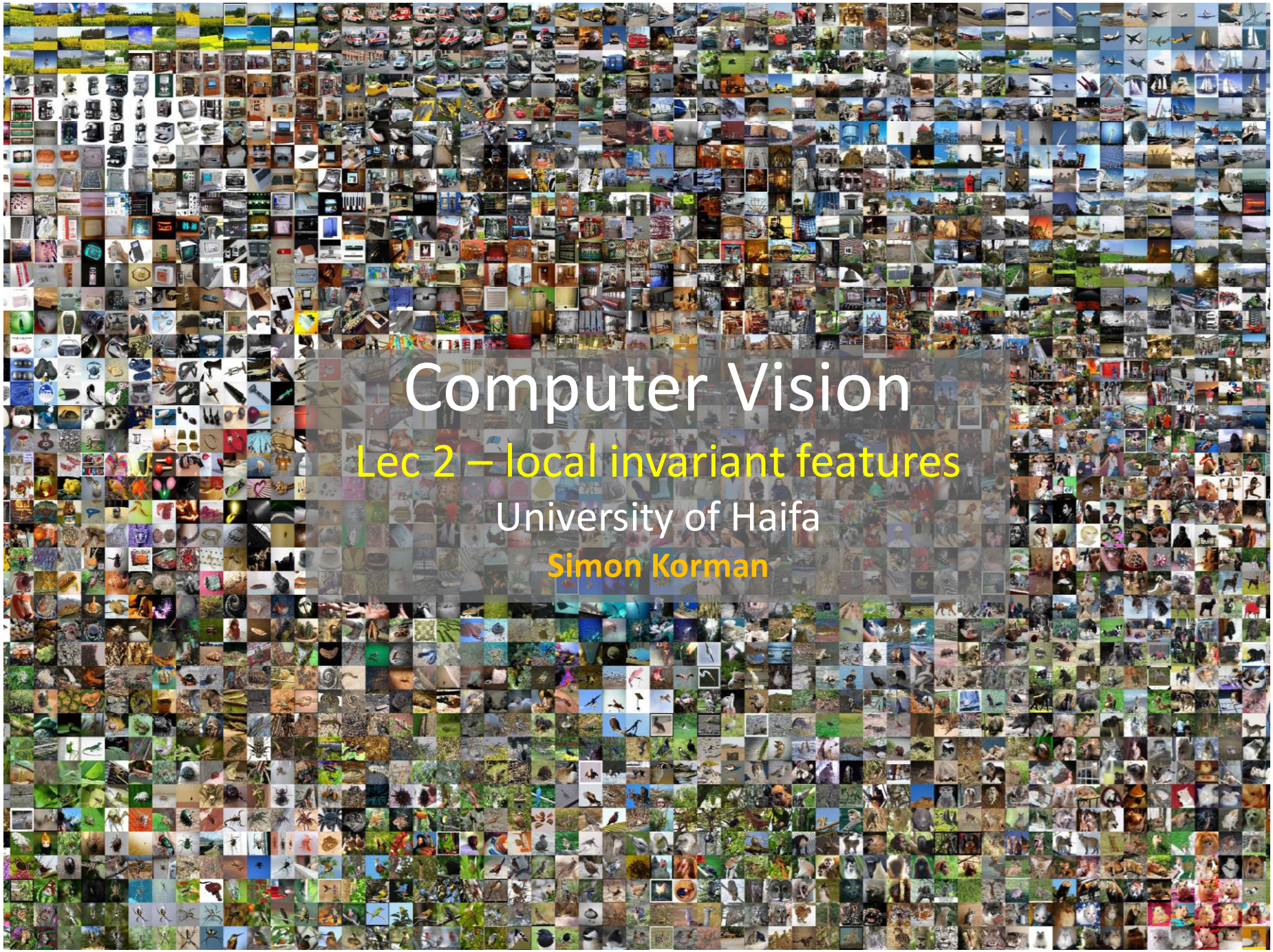


Computer Vision

Lec 2 – local invariant features

University of Haifa

Simon Korman

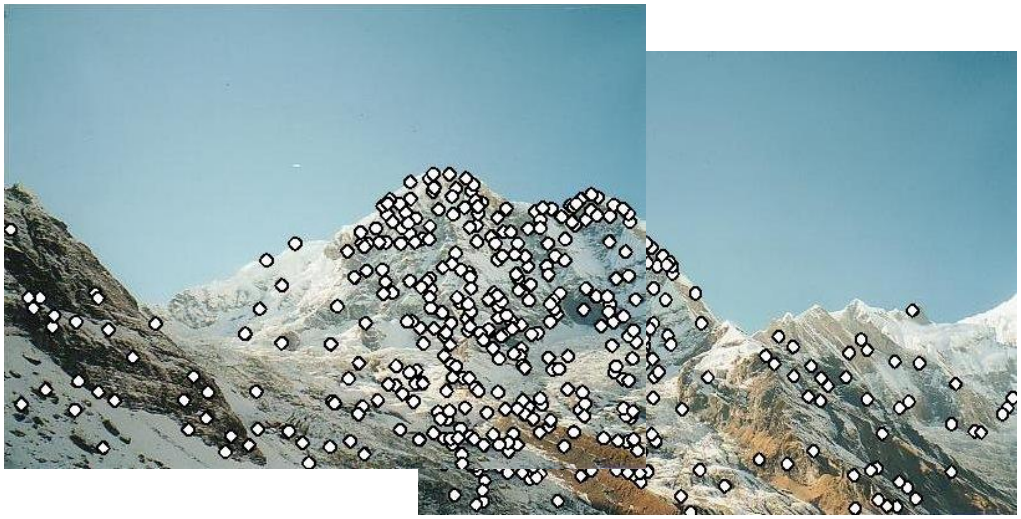
detection, description and matching of local invariant features

slide credit

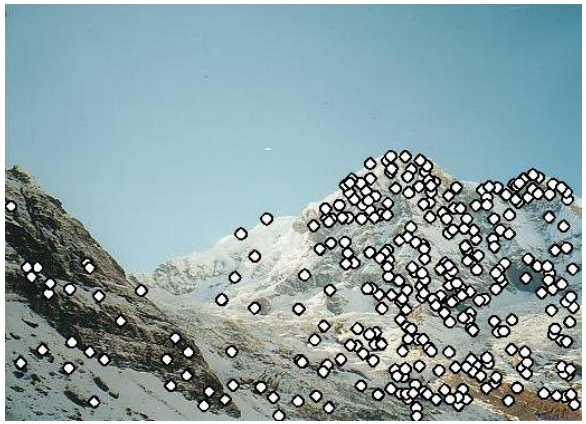
- Svetlana Lazebnik, Sanja Fidler, Kristen Grauman, Ioannis Gkioulekas, Kris Kitani, James Hays, Fredo Durand, Rick Szeliski, Andrew Zisserman, Kyros Kutulakos

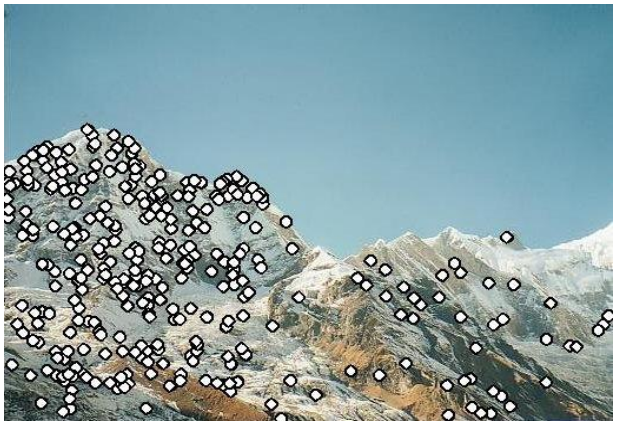
Local invariant features

Important tool for multiple views: Local features



Multi-view matching relies on **local feature** correspondences.

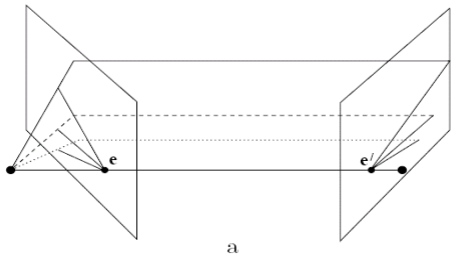




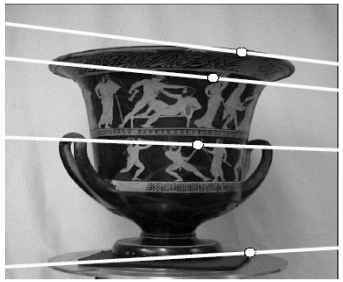
How to detect *which local features* to match?

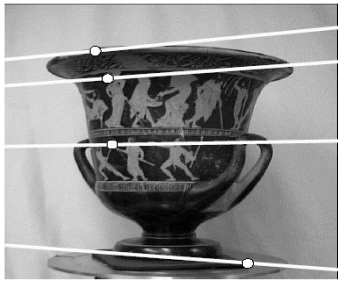
Multiple views

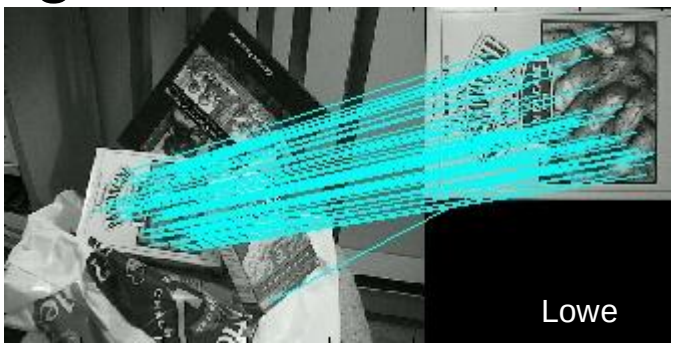
Matching, invariant features, stereo vision, instance recognition



a



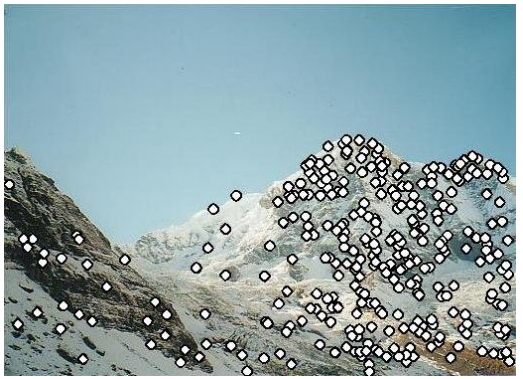




Slide credit: Kristen Grauman

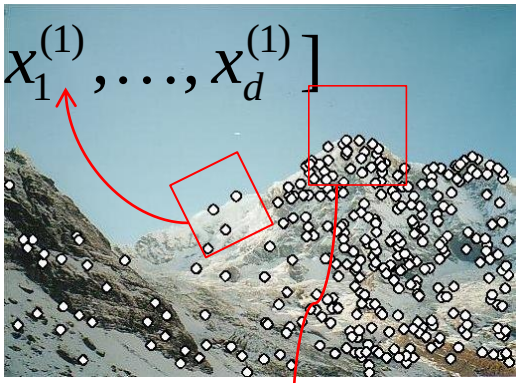
Local features: main components

1) Detection: Identify the interest points



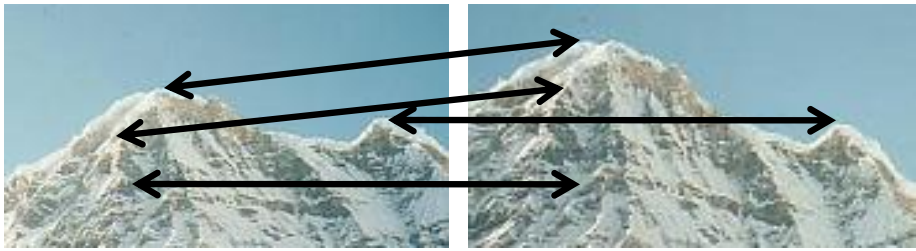
2) Description: Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) Matching: Determine correspondence between descriptors in two views



Slide credit: Kristen Grauman

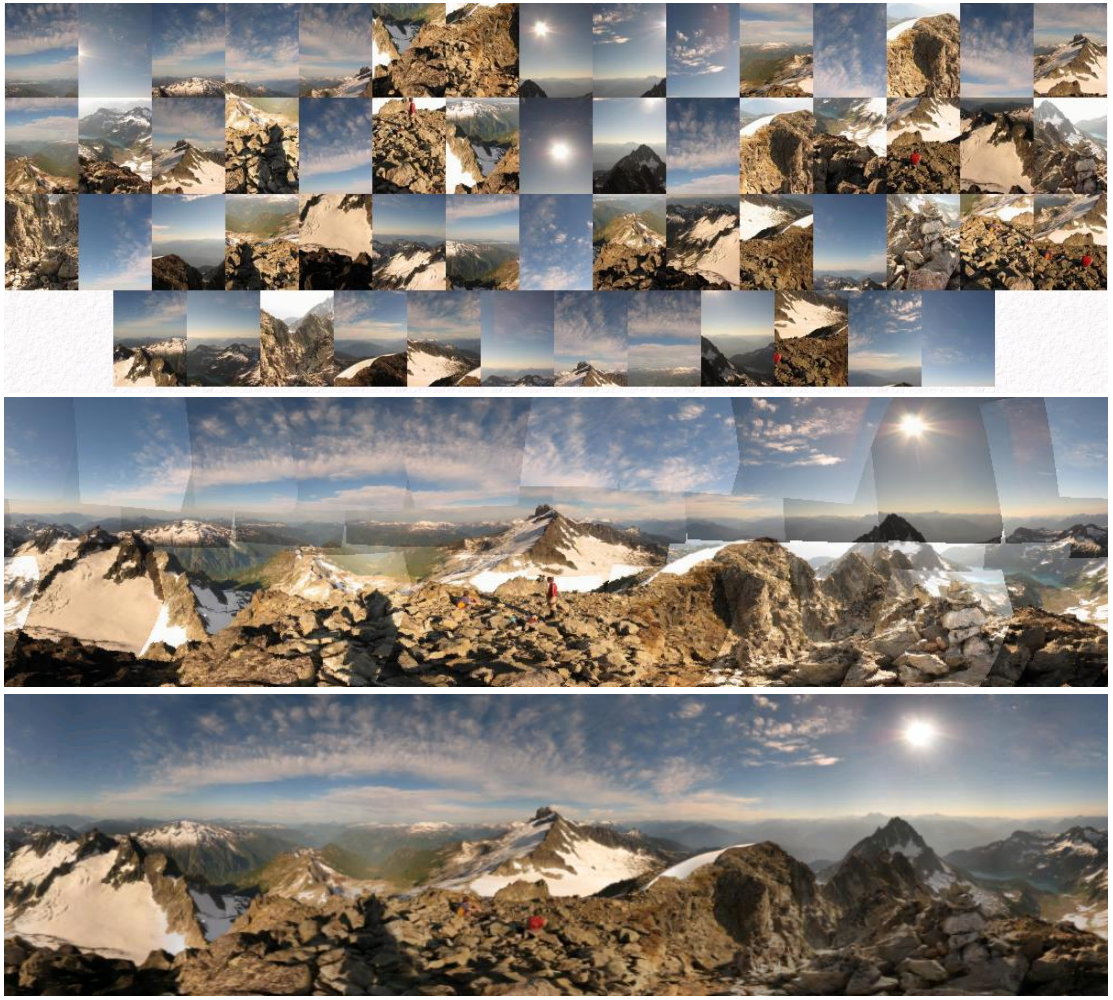
Value of local (invariant) features

- Complexity reduction via selection of distinctive points
- Describe images, objects, parts without requiring segmentation
  - Local character means robustness to clutter, occlusion
- Robustness: similar descriptors in spite of noise, blur, etc.

Applications of local invariant features

- Wide baseline stereo
- Motion tracking
- Panoramas
- Mobile robot navigation
- 3D reconstruction
- Recognition
- …

Automatic mosaicing



Matthew Brown

http://matthewalunbrown.com/autostitch/autostitch.html

Wide baseline stereo

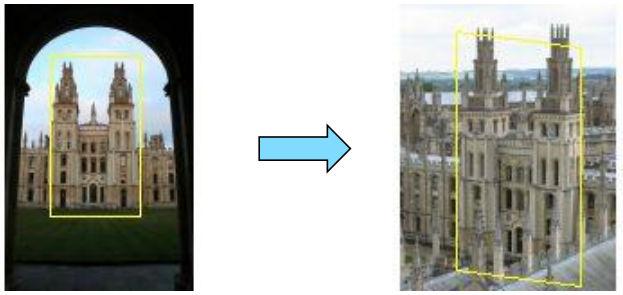
[Image from T. Tuytelaars ECCV 2006 tutorial]

Photo tourism [Snavely et al. 2006/8]

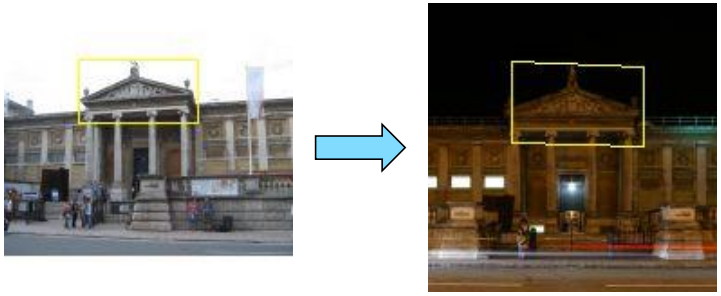
Slide credit: Noah Snavely

Recognition of specific objects, scenes

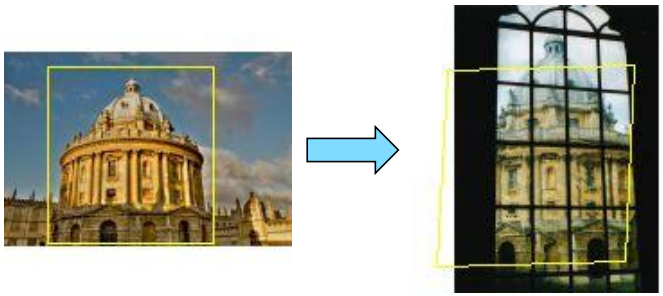
Scale



Viewpoint



Lighting



Occlusion

Slide credit: J. Sivic

Google Goggles

Local features: desired properties

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Goal: interest operator repeatability

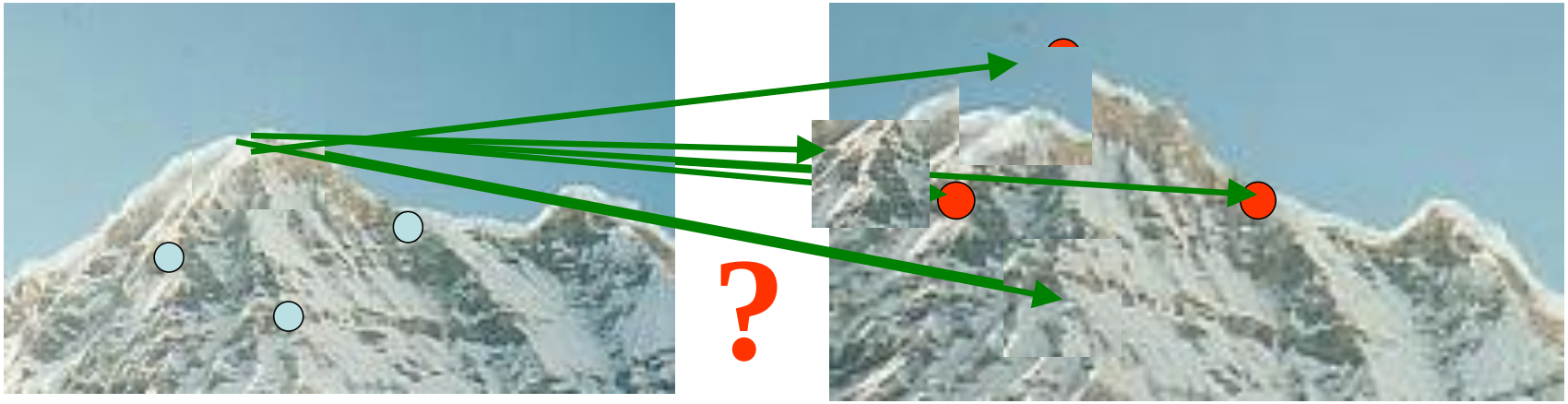
- We want to detect (at least some of) the same points in both images.

No chance to find (all and only) true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

Goal: descriptor distinctiveness

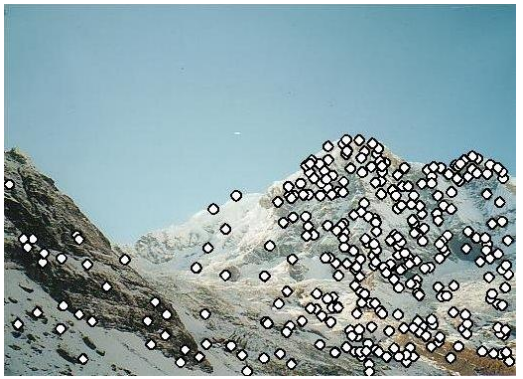
- We want to be able to reliably determine which point goes with which.



- Must provide some invariance to geometric and photometric differences between the two views.

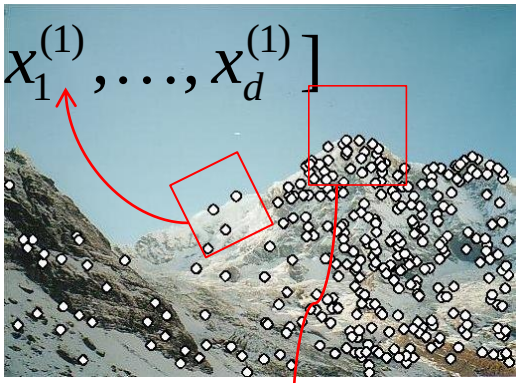
Local features: main components

1) Detection: Identify the interest points



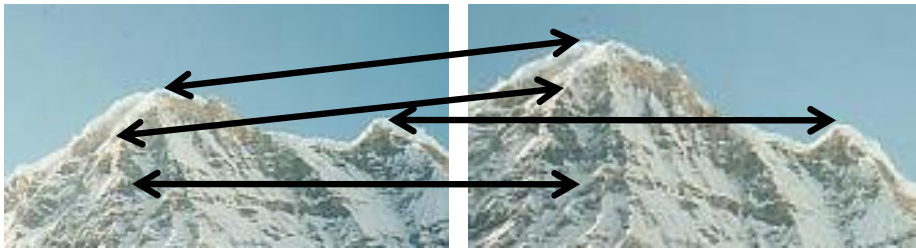
2) Description: Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) Matching: Determine correspondence between descriptors in two views

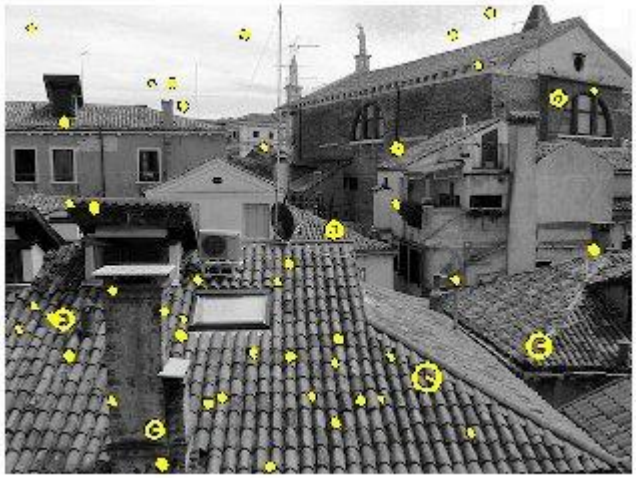


Slide credit: Kristen Grauman

Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

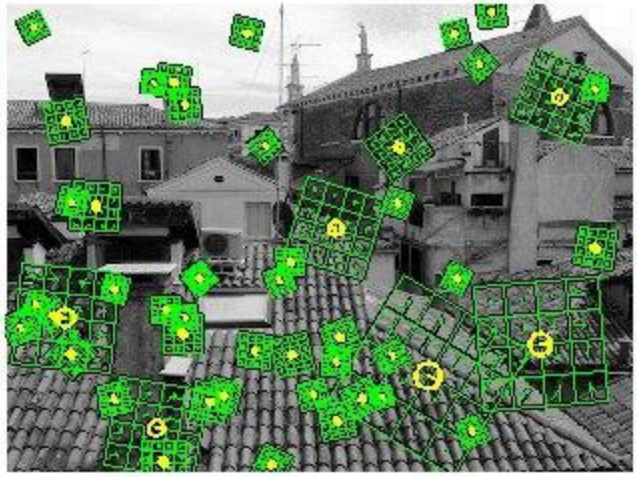
input

detection



Interest points and their scales and orientations (random subset of 50)

description



SIFT descriptors

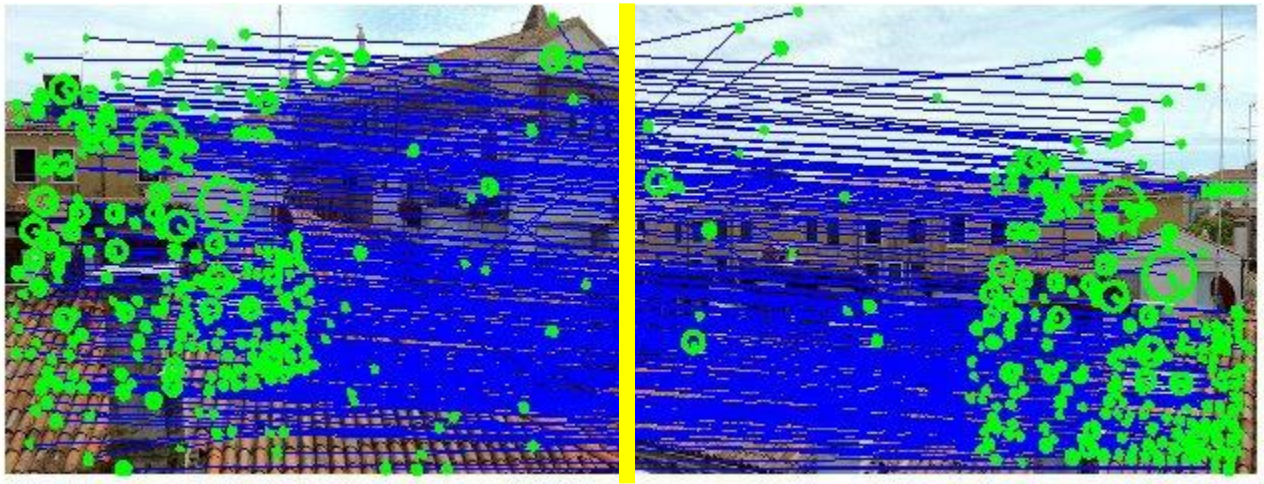
http://www.vlfeat.org/overview/sift.html

SIFT (preliminary) matches

img1

img2

img1

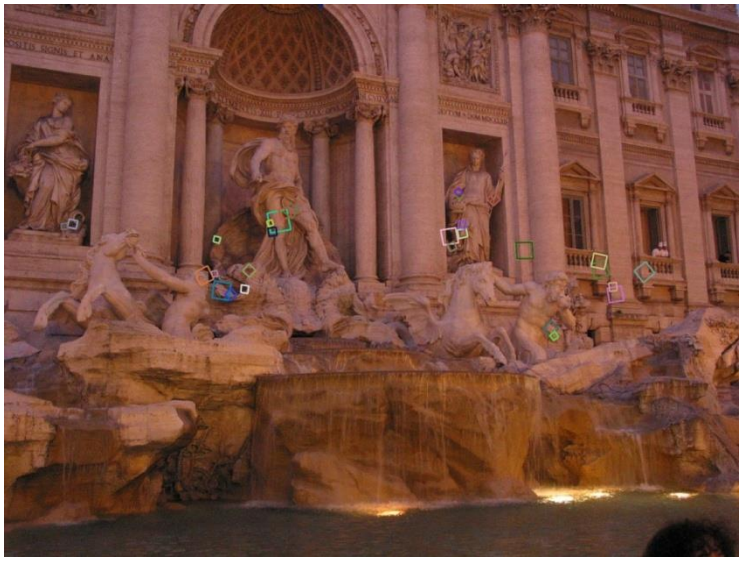


img2

http://www.vlfeat.org/overview/sift.html

SIFT descriptor [Lowe 2004]

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night (below)
  - Fast and efficient—can run in real time
  - Lots of code available, e.g. `http://www.vlfeat.org/overview/sift.html`



Steve Seitz

SIFT properties

- Invariant to
  - Scale
  - Rotation

- Partially invariant to
  - Illumination changes
  - Camera viewpoint
  - Occlusion, clutter

- ASIFT: fully affine invariant image comparison (SIAM 2009)

Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

- introduction to the image matching problem
- image matching using SIFT features
- multi-scale interest-point detection
- the SIFT feature detector
- the SIFT descriptor

Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

- introduction to the image matching problem
- image matching using SIFT features
- multi-scale interest-point detection
- the SIFT feature detector
- the SIFT descriptor

Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

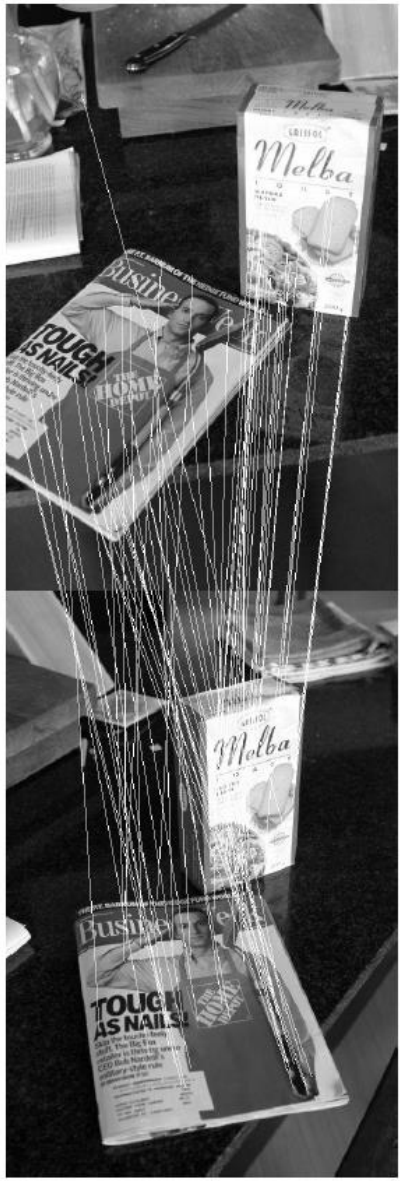
- **introduction to the image matching problem**
- image matching using SIFT features
- multi-scale interest-point detection
- the SIFT feature detector
- the SIFT descriptor

The Image Matching Problem

Image I

Image I'

Goal: Identify "features" or patches in image I that appear in another image, I'



The Image Matching Problem

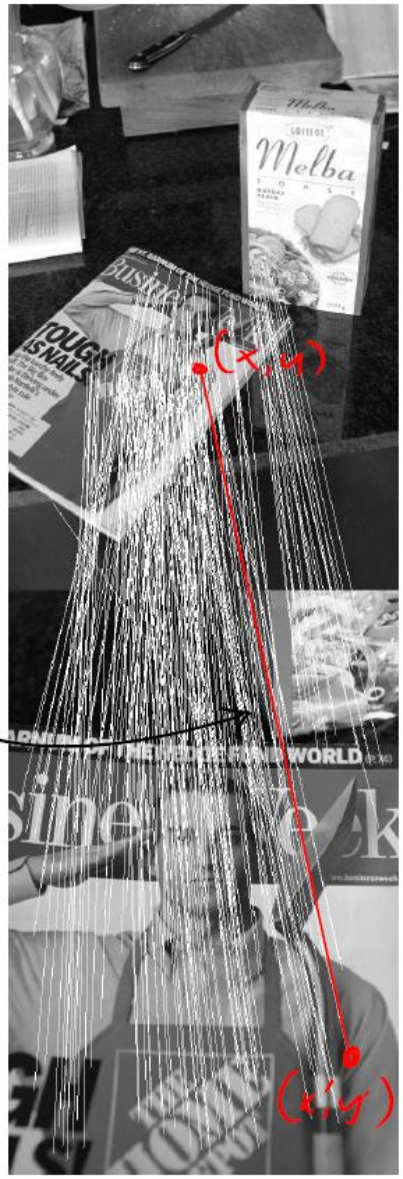
I

Image

I′

Image

Indicates a correspondence between location (x,y) in image I and location (x',y') in image I′



The Image Matching Problem

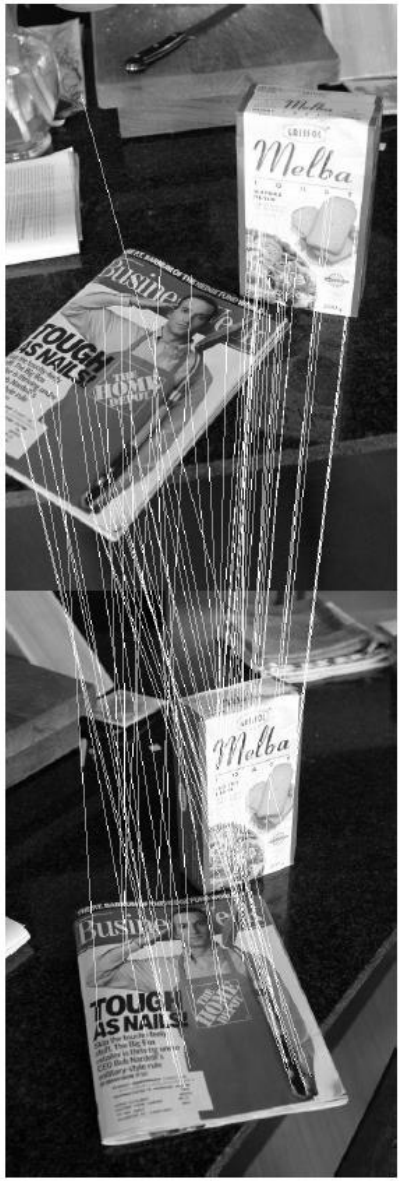
I

Image

I'

Image

Q: Is it possible to solve this problem by direct template matching between the two images?

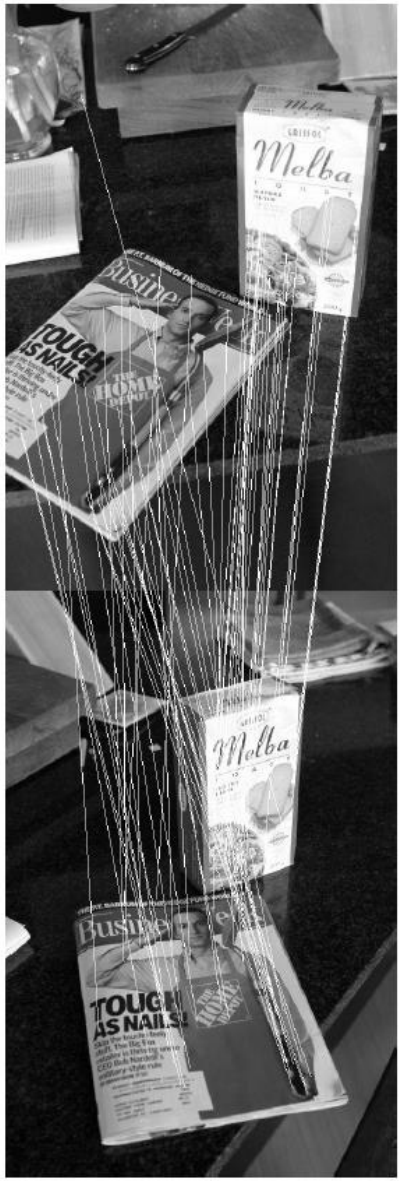


The Image Matching Problem

Image I

Q: Is it possible to solve this problem by direct template matching between the two images?

Image I'



The Image Matching Problem

I (Image 1)

I' (Image 2)

Q: Is it possible to solve this problem by direct template matching between the two images?

Ans: Yes, but it would be impossibly inefficient (i.e. must search over all possible pairs of patches)

Feature-Based Image Matching

I

Image

I'

Image

Feature detection & matching

Feature-Based Image Matching

I

Image

I′

Image

Detect features in I

Detect features in I′

Match features across the two images

Errors in Feature-Based Image Matching

Image I

Image I'

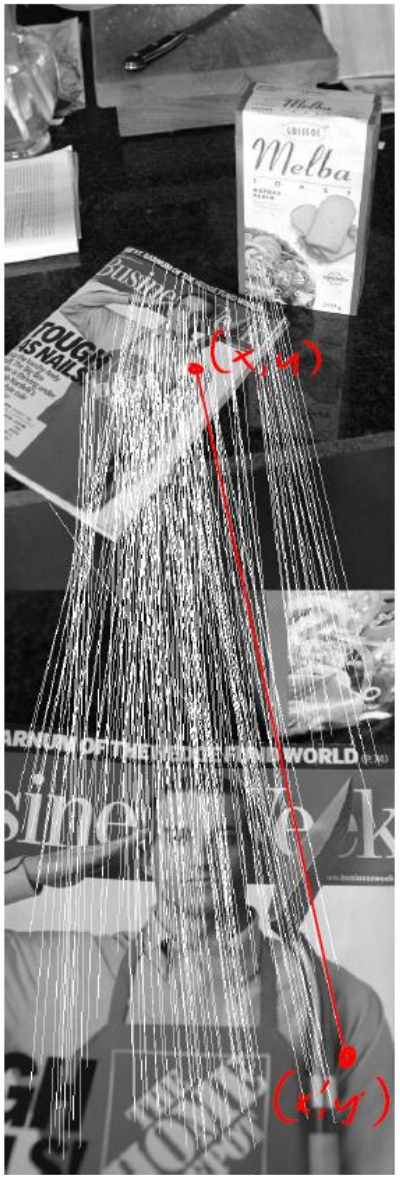
- In general some/many of these correspondences may be incorrect
- Two types of error:

① False positive matches

algorithm returns a correspondence between 2 locations where none exists

② False negative matches

algorithm fails to detect a correspondence between two instances of the same



Errors in Feature-Based Image Matching

Image I

Image I'

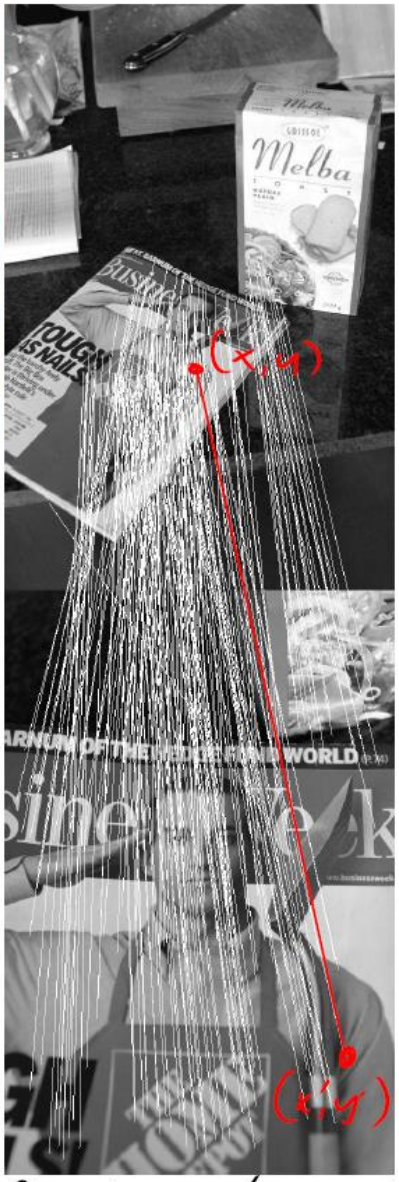
Goal: Minimize false positives AND false negatives across a wide range of imaging conditions

① False positive matches

algorithm returns a correspondence between 2 locations where none exists

② False negative matches

algorithm fails to detect a correspondence between two instances of the same feature/patch



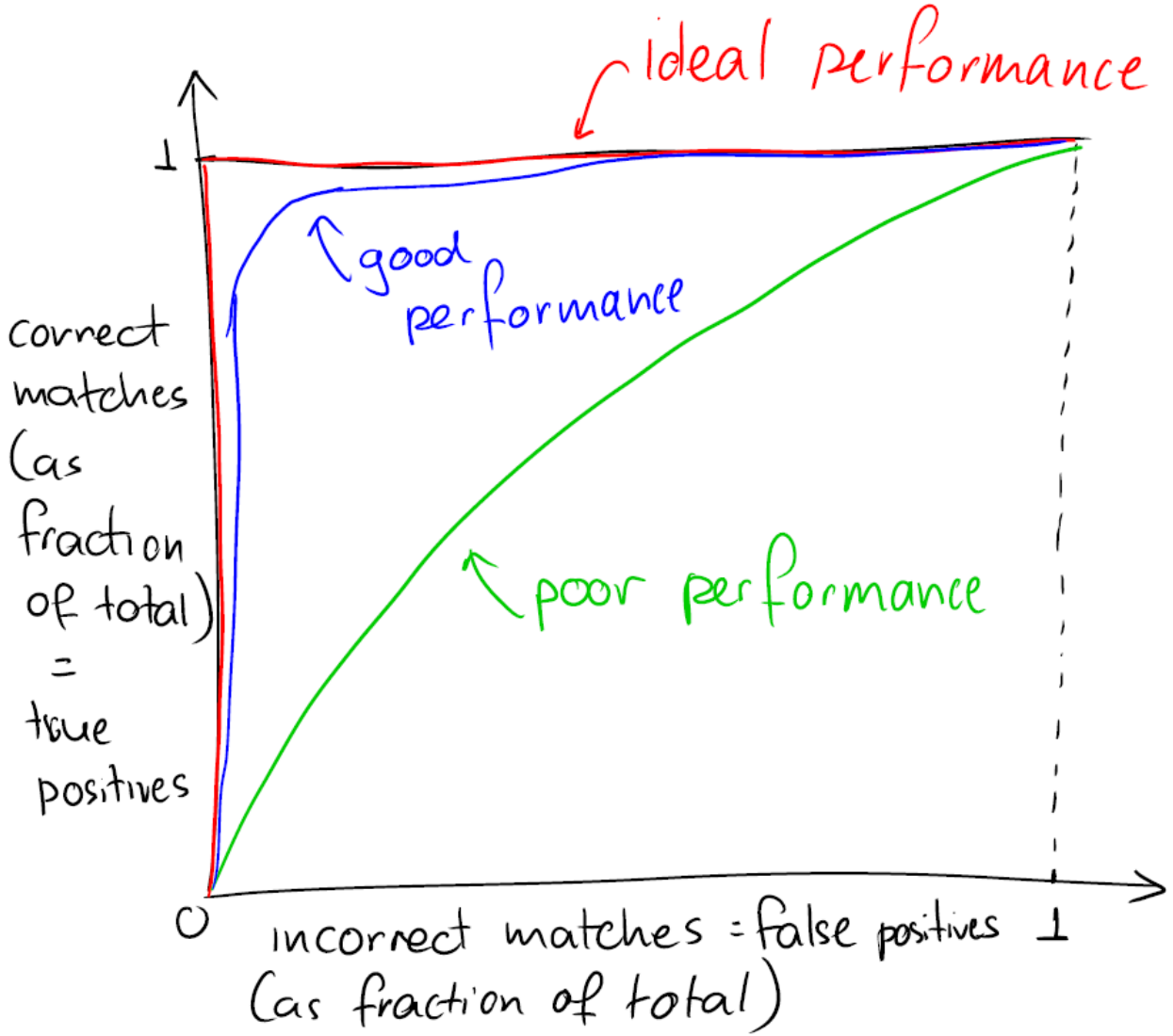
Evaluating a Feature Detector's Performance

I

Image

I'

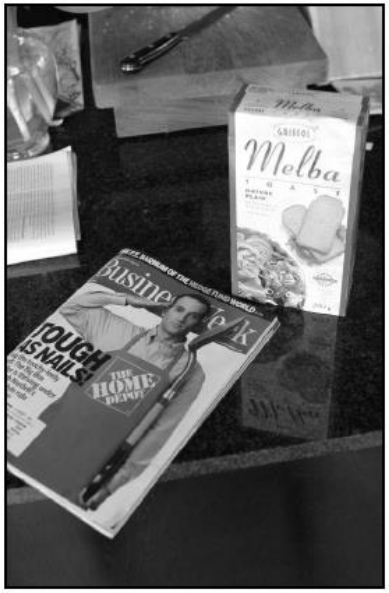
Image



Feature Matching & Transformation Invariance

Source image I

"Transformed" source images

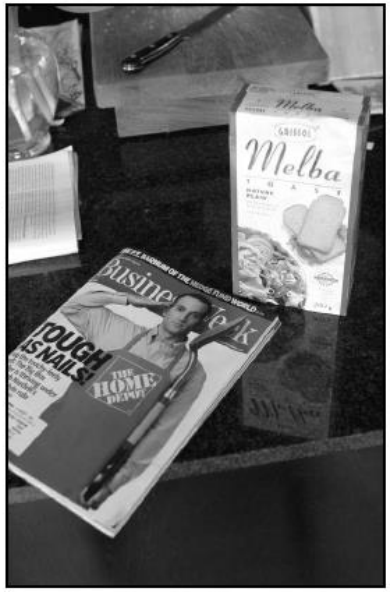


To be most useful, the feature detector & matching algorithm must be insensitive to a wide range of image transformations

Transformation-Invariant Feature Detectors

Source image I

"Transformed" source images



A feature detector is called invariant to a certain image transformation if it can reliably detect features in a transformed version of the source image

Transformation-Invariant Feature Detectors

Source image I

"Transformed" source images

Brightness transformation

A feature detector is called invariant to a certain image transformation if it can reliably detect features in a transformed version of the source image

Transformation-Invariant Feature Detectors

Source image I

"Transformed" source images

?

Distortion due to change in viewpoint

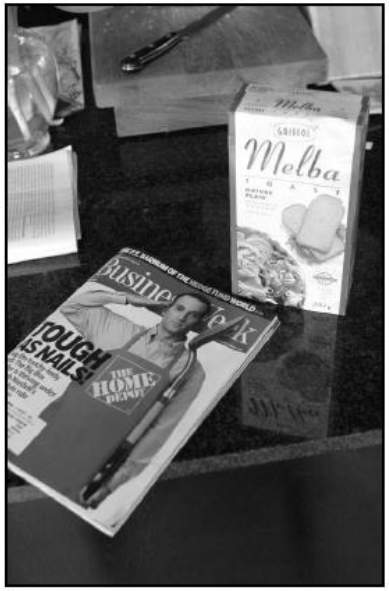
A feature detector is called invariant to a certain image transformation if it can reliably detect features in a transformed version of the source image

Transformation-Invariant Feature Detectors

"Transformed" source images

Source image I

A feature detector is called invariant to a certain image transformation if it can reliably detect features in a transformed version of the source image



Distortion due to change in viewpoint & magnification (i.e. scale)

Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

- introduction to the image matching problem
- **image matching using SIFT features**
- multi-scale interest-point detection
- the SIFT feature detector
- the SIFT descriptor

Image Matching Using SIFT Features

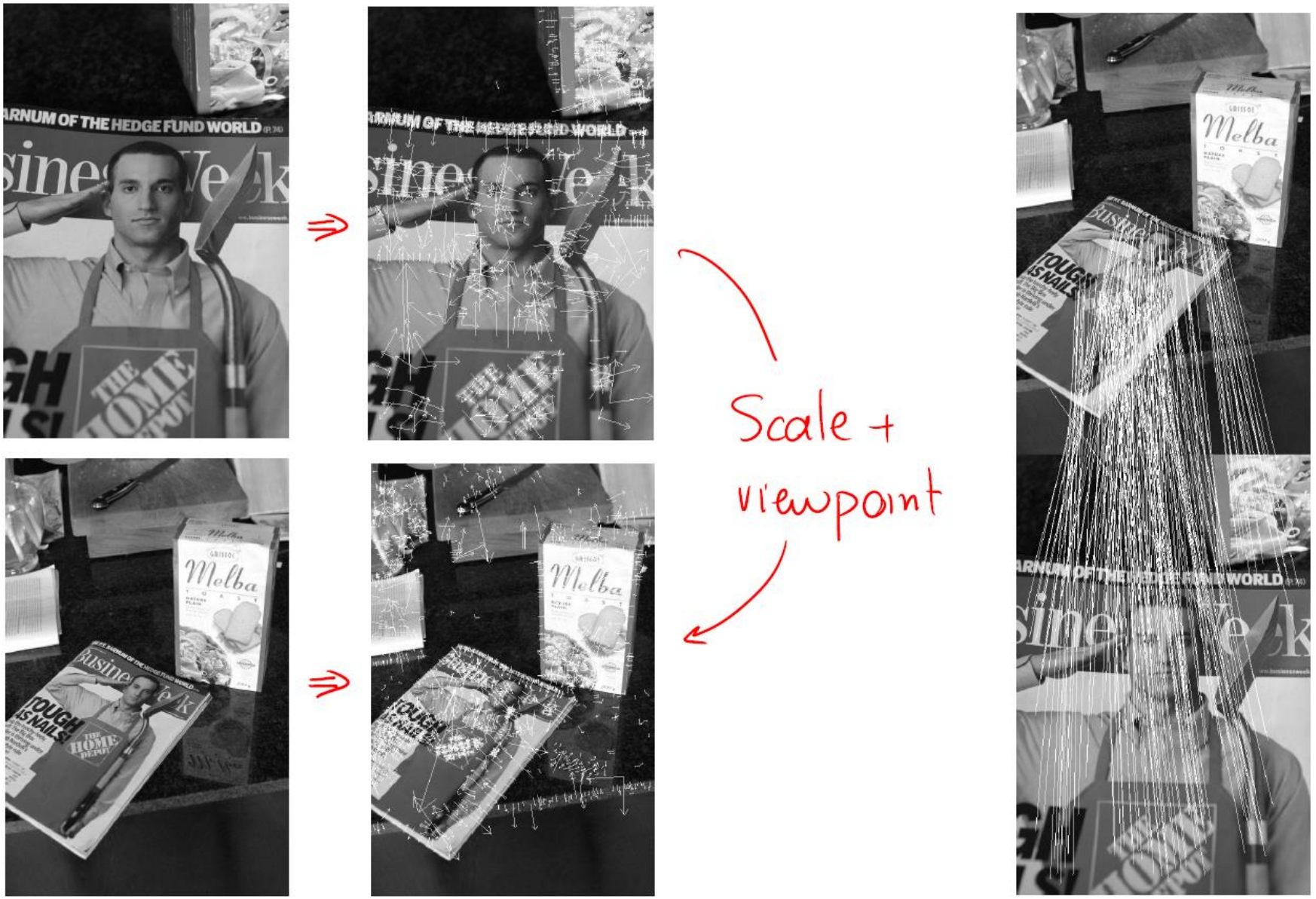


Image Matching Using SIFT Features

Image I

Image I′

Compute SIFT features of I

Compute SIFT features of I′

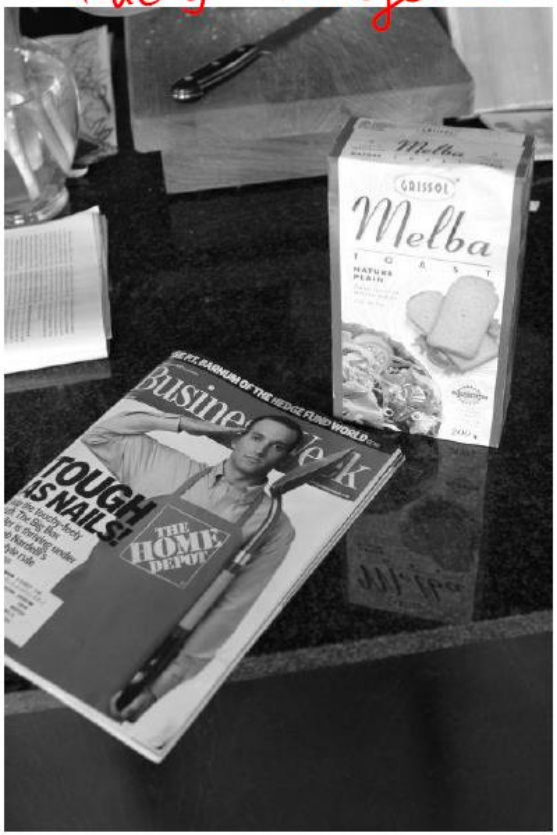
Match SIFT features in images I & I′

The SIFT Feature Detection Algorithm

Goal: Represent an image I as a collection of SIFT features that can be identified reliably in other images where the same (or similar) objects are present

"Source" Image I

"Query" image I'



The SIFT Feature Detection Algorithm

Goal: Represent an image I as a collection of SIFT features that can be identified reliably in other images where the same (or similar) objects are present

Input:

Image I

Output:

A set of k SIFT keypoints

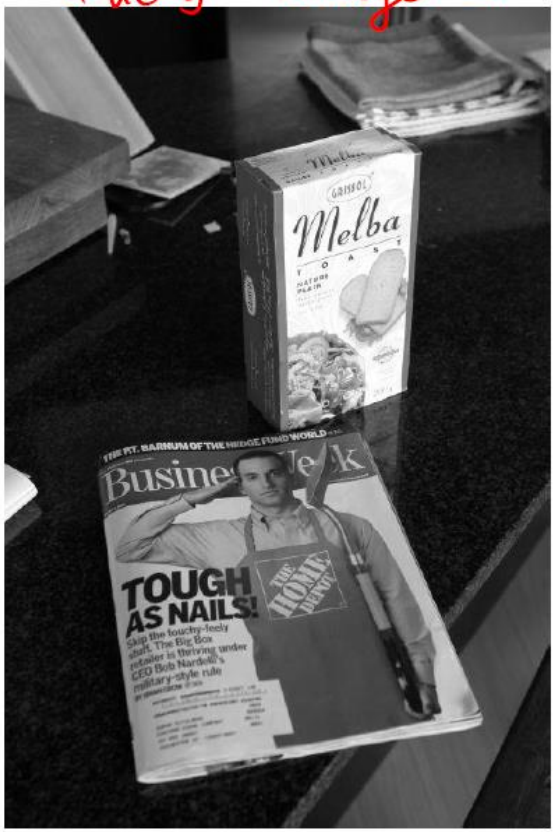
$\{p_1, p_2, \ldots, p_k\}$ &

feature vectors

$\{f_1, f_2, \ldots, f_k\}$

"Source" Image I

"Query" image I'



The SIFT Keypoints

Keypoint: A location (x,y) in the source image, with an associated orientation & scale, that is "visually distinct" from its surroundings

Input:

Image I

Output:

A set of k SIFT keypoints

$$p_i = (x_i, y_i, \sigma_i, \theta_i)$$

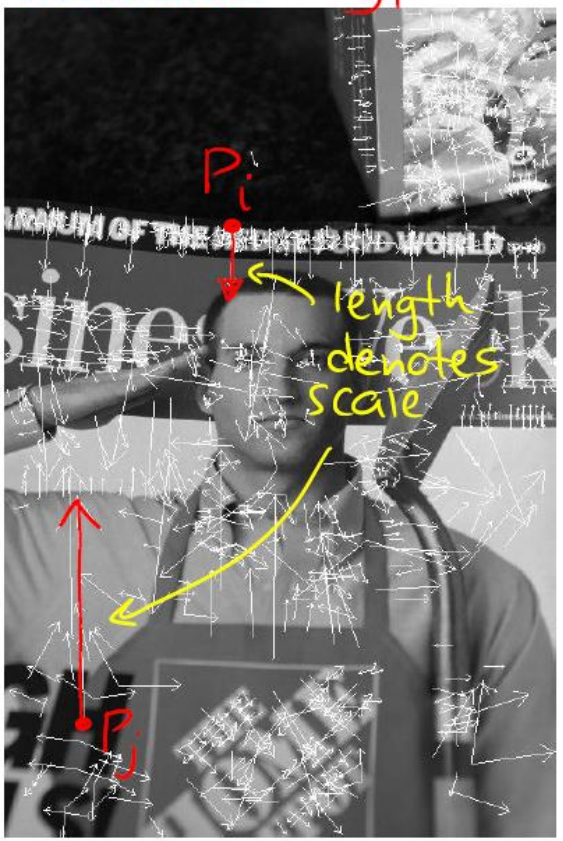
location

scale

orientation

"Source" Image I

Detected keypoints



The SIFT Feature Vectors

Feature vector (of a keypoint p_i): A vector of fixed length that represents the image patch centered at p_i

Input:

Image I

Output:

A set of K keypoints

A set of K feature vectors

$$p_i = (x_i, y_i, \sigma_i, \theta_i)$$

x_i, y_i: location; σ_i: scale; θ_i: orientation

$$f_i = [\;|\;|\;\ldots\;|\;|\;]$$

describes the patch centered at p_i

Detected keypoints

Representing an Image Using SIFT Features: Steps

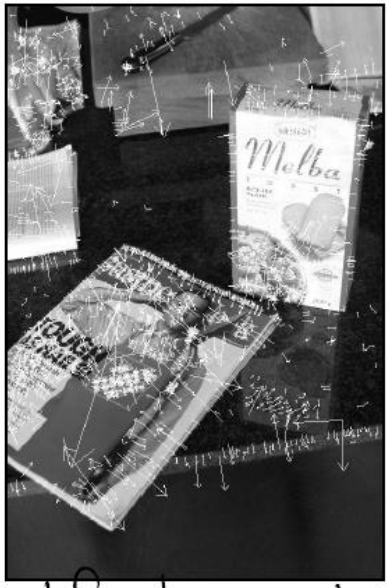
Source Image I

Matching 2 Images Using SIFT Features

Image 1

f_1
f_2
f_3
f_{k-1}
f_k

Image 1'



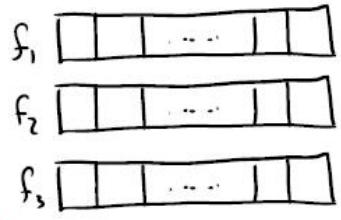
f_1'
f_2'
f_3'
$f_{k'-1}'$
$f_{k'}'$

① Identify keypoints

② Build feature vectors

Matching 2 Images Using SIFT Features

Image 1



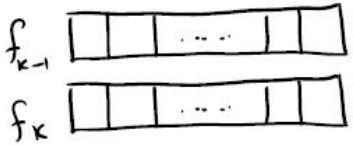
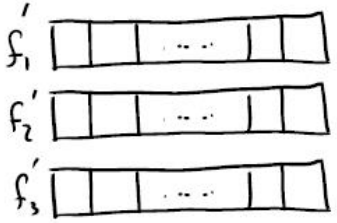
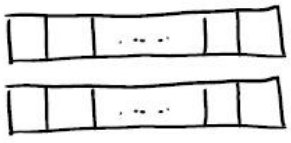


Image 1'





① Identify keypoints

② Build feature vectors

③ Match vectors

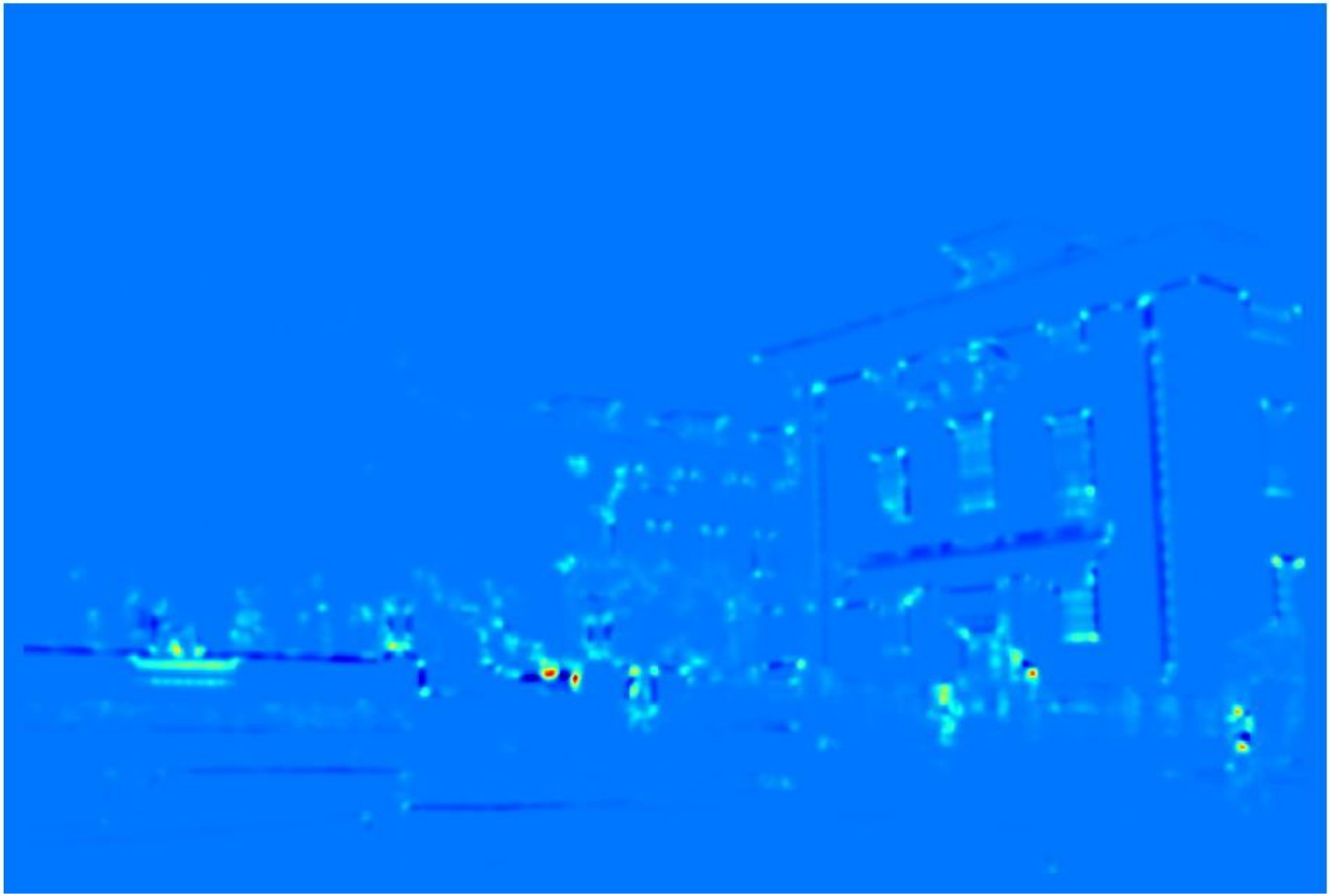
Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

- introduction to the image matching problem
- image matching using SIFT features
- **multi-scale interest-point detection**
- the SIFT feature detector
- the SIFT descriptor

Detecting corners

Slide credit: Kristen Grauman

Detecting corners

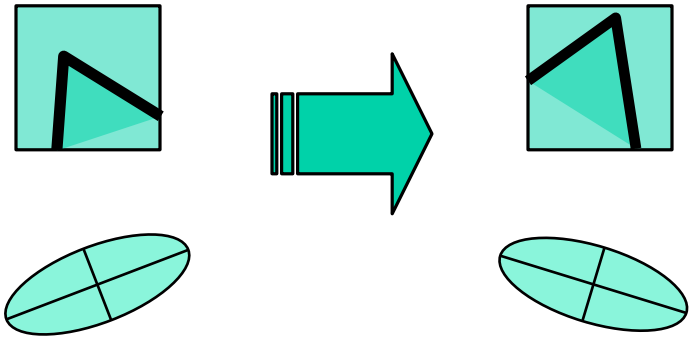


Slide credit: Kristen Grauman

Detecting corners

Slide credit: Kristen Grauman

Harris corner response is invariant to rotation



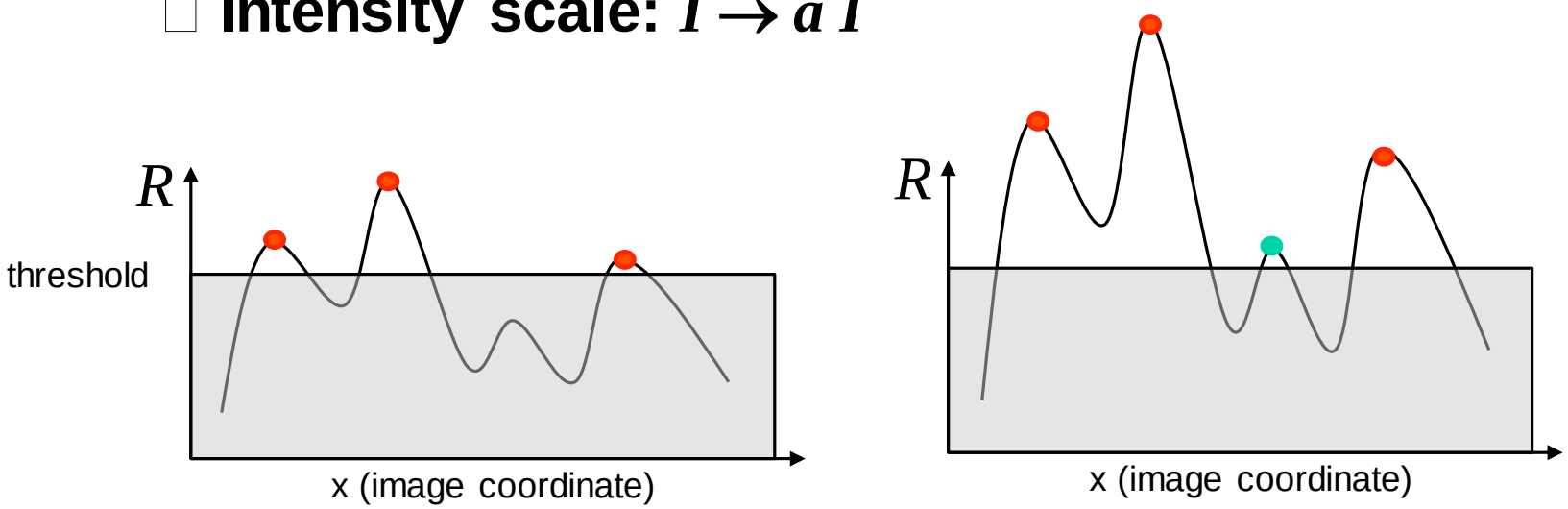
Ellipse rotates but its shape (**eigenvalues**) remains the same

Corner response R is invariant to image rotation

Harris corner response is invariant to intensity changes

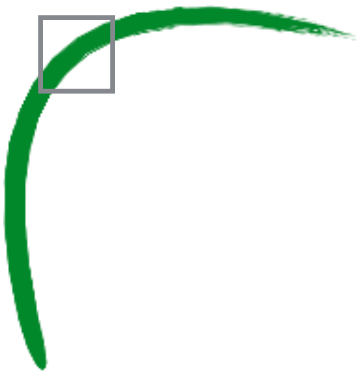
Partial invariance to *affine intensity* change

- **Only derivatives are used** => **invariance to intensity shift** $I \rightarrow I + b$
- **Intensity scale:** $I \rightarrow a\, I$



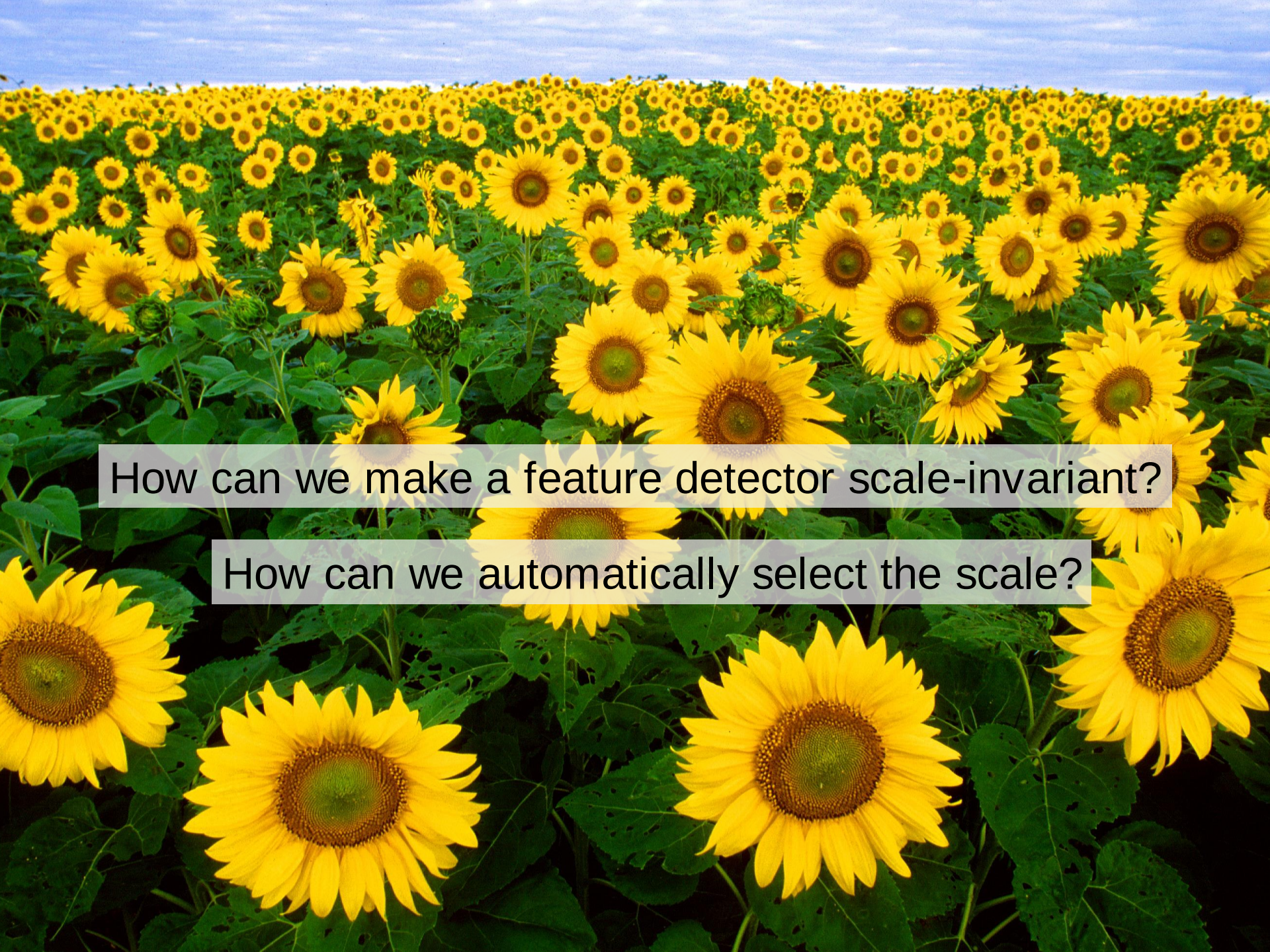
The (Harris) corner detector is not invariant to scale

edge!



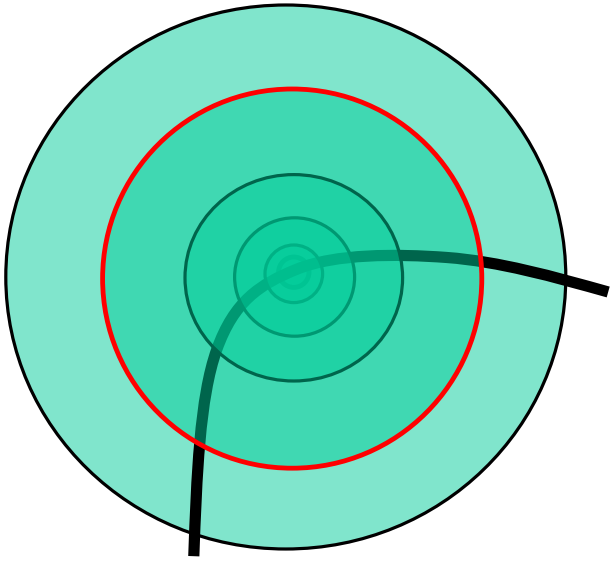
corner!

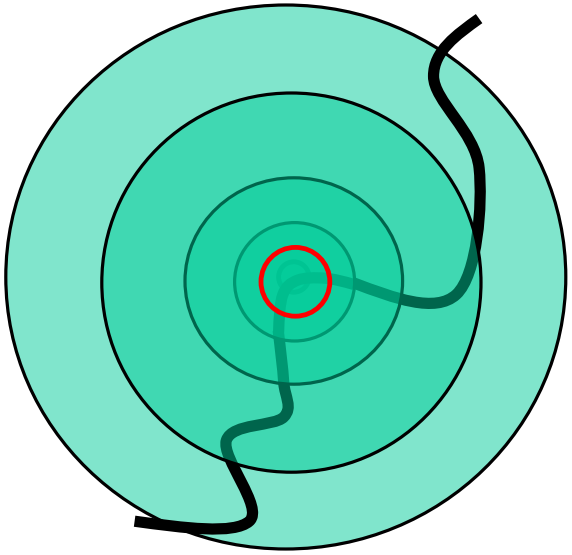
Multi-scale detection

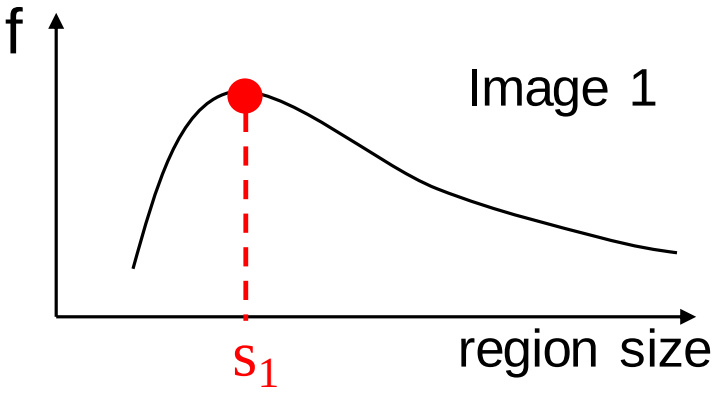

How can we make a feature detector scale-invariant?
How can we automatically select the scale?

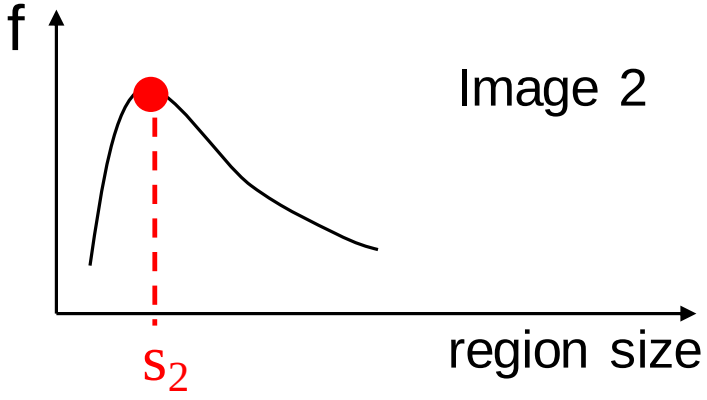
Intuitively…

Find local maxima in both **position** and **scale**



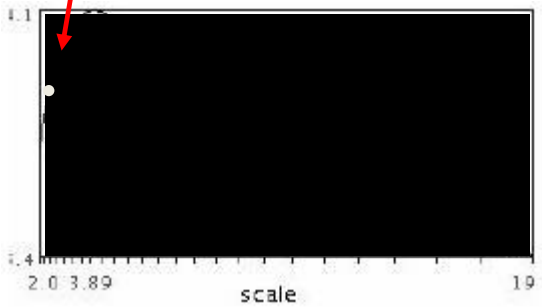




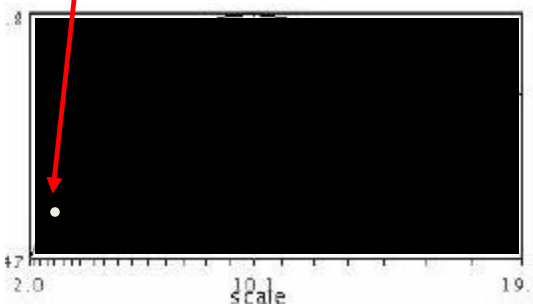


Automatic Scale Selection

- Function responses for increasing scale (scale signature)



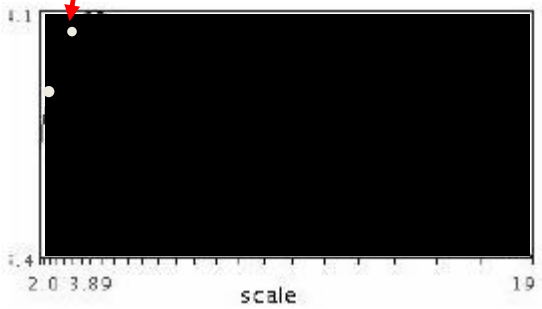
$f(I_{i_1 \ldots i_m}(x,\sigma))$



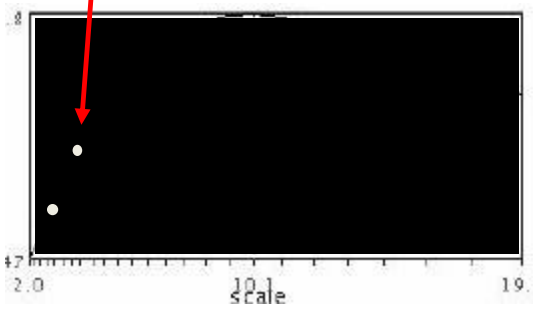
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



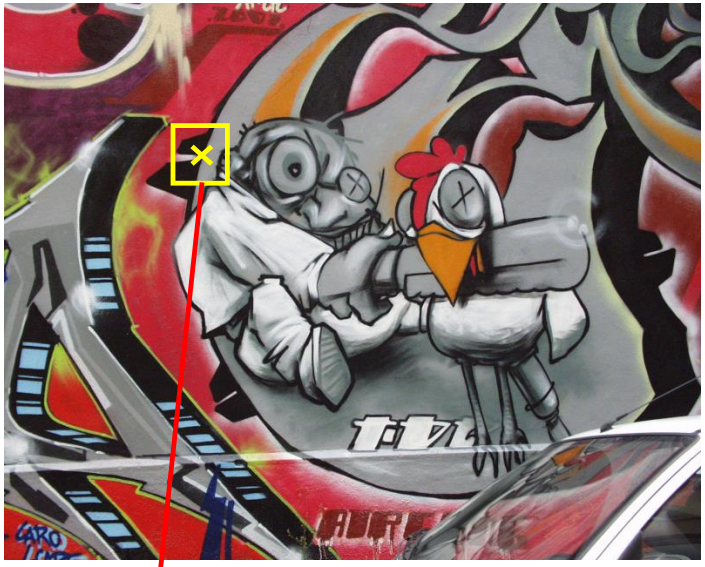
$f(I_{i_1 \ldots i_m}(x,\sigma))$

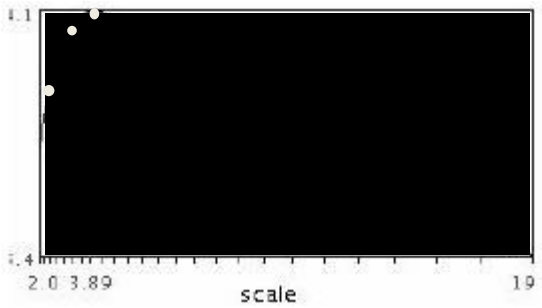


$f(I_{i_1 \ldots i_m}(x',\sigma))$

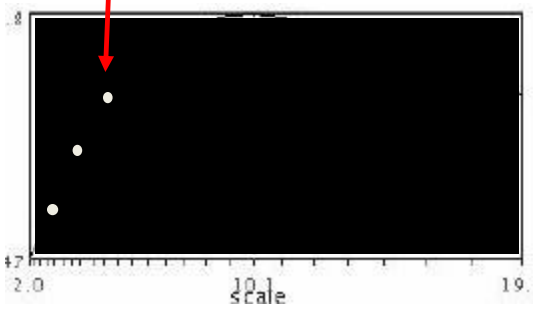
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





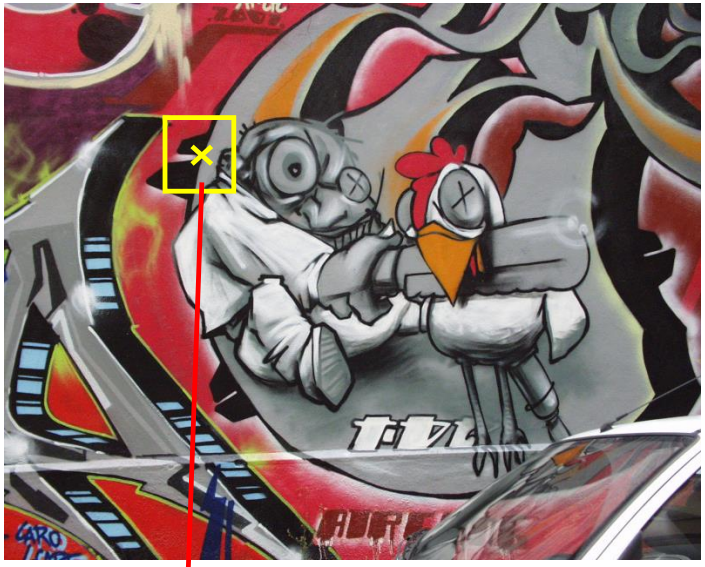
$f(I_{i_1 \ldots i_m}(x,\sigma))$

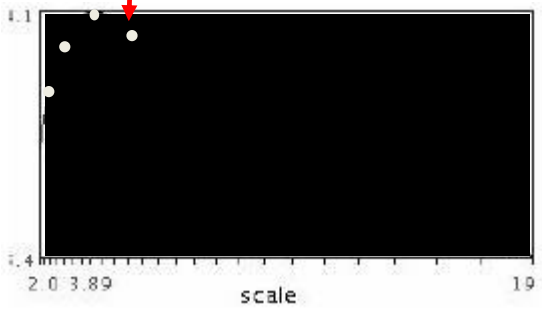


$f(I_{i_1 \ldots i_m}(x',\sigma))$

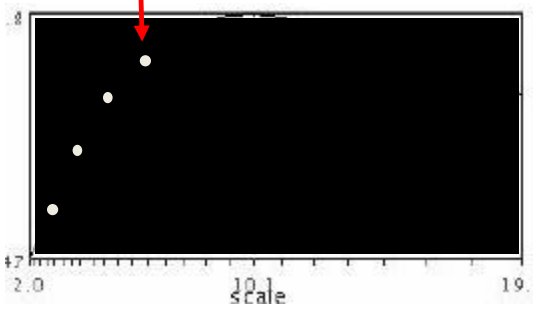
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





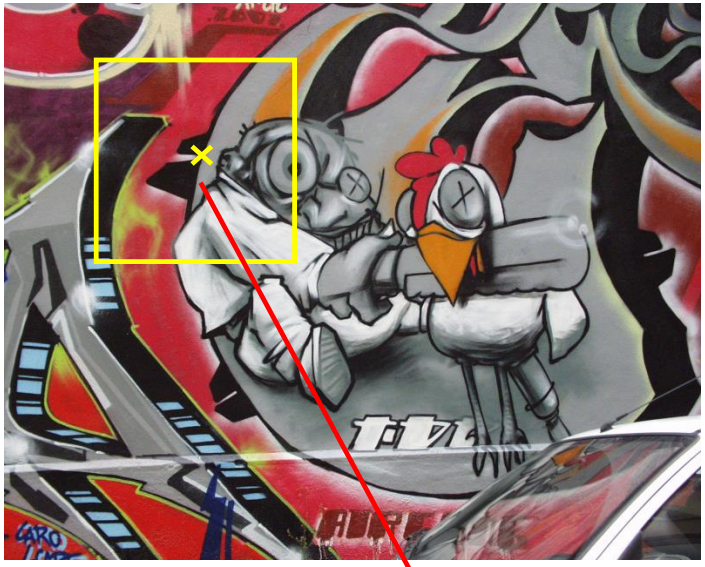
$f(I_{i_1 \ldots i_m}(x,\sigma))$

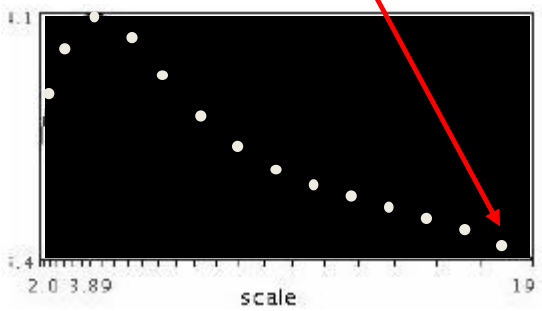


$f(I_{i_1 \ldots i_m}(x',\sigma))$

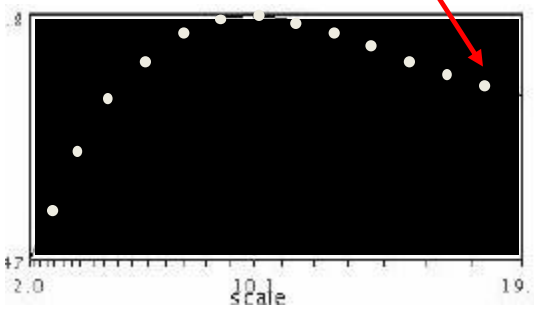
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





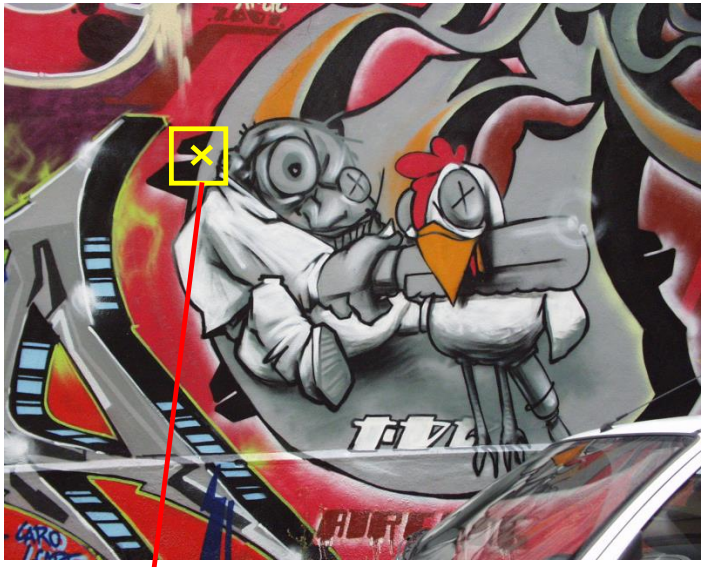
$f(I_{i_1 \ldots i_m}(x,\sigma))$

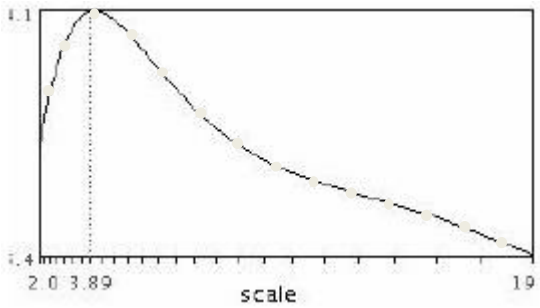


$f(I_{i_1 \ldots i_m}(x',\sigma))$

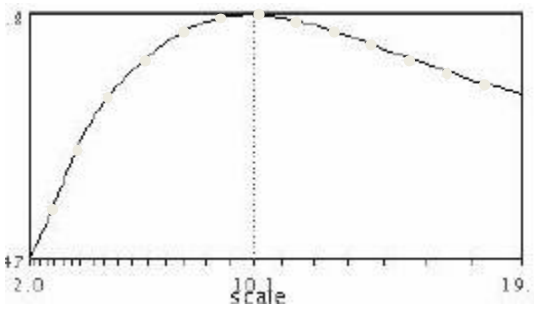
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





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



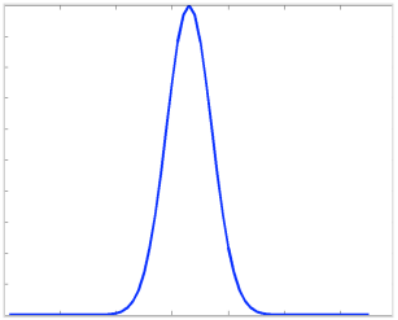
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Second Derivative Filters

Are applied to images to produce “signature” functions

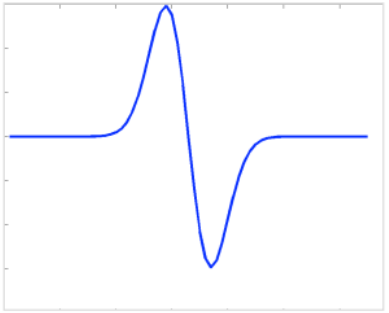
gaussian

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$



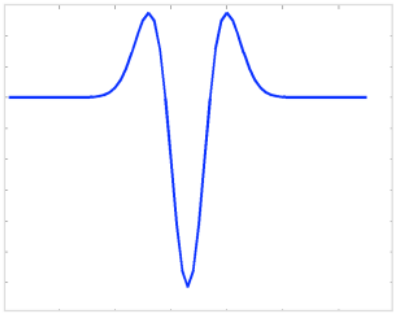
1st derivative

$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$



2nd derivative

$$g''(x) = (\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}) e^{-\frac{x^2}{2\sigma^2}}$$

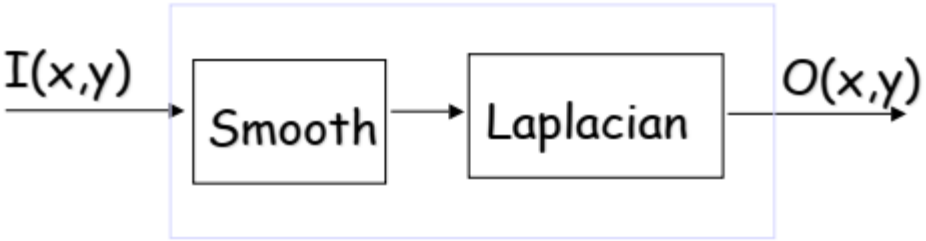


Second Derivative Filters

Laplacian

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * I$$

Second Derivative Filters



Used in practice

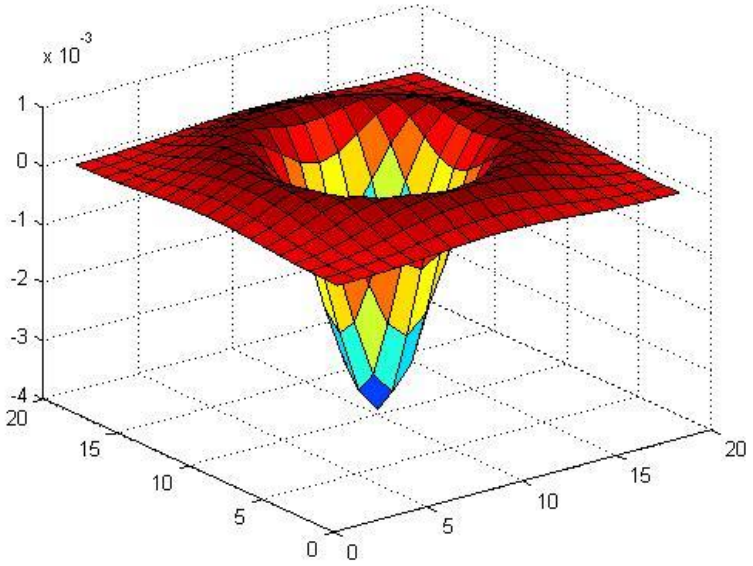
$$\nabla^2(f(x,y)\otimes G(x,y)) = \nabla^2 G(x,y)\otimes f(x,y)$$

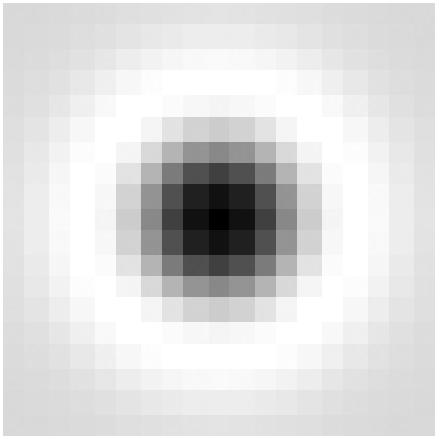
Laplacian of Gaussian-filtered image

Laplacian of Gaussian (LoG) -filtered image

Blob detection in 2D

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

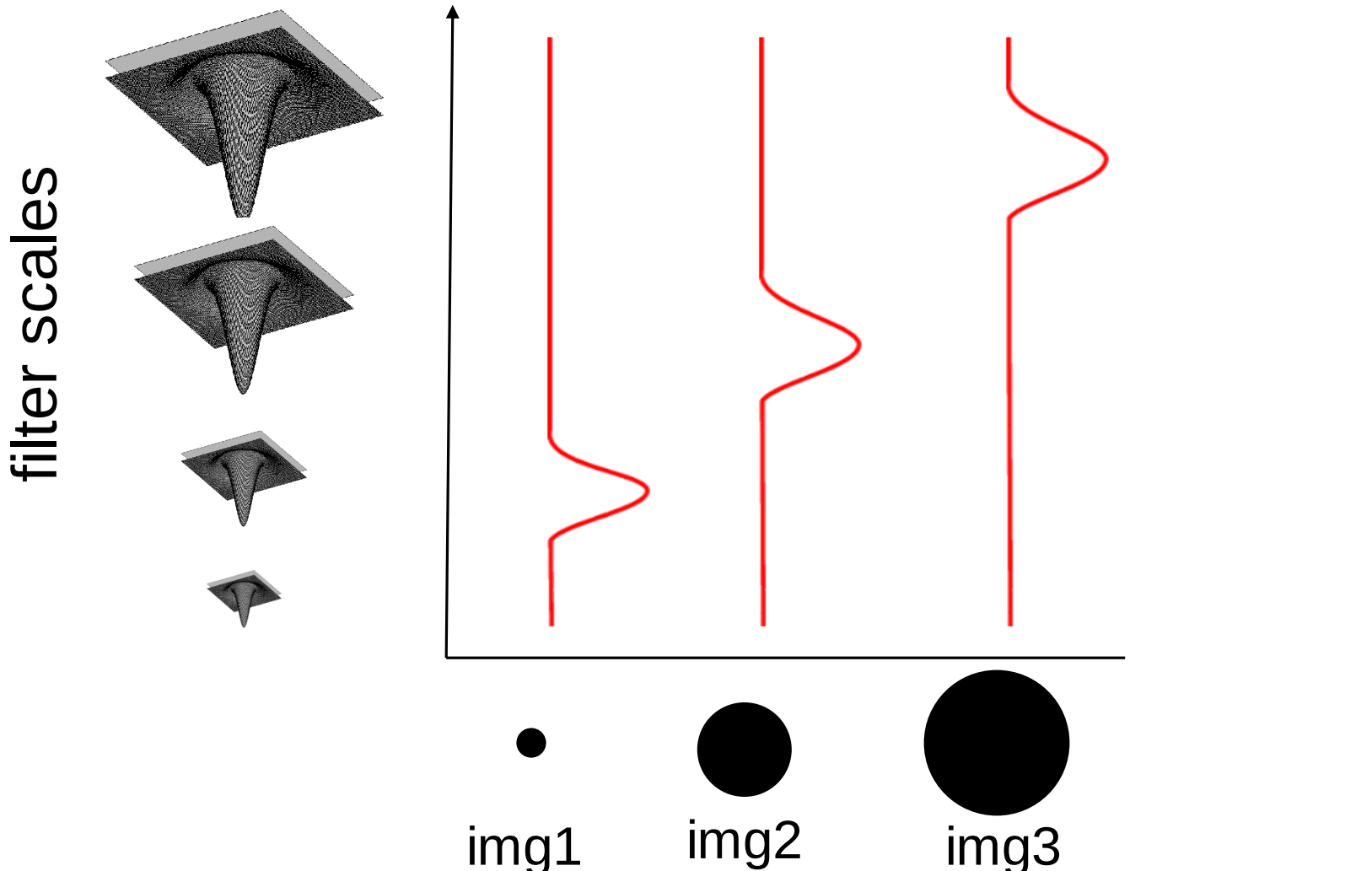




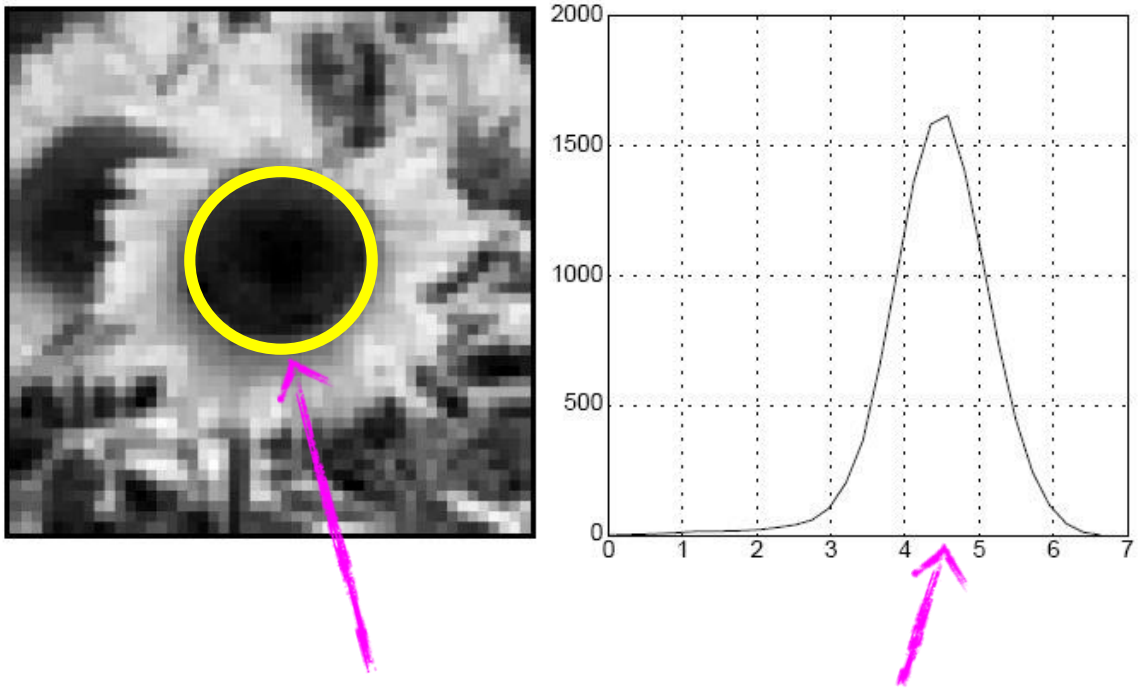
$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Blob detection in 2D: scale selection

Laplacian-of-Gaussian = “blob” detector $\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$



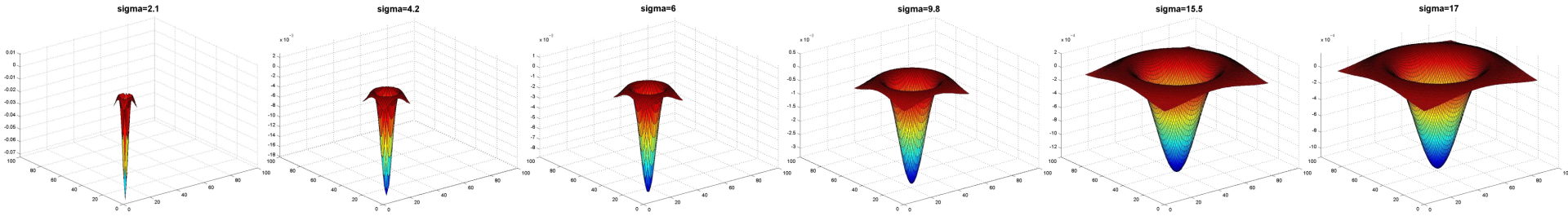
characteristic scale - the scale that produces peak filter response



characteristic scale

we need to search over characteristic scales

What happens if you apply different Laplacian filters?



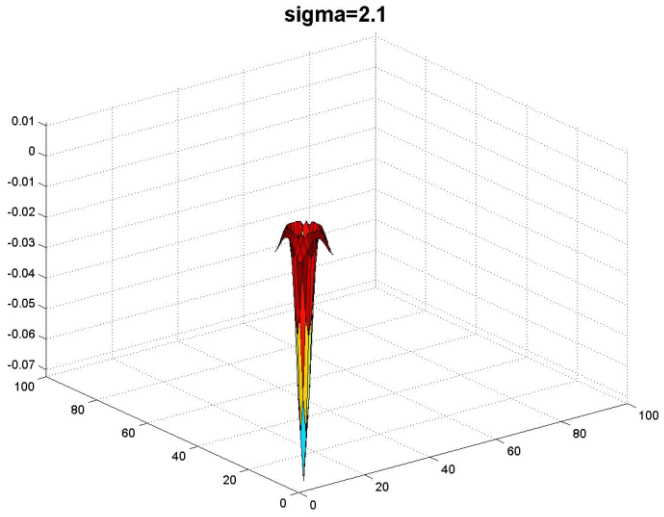
Full size

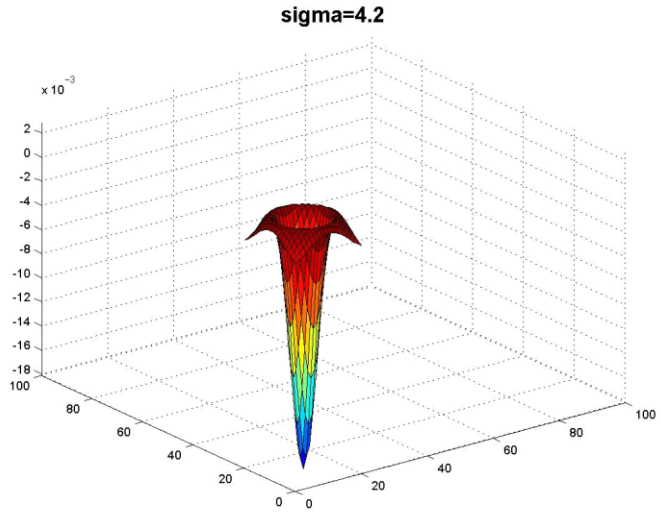
3/4 size

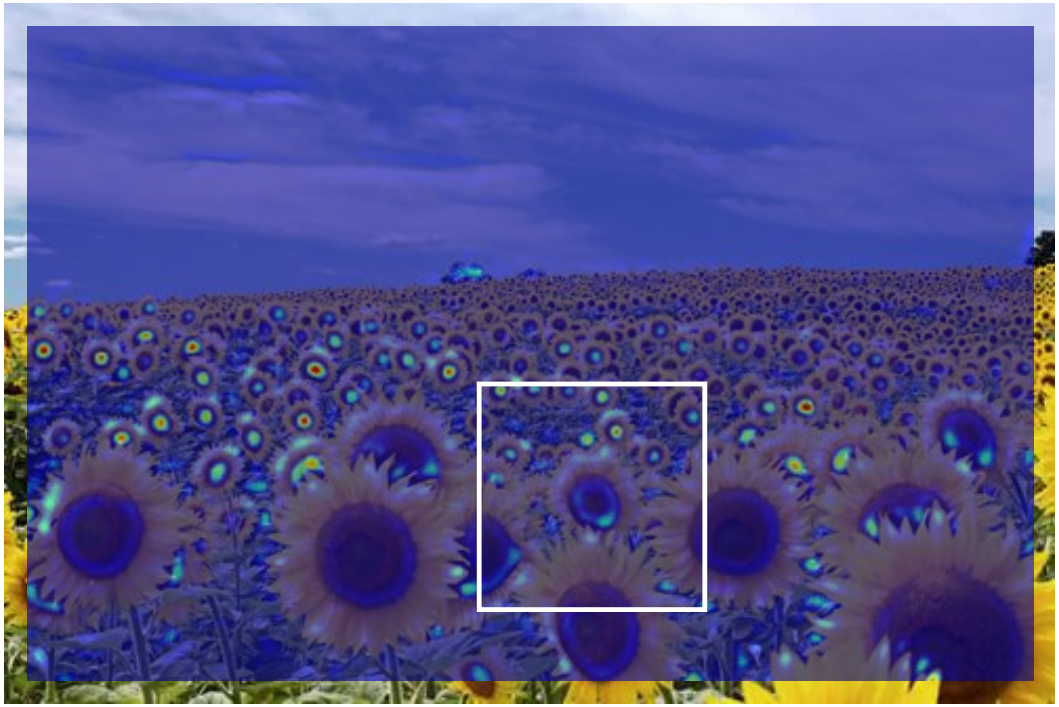
What happened when you applied different Laplacian filters?

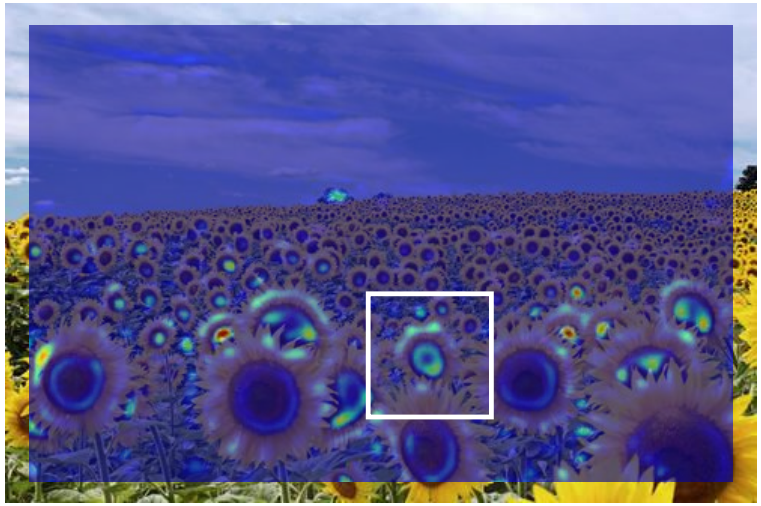
Full size

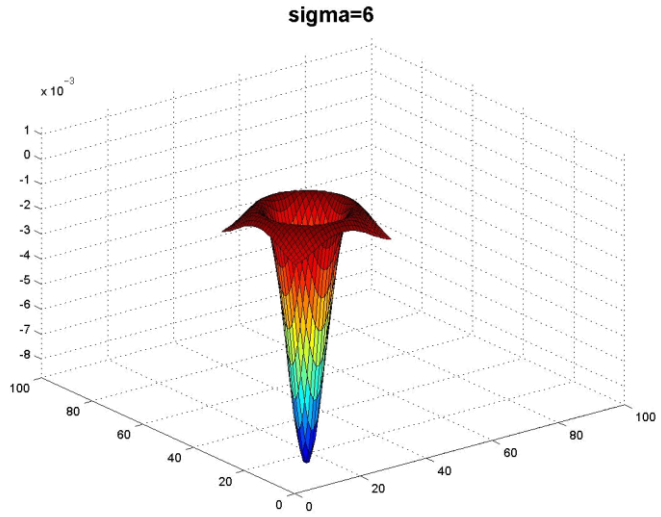
3/4 size

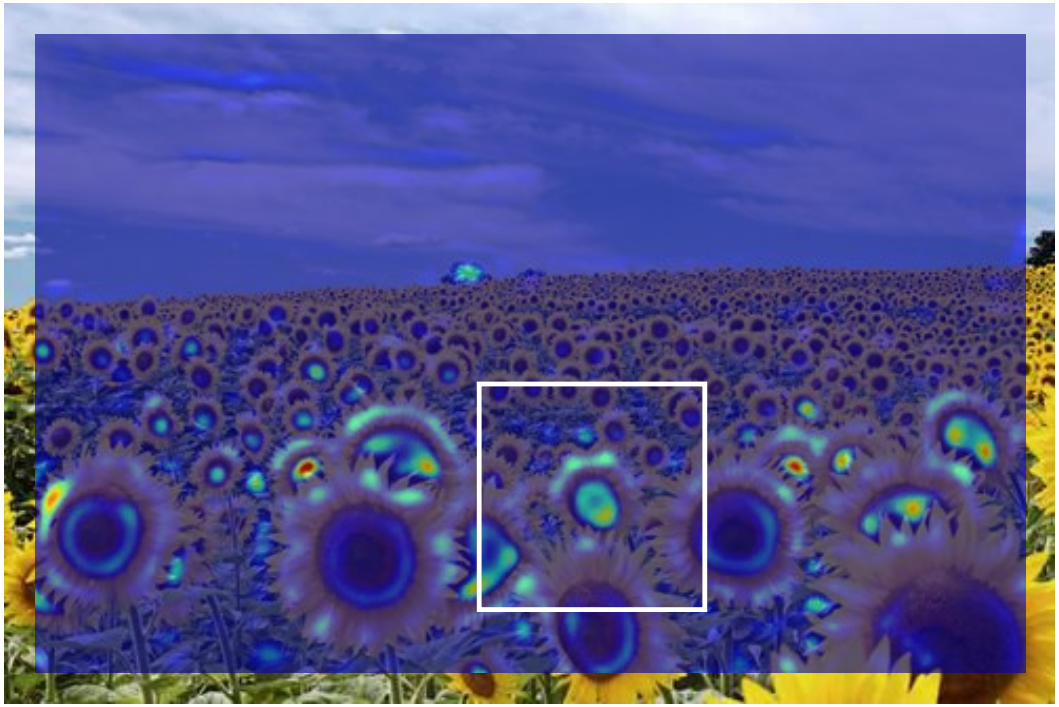

sigma=2.1
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
100
80
60
40
20
0
0
20
40
60
80
100


sigma=4.2
x 10^-3
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
100
80
60
40
20
0
0
20
40
60
80
100

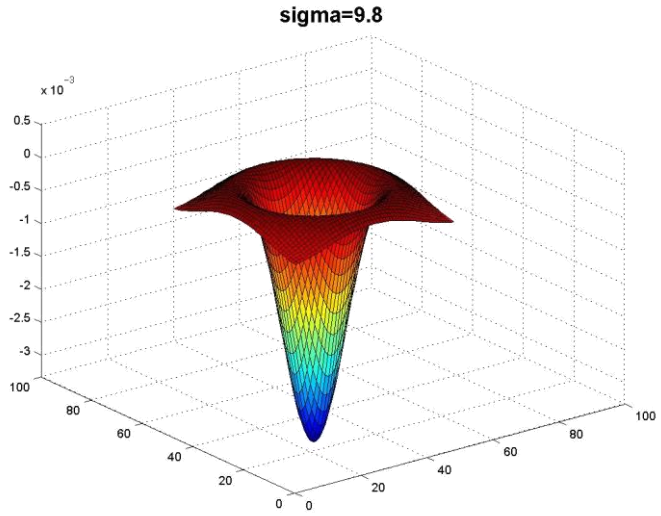




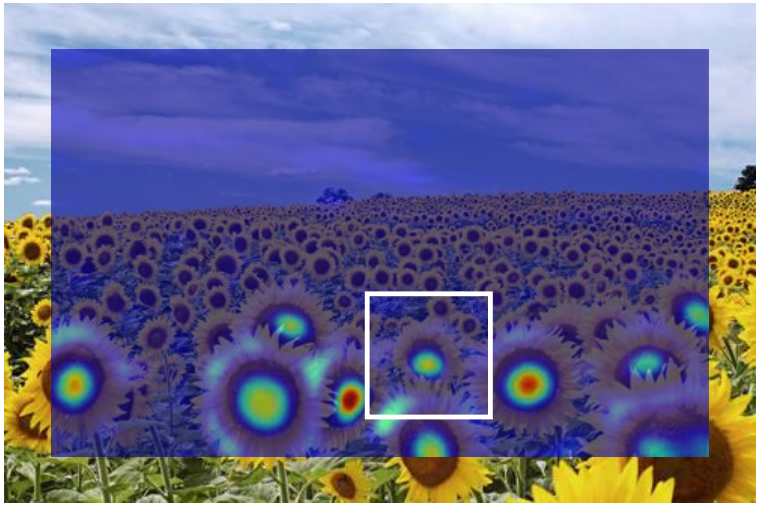

sigma=6
x 10^-3
1
0
-1
-2
-3
-4
-5
-6
-7
-8
100
80
60
40
20
0
0
20
40
60
80
100

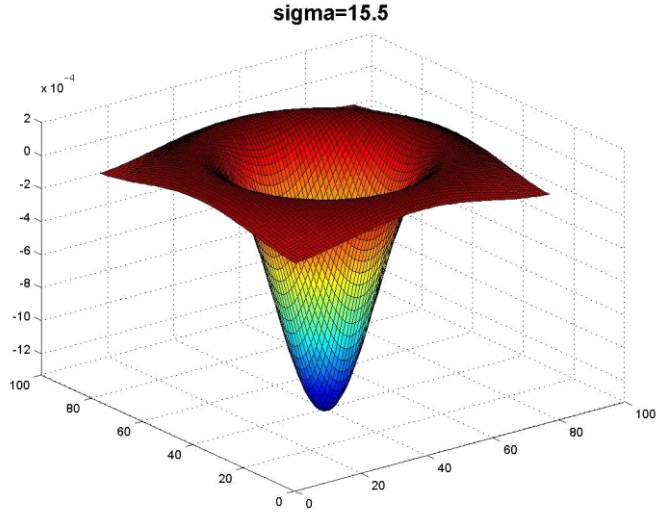


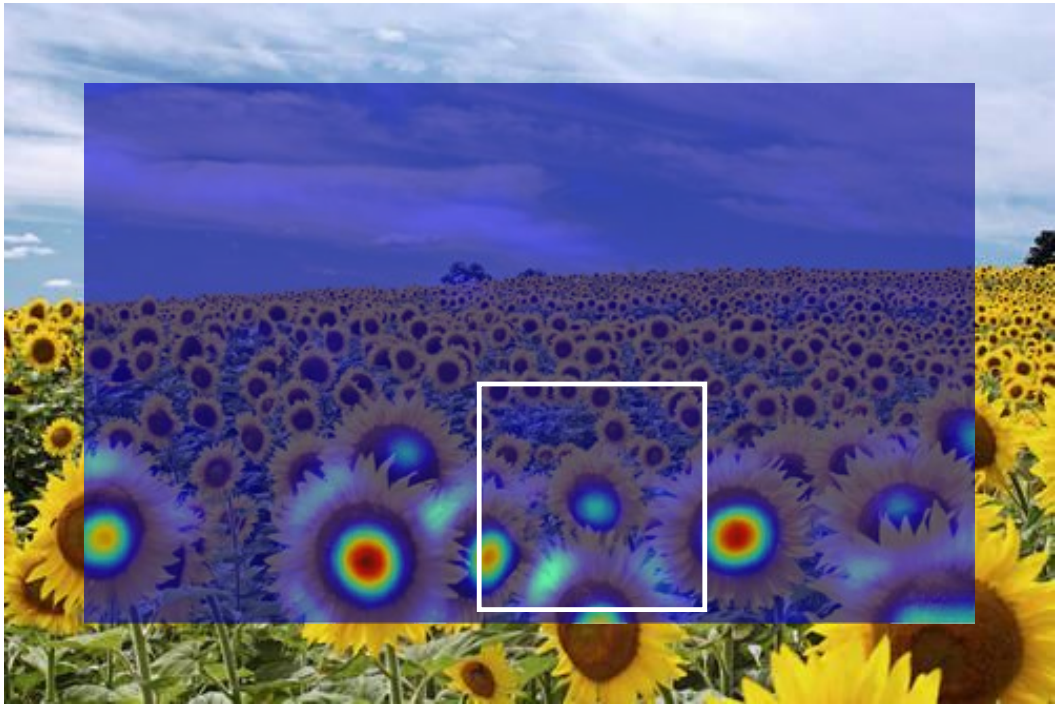

peak!

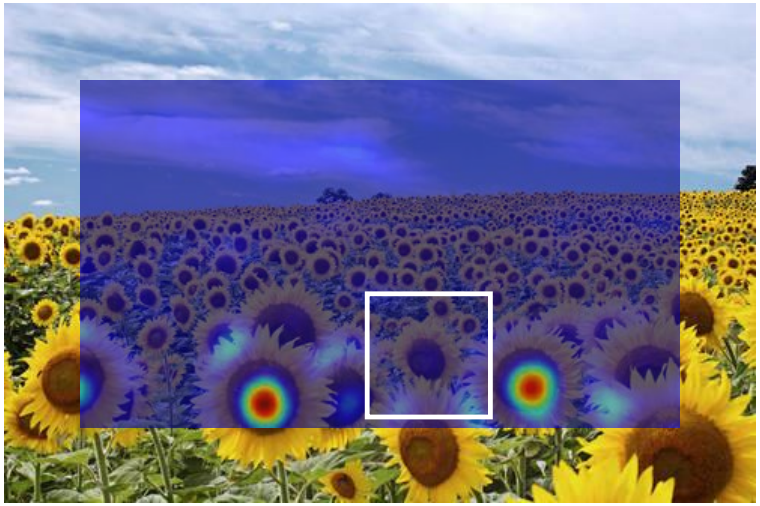

sigma=9.8
x 10^-3
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100

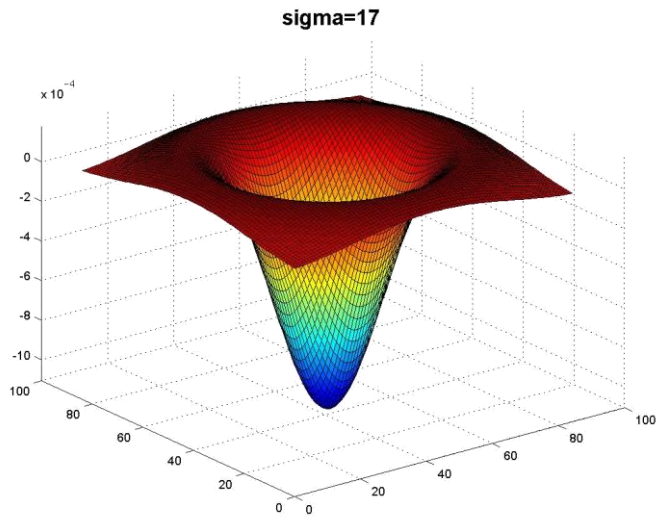

peak!

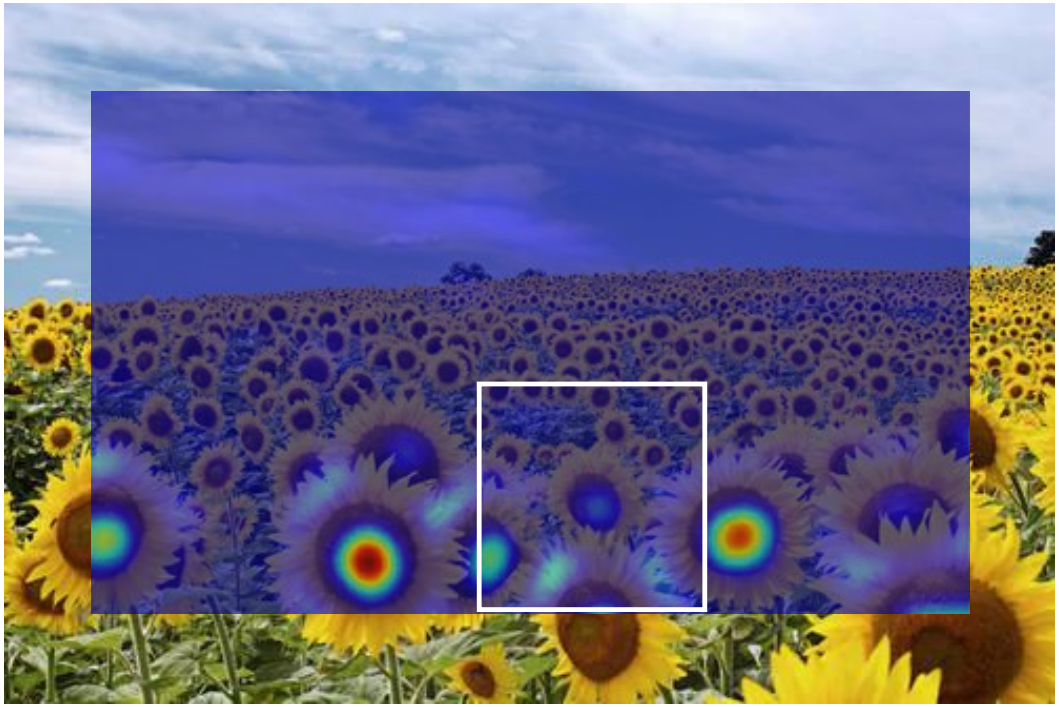


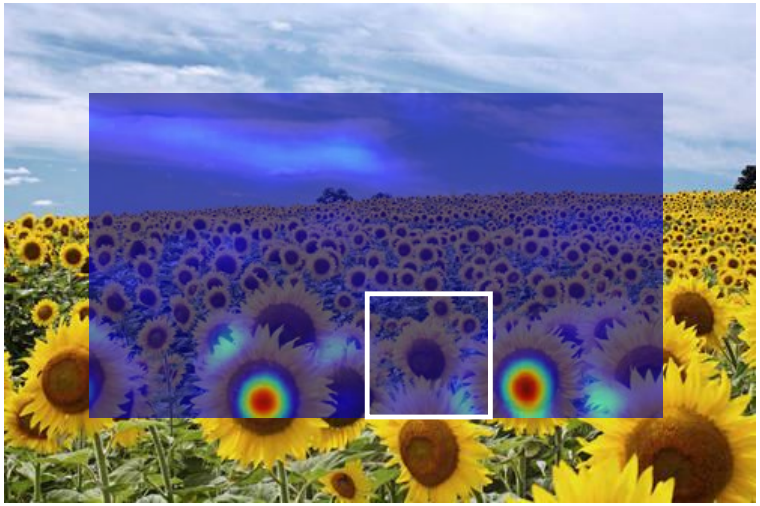

sigma=15.5
x 10^-4
2
0
-2
-4
-6
-8
-10
-12
100
80
60
40
20
0
0
20
40
60
80
100






sigma=17
x 10^-4
0
-2
-4
-6
-8
-10
100
80
60
40
20
0
0
20
40
60
80
100



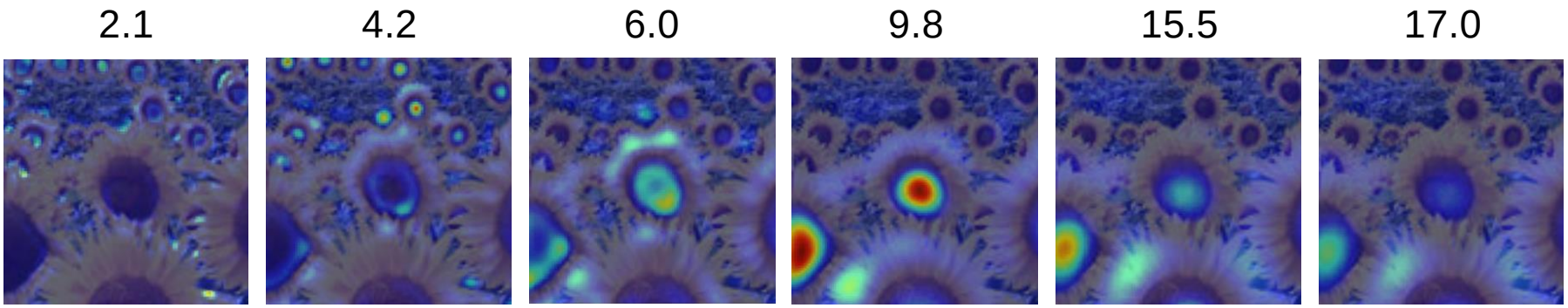


What happened when you applied different Laplacian filters?

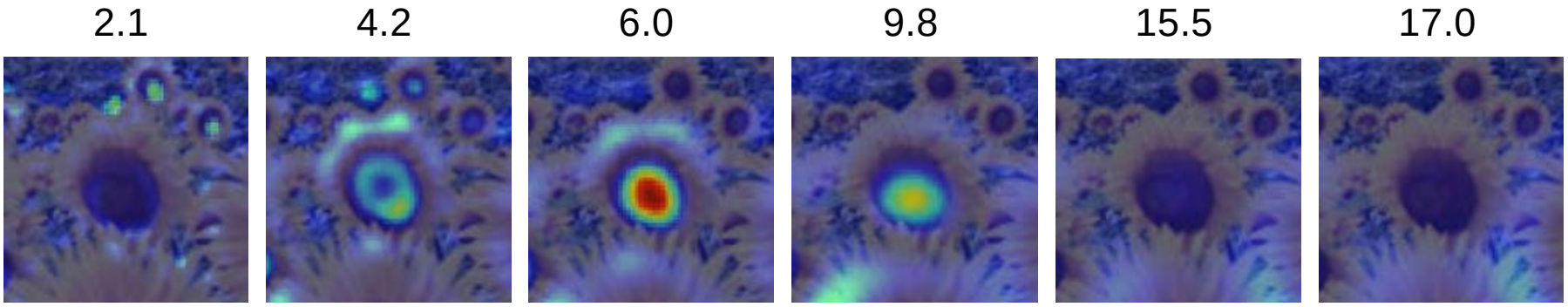
Full size

3/4 size

optimal scale

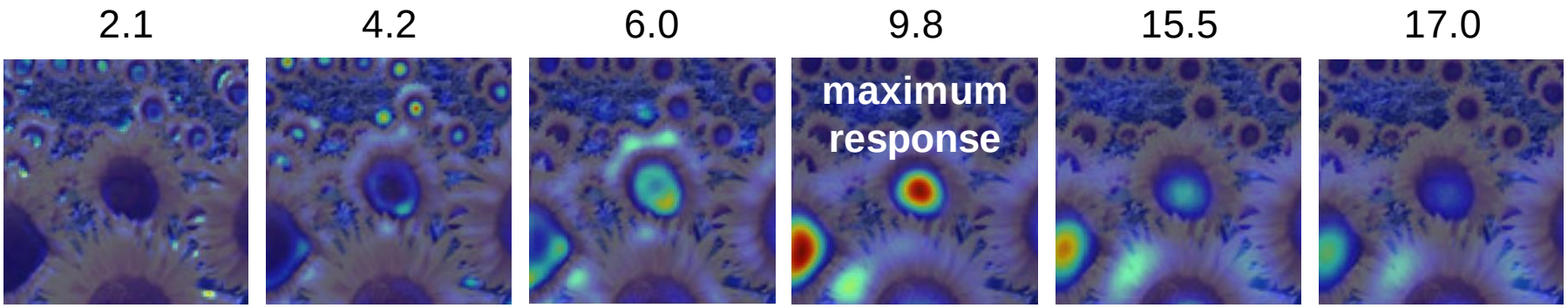


Full size image

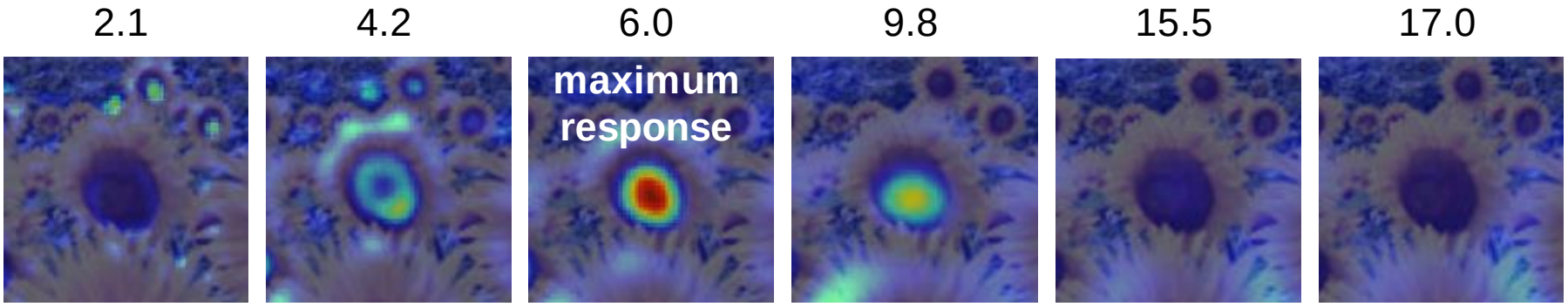


3/4 size image

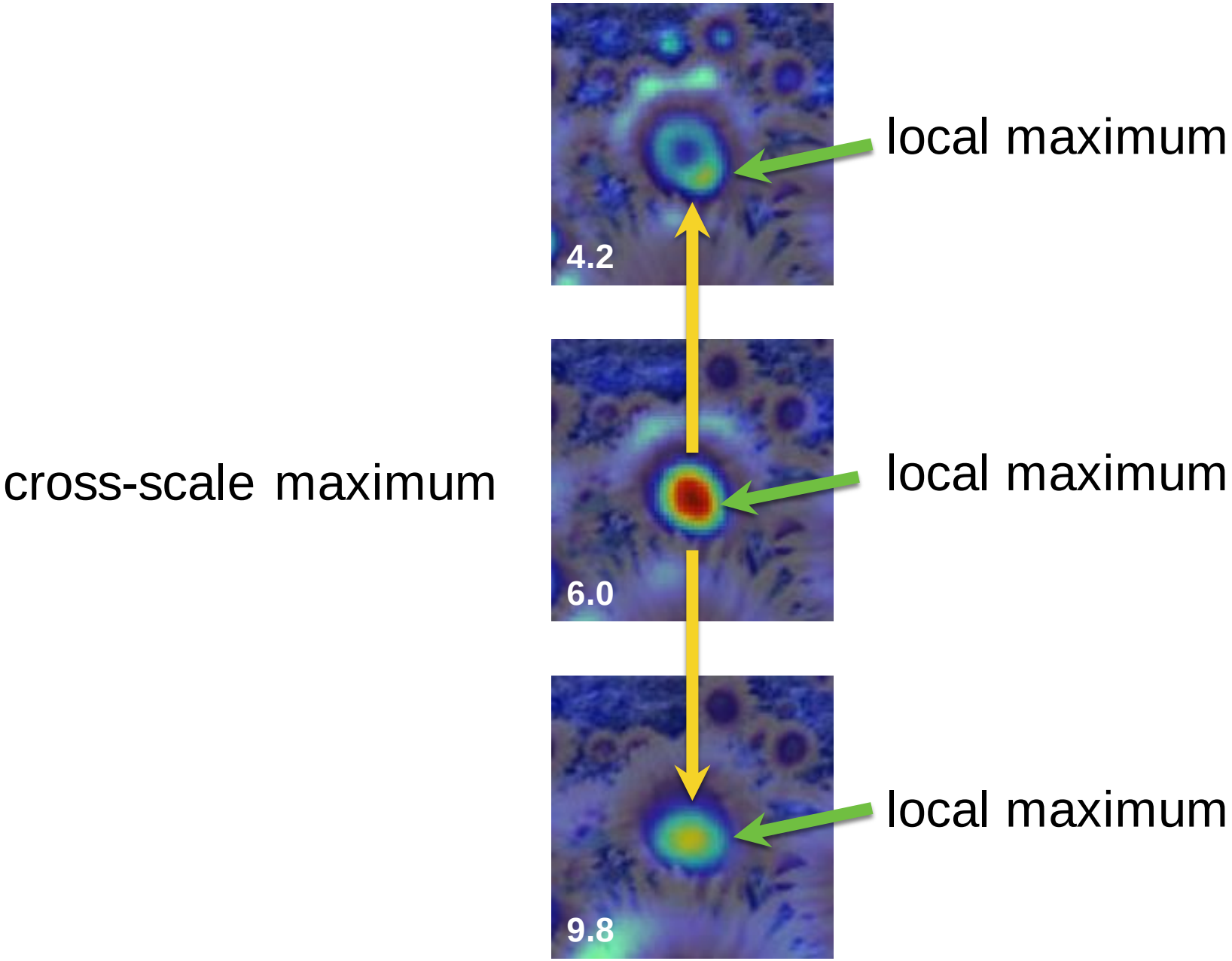
optimal scale



Full size image



3/4 size image


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

implementation

For each level of the Gaussian pyramid:

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

For each level of the Gaussian pyramid:

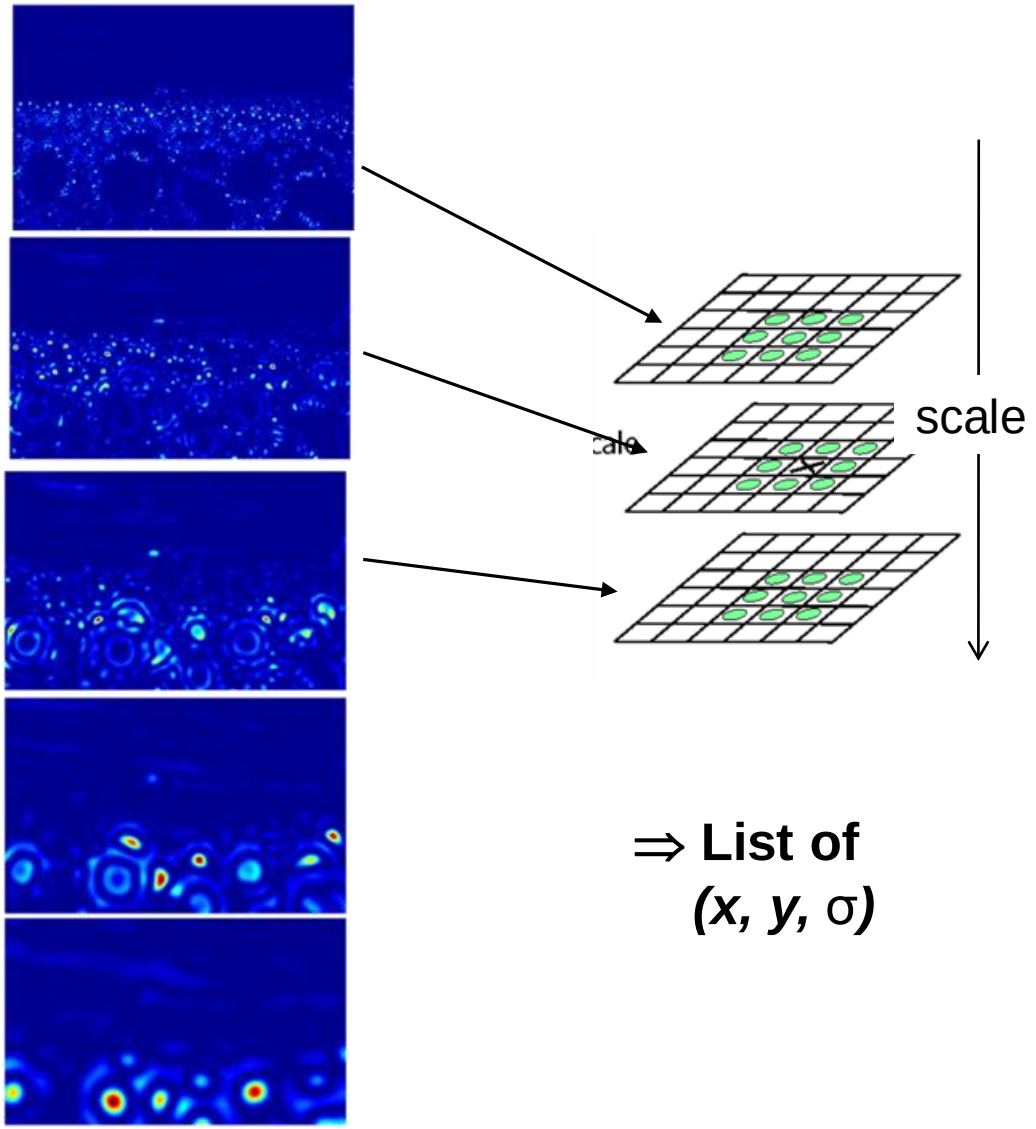
    if local and cross-scale maximum:

        save scale and location of feature (x, y, s)

Scale invariant interest points

Interest points are local maxima in both position and scale.

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



⇒ **List of** ***(x, y,*** σ***)***

Slide credit: Kristen Grauman

Scale-space blob detector: Example

Image credit: Lana Lazebnik

Technical detail

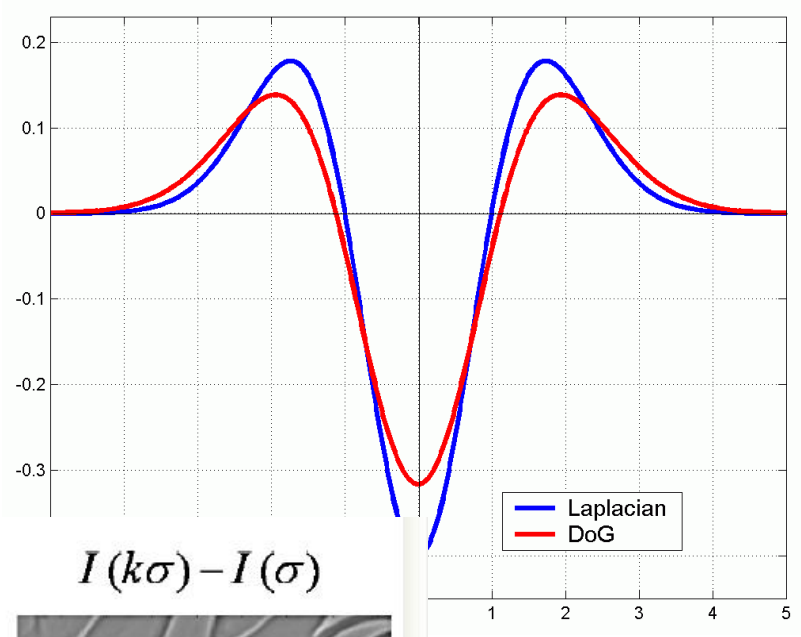
Can approximate the Laplacian with a difference of Gaussians; more efficient

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

(Laplacian of Gaussian)

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

(Difference of Gaussians)



$I(k\sigma)$ $I(\sigma)$ $I(k\sigma) - I(\sigma)$

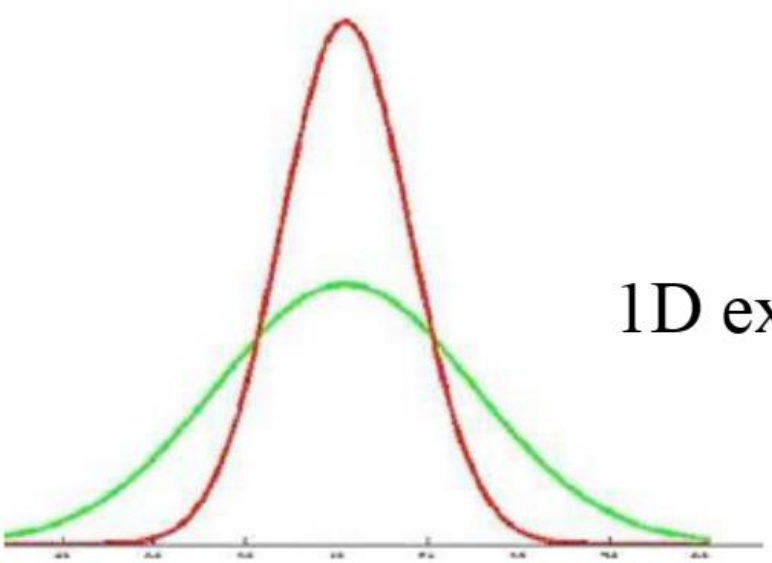
Technical detail

LoG can be approximate by a Difference of two Gaussians (DoG) at different scales

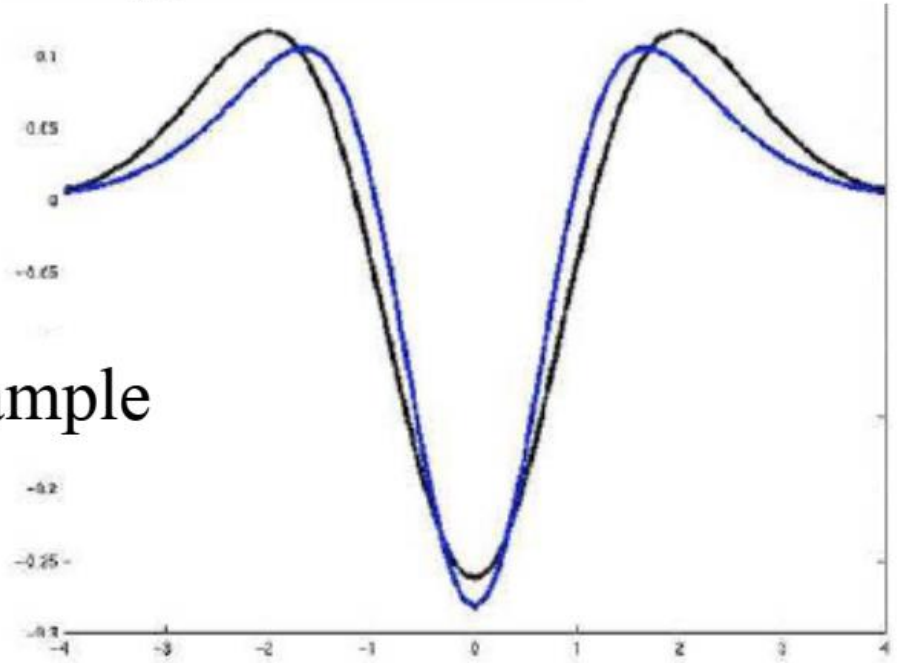
$$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$$

Best approximation when:

$\sigma_1 = \frac{\sigma}{\sqrt{2}}$, $\sigma_2 = \sqrt{2}\sigma$



1D example



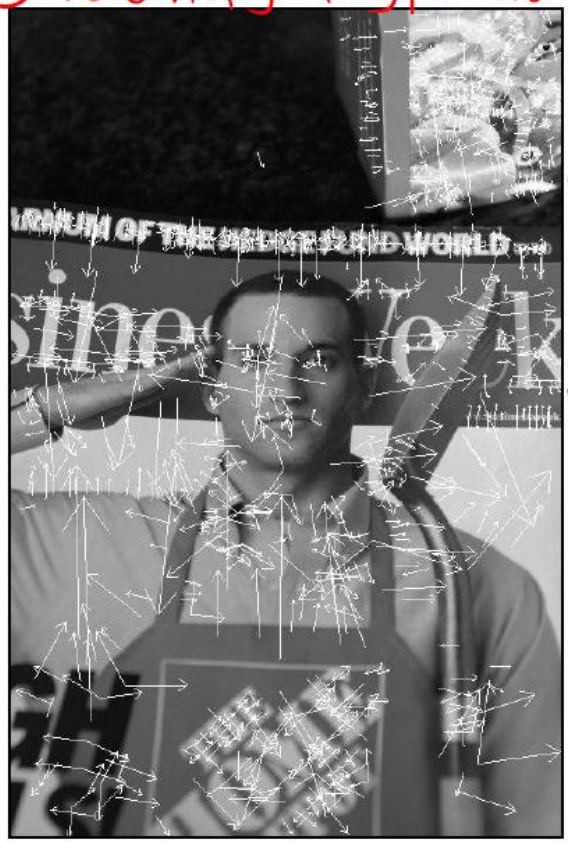
Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

- introduction to the image matching problem
- image matching using SIFT features
- multi-scale interest-point detection
- **the SIFT feature detector**
- the SIFT descriptor

Step 1: Compute a Set of Keypoints

Source Image I

① Identify keypoints



Goals:

- Identify distinctive image locations
- Assign scale & orientation to each keypoint
- Should be able to detect same keypoint in images that vary in magnification brightness, etc

Computing SIFT Keypoints: Basic Steps

Source image I

Gauss-pyramid

Step 1a

Build pyramid of Gauss-smoothed images

DOG pyramid

Step 1b

Buid DOG pyramid

DOG extrema

Step 1c

Locate extrema of DOG pyramid

(x_i, y_i, ρ_i)

Step 1d

Refine location of DOG extrema

$(x_i, y_i, \rho_i) \rightarrow (x_i', y_i', \rho_i')$

Location refinement

Step 1e

Prune set of extrema

Keypoints = { all remaining (x_i', y_i', ρ_i') }

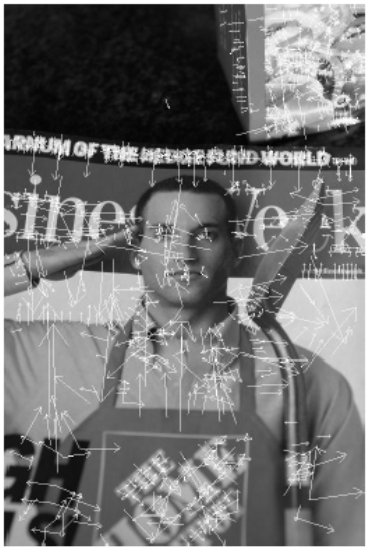
Extremum pruning

Step 1f

Assign orientation to extrema

$P_i = (x_i', y_i', \rho_i', \theta_i)$

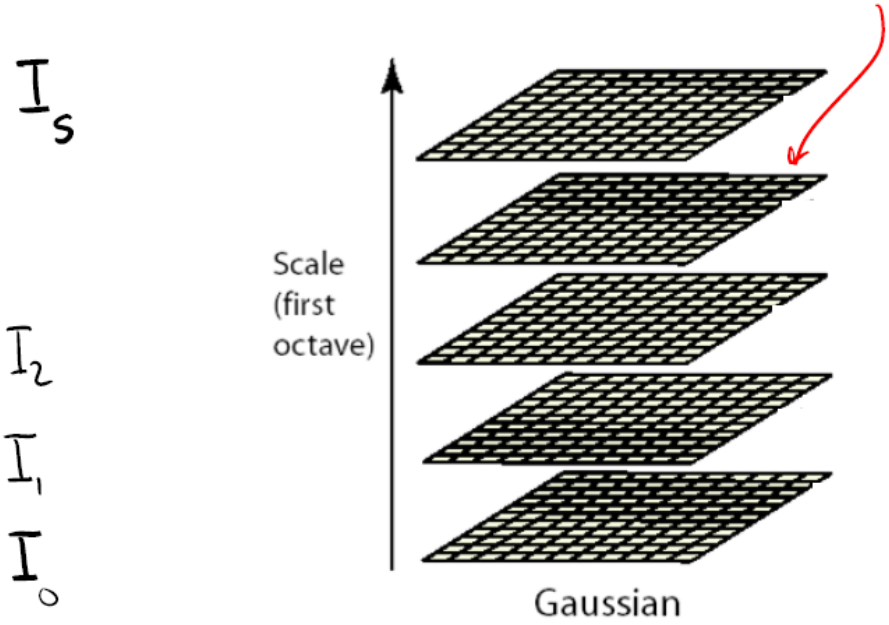
Orientation assign



Step 1a: Construct a Gauss-like Pyramid

Step 1a: Compute a pyramid of Gauss-filtered images, organized into octaves of s+1 images

each image is smoothed by a factor of k more than the image below it



Step 1a: Construct a Gauss-like Pyramid

SIFT terminology- An octave is a set of Gauss-convolved images, $I_1, \ldots, I_s$ representing a doubling of the scale parameter σ between I_1 and I_s

each image is smoothed by a factor of k more than the image below it

each octave contains $s+1$ images

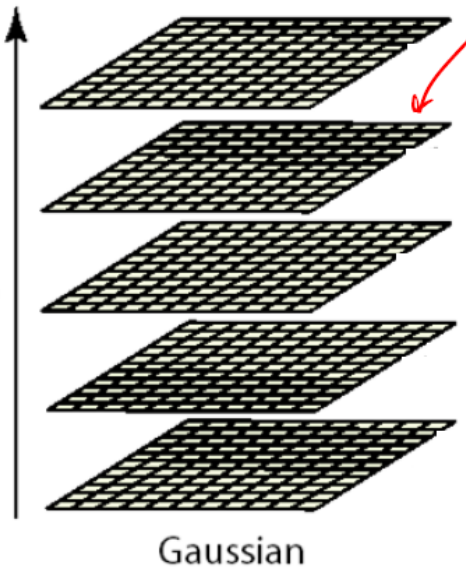
$I_s = I * G_{k\sigma}^{s}$

$\vdots$

$I_2 = I * G_{k \cdot (k\sigma)}$

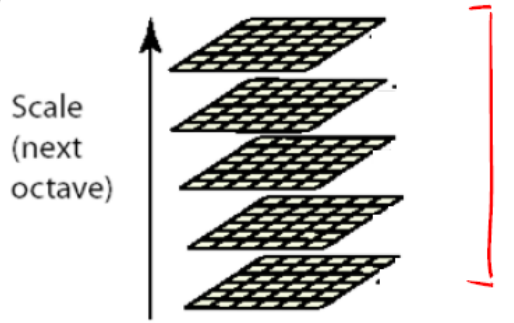
$I_1 = I * G_{k\sigma}$

$I_0 = I * G_{\sigma}$



Step 1a: Construct a Gauss-like Pyramid

Step 1a: Compute a pyramid of Gauss-filtered images, organized into <u>octaves</u> of s+1 images



Images in next octave are subsampled & stored at 1/2 resolution of previous octave

each octave contains s+1 images

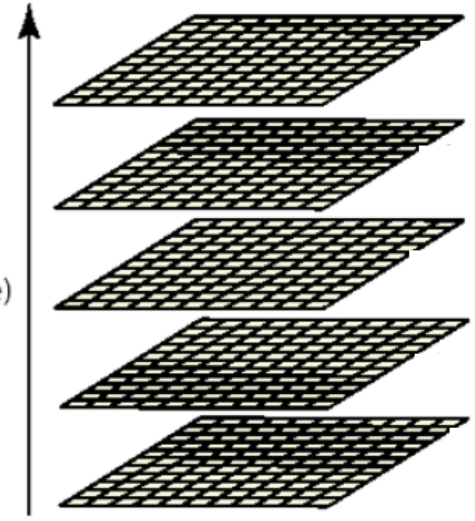
$I_s = I * G_{k^s\sigma}$

$\vdots$

$I_2 = I * G_{k\cdot(k\sigma)}$

$I_1 = I * G_{k\sigma}$

$I_0 = I * G_{\sigma}$



In practice:

- $s = 3$ (3 images/octave)
- $\sigma = 1.6$ (for 1st octave)

Computing SIFT Keypoints: Basic Steps

Source image I

Gauss-pyramid

Step 1a

Build pyramid of Gauss-smoothed images

DOG pyramid

Step 1b

Buid DOG pyramid

DOG extrema

Step 1c

Locate extrema of DOG pyramid

(x_i, y_i, ρ_i)

Step 1d

Refine location of DOG extrema

$(x_i, y_i, \rho_i) \rightarrow (x_i', y_i', \rho_i')$

Location refinement

Step 1e

Prune set of extrema

Keypoints = { all remaining (x_i', y_i', ρ_i') }

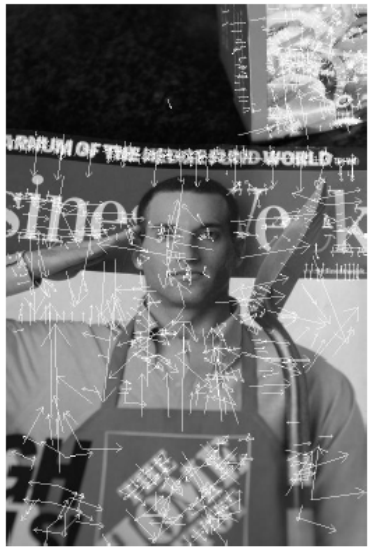
Extremum pruning

Step 1f

Assign orientation to extrema

$p_i = (x_i', y_i', \rho_i', \vartheta_i)$

Orientation assign



Step 1b: Compute Pyramid of DOG Images

Step 1b: Compute a pyramid of DOG-filtered images

$$D(x,y,\rho) = I(x,y) * \left(G(x,y,k\rho) - G(x,y,\rho)\right)$$
$$\text{for } \rho = \sigma, k\sigma, k^2\sigma, \ldots, k^{s-1}\sigma$$

each octave contains s+1 images

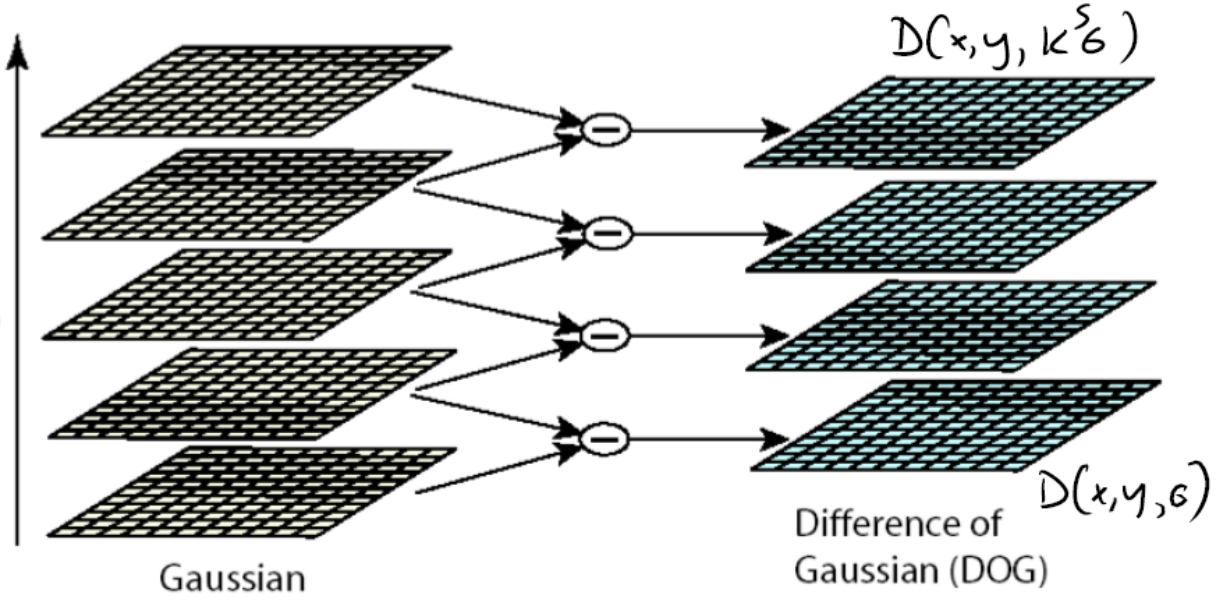
$I_s = I * G_{k^s\sigma}$

$\vdots$

$I_2 = I * G_{k\cdot(k\sigma)}$

$I_1 = I * G_{k\sigma}$

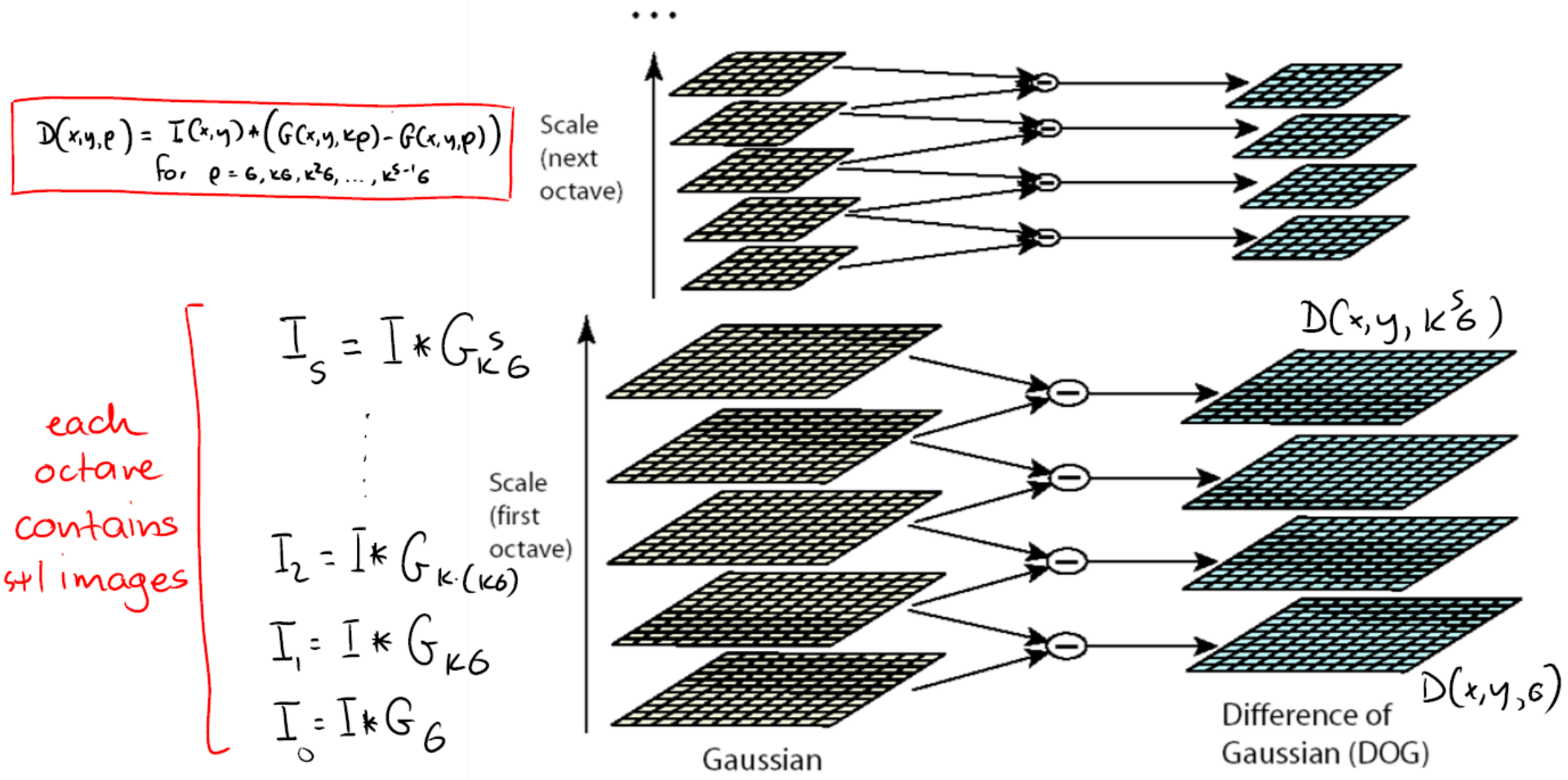
$I_0 = I * G_\sigma$



Step 1b: Compute Pyramid of DOG Images

Step 1b: Compute a pyramid of DOG-filtered images

$$D(x,y,\rho) = I(x,y) * \left(G(x,y,k\rho) - G(x,y,\rho)\right)$$
for $\rho = \sigma, k\sigma, k^2\sigma, \ldots, k^{s-1}\sigma$



each octave contains $s+1$ images

$I_s = I * G_{k^s\sigma}$

$\vdots$

$I_2 = I * G_{k\cdot(k\sigma)}$

$I_1 = I * G_{k\sigma}$

$I_0 = I * G_\sigma$

Reminder: Difference-of-Gaussian Filtering

$I * G_\rho$

Reminder: Difference-of-Gaussian Filtering

$I * G_{k\rho}$

Reminder: Difference-of-Gaussian Filtering

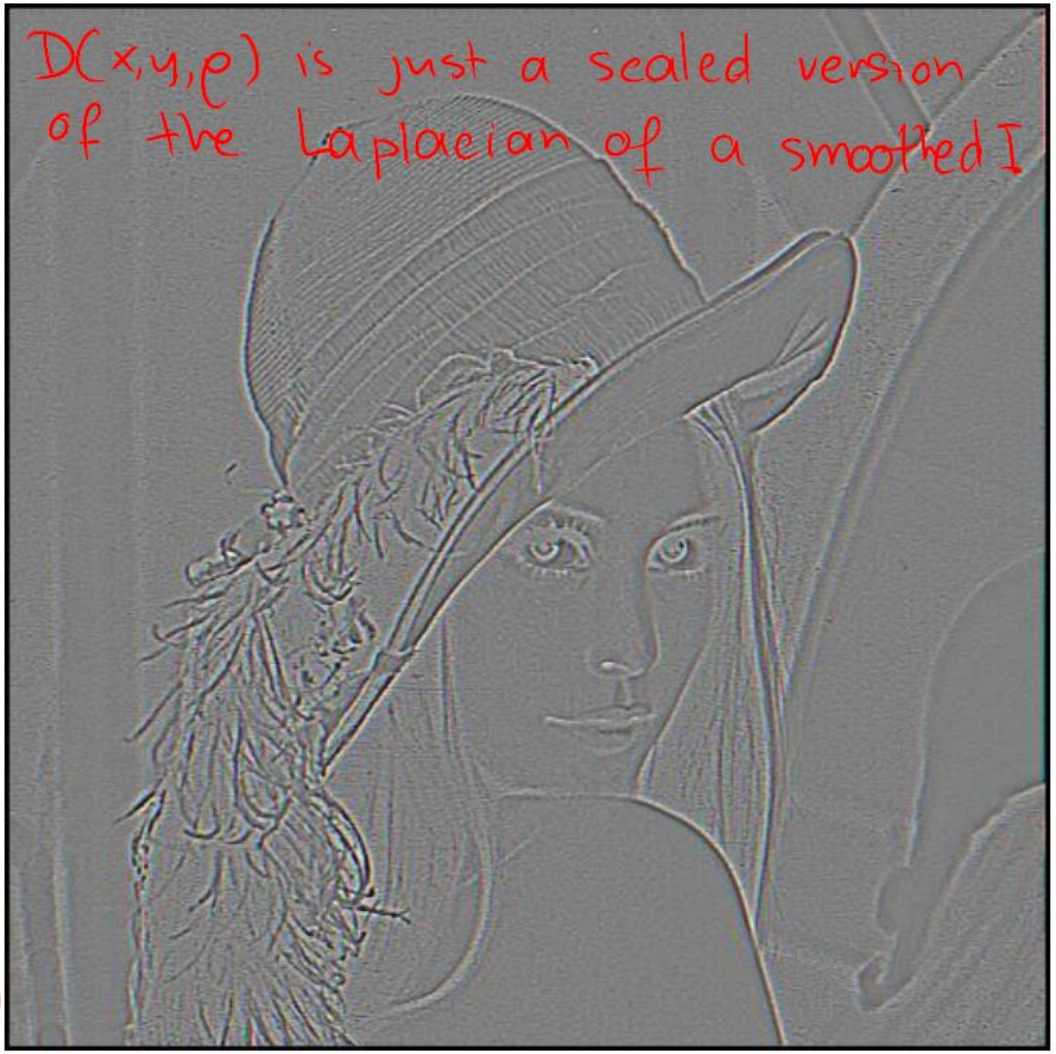
Difference of two Gaussian-smoothed versions of I:

$I * G_{k\rho} -$

$I * G_{\rho} =$

$I * (G_{k\rho} - G_{\rho})$

(just the difference between two Gaussian masks)



But we know that

$G_{k\rho} - G_{\rho} = k\rho(k\rho - \rho)\nabla^2 G_{\rho}$

Computing SIFT Keypoints: Basic Steps

Source image I

Gauss-pyramid

Step 1a

Build pyramid of Gauss-smoothed images

DOG pyramid

Step 1b

Buid DOG pyramid

DOG extrema

Step 1c

Locate extrema of DOG pyramid

(x_i, y_i, ρ_i)

Step 1d

Refine location of DOG extrema

$(x_i, y_i, \rho_i) \rightarrow (x_i', y_i', \rho_i')$

Location refinement

Step 1e

Prune set of extrema

Keypoints = { all remaining (x_i', y_i', ρ_i') }

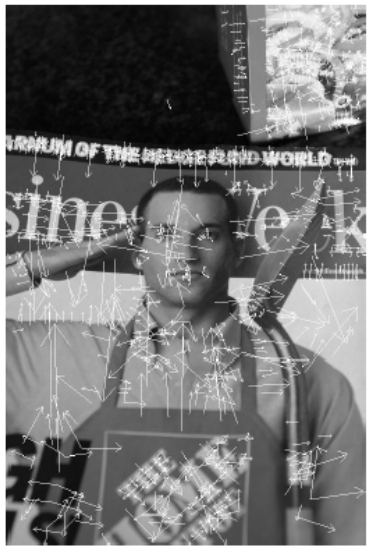
Extremum pruning

Step 1f

Assign orientation to extrema

$p_i = (x_i', y_i', \rho_i', \vartheta_i)$

Orientation assign



Step 1c: Detecting DOG Extrema

Step 1c (Extremum detection): Find all pixels that correspond to extrema of $D(x,y,\rho)$

- Extrema of the image Laplacian are readily distinguishable from their surroundings
- There are usually just a few in each pyramid (a few thousand)

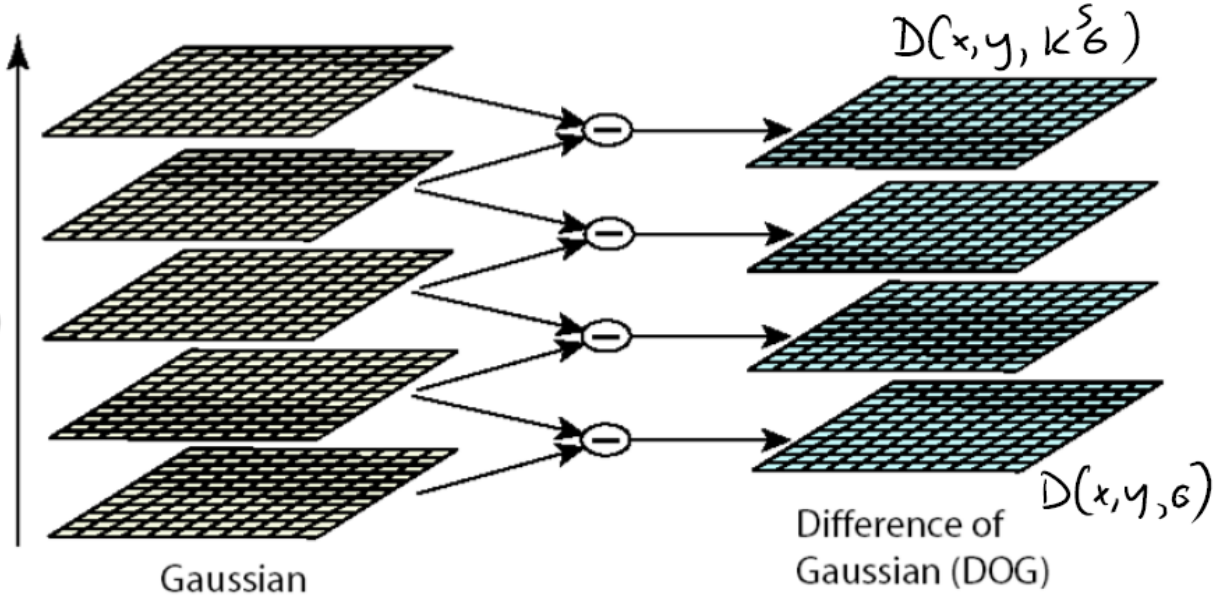
$I_s = I * G_{k^s\sigma}$

$\vdots$

$I_2 = I * G_{k\cdot(k\sigma)}$

$I_1 = I * G_{k\sigma}$

$I_0 = I * G_{\sigma}$

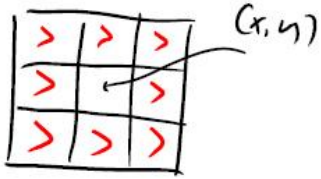


The Difference-Of-Gaussians (DOG) Filter

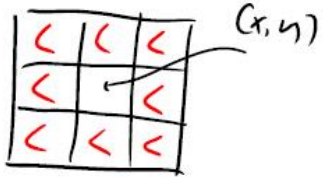
Finding local extrema in a single image

$D(x, y, \rho)$

- minimum at (x, y) if $D(x, y, \rho) <$ all neighbours



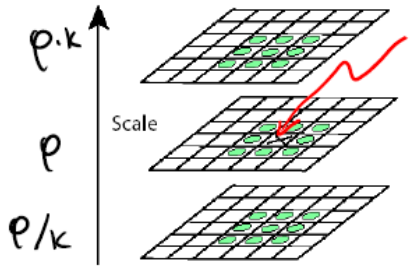
- maximum if $D(x, y, \rho) >$ all neighbours



Step 1c: Detecting DOG Extrema

Step 1c (Extremum detection): Find all pixels that correspond to extrema of $D(x, y, \rho)$

finding extrema in a "stack" of images



must also be < than all neighbors in adjacent scales

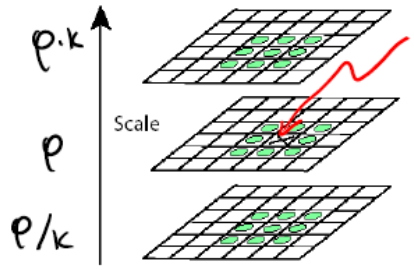
<u>or</u>

must also be > than all neighbors in adjacent scales

Step 1c: Detecting DOG Extrema

Step 1c (Extremum detection): Find all pixels that correspond to extrema of $D(x,y,\rho)$

finding extrema in a "stack" of images



must also be < than all neighbors in adjacent scales

<u>or</u>

must also be > than all neighbors in adjacent scales

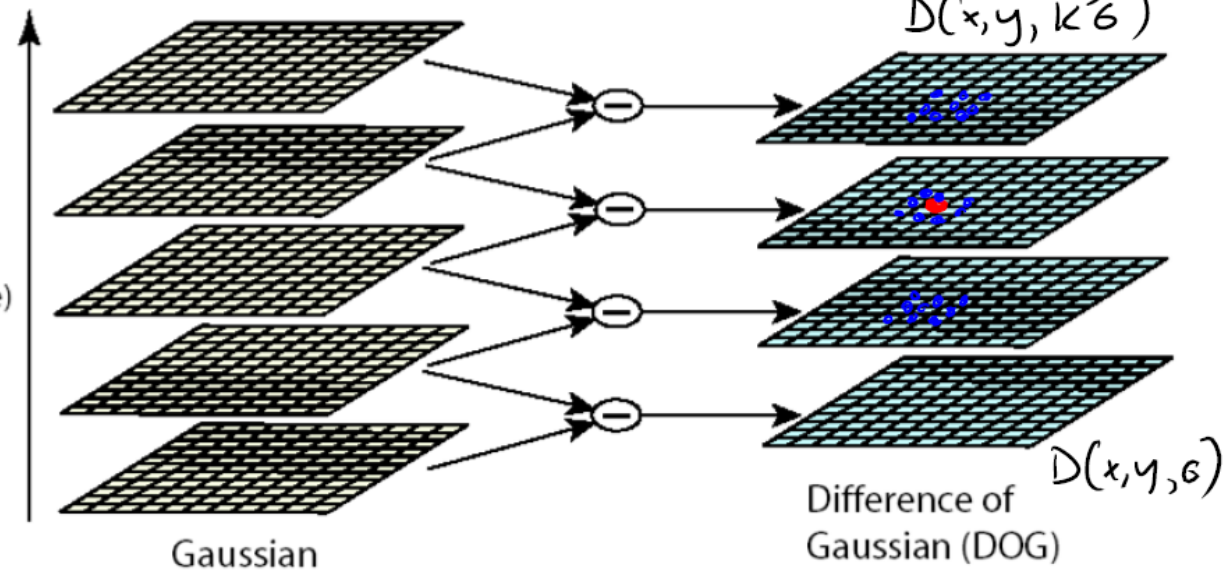
$I_s = I * G_{k^s\sigma}$

$\vdots$

$I_2 = I * G_{k\cdot(k\sigma)}$

$I_1 = I * G_{k\sigma}$

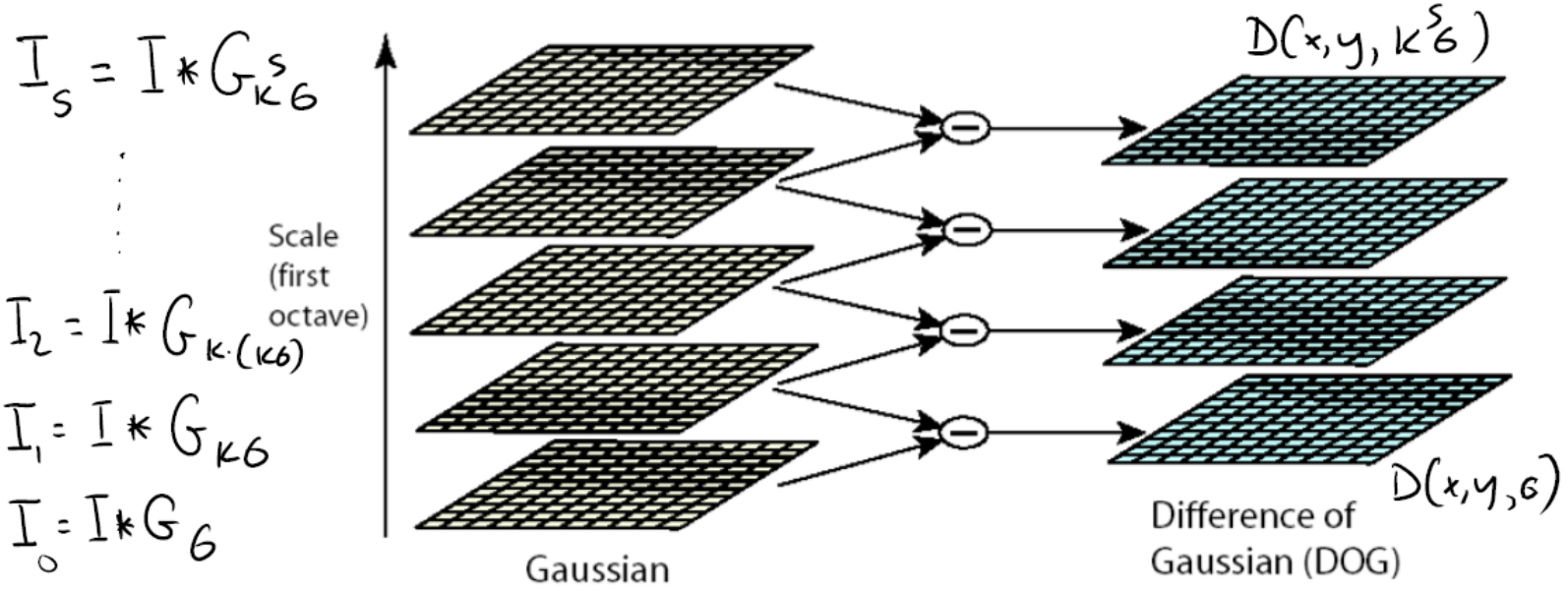
$I_0 = I * G_{\sigma}$



Step 1c: Detecting DOG Extrema: Algorithm

Step 1c (Extremum detection): Find all pixels that correspond to extrema of $D(x,y,\rho)$

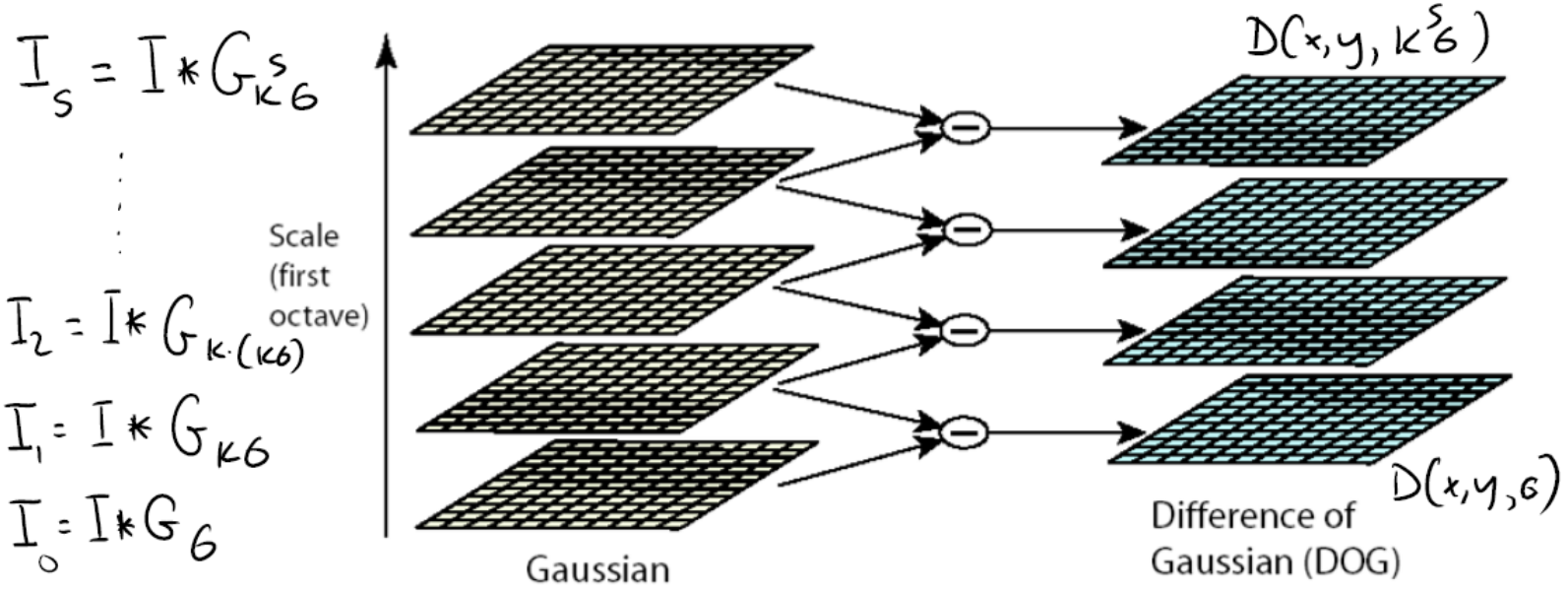
- for each (x,y,ρ), check whether $D(x,y,\rho)$ is greater than (or smaller than) all of its neighbours in current scale and adjacent scales above & below



Step 1c: SIFT Keypoints = DOG Extrema

Step 1c (Extremum detection): Find all pixels that correspond to extrema of $D(x,y,\rho)$

An extremum detected at $D(x,y,\rho)$ defines the keypoint (x,y,ρ)



Computing SIFT Keypoints: Basic Steps

Source image I

Gauss-pyramid

Step 1a

Build pyramid of Gauss-smoothed images

DOG pyramid

Step 1b

Buid DOG pyramid

DOG extrema

Step 1c

Locate extrema of DOG pyramid

(x_i, y_i, ρ_i)

Step 1d

Refine location of DOG extrema

$(x_i, y_i, \rho_i) \rightarrow (x_i', y_i', \rho_i')$

Location refinement

Step 1e

Prune set of extrema

Keypoints = { all remaining (x_i', y_i', ρ_i') }

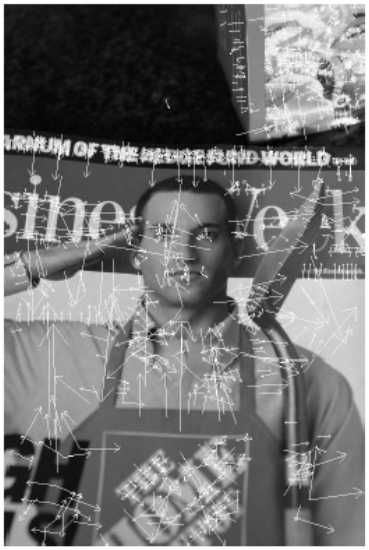
Extremum pruning

Step 1f

Assign orientation to extrema

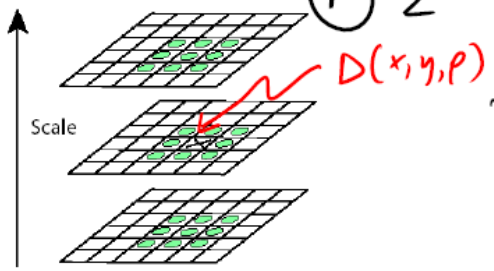
$p_i = (x_i', y_i', \rho_i', \vartheta_i)$

Orientation assign



Step 1d: Refining Location of Extrema

Step 1d (Extremum localization) Refine the location of detected extrema through a quadratic least-sq fit



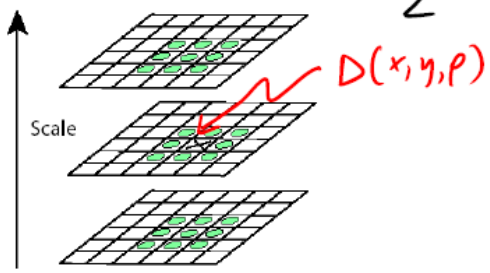
① 2nd order Taylor expansion of D at (x, y, ρ):

$$D(\Delta x, \Delta y, \Delta \rho) = D(x, y, \rho) + \frac{\partial D}{\partial x}\Delta x + \frac{\partial D}{\partial y}\Delta y + \frac{\partial D}{\partial \rho}\Delta \rho + \frac{1}{2}\begin{bmatrix}\Delta x & \Delta y & \Delta \rho\end{bmatrix}\begin{bmatrix}\frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial x \partial \rho} \\ \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial y \partial \rho} \\ \frac{\partial^2 D}{\partial x \partial \rho} & \frac{\partial^2 D}{\partial y \partial \rho} & \frac{\partial^2 D}{\partial \rho^2}\end{bmatrix}\begin{bmatrix}\Delta x \\ \Delta y \\ \Delta \rho\end{bmatrix}$$

* all derivatives approximated using finite differences, e.g.

$$\frac{\partial D}{\partial x} \simeq D(x+1, y, \rho) - D(x, y, \rho)$$

Step 1d: Refining Location of Extrema



2nd order Taylor expansion of D at (x,y,ρ):

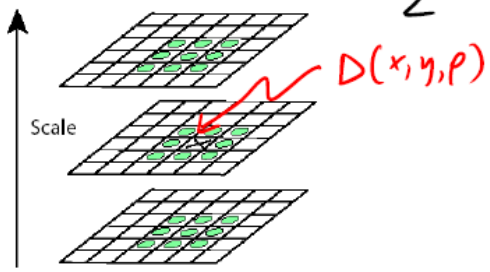
In vector notation

$$\vec{x} = \begin{bmatrix} x \\ y \\ \rho \end{bmatrix} \quad \Delta\vec{x} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \rho \end{bmatrix}$$

$$D(\Delta\vec{x}) = D(\vec{x}) + \left(\frac{\partial D}{\partial \vec{x}}\right)^T \cdot \Delta\vec{x} + \frac{1}{2}(\Delta\vec{x})^T \cdot \frac{\partial^2 D}{\partial \vec{x}^2} \cdot (\Delta\vec{x})$$

Step 1d: Refining Location of Extrema

(1)



2nd order Taylor expansion of D at (x,y,ρ):

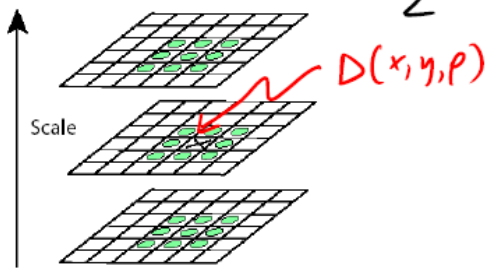
$$D(\Delta\vec{x}) = D(\vec{x}) + \left(\frac{\partial D}{\partial \vec{x}}\right)^T \cdot \Delta\vec{x} + \frac{1}{2}(\Delta\vec{x})^T \cdot \frac{\partial^2 D}{\partial \vec{x}^2} \cdot (\Delta\vec{x})$$

(2) Take derivatives with respect to $\Delta\vec{x}$

$$\frac{\partial D}{\partial(\Delta\vec{x})} = \left(\frac{\partial D}{\partial \vec{x}}\right)^T + \left(\frac{\partial^2 D}{\partial \vec{x}^2}\right)(\Delta\vec{x})$$

Step 1d: Refining Location of Extrema

(1)



2nd order Taylor expansion of D at (x, y, ρ):

$$D(\Delta\vec{x}) = D(\vec{x}) + \left(\frac{\partial D}{\partial \vec{x}}\right)^{T} \cdot \Delta\vec{x} + \frac{1}{2}(\Delta\vec{x})^{T} \cdot \frac{\partial^2 D}{\partial \vec{x}^2} \cdot (\Delta\vec{x})$$

(2) Take derivatives with respect to $\Delta\vec{x}$

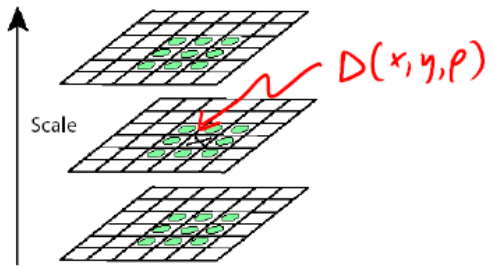
$$\frac{\partial D}{\partial(\Delta\vec{x})} = \left(\frac{\partial D}{\partial \vec{x}}\right) + \left(\frac{\partial^2 D}{\partial \vec{x}^2}\right)(\Delta\vec{x})$$

(3) Extremum $\Leftrightarrow$ derivative is zero $\Rightarrow$ solve for $\frac{\partial D}{\partial \Delta\vec{x}} = 0$

$$(\Delta\vec{x}) = -\left(\frac{\partial^2 D}{\partial \vec{x}^2}\right)^{-1} \cdot \left(\frac{\partial D}{\partial \vec{x}}\right)$$

Step 1d: Refining Location of Extrema

Step 1d (Extremum localization) Refine the location of detected extrema through a quadratic least-sq fit



Location refinement:

extremum (x, y, ρ)

moved to

$(x + \Delta x, y + \Delta y, \rho + \Delta \rho)$

$$\vec{\Delta x} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \rho \end{bmatrix}$$

$$\frac{\partial D}{\partial \vec{x}} = \begin{bmatrix} \frac{\partial D}{\partial x} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial \rho} \end{bmatrix}$$

where

$$(\vec{\Delta x}) = \left(\frac{\partial^2 D}{\partial \vec{x}^2}\right)^{-1} \cdot \left(\frac{\partial D}{\partial \vec{x}}\right) \qquad \frac{\partial D^2}{\partial \vec{x}^2} = \begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial D^2}{\partial x \partial \rho} \\ \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial D^2}{\partial y \partial \rho} \\ \frac{\partial^2 D}{\partial x \partial \rho} & \frac{\partial^2 D}{\partial y \partial \rho} & \frac{\partial D^2}{\partial \rho^2} \end{bmatrix}$$

Computing SIFT Keypoints: Basic Steps

Source image I

Gauss-pyramid

Step 1a

Build pyramid of Gauss-smoothed images

DOG pyramid

Step 1b

Buid DOG pyramid

DOG extrema

Step 1c

Locate extrema of DOG pyramid

(x_i, y_i, ρ_i)

Step 1d

Refine location of DOG extrema

$(x_i, y_i, \rho_i) \rightarrow (x_i', y_i', \rho_i')$

Location refinement

Step 1e

Prune set of extrema

Keypoints = { all remaining (x_i', y_i', ρ_i') }

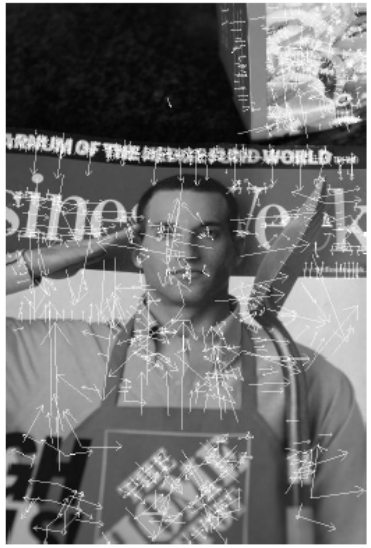
Extremum pruning

Step 1f

Assign orientation to eatrema

$p_i = (x_i', y_i', \rho_i', \vartheta_i)$

Orientation assign



Step 1e: Pruning "Insignificant" Extrema

Step 1e (Extremum pruning): Prune all extrema that are weak or that correspond to edges

Condition for detecting a "strong" extremum (x_i', y_i', σ_i'):

$$|D(x_i', y_i', \sigma_i')| = \text{large}$$

in practice > 0.03

assumes image I has pixel intensities in the range $[0,1]$

$I_s = I * G_{k^s\sigma}$

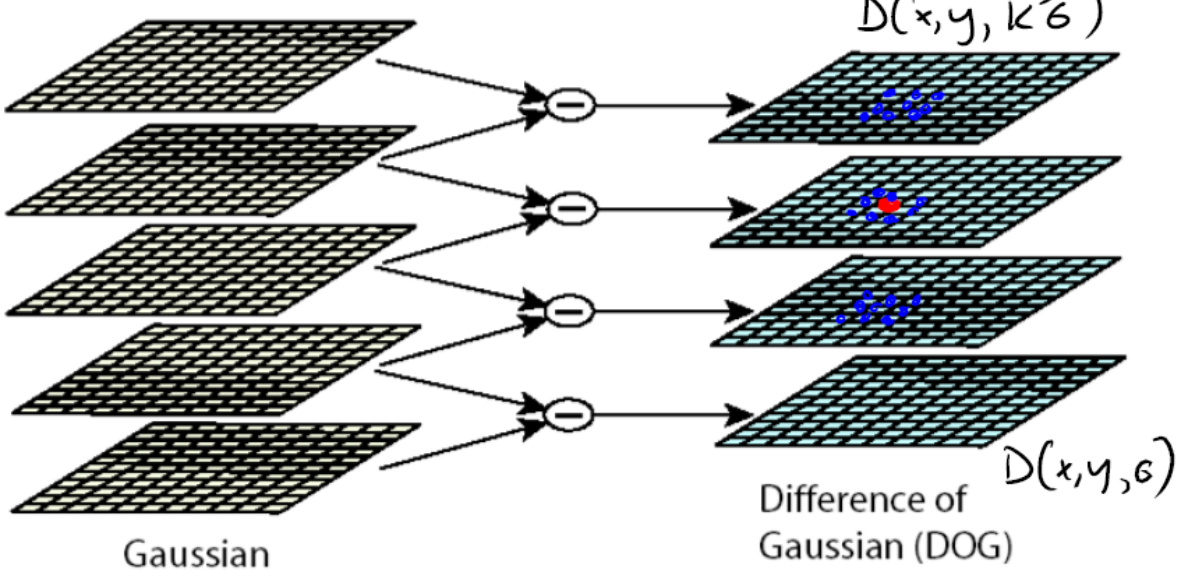
$\vdots$

$I_2 = I * G_{k^2(k\sigma)}$

$I_1 = I * G_{k\sigma}$

$I_0 = I * G_\sigma$

Scale (first octave)

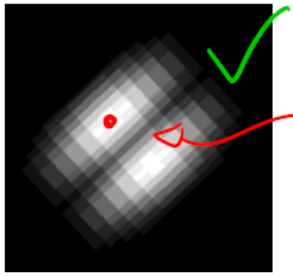


Step 1e: Pruning "Insignificant" Extrema

Step 1e (Extremum pruning): Prune all extrema that are weak or that correspond to edges

corner-like extremum

position is well-constrained

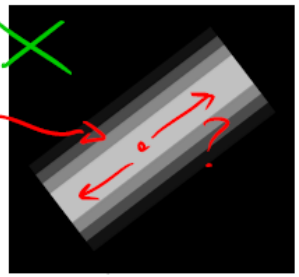


keep

$D(x_i', y_i', \rho_i')$

edge-like extremum

Prune



Position along edge is not constrained

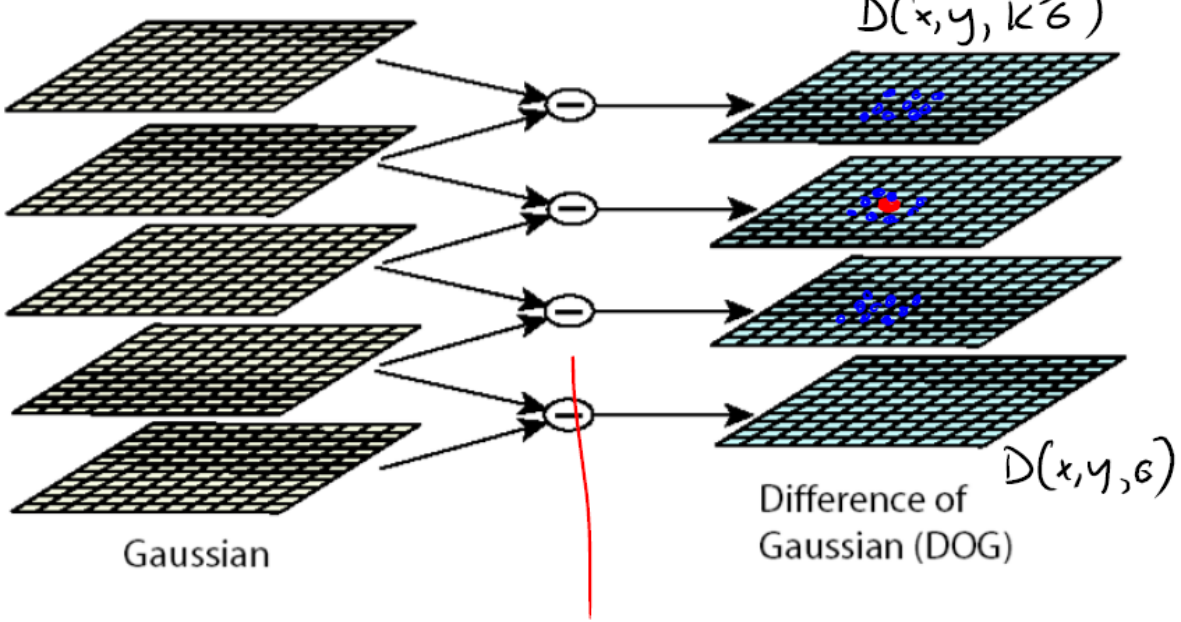
$I_s = I * G_{k^s\sigma}$

$I_2 = I * G_{k^2\sigma}$

$I_1 = I * G_{k\sigma}$

$I_0 = I * G_{\sigma}$

Scale (first octave)



Step 1e: Pruning "Insignificant" Extrema

Step 1e (Extremum pruning): Prune all extrema that are weak or that correspond to edges

$$H = \begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} \\ \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial y^2} \end{bmatrix}$$

- Compute hessian H of $D(x, y, \sigma_i')$ at $(x, y) = (x_i', y_i')$
- prune if $\frac{Tr^2(H)}{Det(H)} > \left(\frac{11}{10}\right)^2$

$I_s = I * G_{k^s\sigma}$

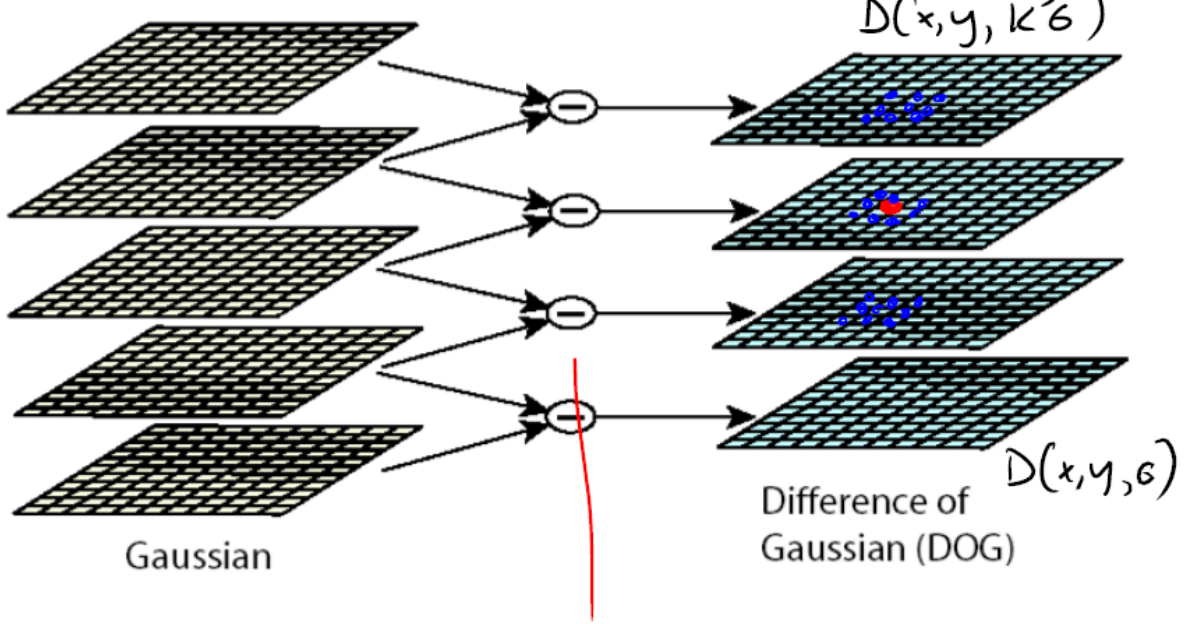
$\vdots$

$I_2 = I * G_{k^2\sigma}$

$I_1 = I * G_{k\sigma}$

$I_0 = I * G_\sigma$

Scale (first octave)



Computing SIFT Keypoints: Basic Steps

Source image I

Gauss-pyramid

Step 1a

Build pyramid of Gauss-smoothed images

DOG pyramid

Step 1b

Buid DOG pyramid

DOG extrema

Step 1c

Locate extrema of DOG pyramid

(x_i, y_i, ρ_i)

Step 1d

Refine location of DOG extrema

$(x_i, y_i, \rho_i) \rightarrow (x_i', y_i', \rho_i')$

Location refinement

Step 1e

Prune set of extrema

Keypoints = { all remaining (x_i', y_i', ρ_i') }

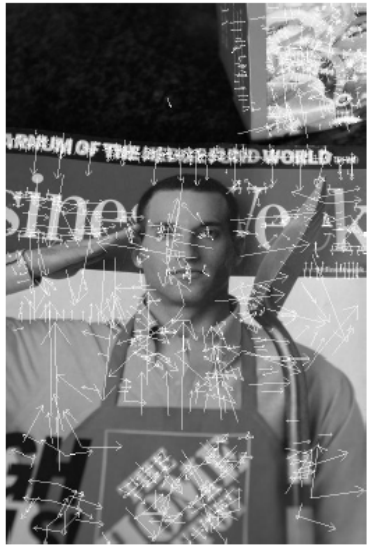
Extremum pruning

Step 1f

Assign orientation to extrema

$p_i = (x_i', y_i', \rho_i', \vartheta_i)$

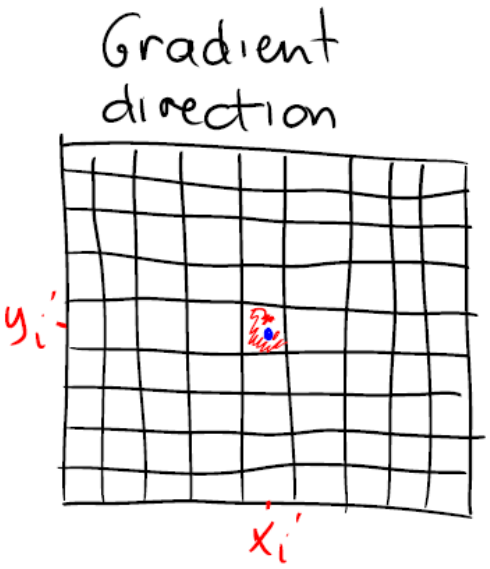
Orientation assign



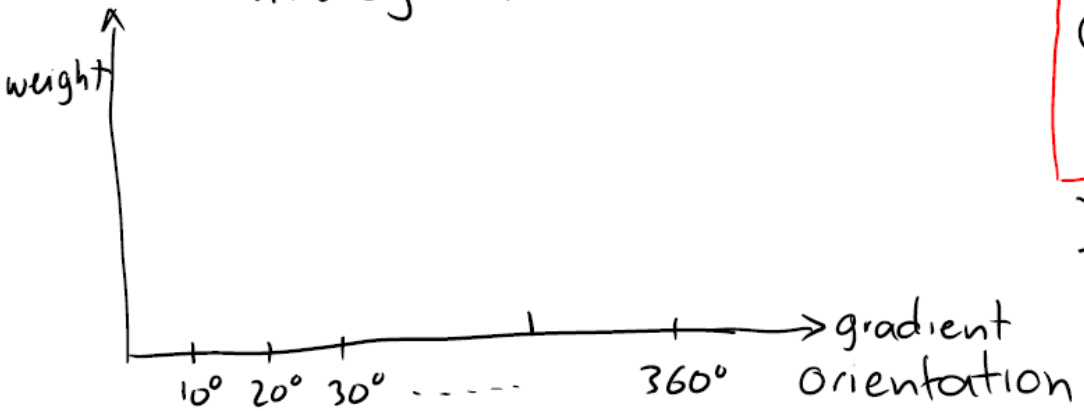
Step 1f: Keypoint Orientation Assignment

Assigning an orientation θ_i to keypoint (x_i', y_i', ρ_i'):

$I * G_{\rho_i'}$



Histogram



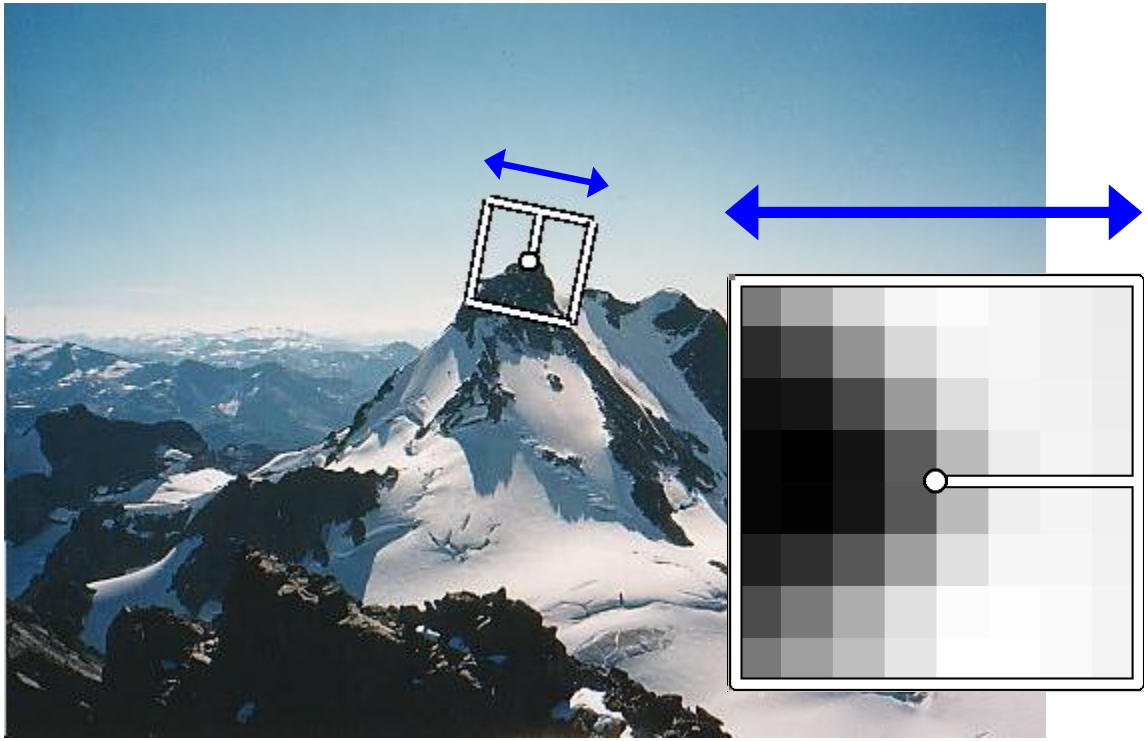
A. Compute smoothed image $I * G_{\rho_i'}$

B. Compute gradient magnitude & orientation in neighborhood of (x_i', y_i') in $I * G_{\rho_i'}$

C. Compute histogram of orientations

D. Assigned orientation θ_i = highest peak in hist

Making descriptor rotation invariant



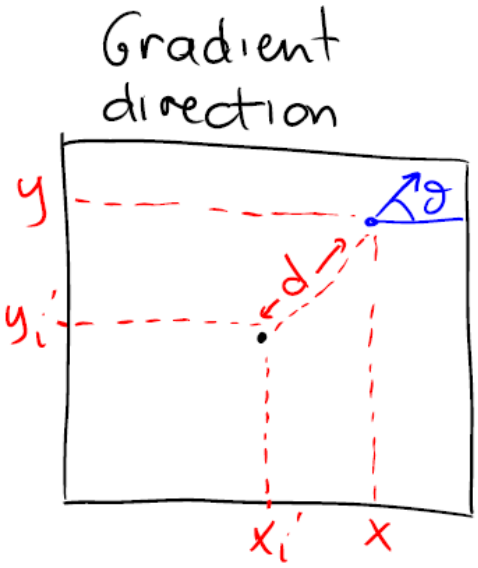
- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation.

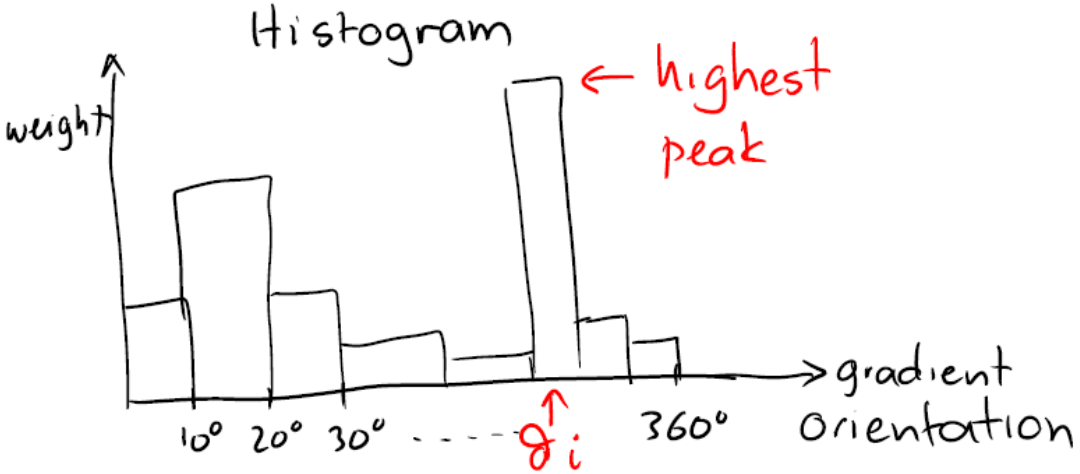
Image from Matthew Brown

Step 1f: Keypoint Orientation Assignment

Computing Histogram of Orientations

$I * G_{p_i'}$





- Orientations divided into 36 bins (one every 10 degrees)
- Pixel (x,y) contributes to the bin corresponding to the gradient orientation θ at (x,y)
- Contribution to the bin is equal to

$$|\nabla I(x,y)| \cdot G_{1.5p_i'}(d)$$

- Total bin weight = sum of contributions from all pixels

Computing SIFT Keypoints: Basic Steps

Source image I

Gauss-pyramid

Step 1a

Build pyramid of Gauss-smoothed images

DOG pyramid

Step 1b

Buid DOG pyramid

DOG extrema

Step 1c

Locate extrema of DOG pyramid

(x_i, y_i, ρ_i)

Step 1d

Refine location of DOG extrema

$(x_i, y_i, \rho_i) \rightarrow (x_i', y_i', \rho_i')$

Location refinement

Step 1e

Prune set of extrema

Keypoints = { all remaining (x_i', y_i', ρ_i') }

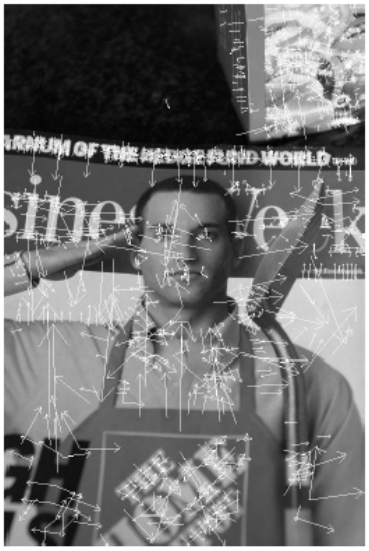
Extremum pruning

Step 1f

Assign orientation to extrema

$P_i = (x_i', y_i', \rho_i', \theta_i)$

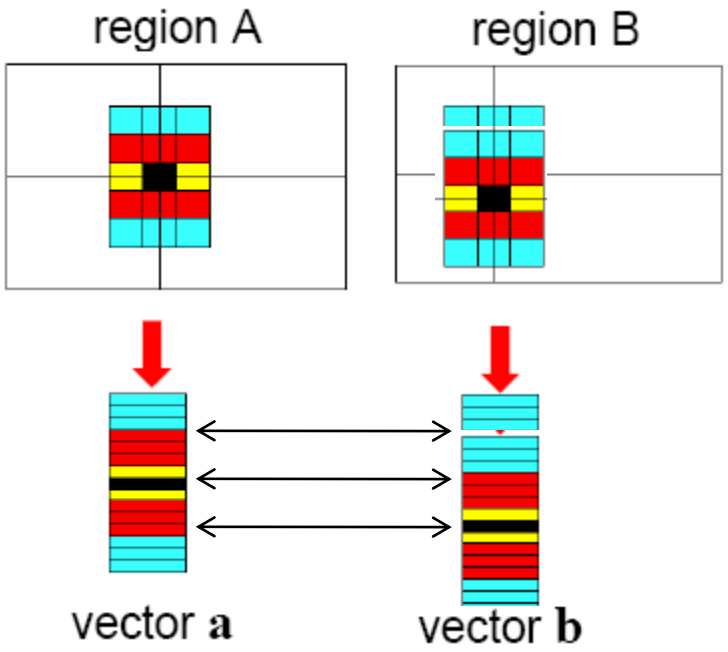
Orientation assign



Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

- introduction to the image matching problem
- image matching using SIFT features
- multi-scale interest-point detection
- the SIFT feature detector
- **the SIFT descriptor**

Raw patches as local descriptors



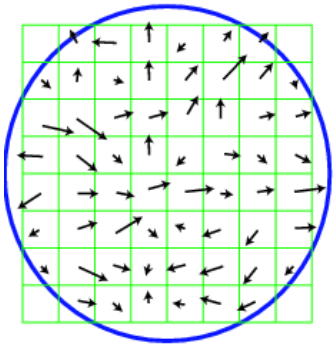
The simplest way to describe the neighborhood around an interest point is to write down the list of intensities to form a feature vector.

But this is very sensitive to even small shifts, rotations.

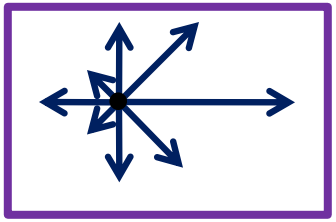
Figure: Andrew Zisserman

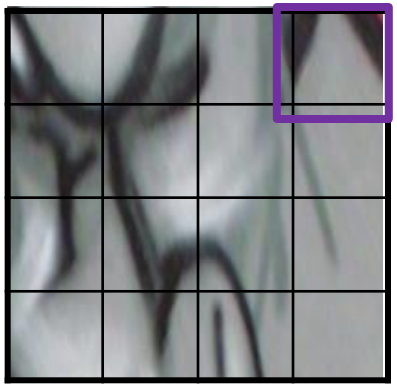
Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

- Use histograms to bin pixels within sub-patches according to their orientation.

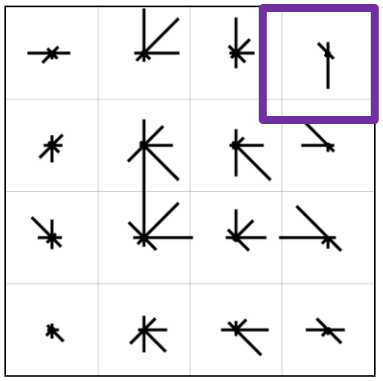


gradients





subdivided local patch



histogram per grid cell

Final descriptor = concatenation of all histograms, normalized

The SIFT Keypoints

Keypoint: A location (x,y) in the source image, with an associated orientation & scale, that is "visually distinct" from its surroundings

Input:

Image I

Output:

A set of k SIFT keypoints

$$p_i = (x_i, y_i, \sigma_i, \theta_i)$$

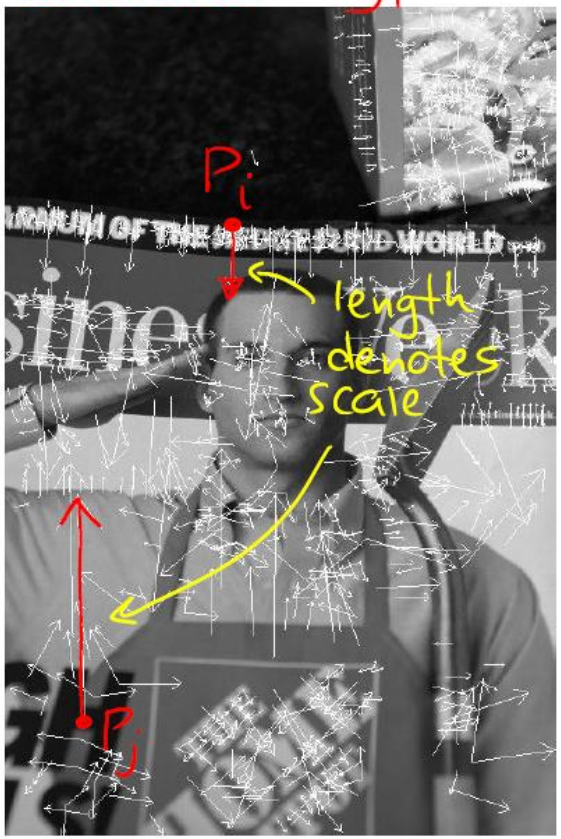
location

scale

orientation

"Source" Image I

Detected keypoints



The SIFT Feature Vectors

Feature vector (of a keypoint p_i): A vector of fixed length that represents the image patch centered at p_i

Input:

Image I

Output:

A set of k keypoints

$$p_i = (x_i, y_i, \sigma_i, \theta_i)$$

location, scale, orientation

A set of k feature vectors

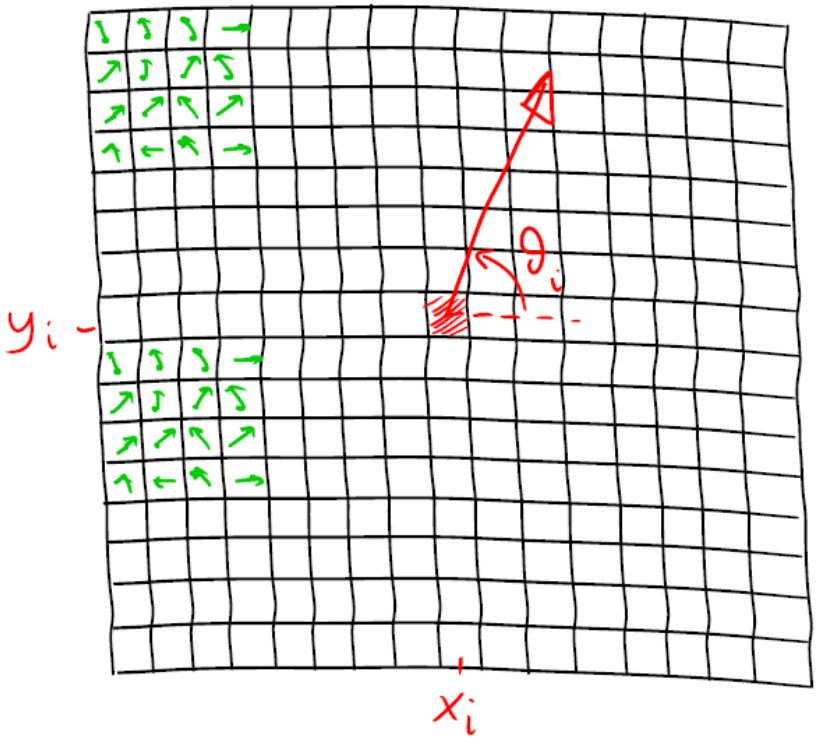
$$f_i = [\quad \cdots \quad]$$

describes the patch centered at p_i

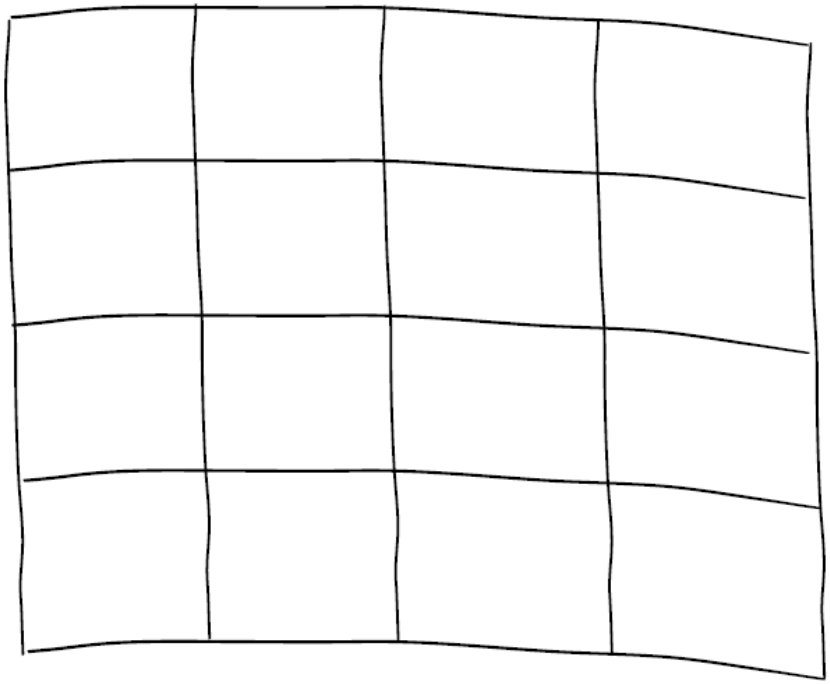
Detected keypoints

Building the SIFT Descriptor

Image patch centered at (x_i, y_i)



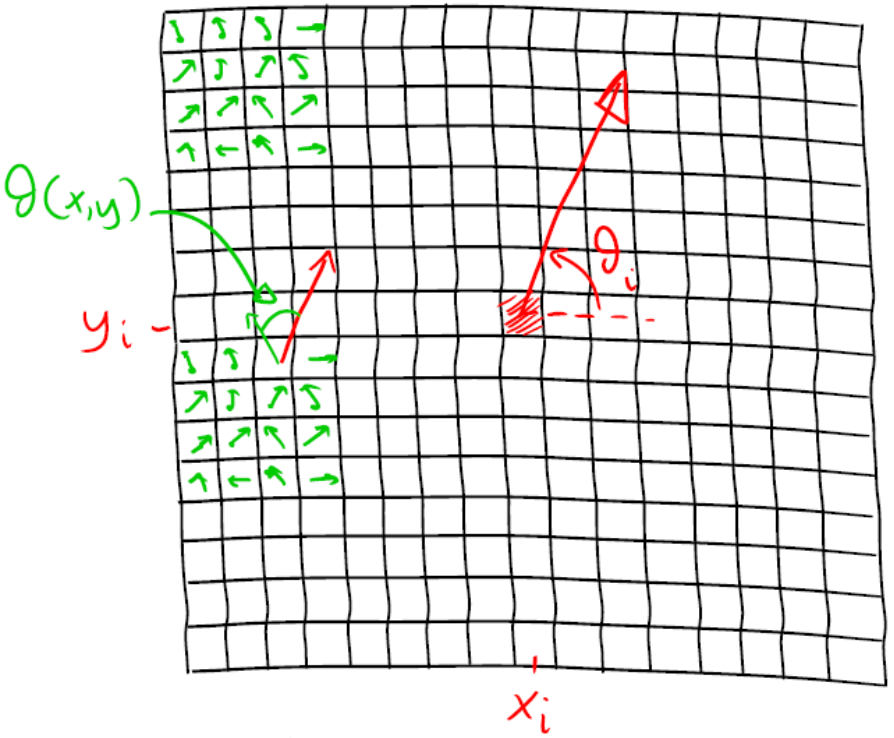
SIFT Descriptor



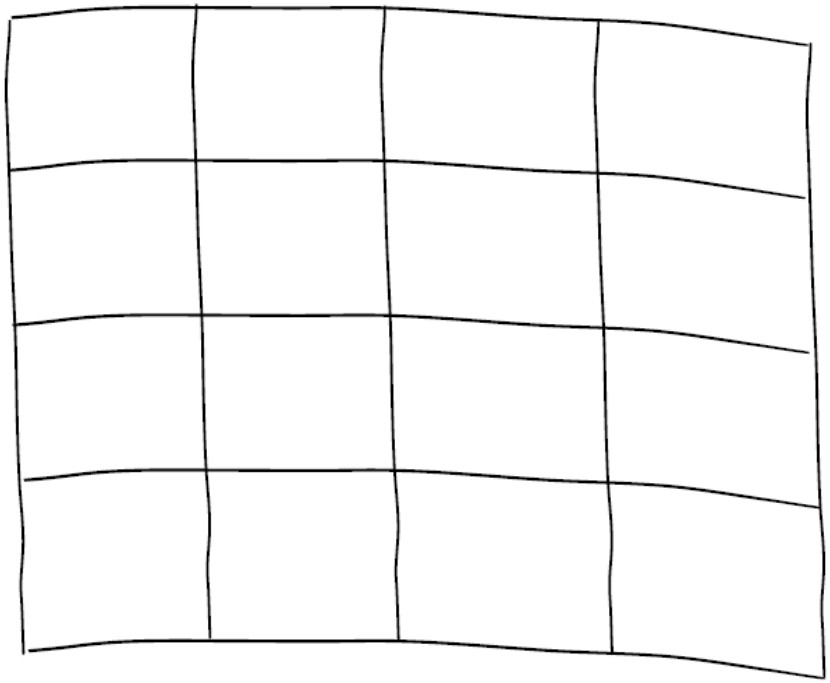
1. Compute gradients in 16x16 pixel patch of image $I * G_{6_i}$ ← Gaussian-smoothed image at scale of the keypoint centered at (x_i, y_i)

Building the SIFT Descriptor

Image patch centered at (x_i, y_i)



SIFT Descriptor

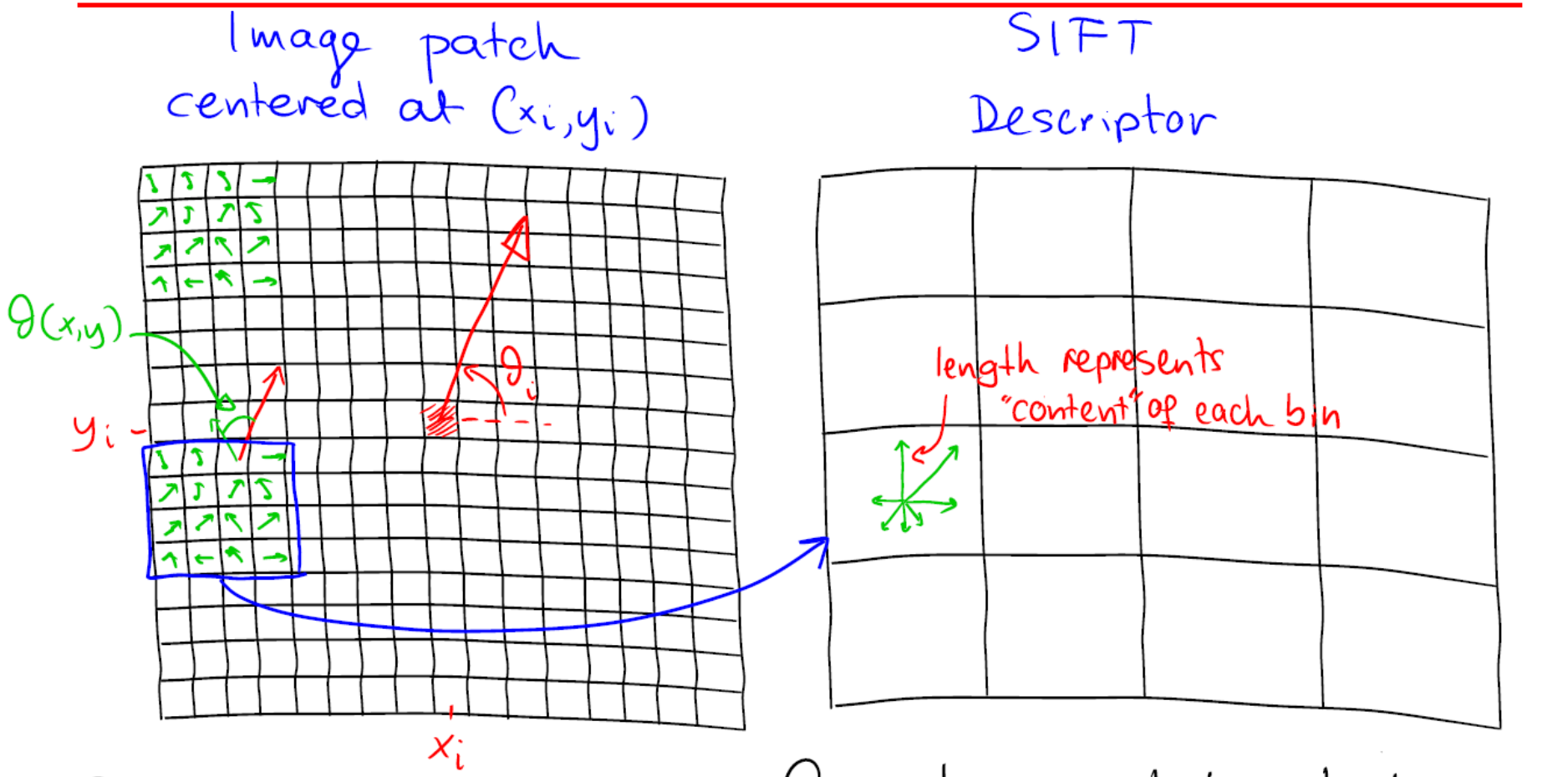


1. Compute gradients in 16×16 pixel patch of image $I * G_{\sigma_i}$ centered at (x_i, y_i)

2. Compute gradient orientation relative to keypoint orientation

$$\theta(x,y) = \tan^{-1}\left[\frac{\partial(I*G_{\sigma_i})}{\partial y} \Big/ \frac{\partial(I*G_{\sigma_i})}{\partial x}\right] - \theta_i$$

Building the SIFT Descriptor



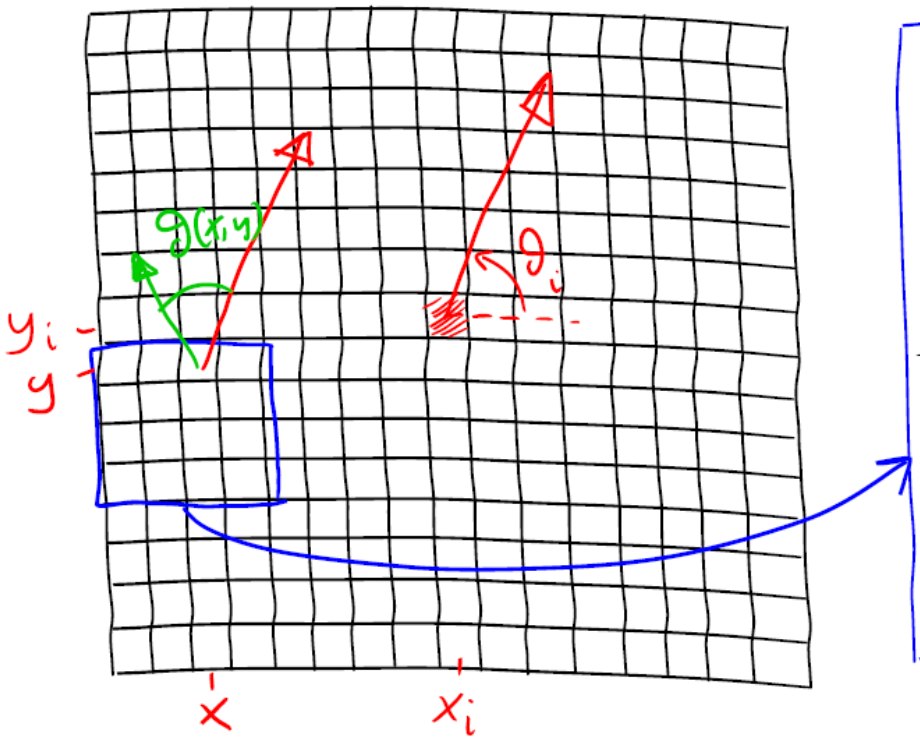
1. Compute gradients
2. Compute relative gradient orientations

3. Compute orientation histogram of each 4x4 pixel block

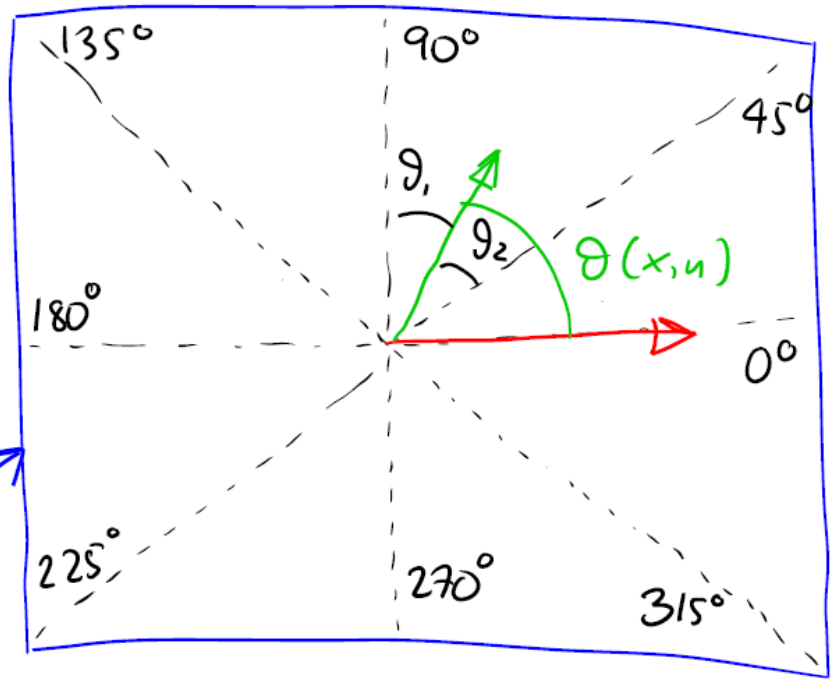
   histogram contains 8 bins, each covering 45°

The 4x4 Orientation Histogram

Image patch centered at (x_i, y_i)



The Orientation Histogram of a 4x4 pixel block



- Weight of (x,y):

$$w = G_{\frac{G_i}{2}}\left(\|(x - x_i, y - y_i)\|\right)$$

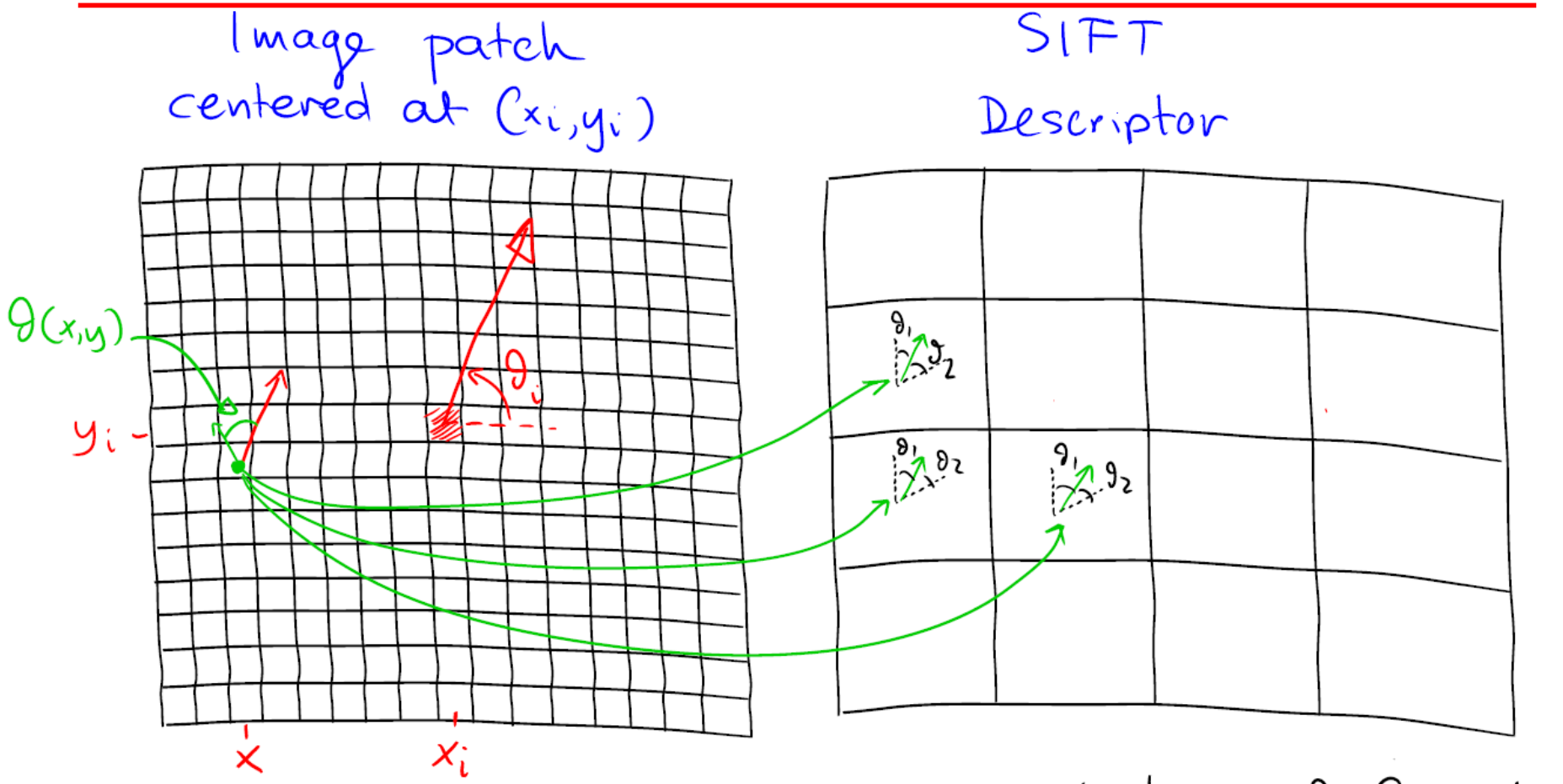
⇒ pixels closer to keypoint center have higher weight

- Total contribution of (x,y) to the orientation histograms:

$$c(x,y) = w \cdot \|\nabla I * G_{G_i}(x,y)\|$$

gradient magnitude of smoothed image

Building the SIFT Descriptor



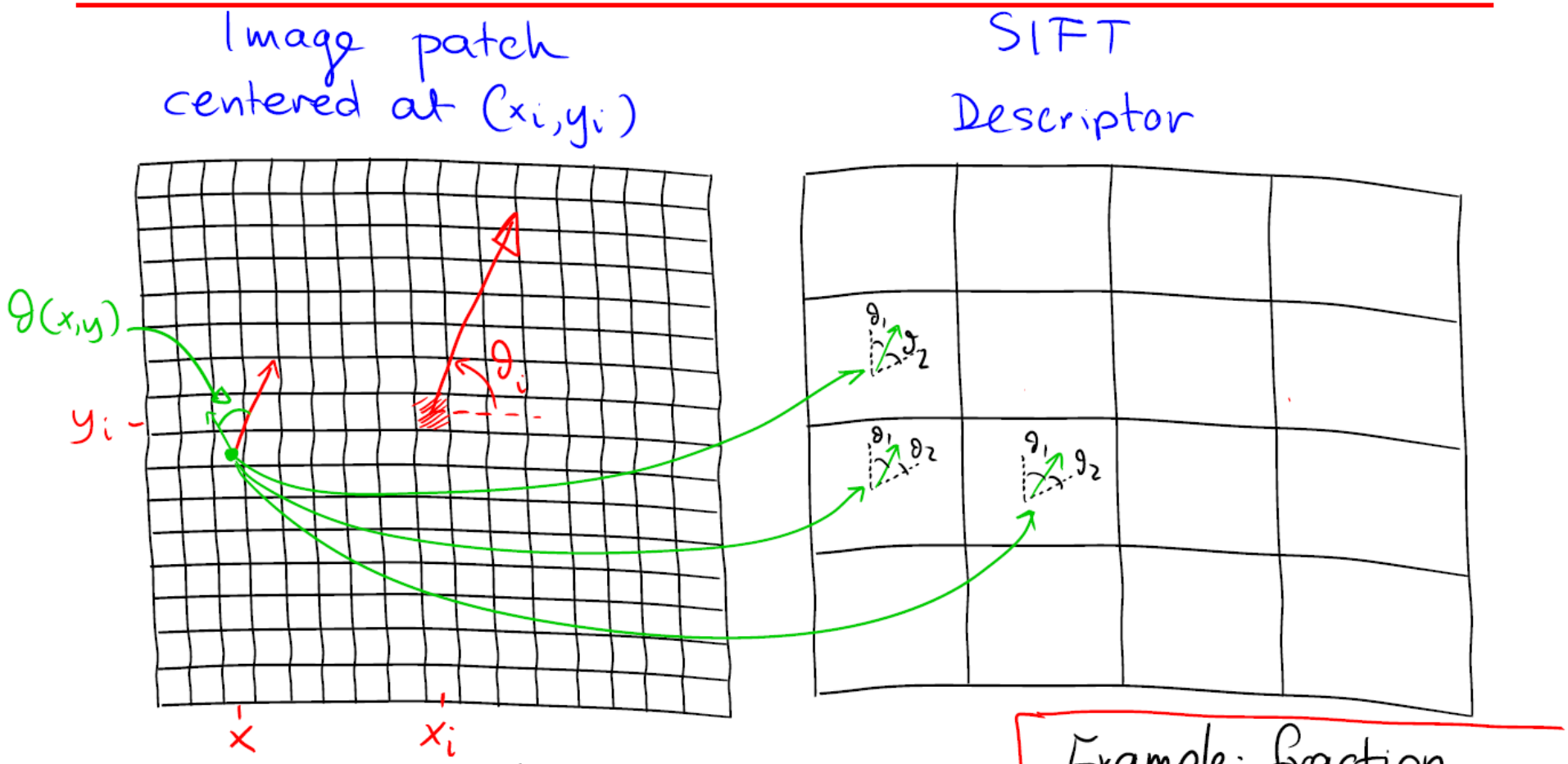
- Contribution spread across 2 closest orientations & 3 closest histograms

⇒ no abrupt changes in histogram if keypoint center displaced by 3-4 pixels

- Total contribution of (x,y) to the orientation histograms:

$$c(x,y) = w \cdot \| \nabla I * G_{\sigma_i}(x,y) \|$$

Building the SIFT Descriptor



- Contribution spread linearly across 2 closest orientations & 3 closest histograms

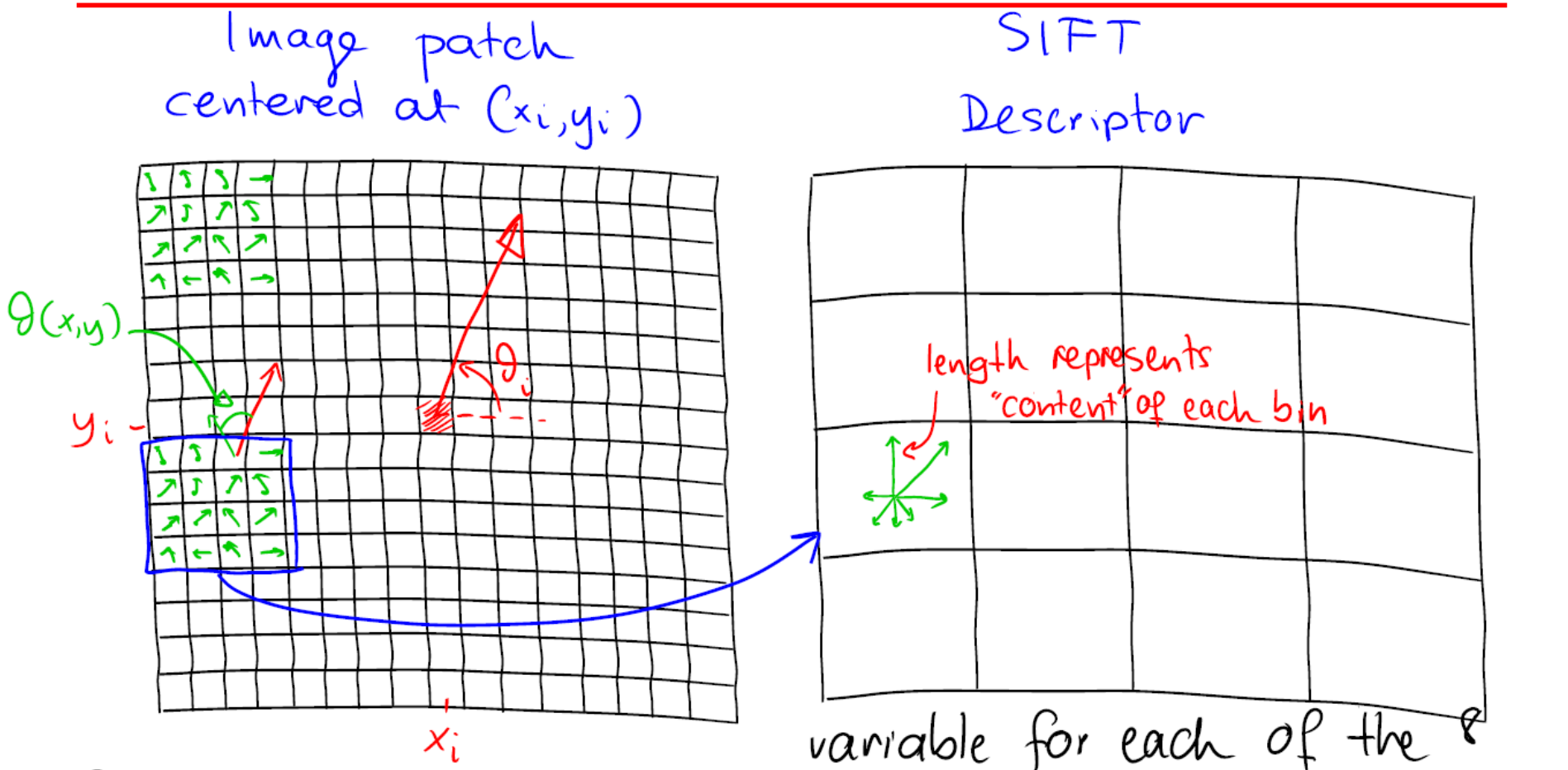
⇒ no abrupt changes in histogram if keypoint center displaced by 3-4 pixels

Example: fraction allocated to

orientation ①: $\frac{\theta_1}{\theta_1+\theta_2}$

orientation ②: $\frac{\theta_2}{\theta_1+\theta_2}$

Building the SIFT Descriptor: Complete Algorithm



1. Compute gradients
2. Compute relative gradient orientations
3. Define an "accumulator" variable for each of the 8 orientations in each of the 16 histograms (128 total)
4. For each pixel, calculate the pixel's contribution to each accumulator variable

Converting SIFT Descriptors to 128-dim Vectors

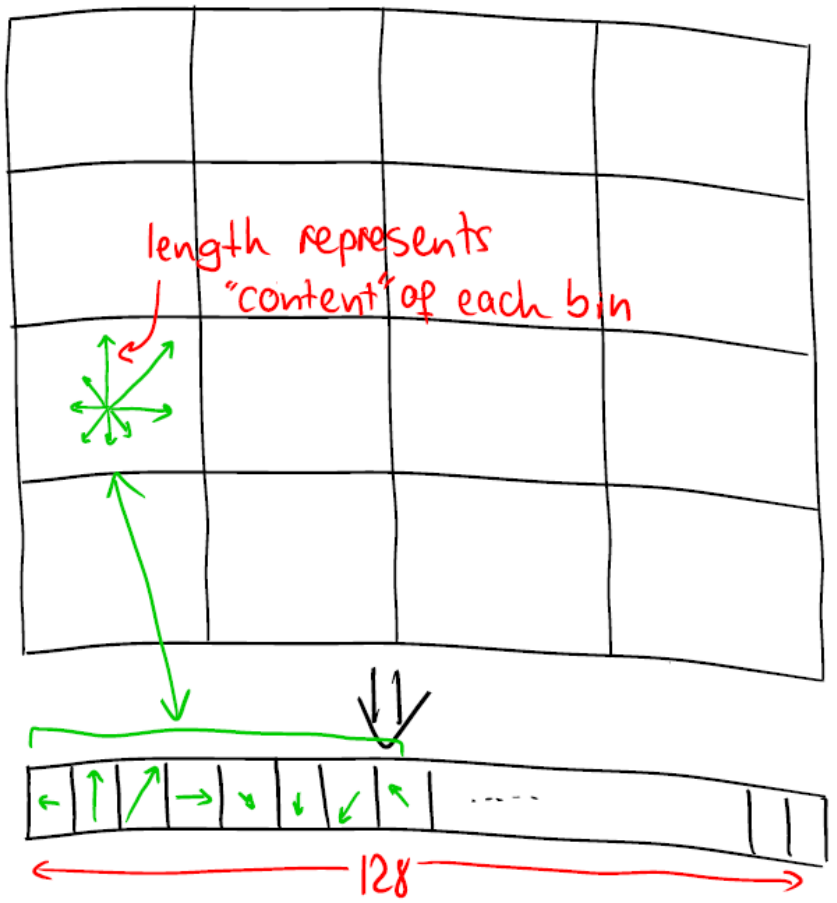
Post processing:

① Normalize f_i:

$$f_i \rightarrow \frac{f_i}{\|f_i\|}$$

⇒ gives invariance to linear lighting variations across images, i.e. when matching image I and image $a \cdot I + b$ (because f_i will be the same in both images)

SIFT Descriptor



② Clamp f_i

Clamp all elements of f_i at 0.2

⇒ gives less weight to very large gradient magnitudes

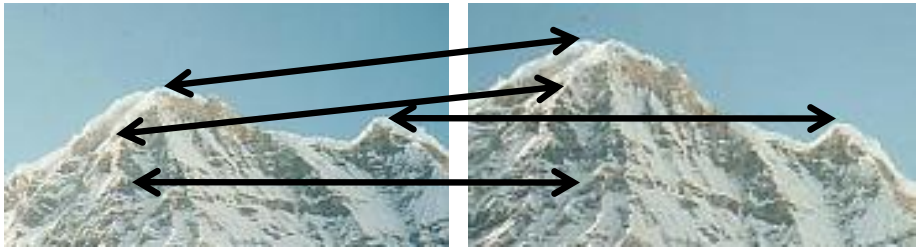
③ Re-normalize

Local features: main components

1) Detection: Identify the interest points

2) Description: Extract vector feature descriptor surrounding each interest point.

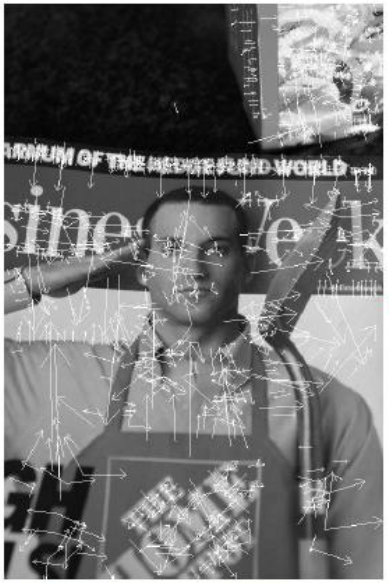
3) Matching: Determine correspondence between descriptors in two views



Slide credit: Kristen Grauman

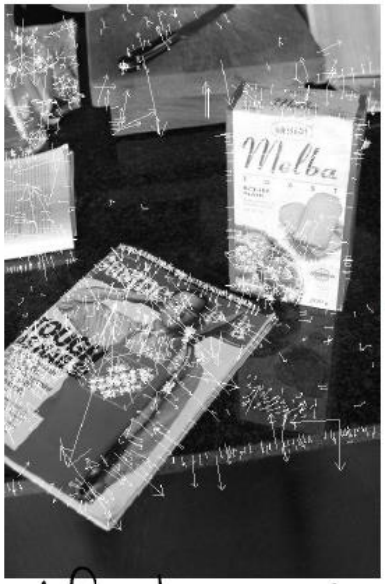
Matching 2 Images Using SIFT Features

Image I



f_1
f_2
f_3
f_{k-1}
f_k

Image I'



f_1'
f_2'
f_3'
f_{k-1}'
f_k'

① Identify keypoints

② Build feature vectors

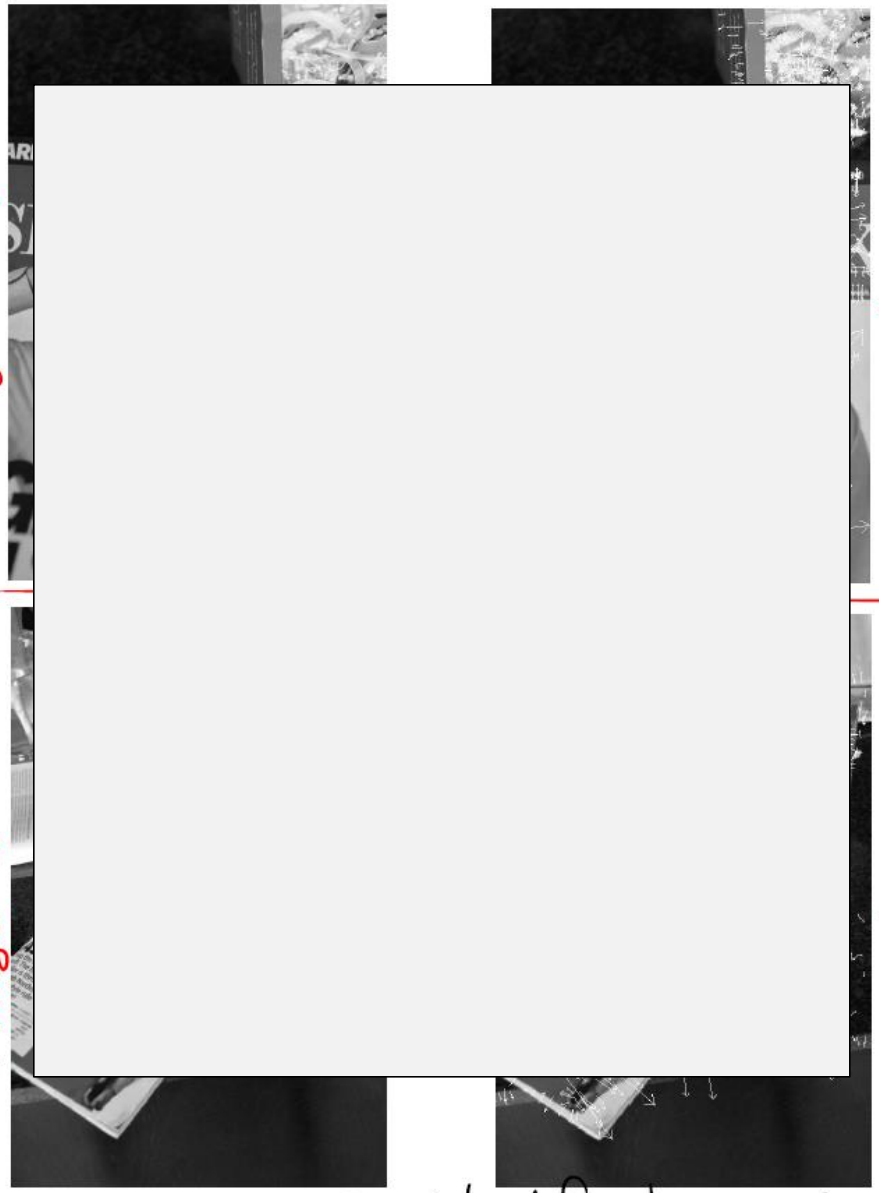
③ Match feature vectors in the two sets

$\{f_1, f_2, \ldots, f_k\}$

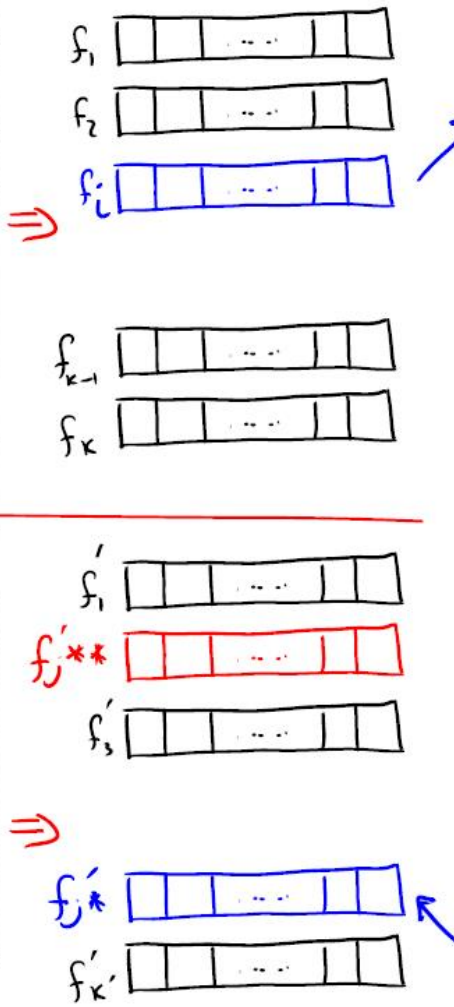
and

$\{f_1', f_2', \ldots, f_k'\}$

SIFT Feature Matching Algorithm



① Identify keypoints



② Build feature vectors

③ Match f_i

a. Compute $\|f_i - f_j'\|$ for all j

b. Compute fraction

$$\phi = \frac{\|f_i - f_j'^*\|}{\|f_i - f_j'^{**}\|}$$

where $f_j'^*$ is closest descriptor in I' and $f_j'^{**}$ is 2nd-closest

c. Match f_i to $f_j'^*$ if $\phi < 0.8$

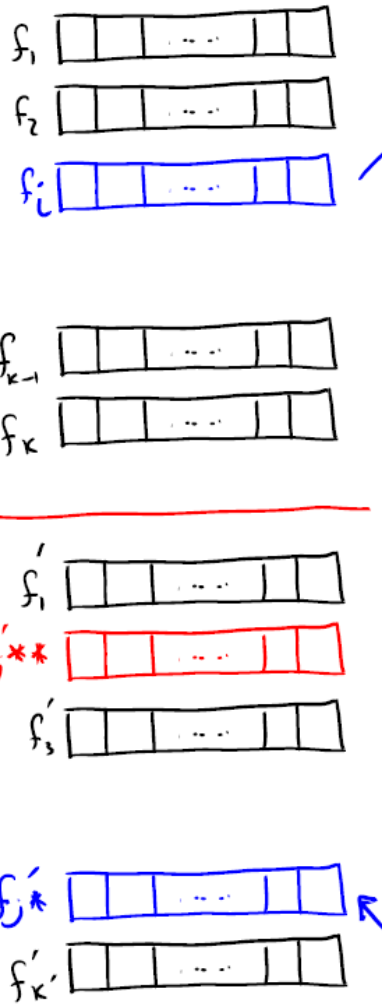
Matching 2 Images Using SIFT Features

Image I

Image I'

Intuition for matching algorithm:

match established only if it is deemed reliable, i.e. if there is only <u>one</u> very similar feature in image I'



① Identify keypoints

② Build feature vectors

③ Match f_i

a. Compute $\|f_i - f_j'\|$ for all j

b. Compute fraction

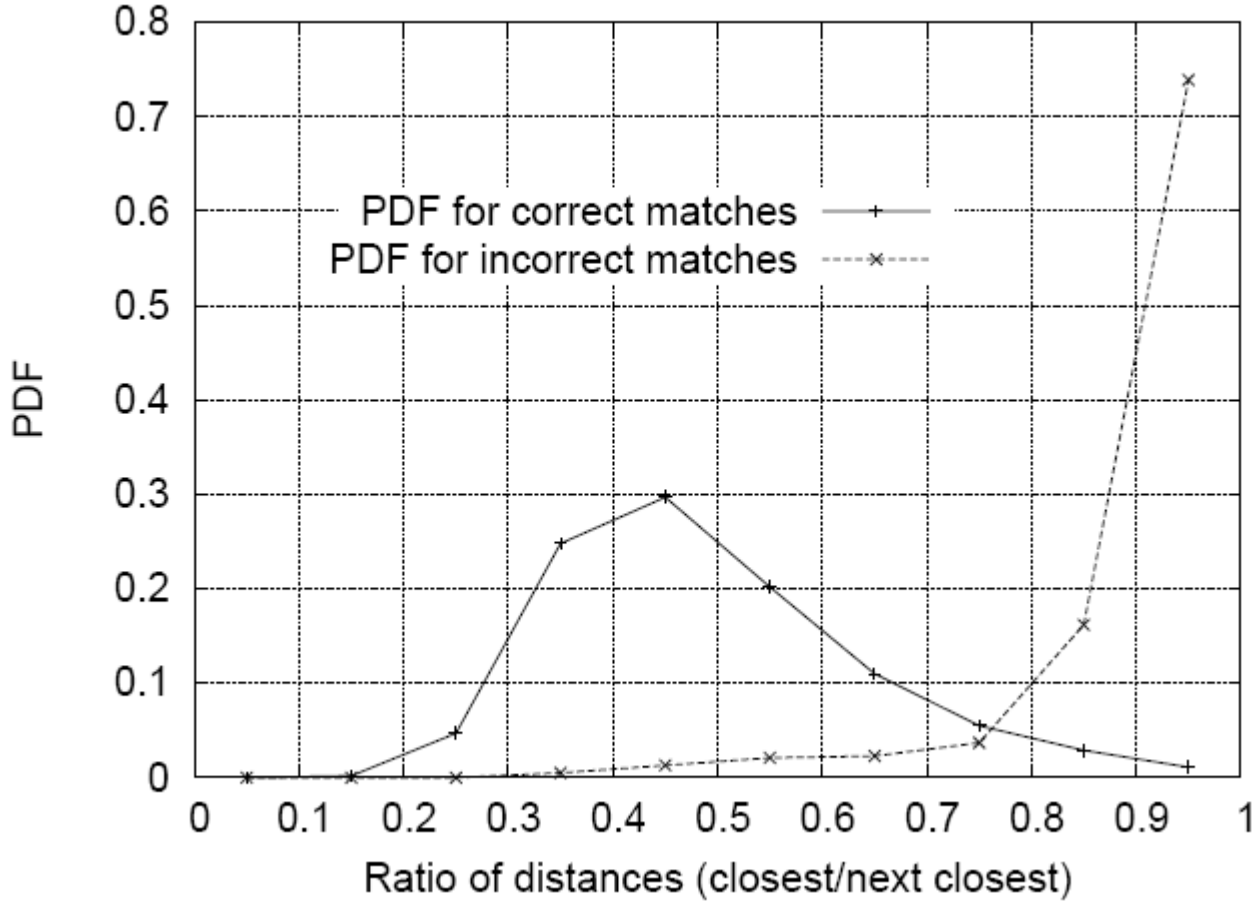
$$\phi = \frac{\|f_i - f_j'^{*}\|}{\|f_i - f_j'^{**}\|}$$

where $f_j'^{*}$ is closest descriptor in I' and $f_j'^{**}$ is 2nd-closest

c. Match f_i to $f_j'^{*}$ if $\phi < 0.8$

Matching SIFT Descriptors

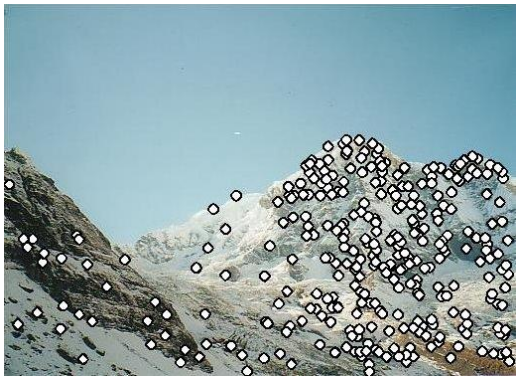
- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2nd nearest descriptor



Lowe IJCV 2004

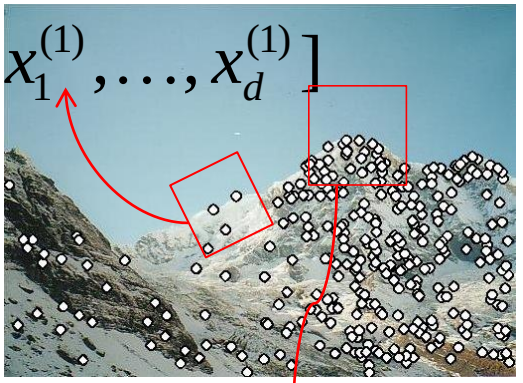
Local features: main components

1) Detection: Identify the interest points



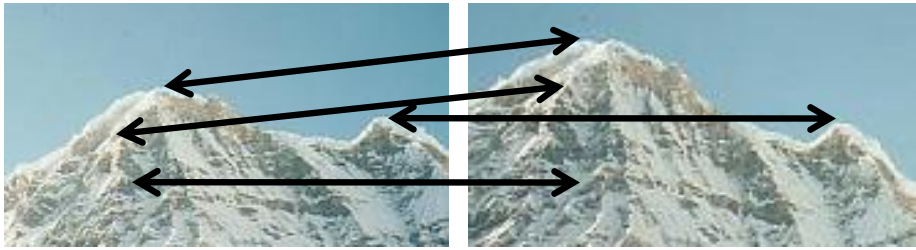
2) Description: Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) Matching: Determine correspondence between descriptors in two views



Slide credit: Kristen Grauman

Value of local (invariant) features

- Complexity reduction via selection of distinctive points
- Describe images, objects, parts without requiring segmentation
  - Local character means robustness to clutter, occlusion
- Robustness: similar descriptors in spite of noise, blur, etc.

Applications of local invariant features

- Wide baseline stereo
- Motion tracking
- Panoramas
- Mobile robot navigation
- 3D reconstruction
- Recognition
- …

Summary

- Interest point detection
  - Harris corner detector
  - Laplacian of Gaussian, automatic scale selection
- Invariant descriptors
  - Rotation according to dominant gradient direction
  - Histograms for robustness to small shifts and translations (SIFT descriptor)