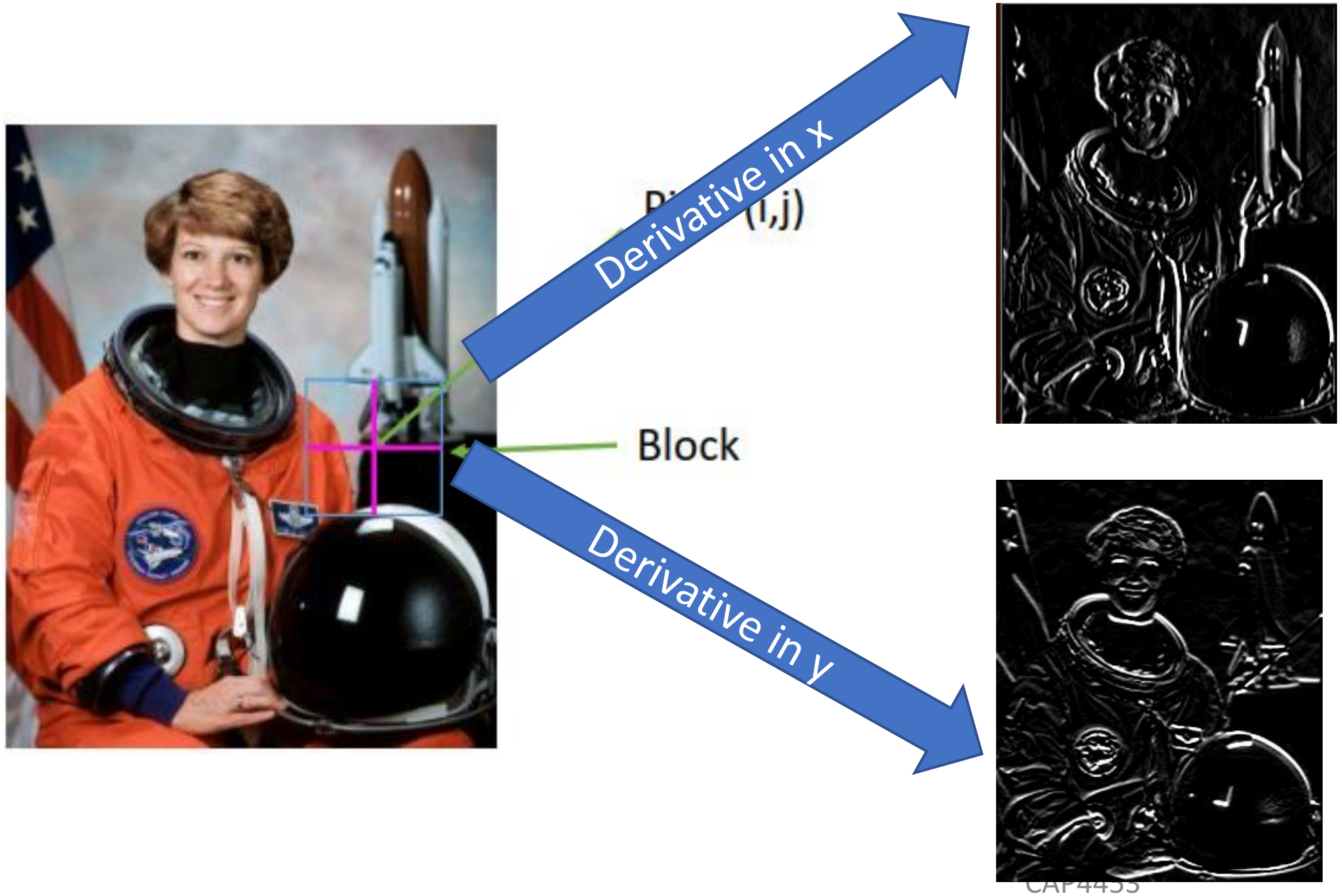
Histogram of Oriented Gradients (HOG)

- Step 2: Divide block into a square grid of sub-blocks (called “cells”) (2x2 grid in our example, resulting in four cells).



Histogram of Oriented Gradients (HOG)

- Step 3: Compute orientation histogram of each cell.

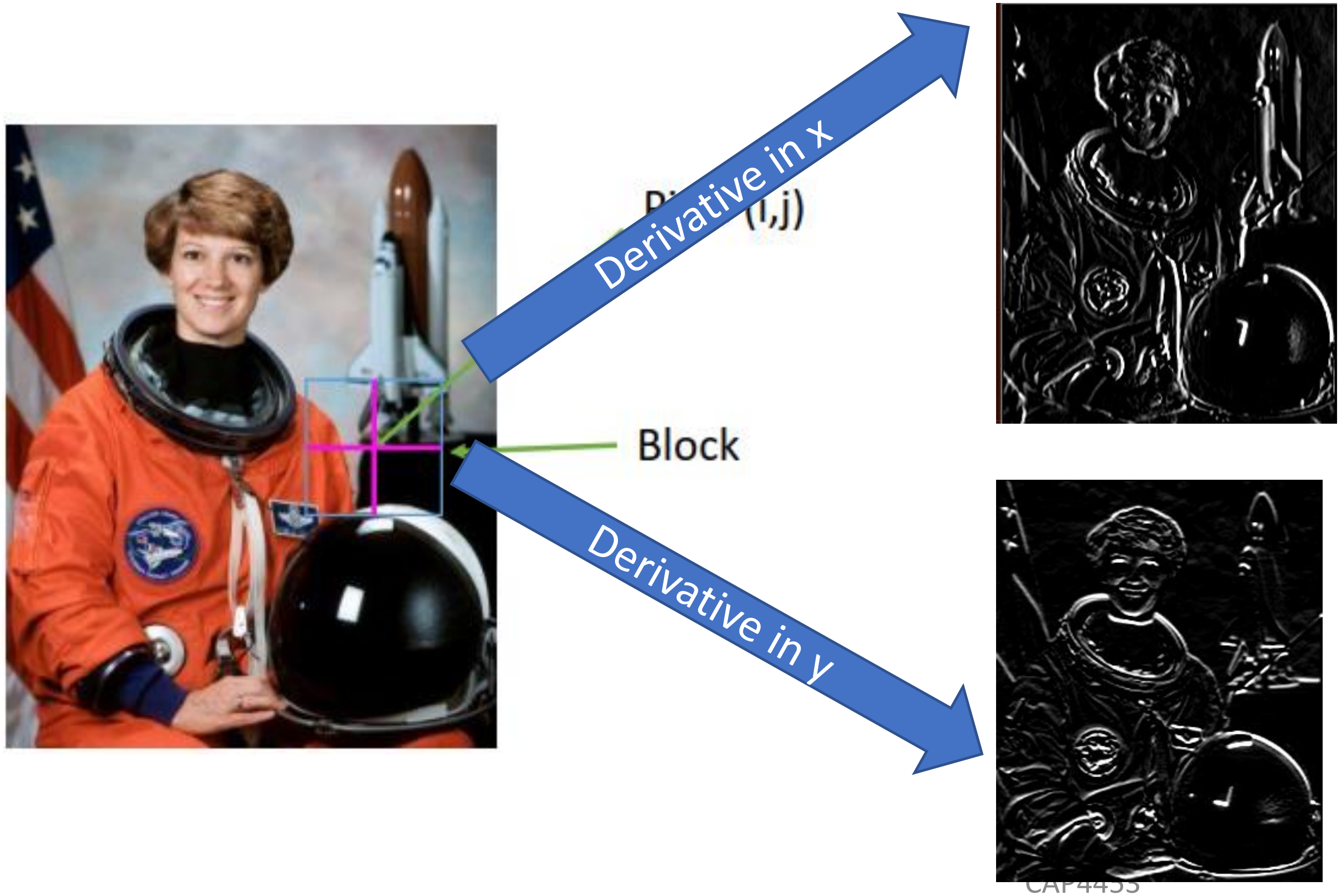


Gradient direction

$$\theta = \tan^{-1} \frac{f_x}{f_y}$$

Histogram of Oriented Gradients (HOG)

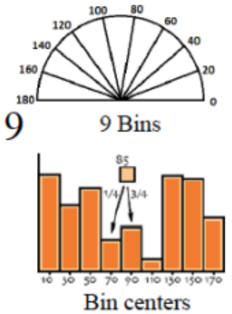
- Step 3: Compute orientation histogram of each cell.



Gradient direction $\theta = \tan^{-1}\dfrac{f_x}{f_y}$

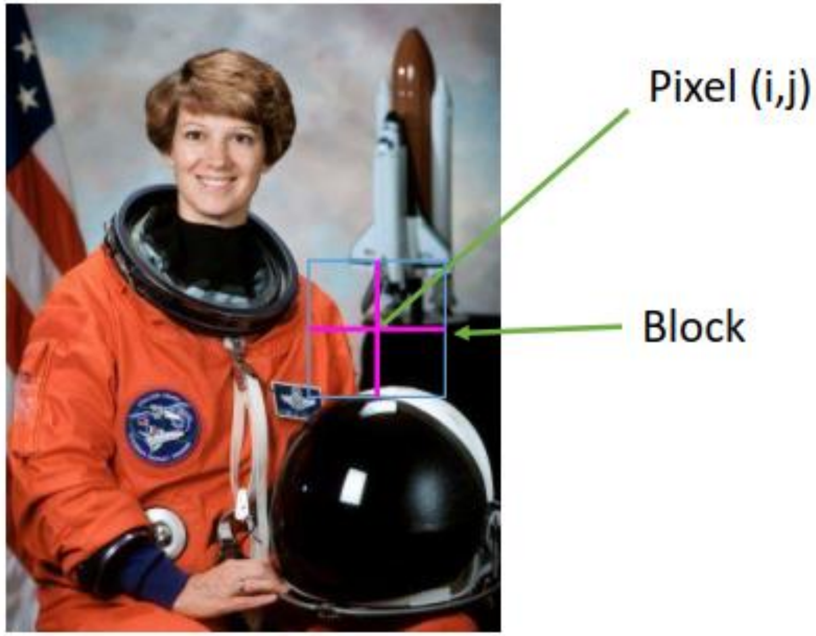
- Cell size is 8x8

- Quantize the gradient orientation into 9 bins (0-180)
  - The vote is the gradient magnitude



Histogram of Oriented Gradients (HOG)

- Step 4: Concatenate the four histograms of each block.



Histogram of Oriented Gradients (HOG)

Let vector v be concatenation of the four histograms from step 4.

- Step 5: Normalize v.

Here we have three options for how to do it:

  - Option 1: Divide v by its Euclidean norm.
  - Option 2: Divide v by its L1 norm (the L1 norm is the sum of all absolute values of v).
  - Option 3:
    - Divide v by its Euclidean norm.
    - In the resulting vector, clip any value over 0.2
    - Then, renormalize the resulting vector by dividing again by its Euclidean norm.

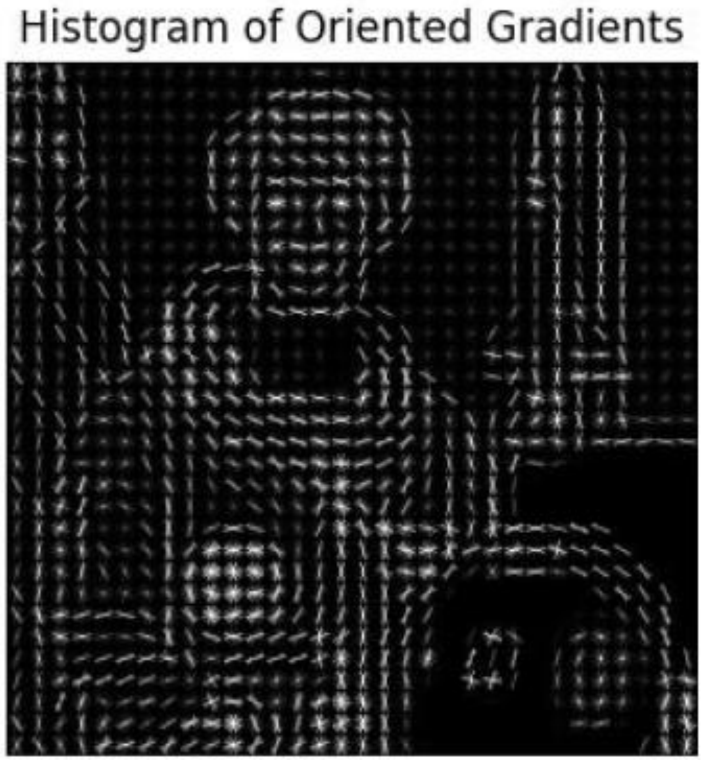
Summary of HOG computation

- Step 1: Extract a square window (called “block”) of some size around the pixel location of interest.
- Step 2: Divide block into a square grid of sub-blocks (called “cells”) (2x2 grid in our example, resulting in four cells).
- Step 3: Compute orientation histogram of each cell.
- Step 4: Concatenate the four histograms.
- Step 5: normalize v using one of the three options described previously.

Histogram of Oriented Gradients (HOG)

- Parameters and design options:
- Angles range from 0 to 180 or from 0 to 360 degrees?
  - In the Dalal & Triggs paper, a range of 0 to 180 degrees is used, and
  - HOGs are used for detection of pedestrians.
- Number of orientation bins.
  - Usually 9 bins, each bin covering 20 degrees.
- Cell size.
  - Cells of size 8x8 pixels are often used.
- Block size.
  - Blocks of size 2x2 cells (16x16 pixels) are often used.
- Usually a HOG feature has 36 dimensions.
  - 4 cells * 9 orientation bins.

Histogram of Oriented Gradients (HOG)



Scale Invariant Feature Transform (SIFT)

A feature detector and a feature descriptor

Scale Invariant Feature Transform (SIFT)

- Lowe., D. 2004, IJCV

cited > 58K

Distinctive Image Features from Scale-Invariant Keypoints

DAVID G. LOWE
Computer Science Department, University of British Columbia, Vancouver, B.C., Canada

Received January 10, 2003; Revised January 7, 2004; Accepted January 22, 2004

Abstract. This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition can robustly identify objects among clutter and occlusion while achieving near real-time performance.

Keywords: invariant features, object recognition, scale invariance, image matching

1. Introduction

Image matching is a fundamental aspect of many problems in computer vision, including object or scene recognition, solving for 3D structure from multiple images, stereo correspondence, and motion tracking. This paper describes image features that have many properties that make them suitable for matching differing images of an object or scene. The features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. They are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for object and scene recognition.

The cost of extracting these features is minimized by taking a cascade filtering approach, in which the more expensive operations are applied only at locations that pass an initial test. Following are the major stages of computation used to generate the set of image features:

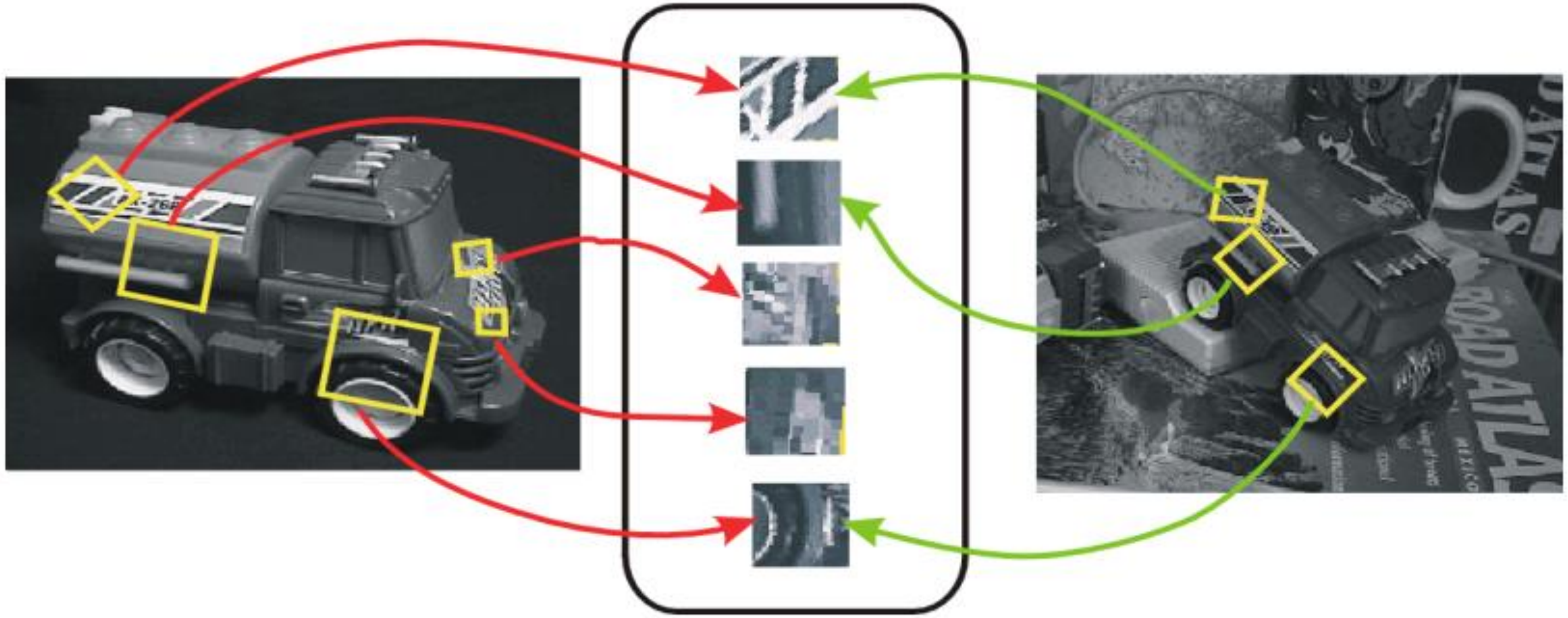
1. *Scale-space extrema detection:* The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.
2. *Keypoint localization:* At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
3. *Orientation assignment:* One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

Scale Invariant Feature Transform (SIFT)

- Image content is transformed into local feature coordinates
- Invariant to
  - translation
  - rotation
  - scale, and
  - other imaging parameters

Scale Invariant Feature Transform (SIFT)

- Image content is transformed into local feature coordinates



Scale Invariant Feature Transform (SIFT)

- Procedure at High Level

Scale-Space Extrema Detection	Search over multiple scales and image locations
KeyPoint Localization	Fit a model to determine location and scale. Select KeyPoints based on a measure of stability.
Orientation Assignment	Compute best orientation(s) for each keyPoint region.
KeyPoint Description	Use local image gradients at selected scale and rotation to describe each keyPoint region.

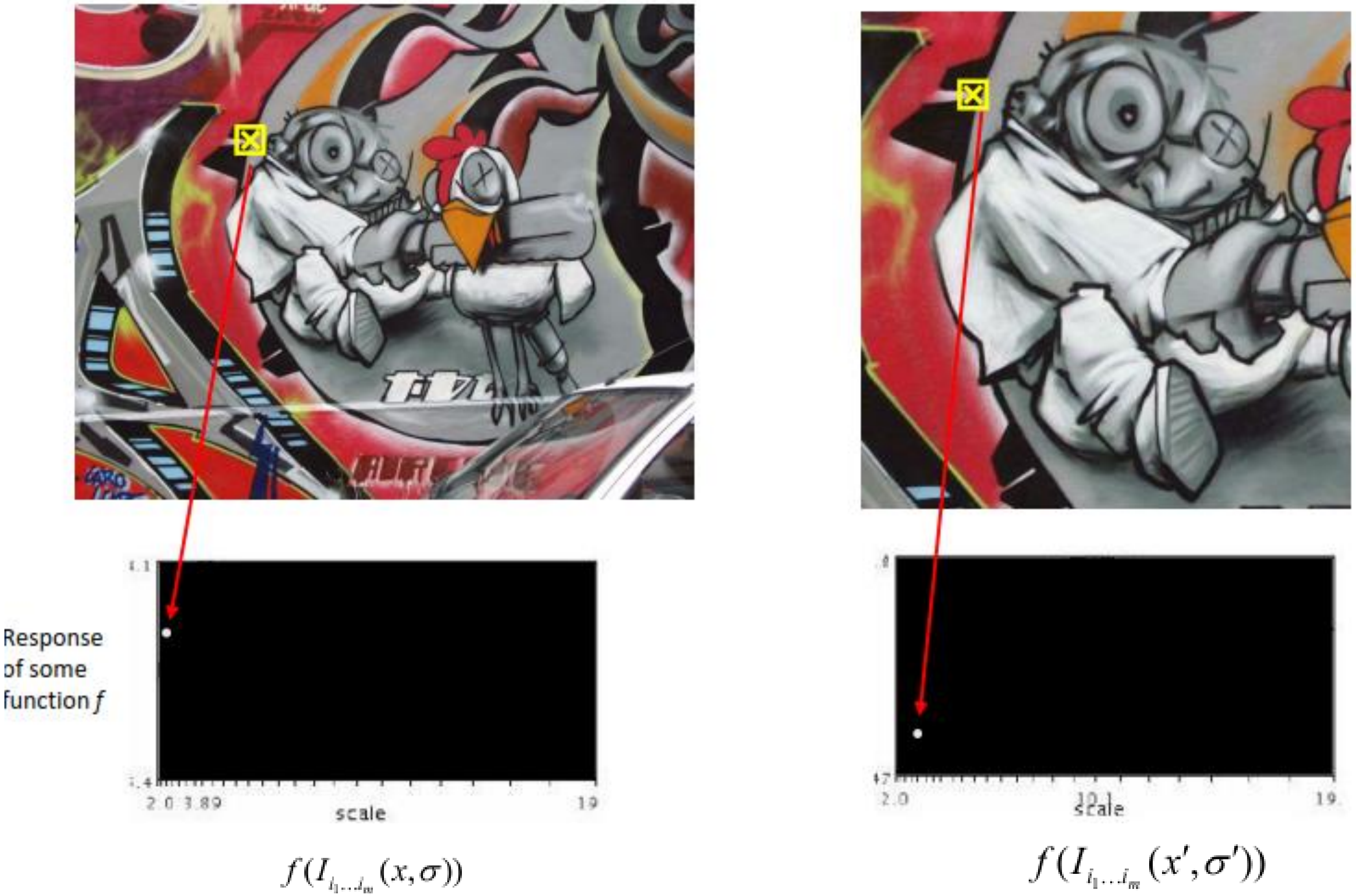
SIFT. Automatic scale selection

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

How to find patch sizes at which *f* response is equal?

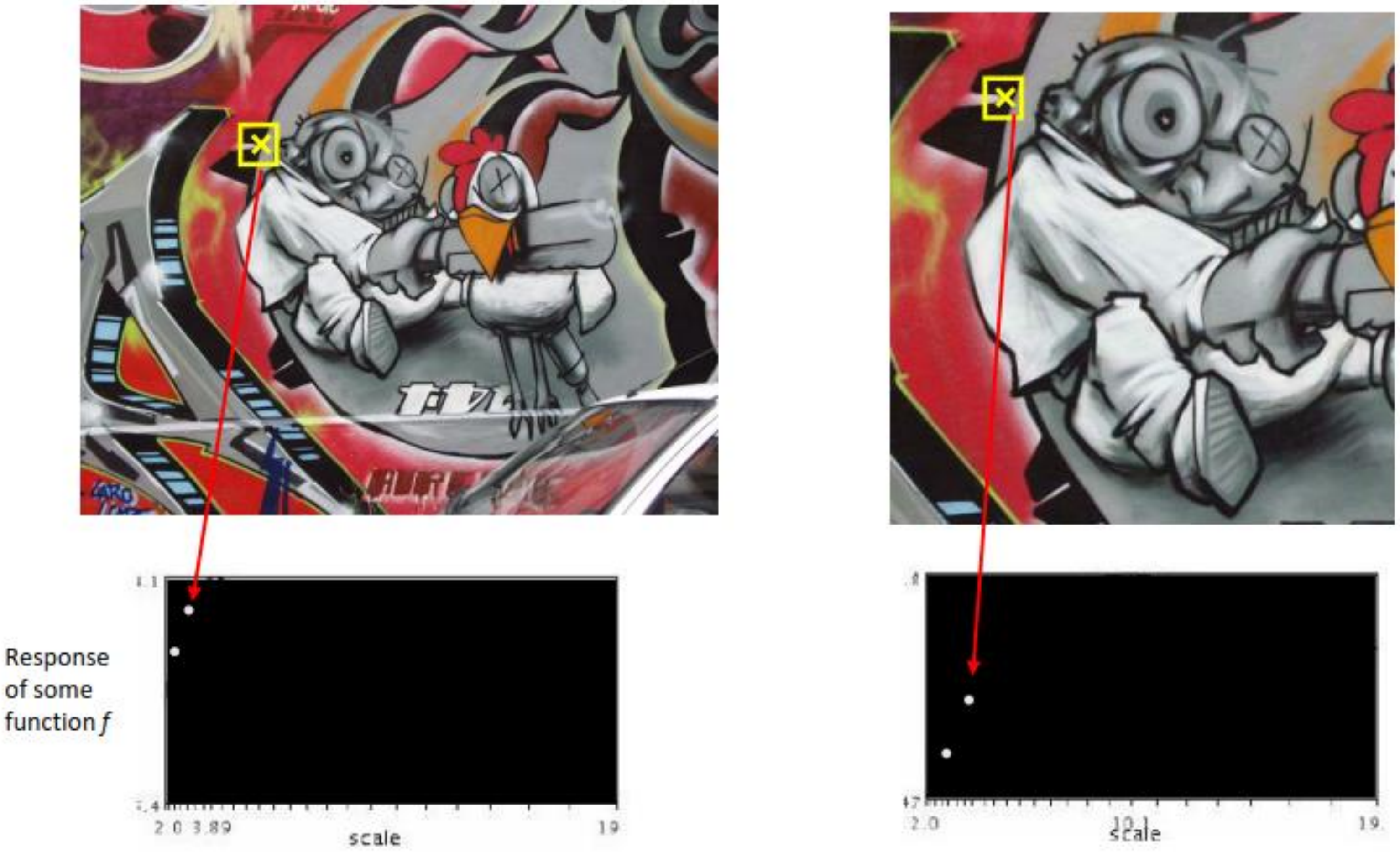
What is a good *f* ?

SIFT. Automatic scale selection



Function responses for increasing scale (scale signature)

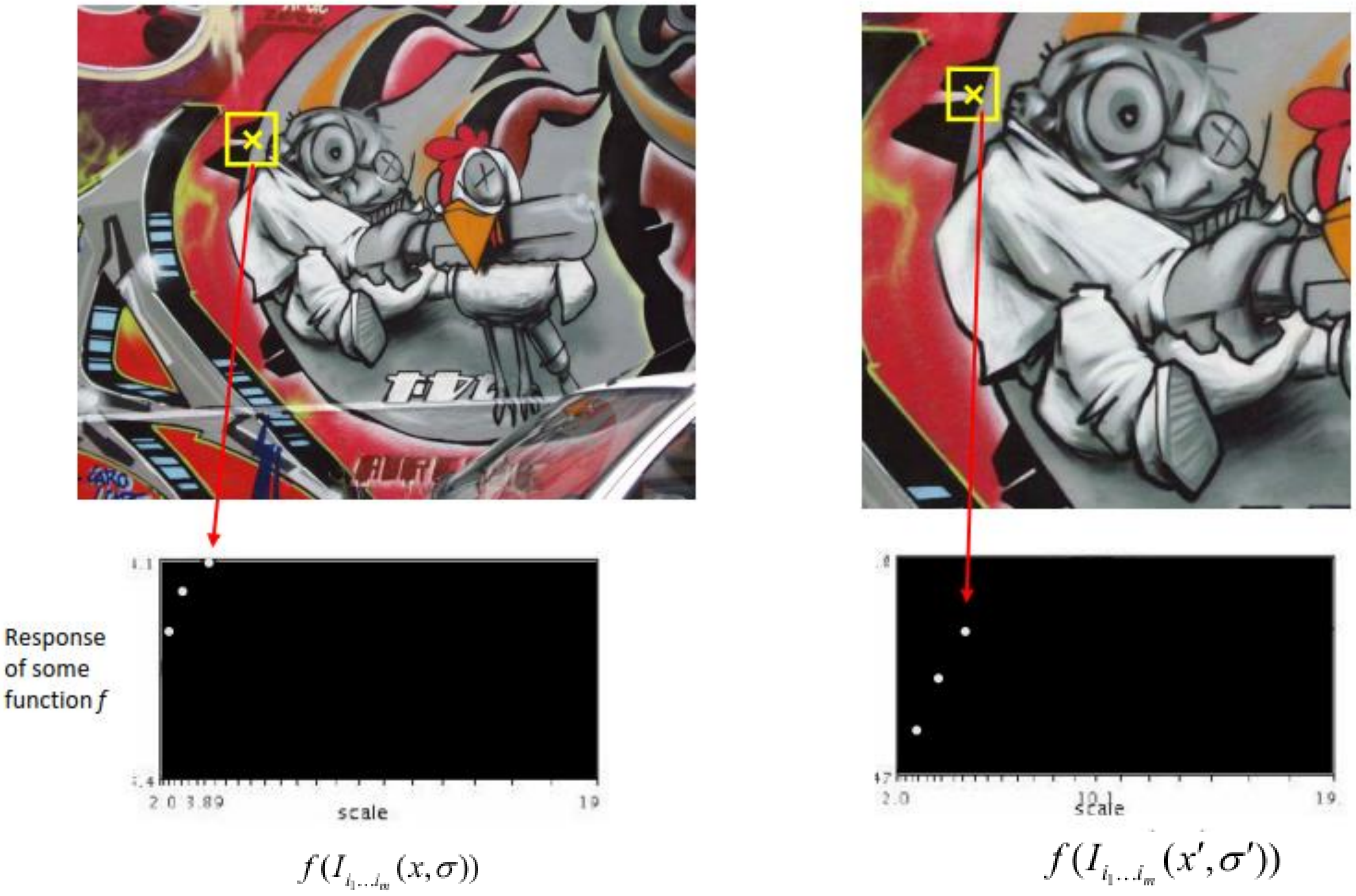
SIFT. Automatic scale selection



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

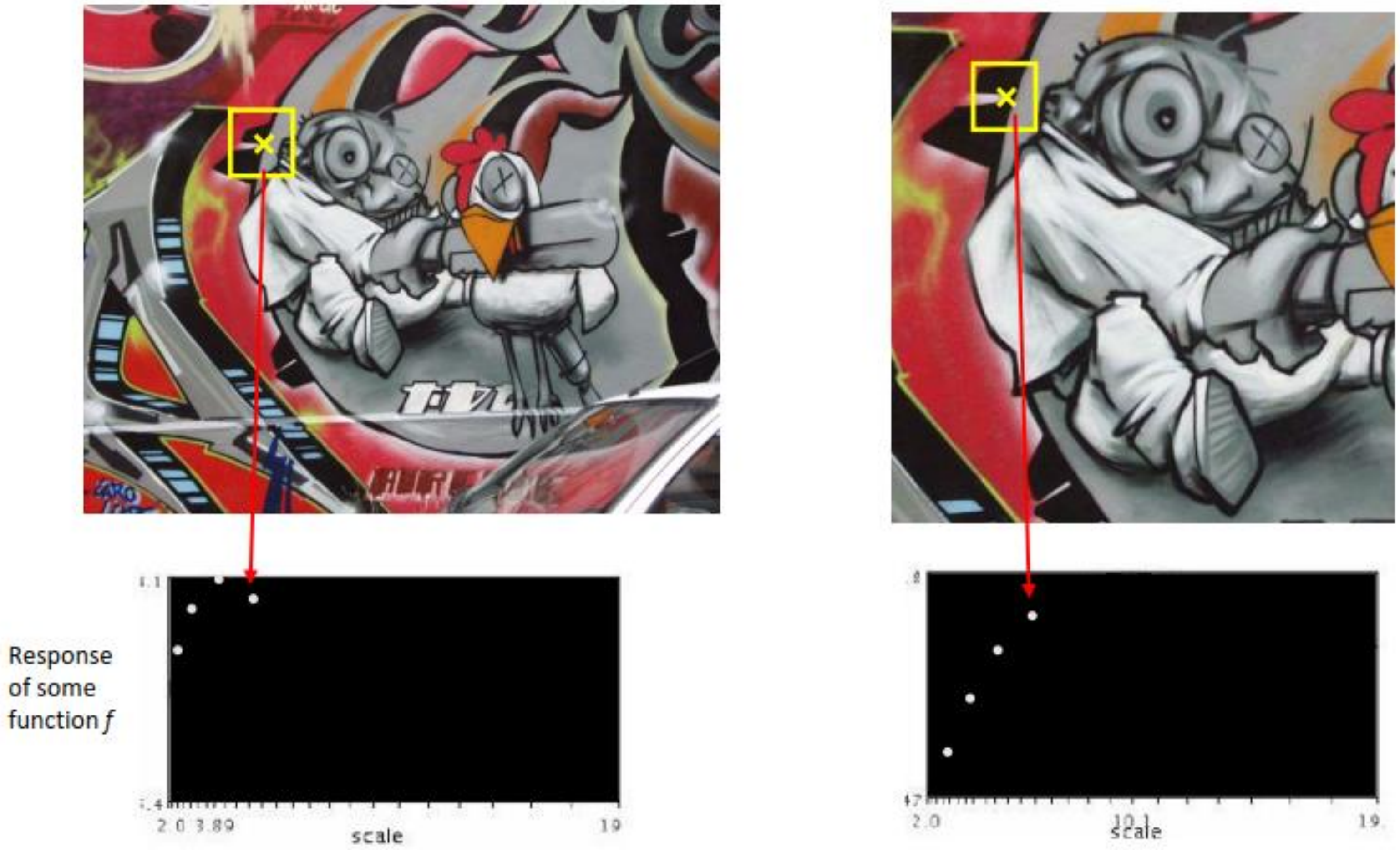
Function responses for increasing scale (scale signature)

SIFT. Automatic scale selection



Function responses for increasing scale (scale signature)

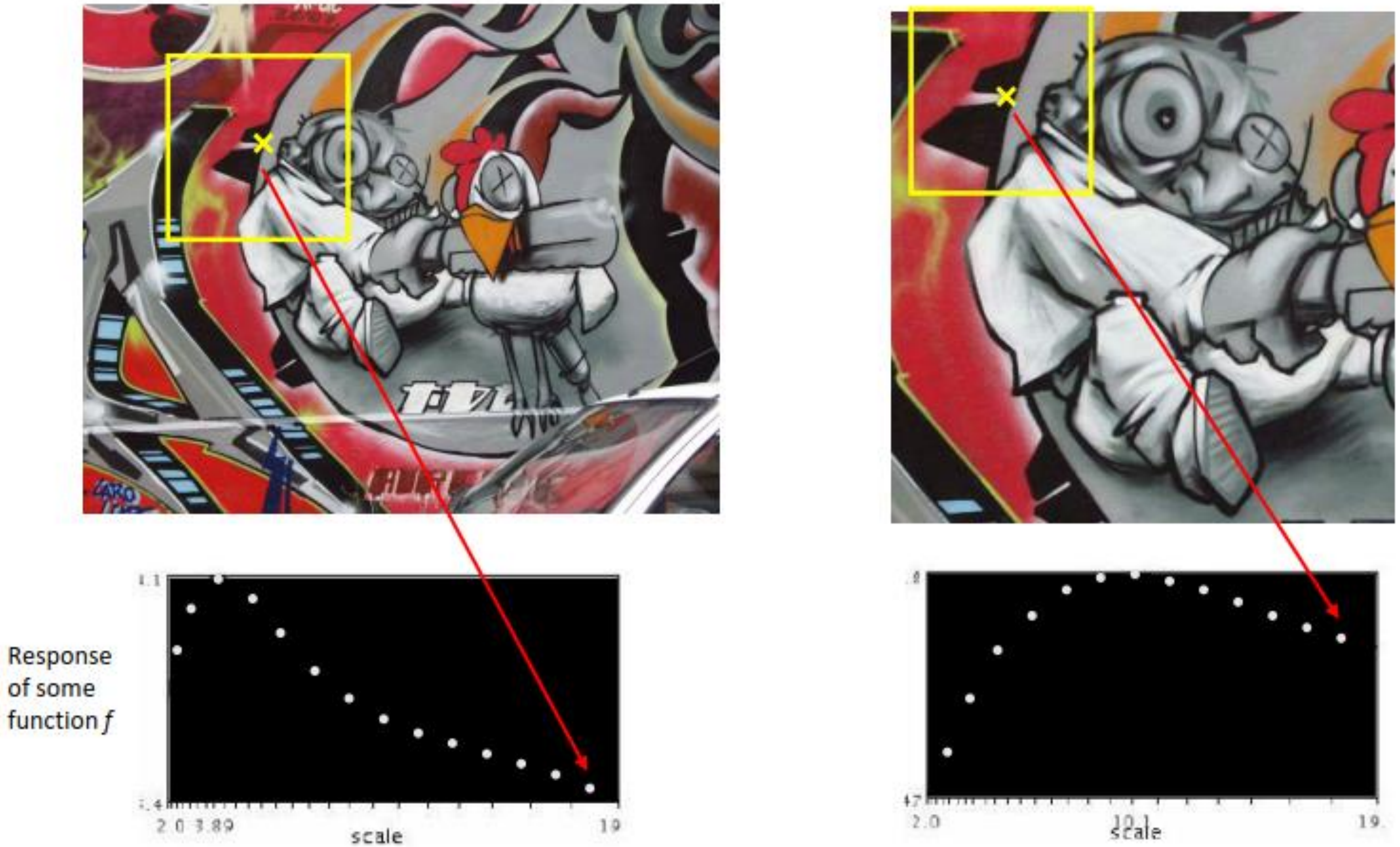
SIFT. Automatic scale selection



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

Function responses for increasing scale (scale signature)

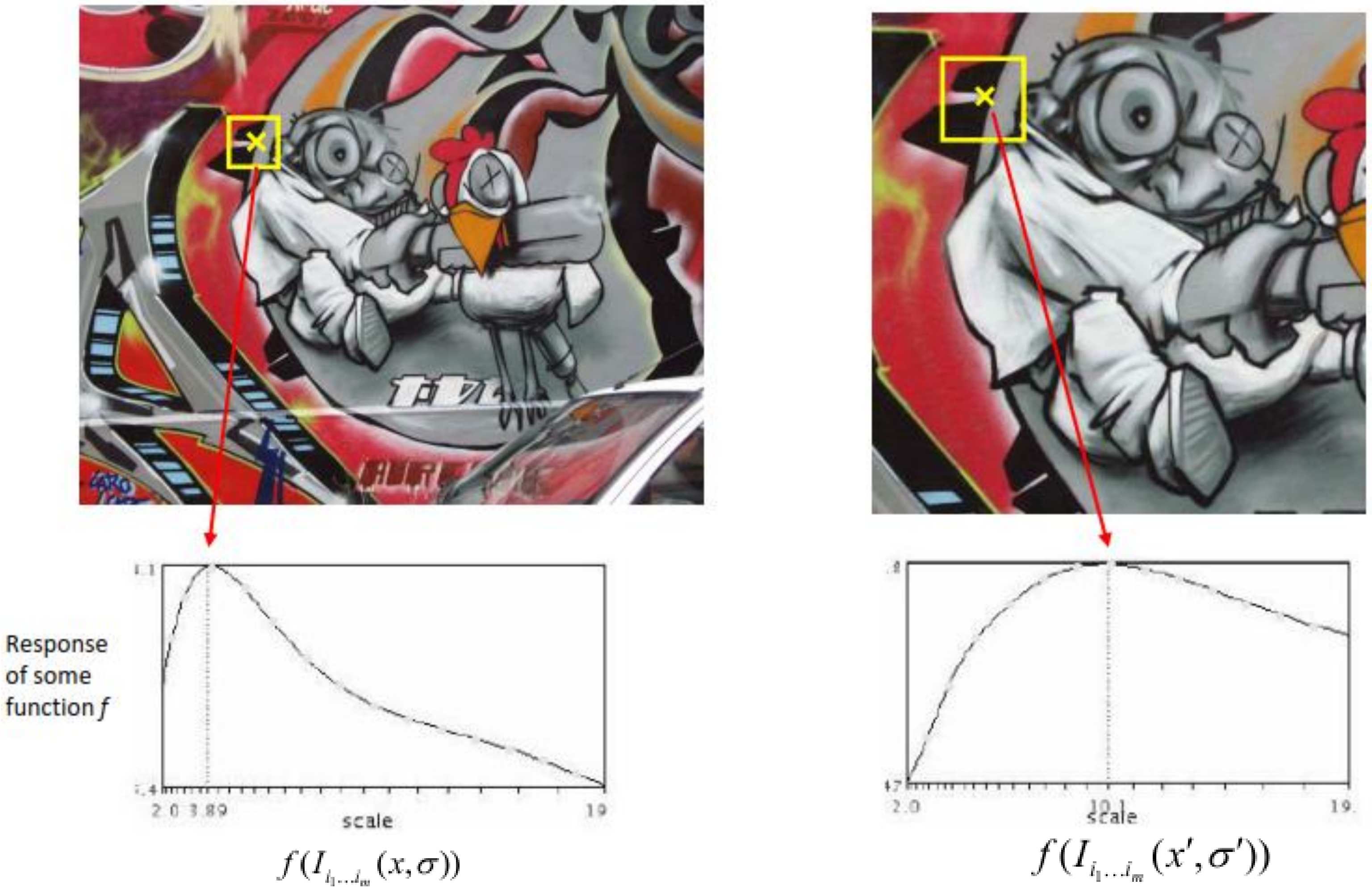
SIFT. Automatic scale selection



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

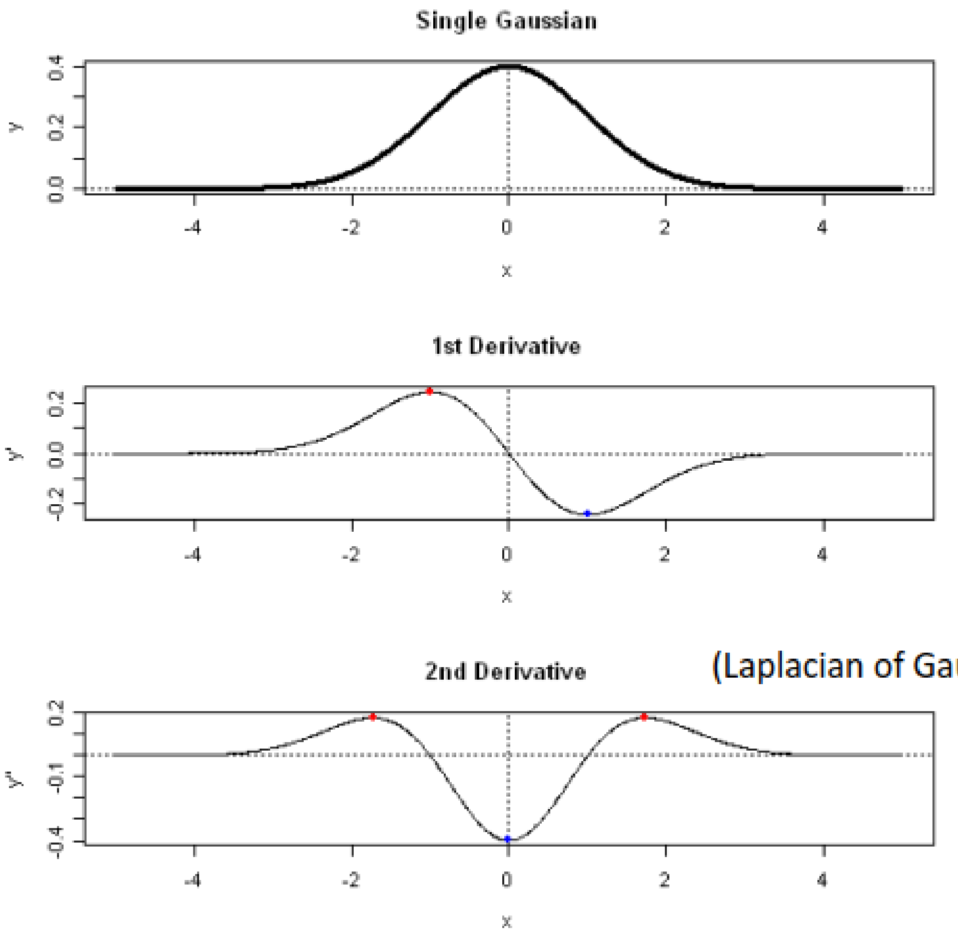
Function responses for increasing scale (scale signature)

SIFT. Automatic scale selection



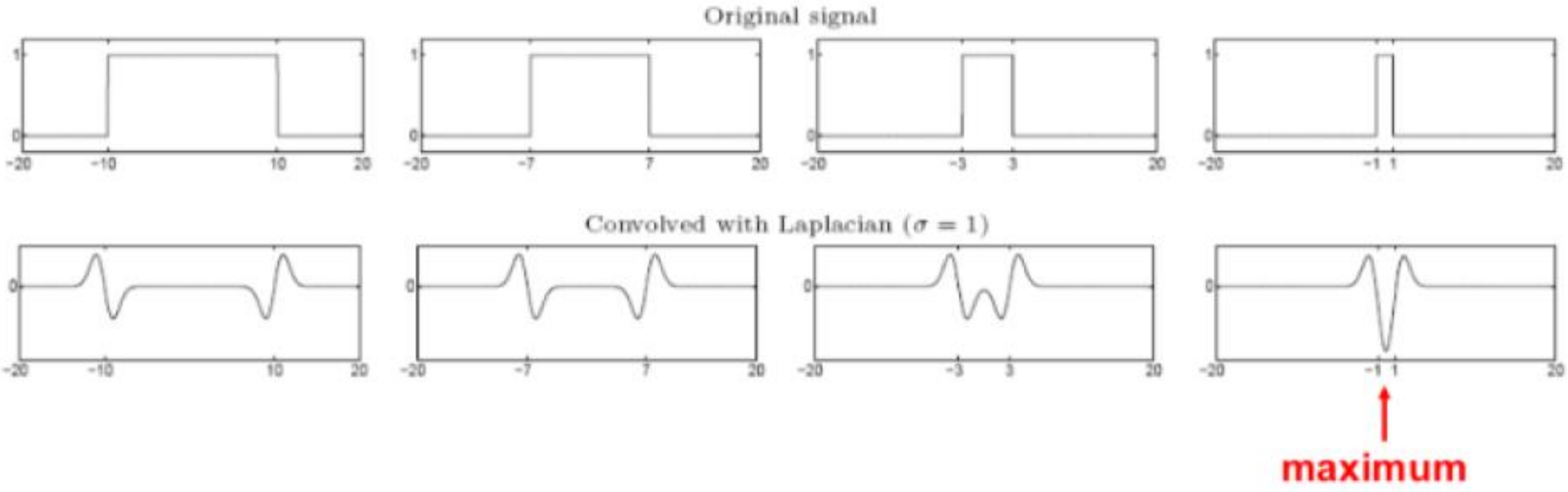
Function responses for increasing scale (scale signature)

What is a useful signature function f?



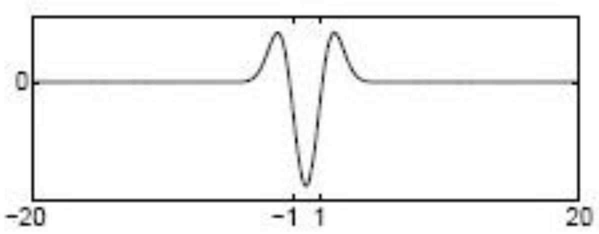
(Laplacian of Gaussian)

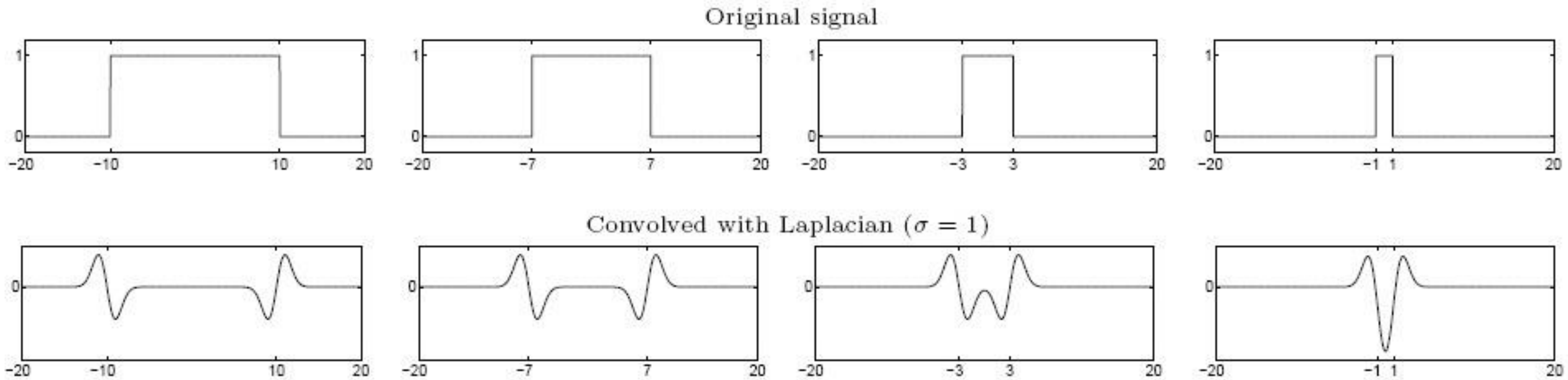
Blob detection



Formally…

Laplacian filter



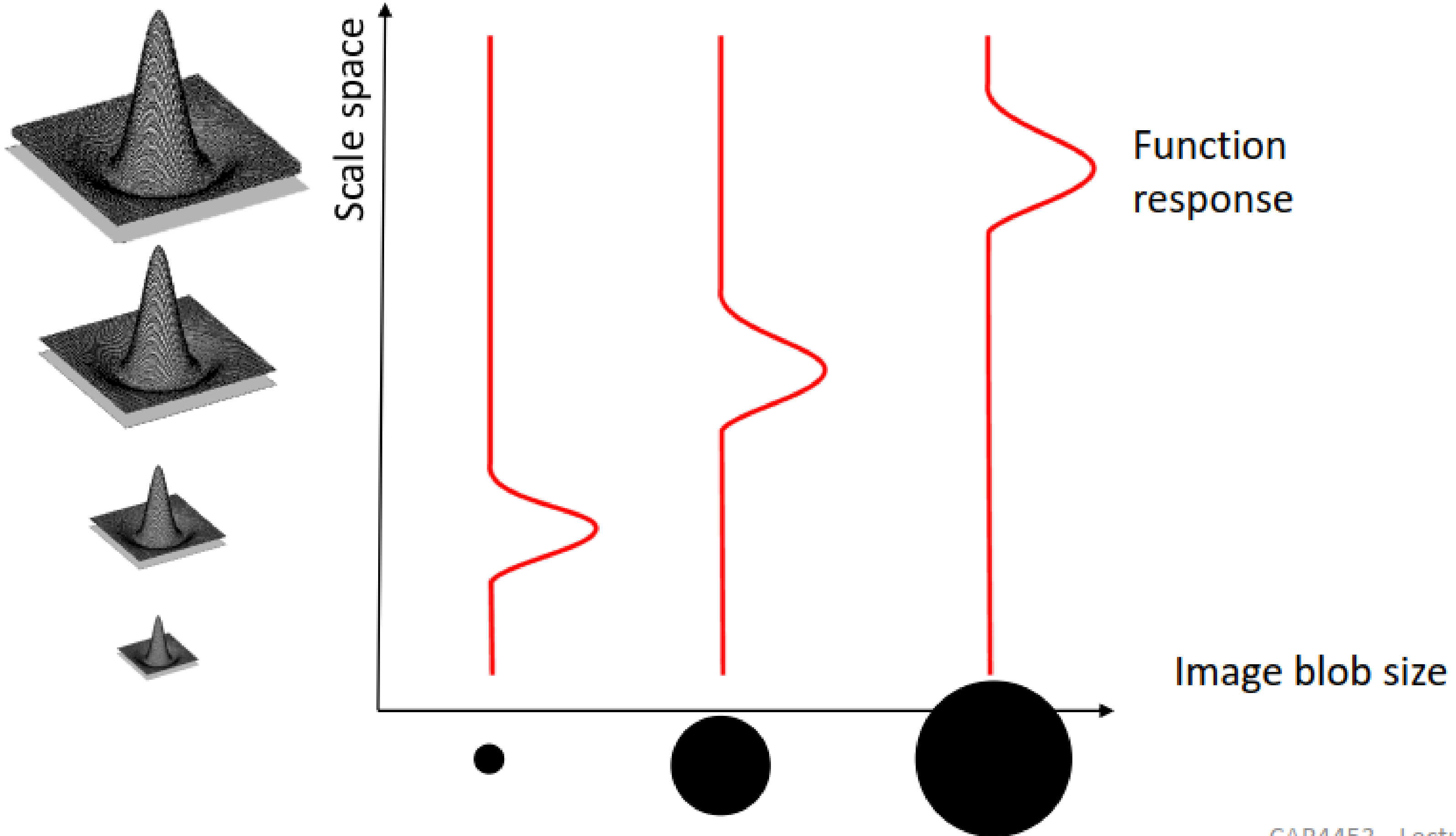


Highest response when the signal has the same **characteristic scale** as the filter

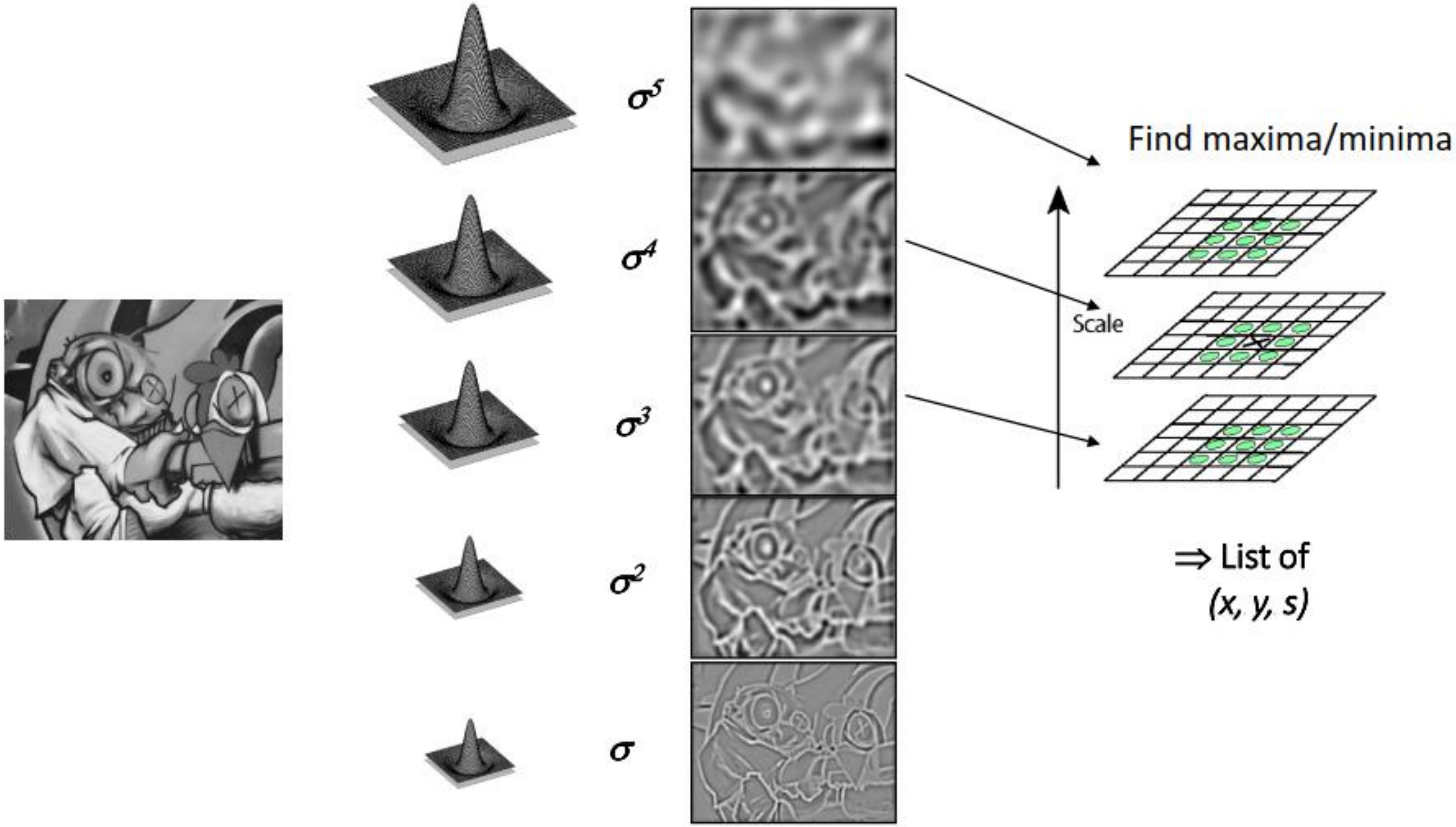
What is a useful signature function f?

"Blob" detector is common for corners

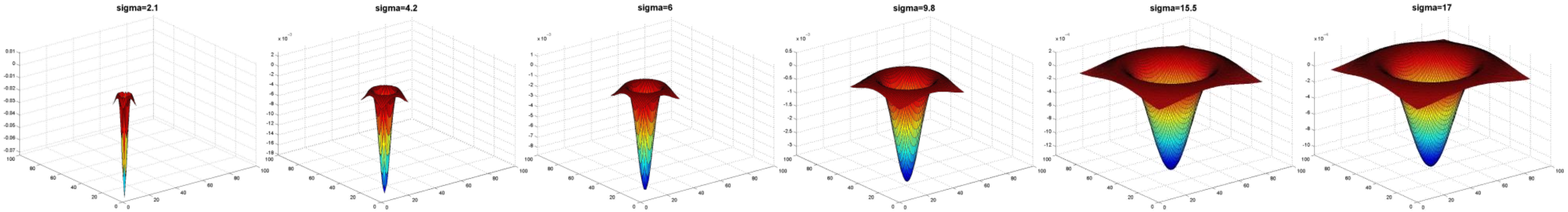
- Laplacian (2nd derivative) of Gaussian (LoG)



Find local maxima in position-scale space



What happens if you apply different Laplacian filters?



Full size

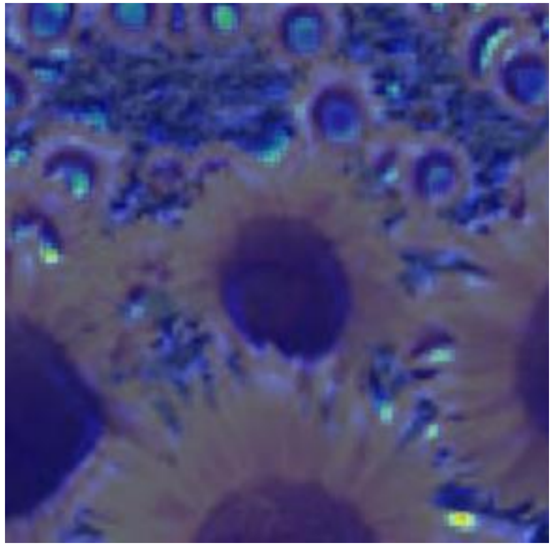
3/4 size

What happened when you applied different Laplacian filters?

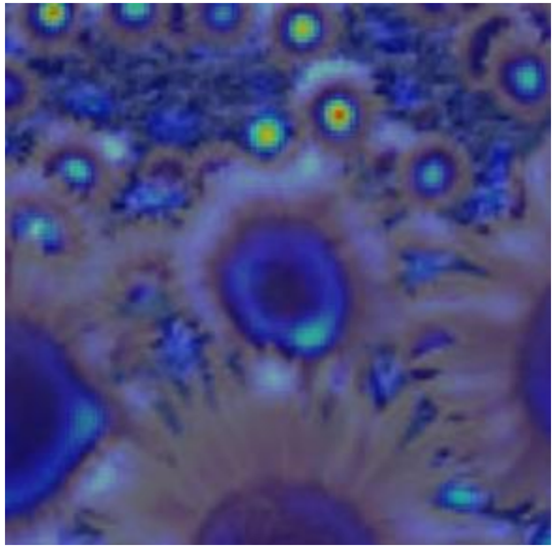
Full size

3/4 size

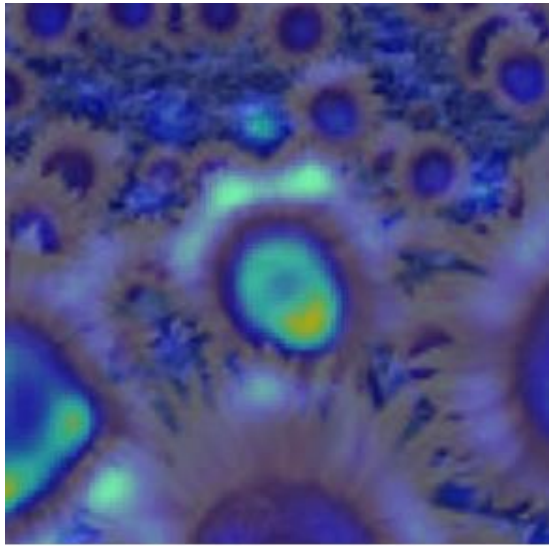
2.1



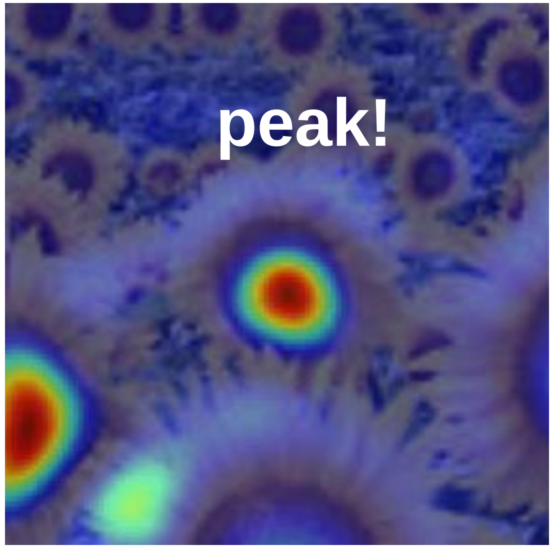
4.2



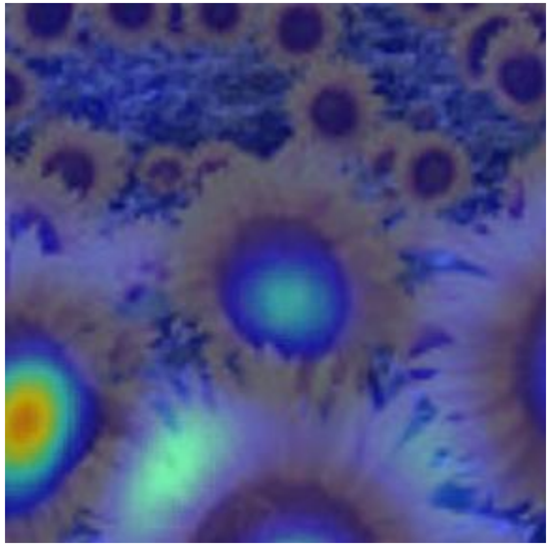
6.0



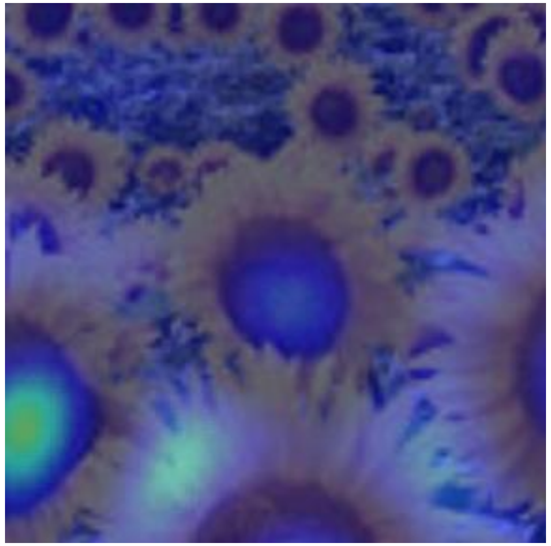
9.8

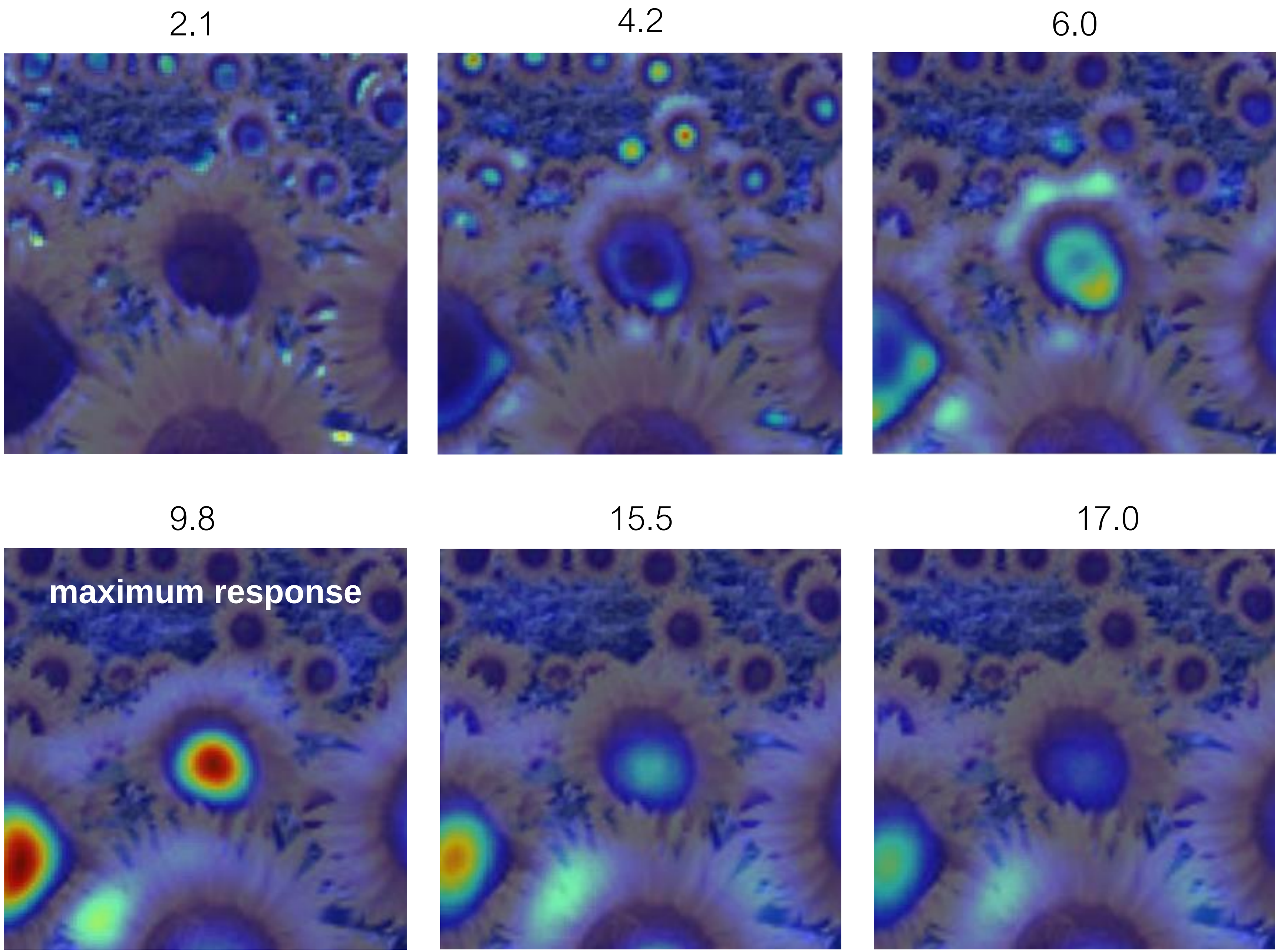


15.5



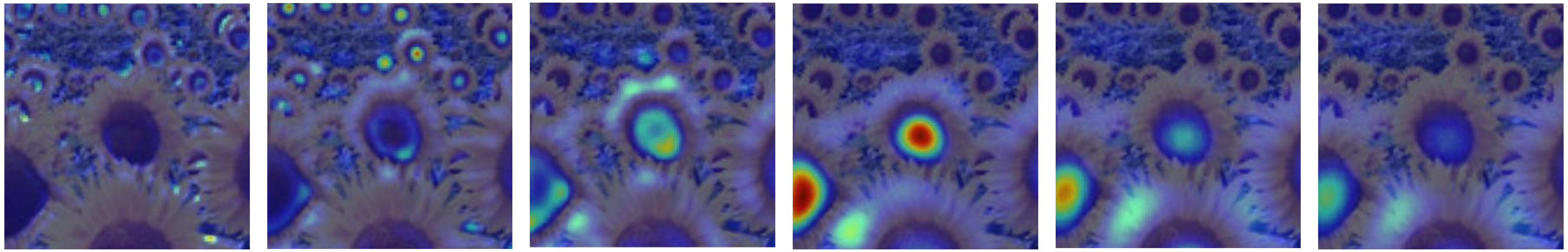
17.0




2.1
4.2
6.0
9.8
maximum response
15.5
17.0

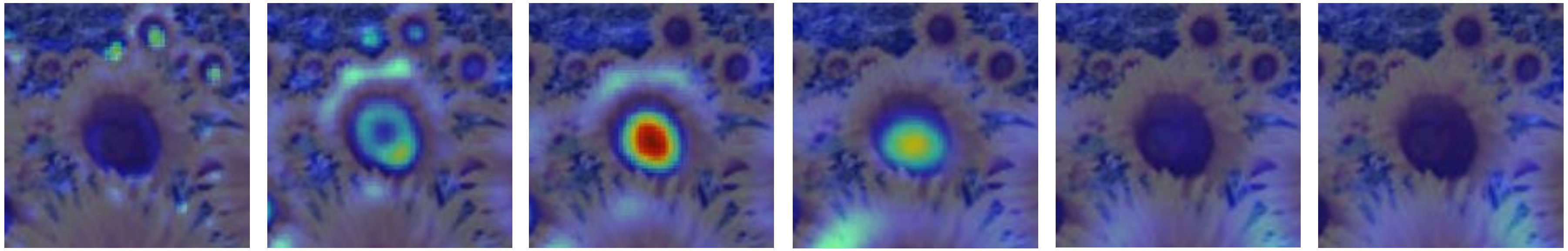
optimal scale

2.1 4.2 6.0 9.8 15.5 17.0



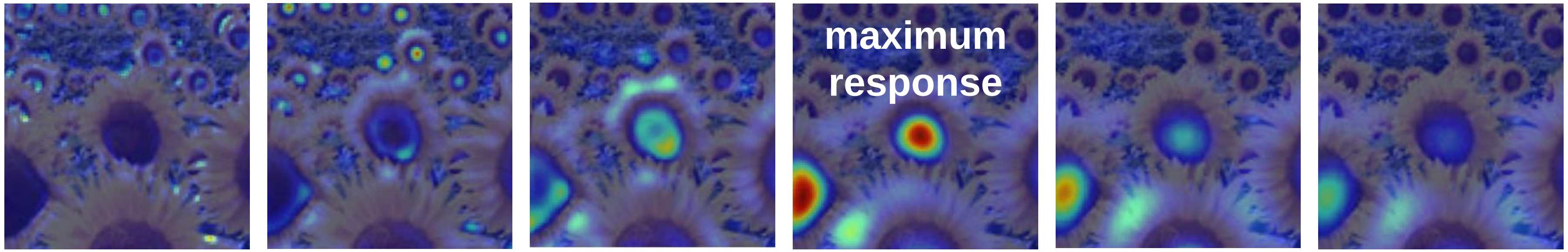
Full size image

2.1 4.2 6.0 9.8 15.5 17.0

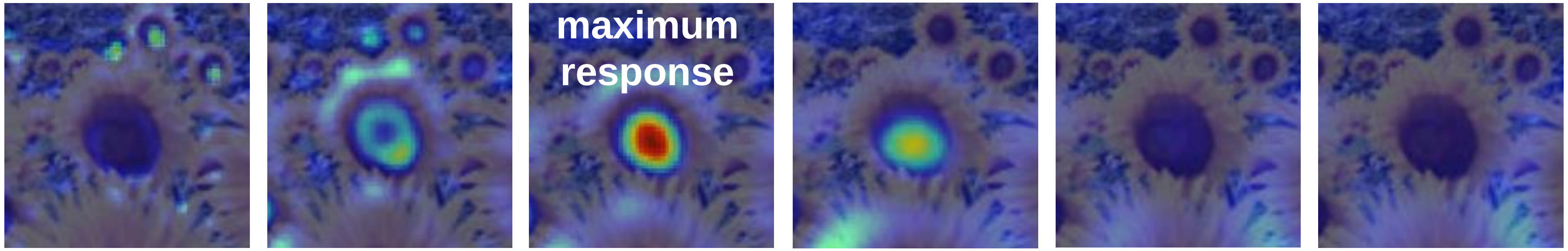


3/4 size image

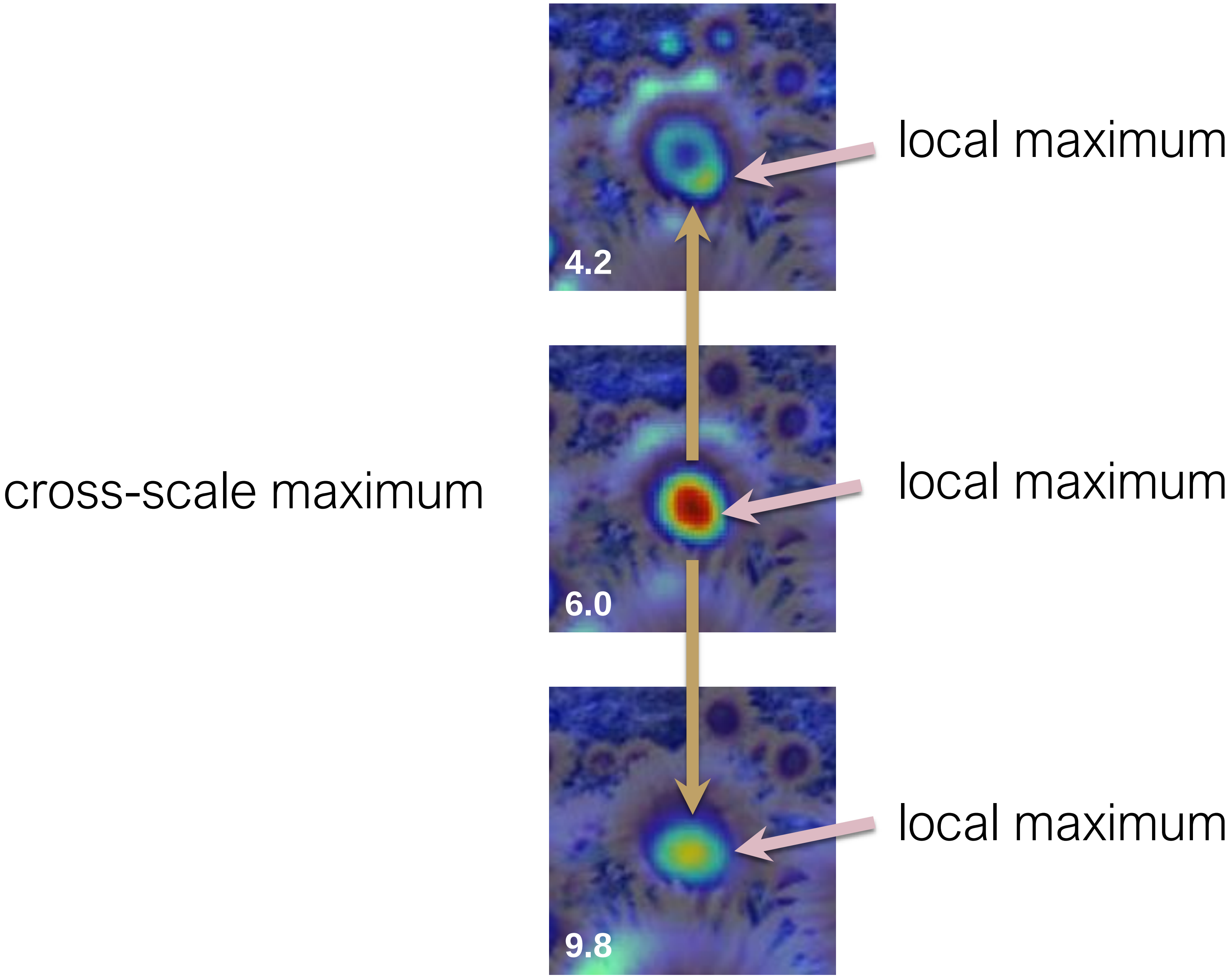
optimal scale



Full size image



3/4 size image


local maximum
4.2
cross-scale maximum
local maximum
6.0
local maximum
9.8

Scale Invariant Detection

- Functions for determining scale $f = \text{Kernel} * \text{Image}$

Kernels:

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

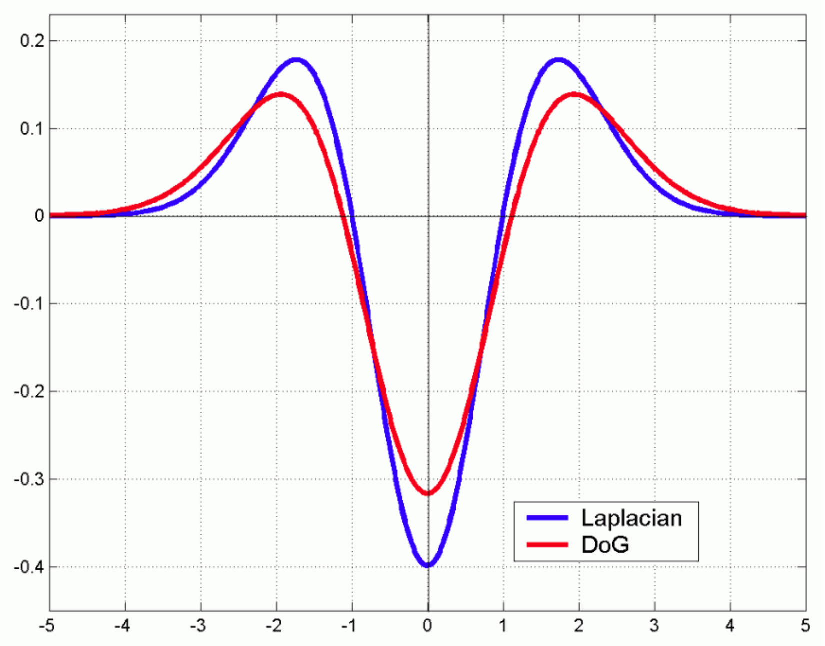
(Laplacian)

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

(Difference of Gaussians)

where Gaussian

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



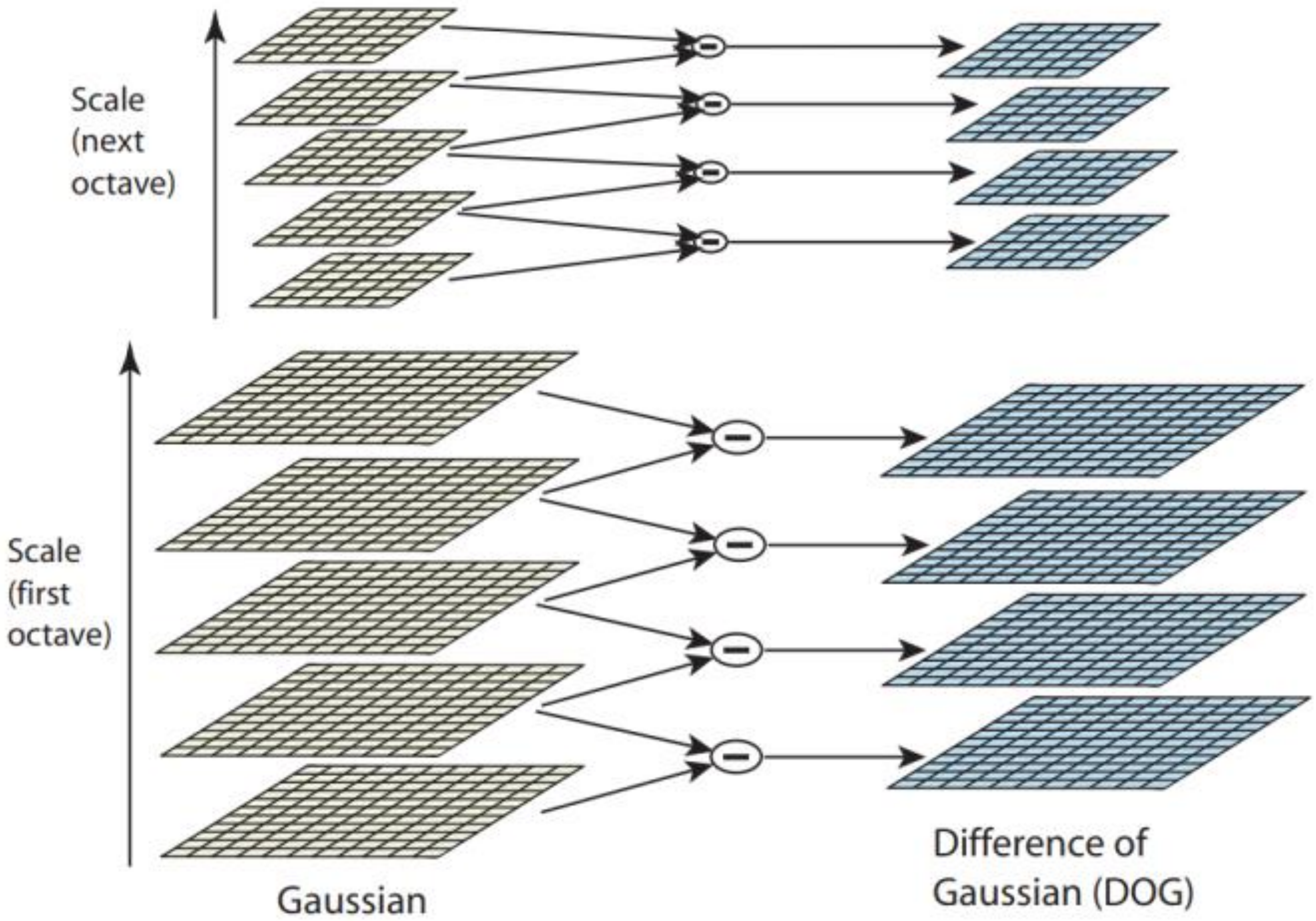
Note: The LoG and DoG operators are both rotation equivariant

Alternative to compute Laplacian of Gaussian

- Approximate LoG with Difference-of-Gaussian (DoG).

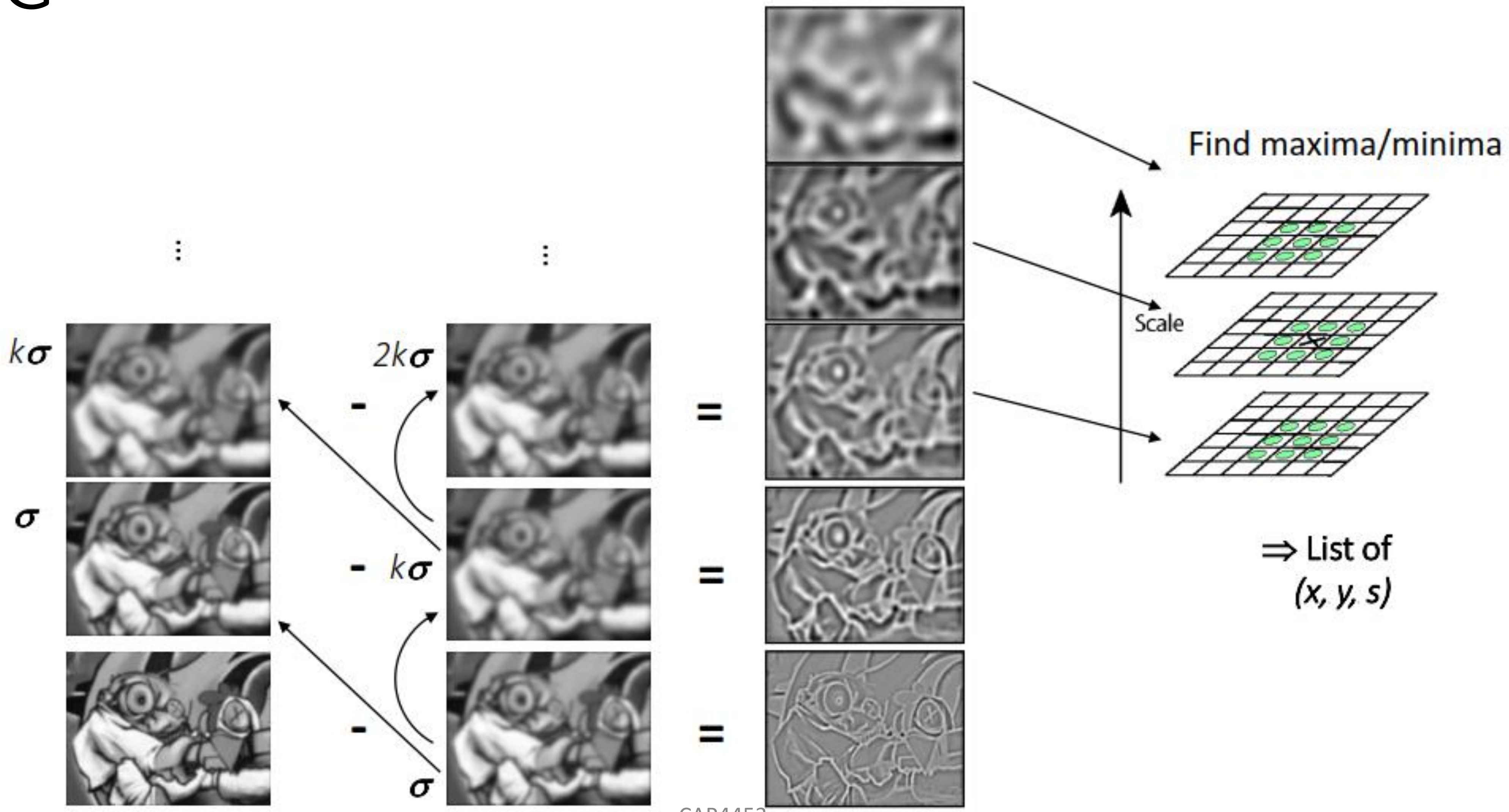
1. Blur image with σ Gaussian kernel
2. Blur image with kσ Gaussian kernel
3. Subtract 2. from 1.

Scale-Space



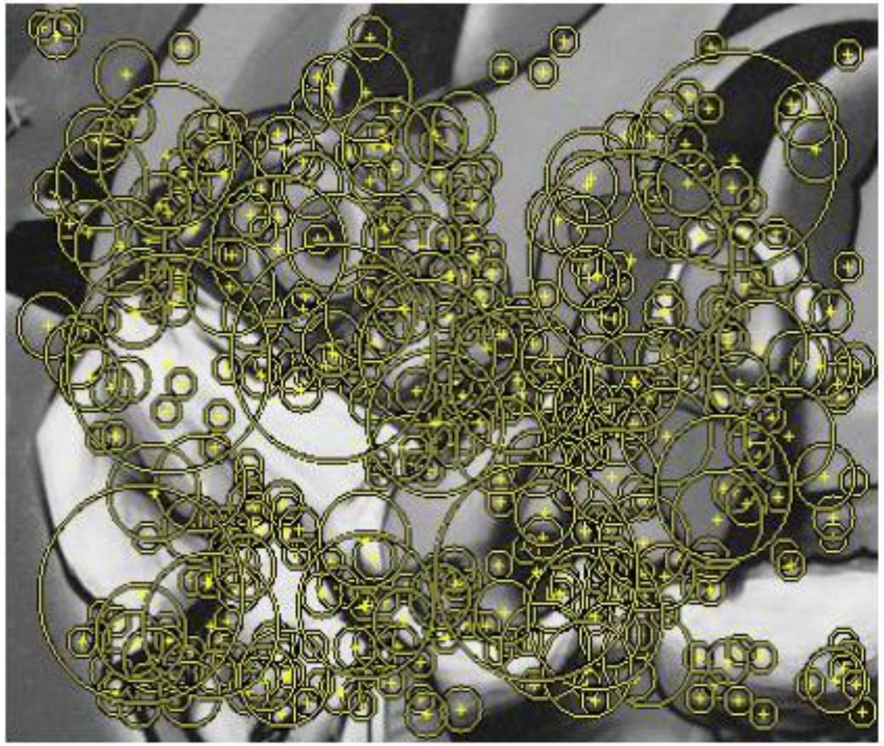
Scale
(next
octave)
Scale
(first
octave)
Gaussian
Difference of
Gaussian (DOG)

Find local maxima in position-scale space of DoG



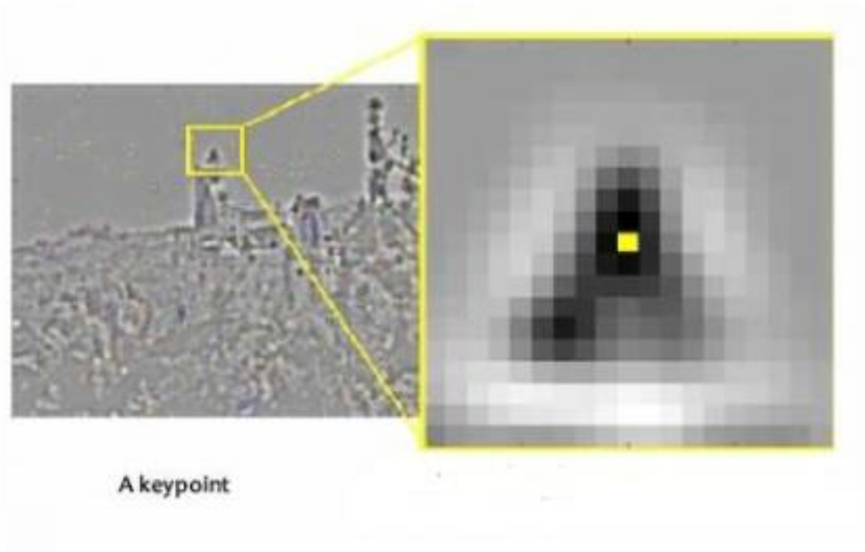
Results: Difference of Gaussians

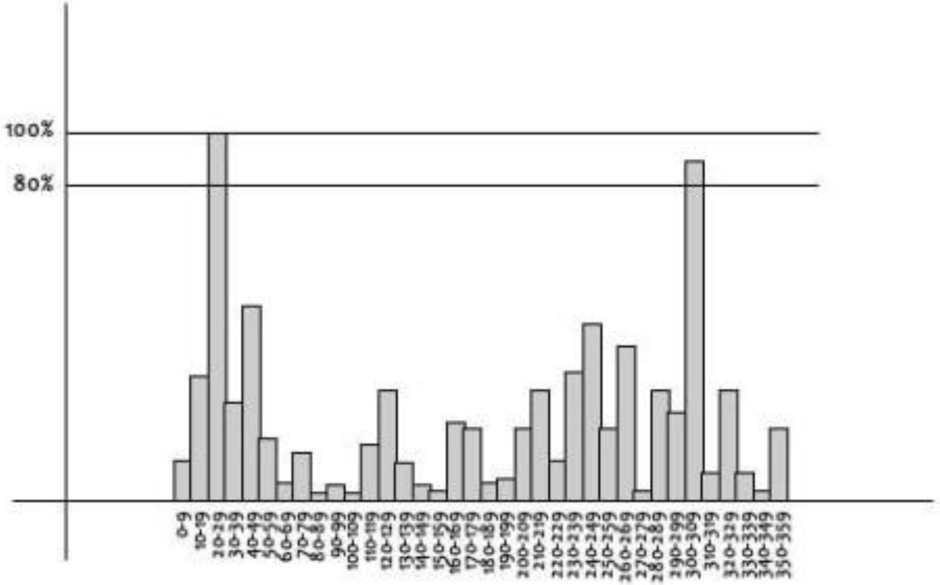
- Larger circles = larger scale
- Descriptors with maximal scale response



SIFT Orientation estimation

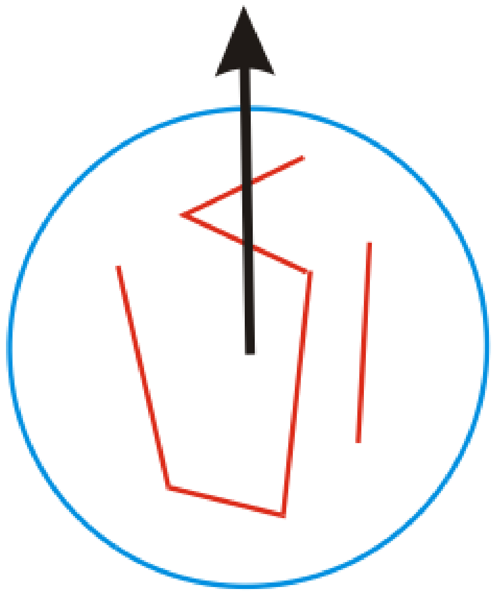
- Compute gradient orientation histogram
- Select dominant orientation ϴ

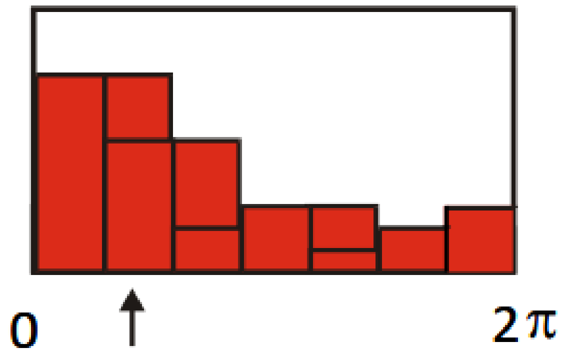




SIFT Orientation Normalization

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



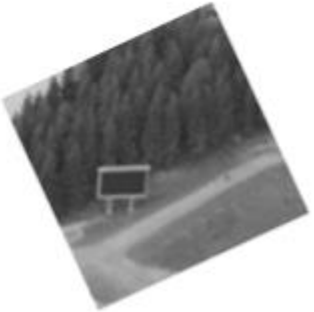


SIFT Detector

- In addition to position x,y of the feature,
  - Scale σ (determined by smoothing value)
  - Orientation of dominant gradient θ

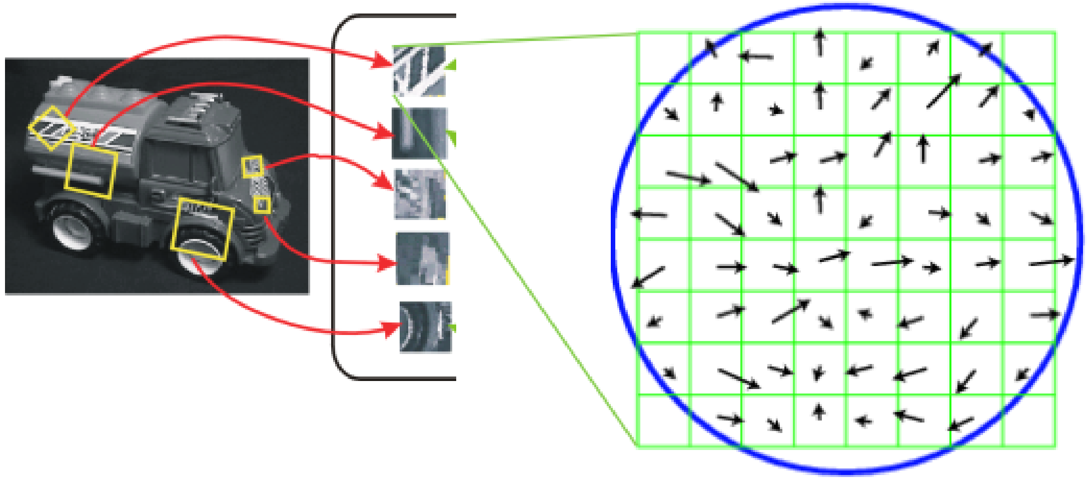
SIFT detections

Patch at detected position, scale, orientation



SIFT descriptor

- Compute on local 16 x 16 window around detection.
- Rotate and scale window according to discovered orientation Θ and scale σ (gain invariance).
- Compute gradients weighted by a Gaussian of variance half the window (for smooth falloff).

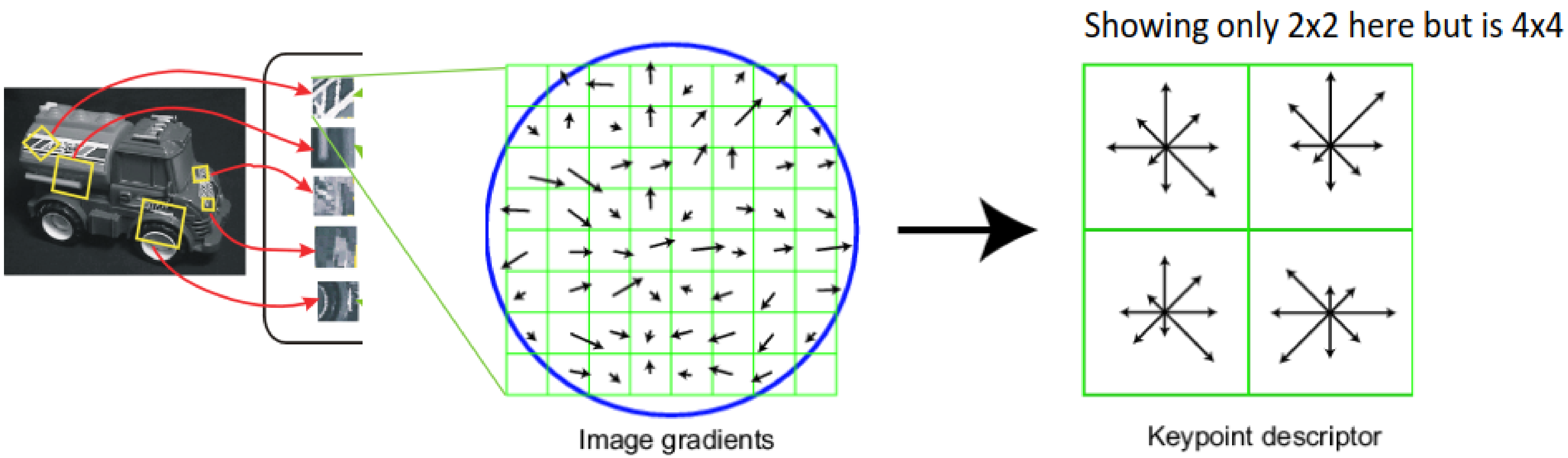


Actually 16x16, only showing 8x8

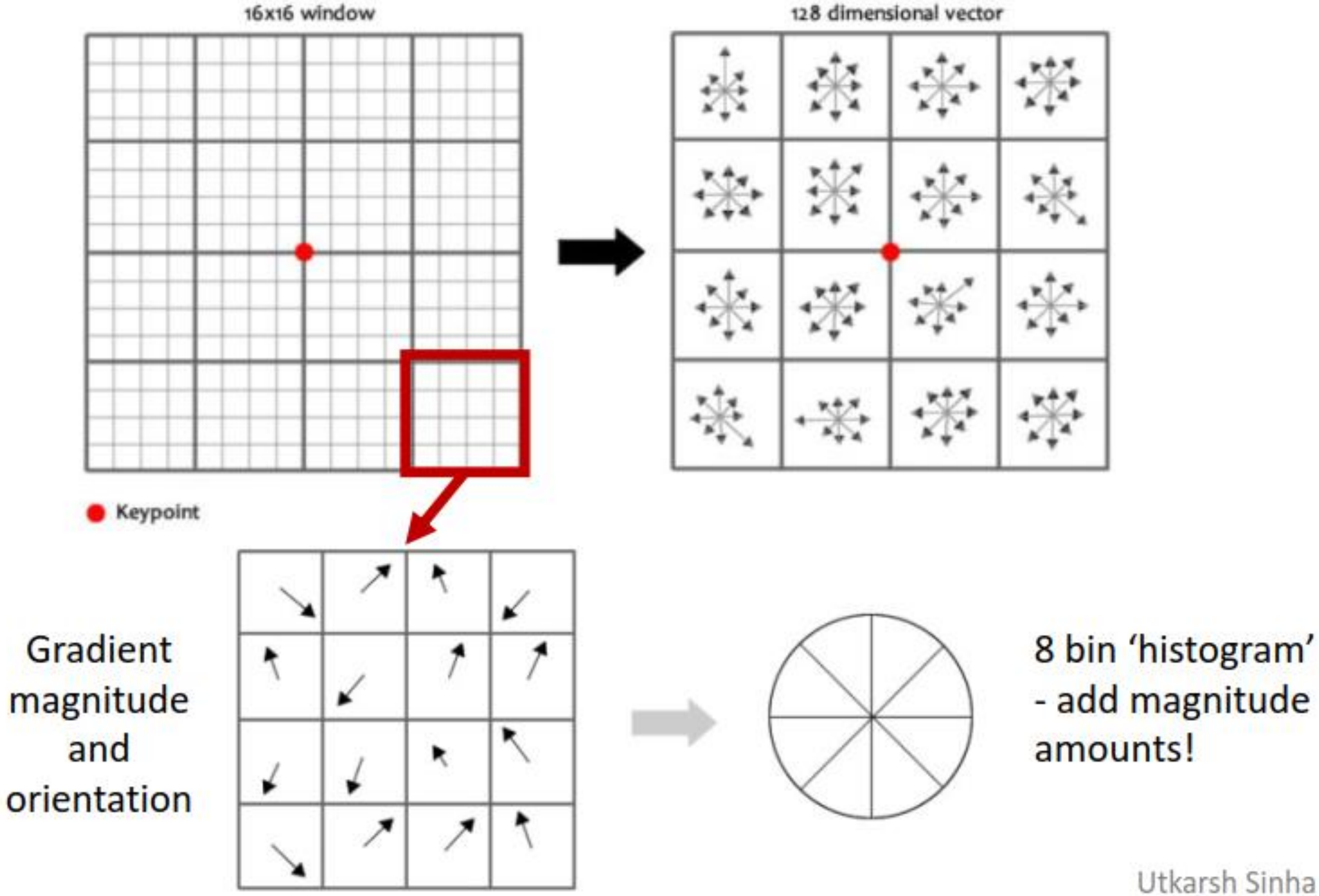
Image gradients

SIFT descriptor

- 4x4 array of gradient orientation histograms weighted by gradient magnitude.
- Bin into 8 orientations x 4x4 array = 128 dimensions.



SIFT Descriptor Extraction

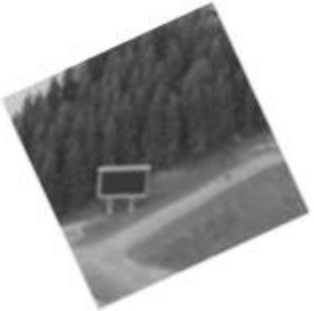


Utkarsh Sinha

SIFT descriptor

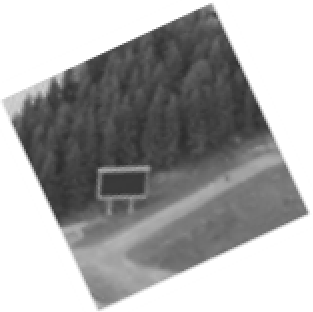
- Extract patch around detected keypoint

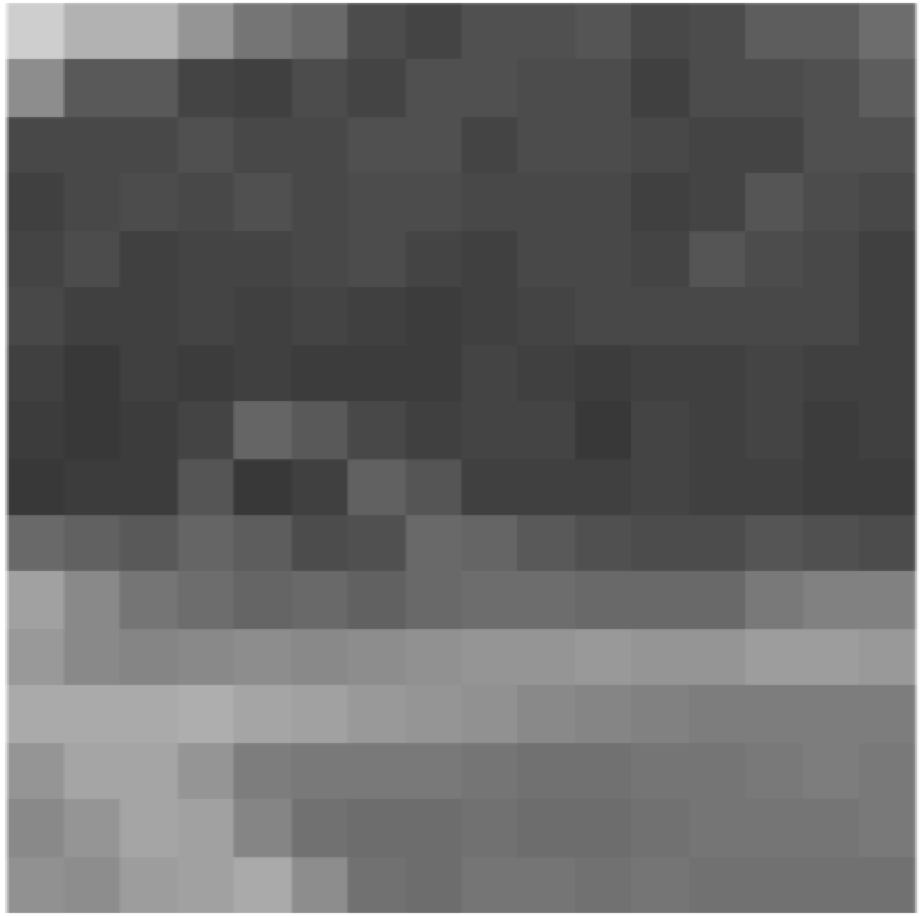
- Normalize the patch
  to canonical scale and orientation



SIFT descriptor

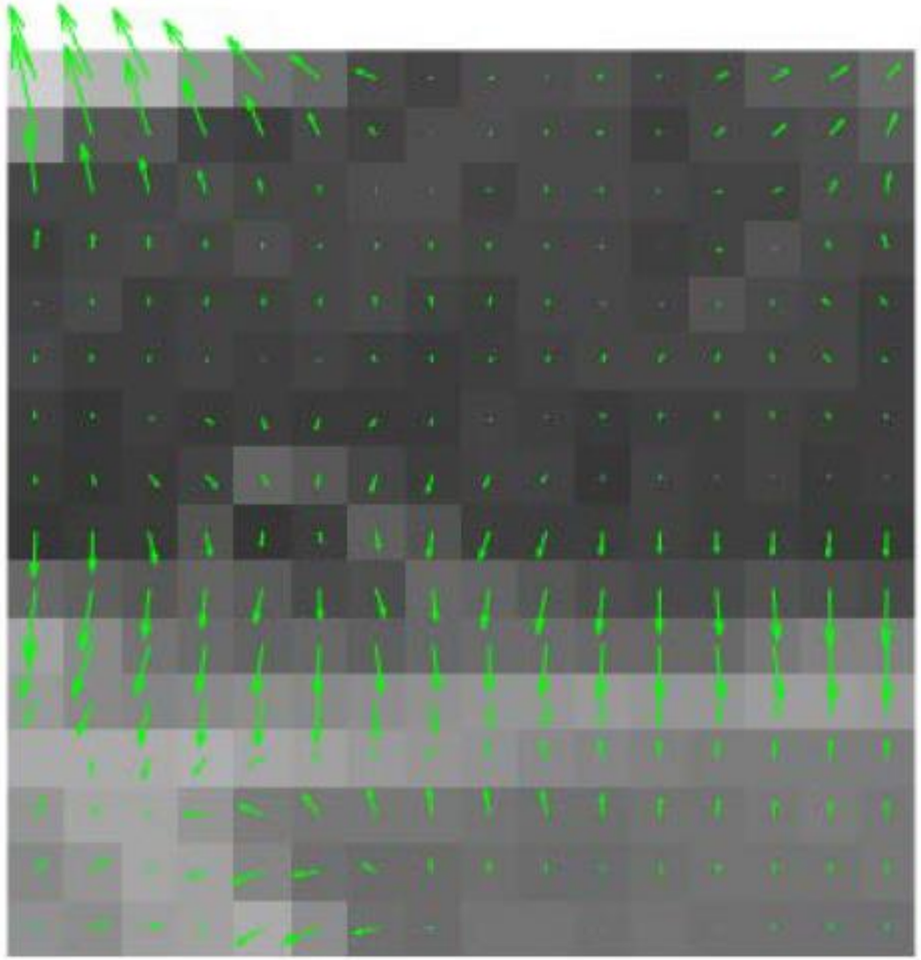
- Extract patch around detected keypoint
- Normalize the patch
  to canonical scale and orientation
- Resize patch to 16x16 pixels





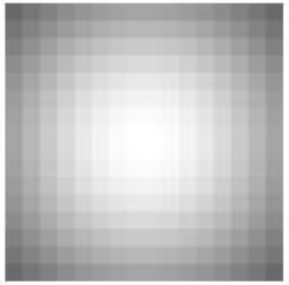
SIFT descriptor

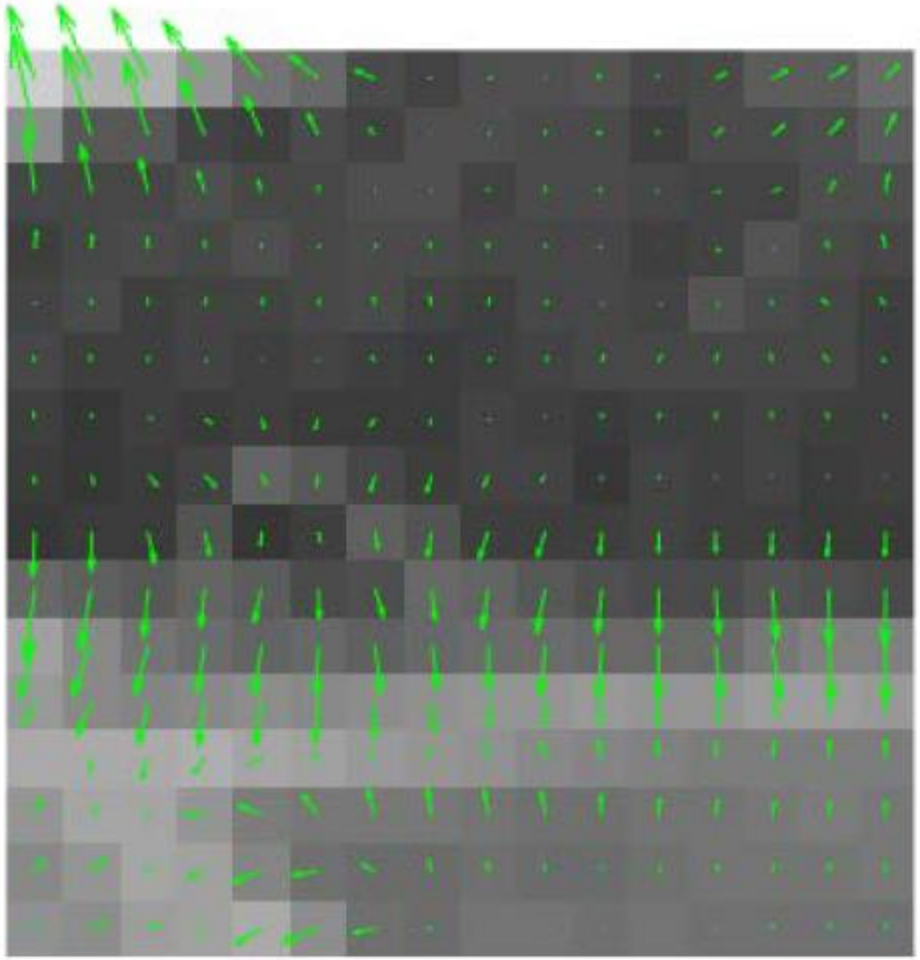
- Compute the gradients



SIFT descriptor

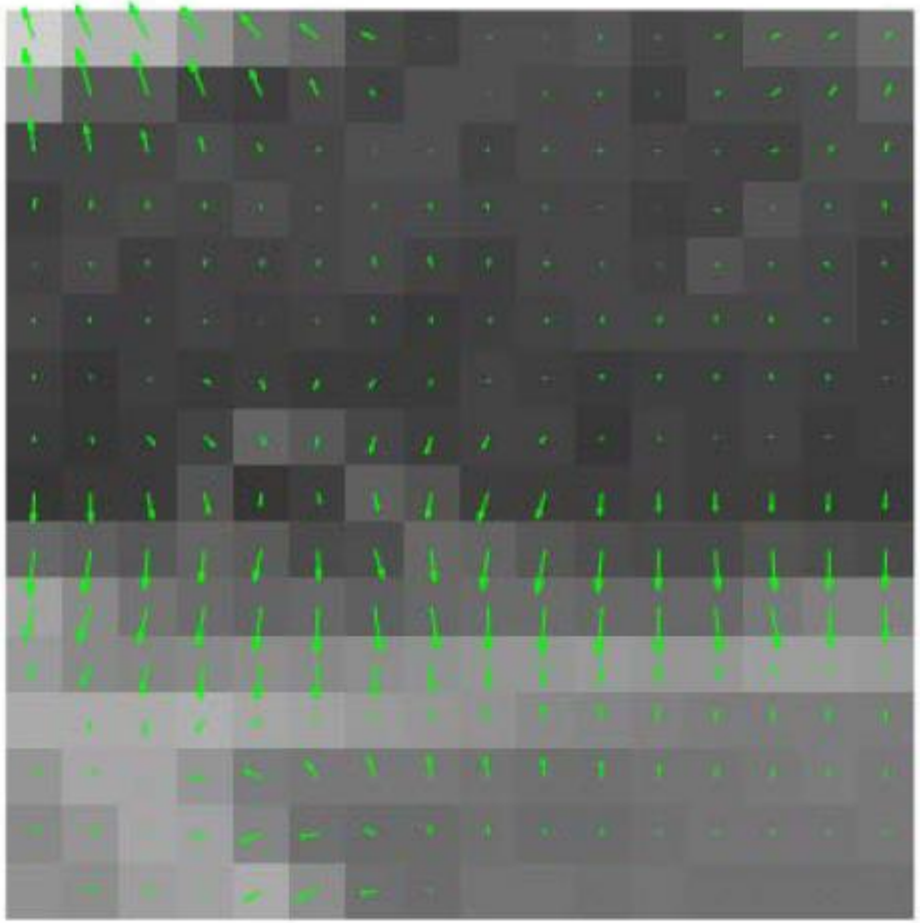
- Compute the gradients
  - Unaffected by additive intensity change
- Apply a Gaussian weighting function





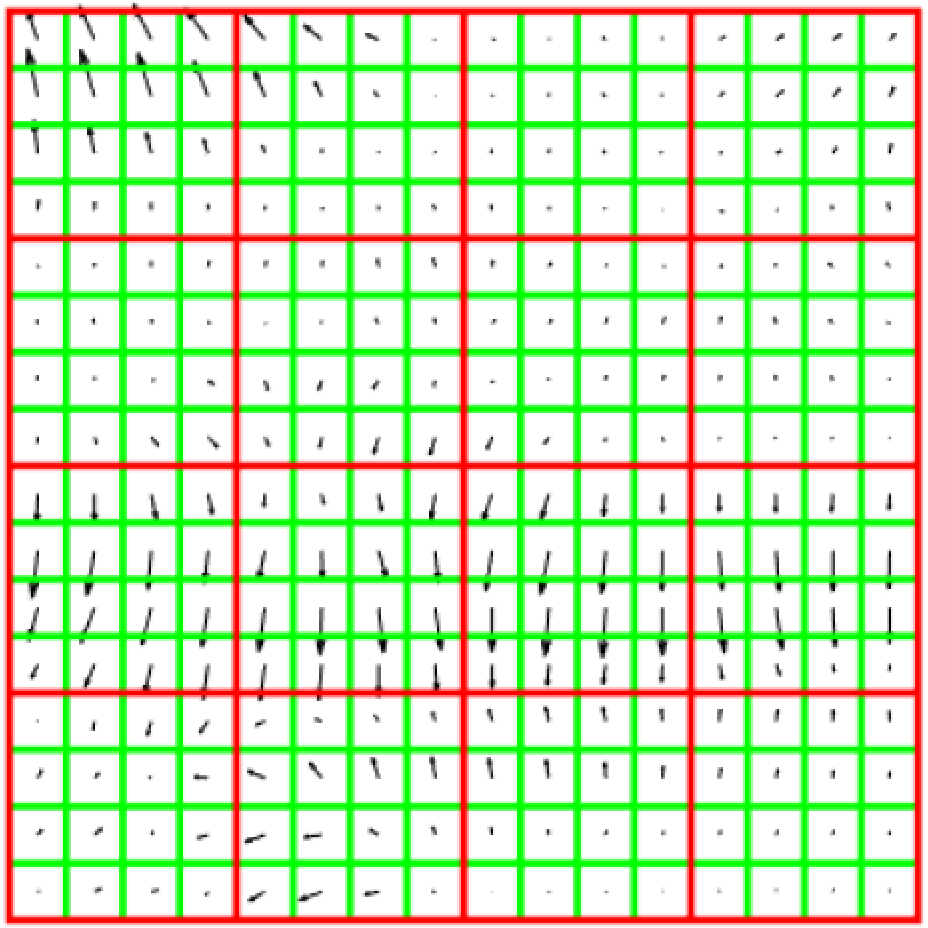
SIFT descriptor

- Compute the gradients
  - Unaffected by additive intensity change
- Apply a Gaussian weighting function
  - Weighs down gradients far from the centre
  - Avoids sudden changes in the descriptor with small changes in the window position



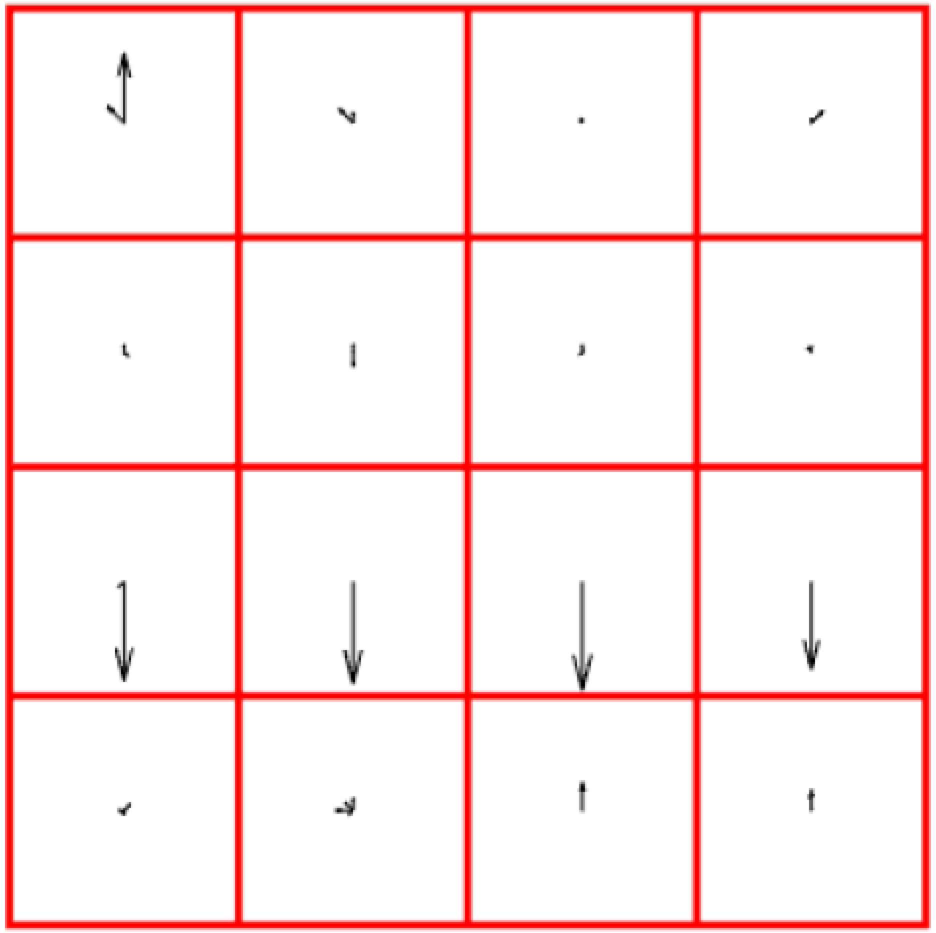
SIFT descriptor

- Compute the gradients
  - Unaffected by additive intensity change
- Apply a Gaussian weighting function
  - Weighs down gradients far from the centre
  - Avoids sudden changes in the descriptor with small changes in the window position
- Divide the patch into 16 4x4 pixels squares



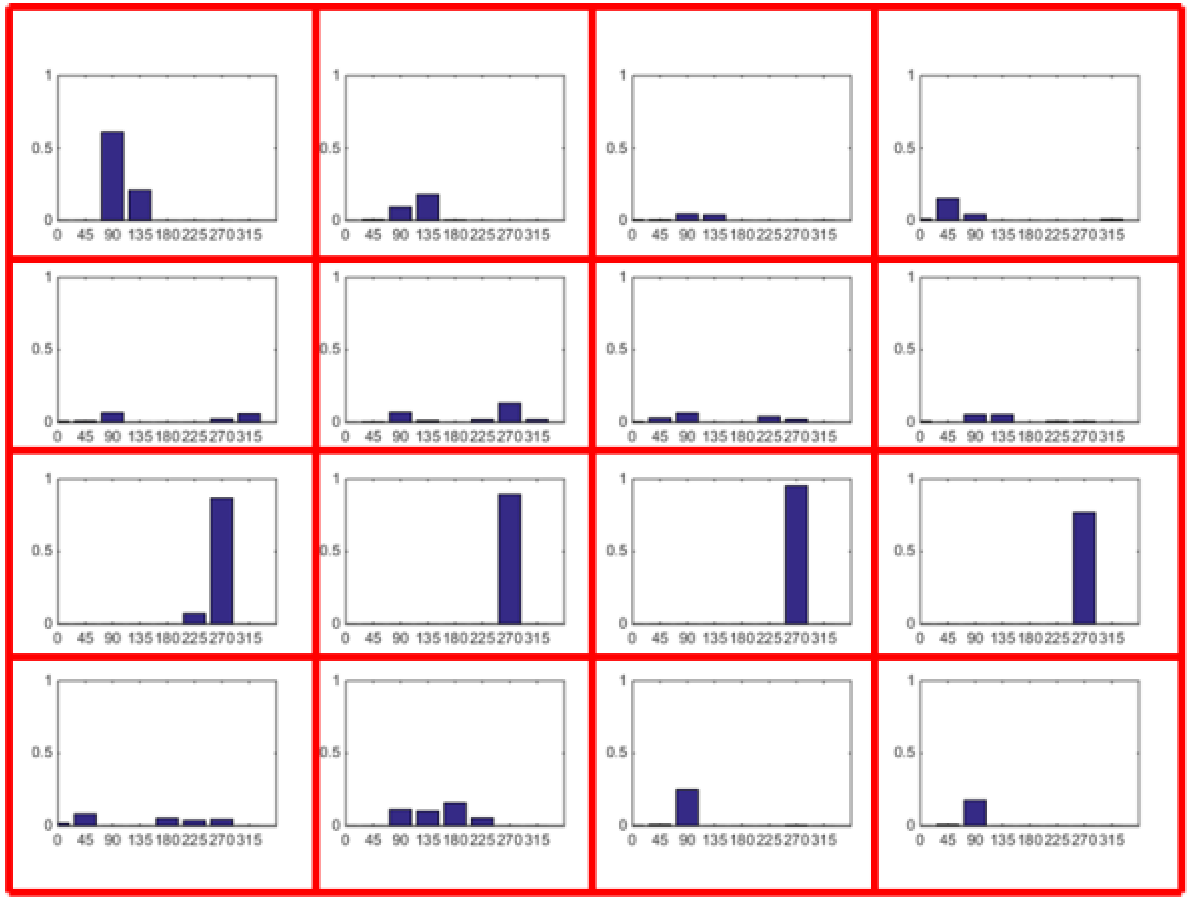
SIFT descriptor

- Compute gradient direction histograms over 8 directions in each square
  - Trilinear interpolation
  - Robust to small shifts, while preserving some spatial information



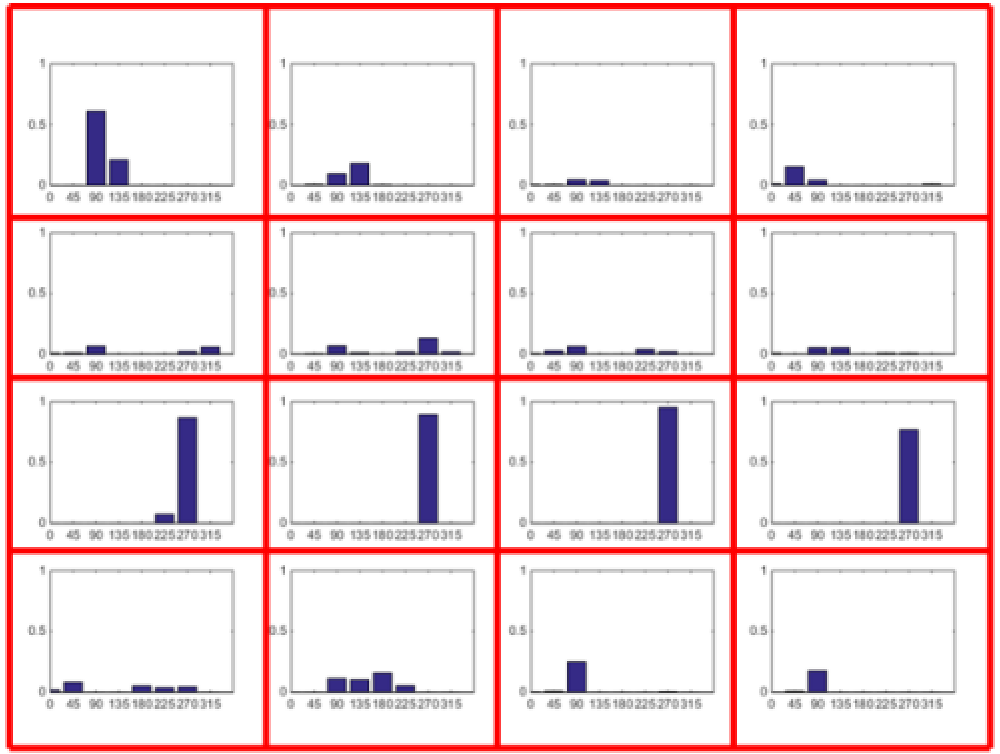
SIFT descriptor

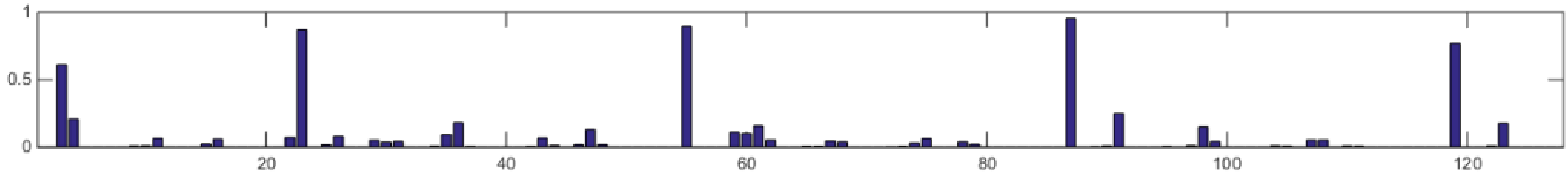
- Compute gradient direction histograms over 8 directions in each square
  - Trilinear interpolation
  - Robust to small shifts, while preserving some spatial information



SIFT descriptor

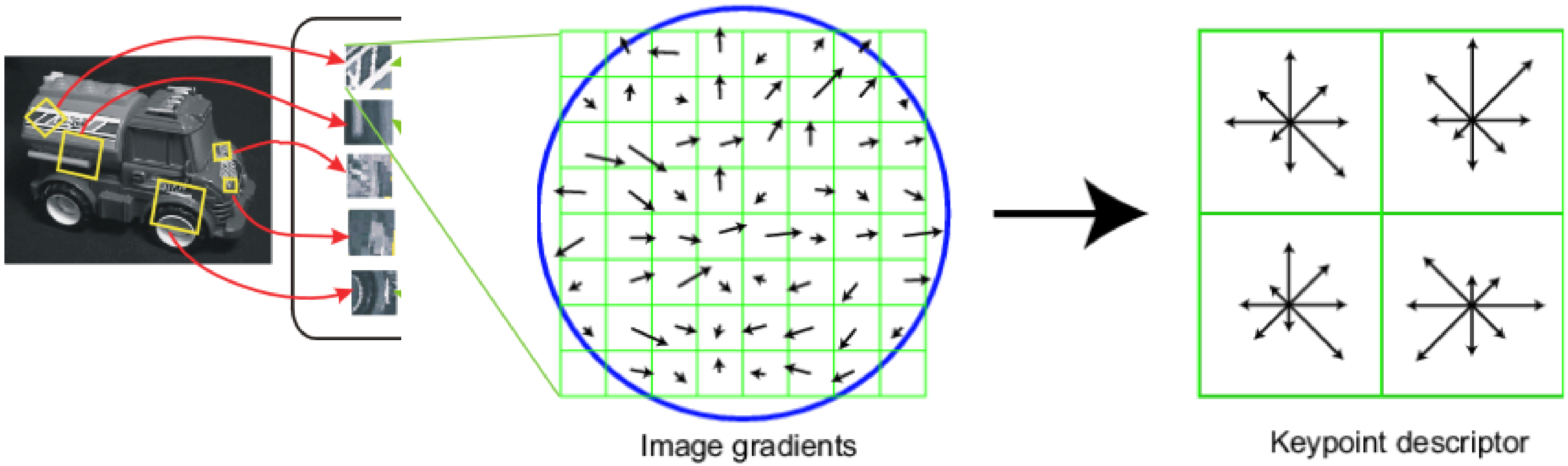
- Concatenate the histograms to obtain a 128 dimensional feature vector





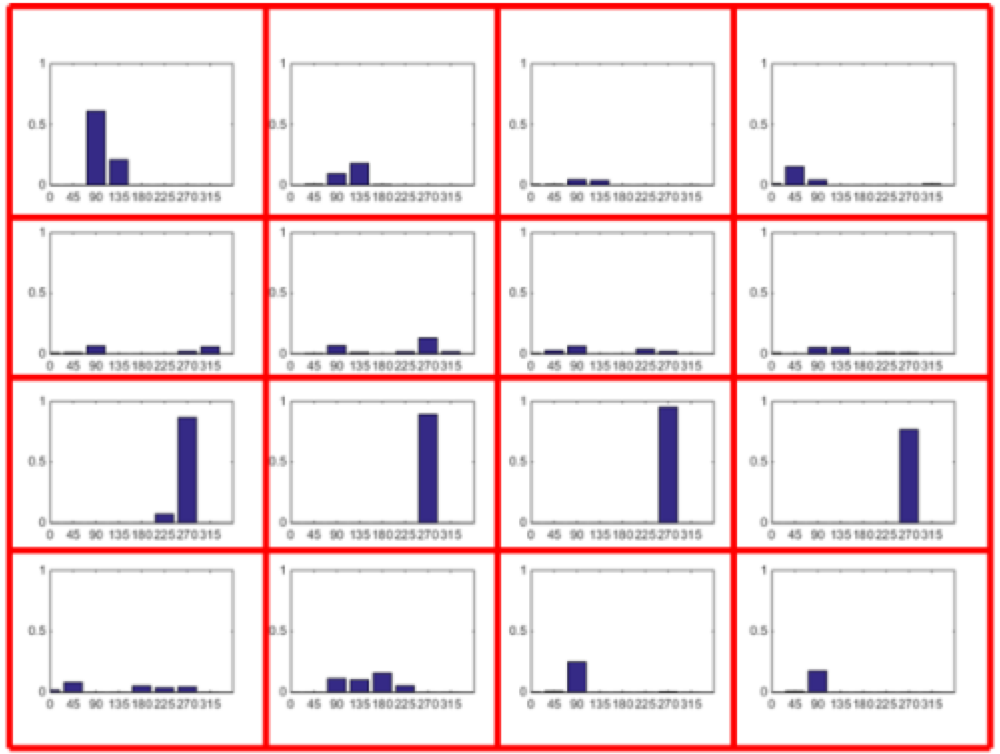
Reduce effect of illumination

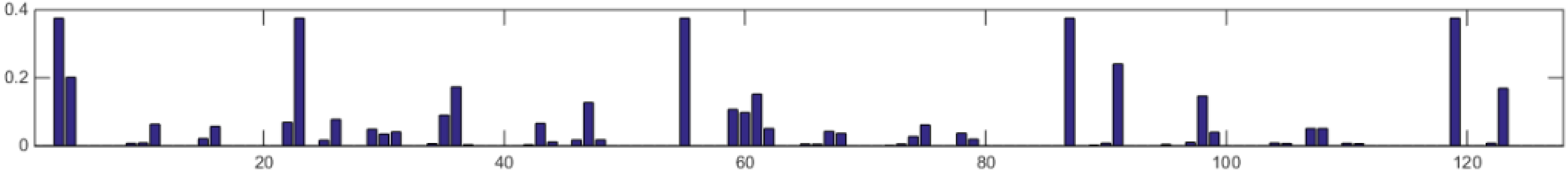
- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - After normalization, clamp gradients > 0.2
  - Renormalize



SIFT descriptor

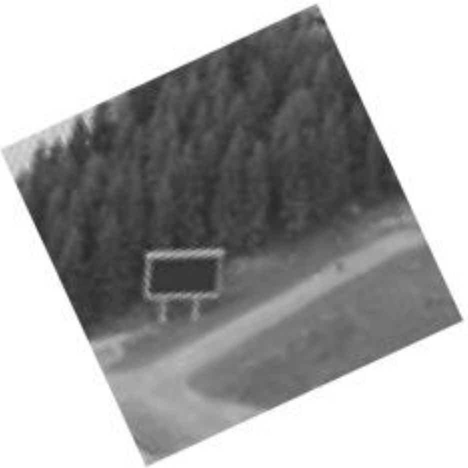
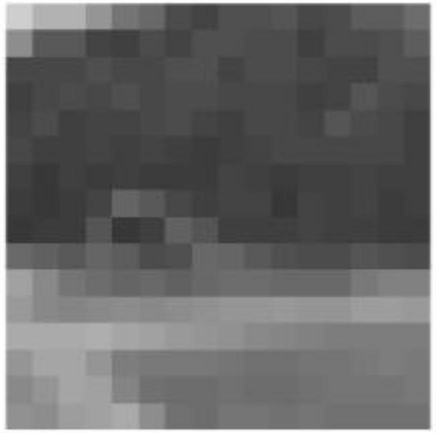
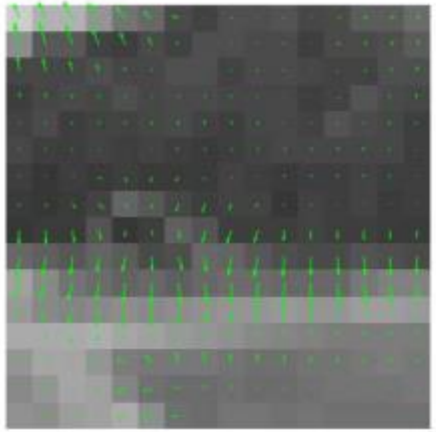
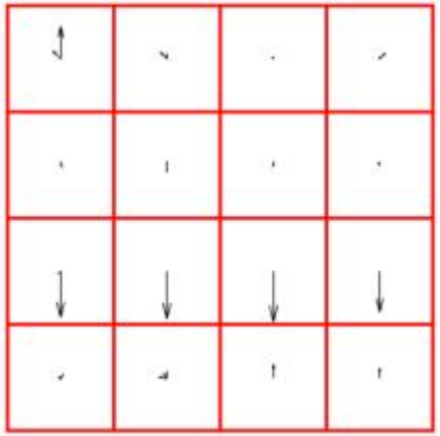
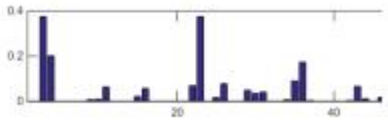
- Concatenate the histograms to obtain a 128 dimensional feature vector
- Normalize to unit length
  - Invariant to multiplicative contrast change
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - Clamp gradients > 0.2
  - Renormalize





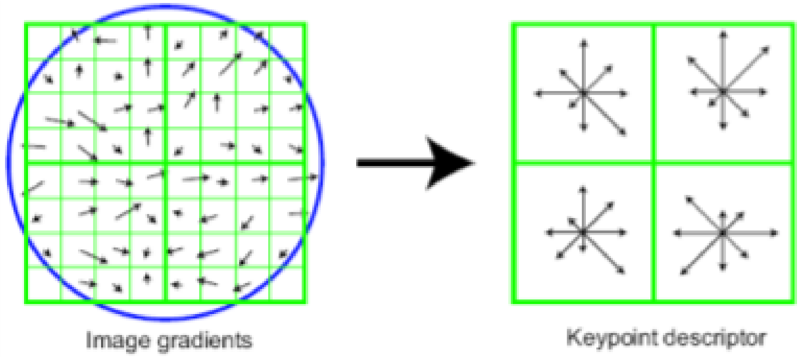
SIFT summary

- Extract a 16x16 patch around detected keypoint
- Compute the gradients and apply a Gaussian weighting function
- Divide the window into a 4x4 grid of cells
- Compute gradient direction histograms over 8 directions in each cell
- Concatenate the histograms to obtain a 128 dimensional feature vector
- Normalize to unit length

Review: Local Descriptors

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



Binary descriptors

- Extremely efficient construction and comparison

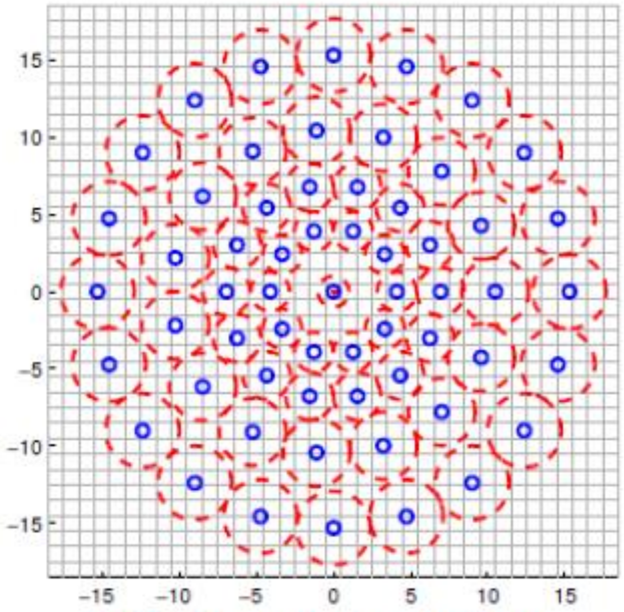
- Based on pairwise intensity comparisons
  - Sampling pattern around keypoint
  - Set of sampling pairs
  - Feature descriptor vector is a binary string:

$$F = \sum_{0 \le a \le N} 2^a T(P_a)$$

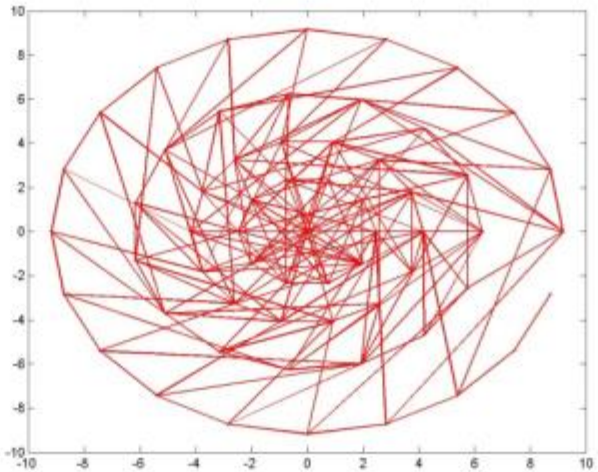
$$T(P_a) = \begin{cases} 1 & \text{if } I(P_a^{r1}) > I(P_a^{r2}) \\ 0 & \text{otherwise} \end{cases}$$

- Matching using Hamming distance:

$$L = \sum_{0 \le a \le N} XOR(F_a^1, F_a^2)$$



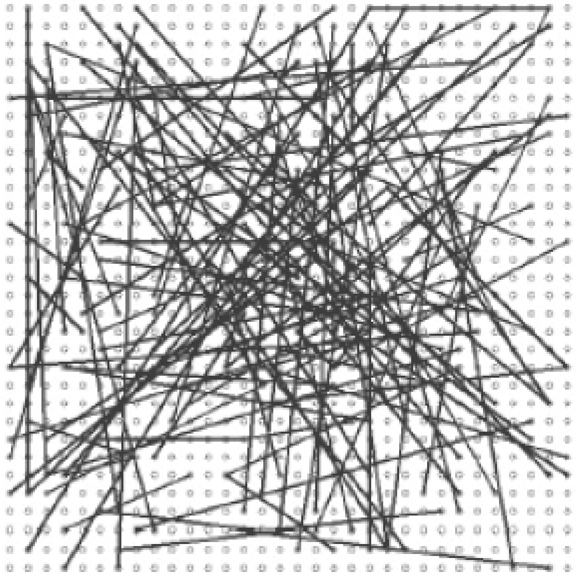
BRISK sampling pattern



BRISK sampling pairs

Binary descriptors

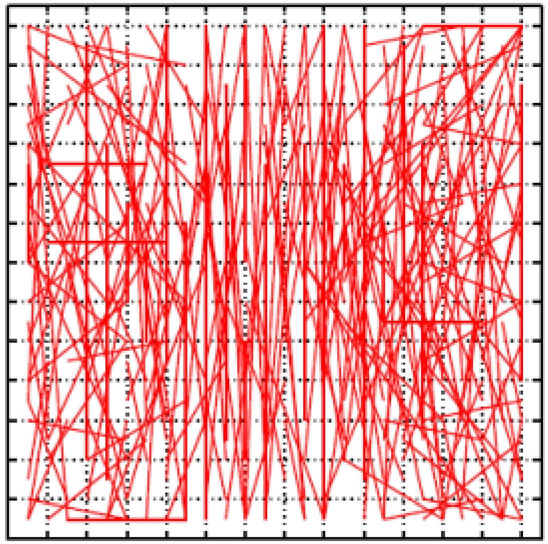
Method	Sampling pattern	Orientation calculation	Sampling pairs
BRIEF	None	None	Random
ORB	None	Moments	Learned pairs
BRISK	Concentric circles, More points on outer rings	Comparing gradients of long pairs	Short pairs
FREAK	Overlapping concentric circles, more points on inner rings	Comparing gradients of preselected 45 pairs	Learned pairs



BRIEF sampling pairs

Binary descriptors

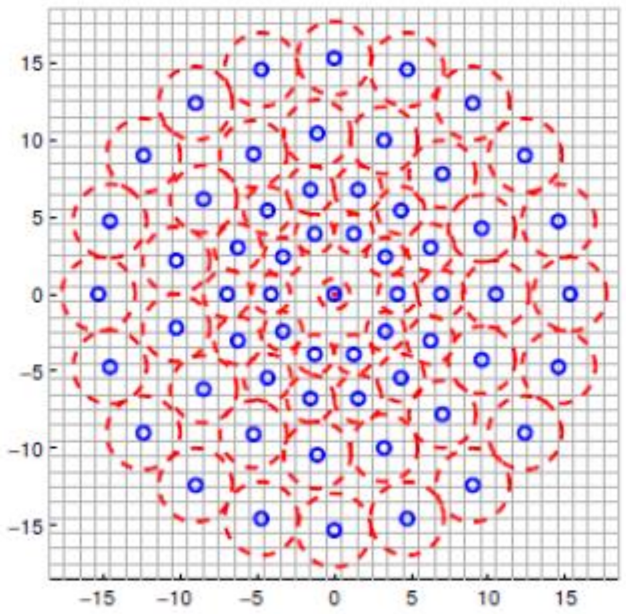
Method	Sampling pattern	Orientation calculation	Sampling pairs
BRIEF	None	None	Random
ORB	None	Moments	Learned pairs
BRISK	Concentric circles, More points on outer rings	Comparing gradients of long pairs	Short pairs
FREAK	Overlapping concentric circles, more points on inner rings	Comparing gradients of preselected 45 pairs	Learned pairs



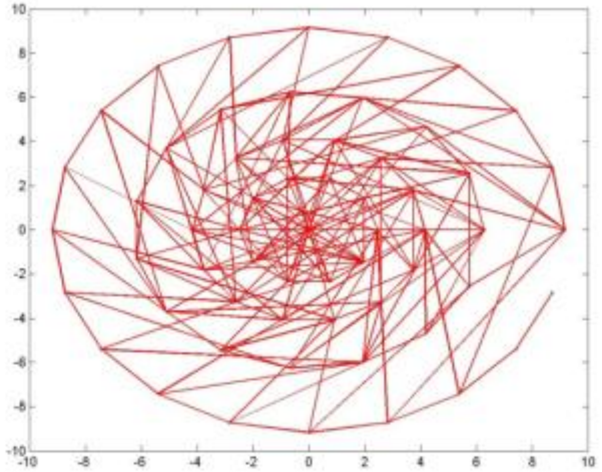
ORB sampling pairs

Binary descriptors

Method	Sampling pattern	Orientation calculation	Sampling pairs
BRIEF	None	None	Random
ORB	None	Moments	Learned pairs
BRISK	Concentric circles, More points on outer rings	Comparing gradients of long pairs	Short pairs
FREAK	Overlapping concentric circles, more points on inner rings	Comparing gradients of preselected 45 pairs	Learned pairs



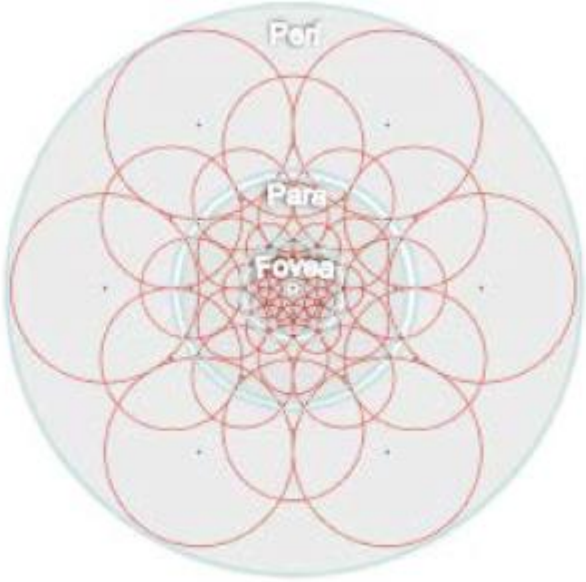
BRISK sampling pattern



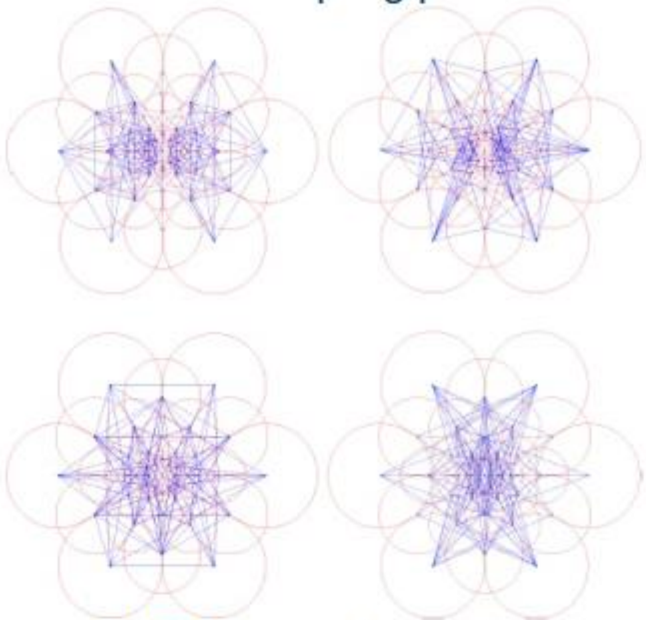
BRISK sampling pairs

Binary descriptors

Method	Sampling pattern	Orientation calculation	Sampling pairs
BRIEF	None	None	Random
ORB	None	Moments	Learned pairs
BRISK	Concentric circles, More points on outer rings	Comparing gradients of long pairs	Short pairs
FREAK	Overlapping concentric circles, more points on inner rings	Comparing gradients of preselected 45 pairs	Learned pairs

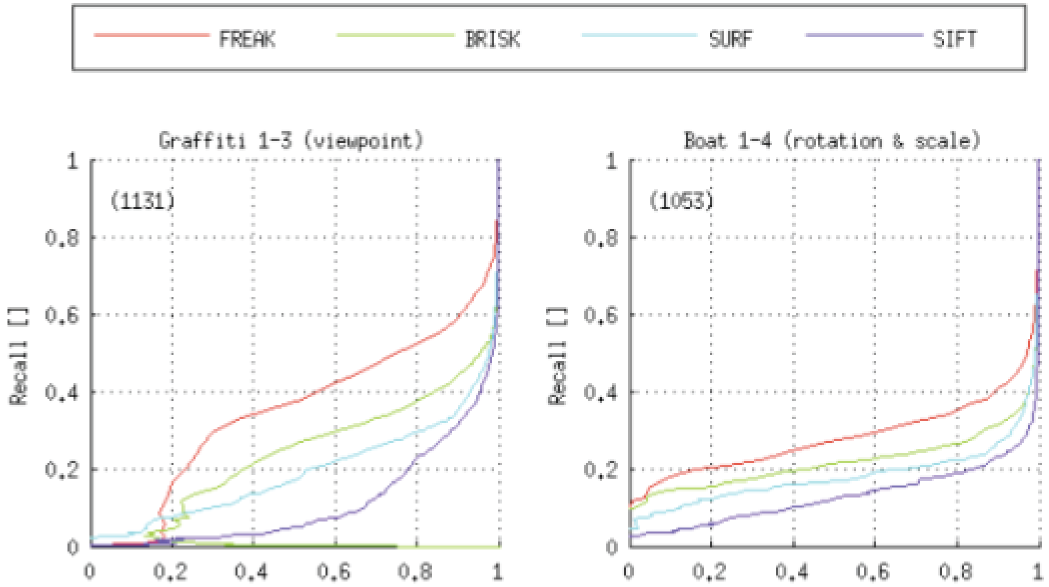


FREAK sampling pattern


FREAK sampling pairs

Binary descriptors

- Often achieves very good performance compared to SIFT/SURF
- Much faster than SIFT/SURF



Time per keypoint	SIFT	SURF	BRISK	FREAK
Description in [ms]	2.5	1.4	0.031	0.018
Matching time in [ns]	1014	566	36	25

Table 1: Computation time on 800x600 images where approximately 1500 keypoints are detected per image. The computation times correspond to the description and matching of all keypoints.

A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In *IEEE Conference on Computer Vision and Pattern Recognition,*

References

Basic reading:
- Szeliski textbook, Sections 4.1.

- Gil's CV blog | Gil's Computer vision blog (gilscvblog.com)

Questions?